CAN Bus

neue Themen / ToDo

- U2C mit Network Manager einrichten
  - Rene melden
  - https://www.pragmaticlinux.com/2021/07/automatically-bring-up-a-socketcan-interface-on-boot/
  - https://askubuntu.com/questions/1337826/18-04-server-with-can-bus-and-netplan-configuration
- wenn mit CAN geflasht dann keine Kommunikation mehr über die serielle !
- CAN Support für RP2040 in github klipper !
  - inkl. USB Bridge Mode für 2040 !
- Sternverdrahtung vermeiden
  - Beispiel Pi, Spider, Head … Und dann Y Verkabelt mit 3 widerständen …
- Hinweisen das man die Widerstände von den Boards auch ablöten kann
  -
- BTT USB Board per CAN Bus sichtbar?
  - Wie finde ich das im System überhaupt ?
- https://canable.io/
- https://github.com/normaldotcom/candleLight_fw
- Tip: Zeichnung vom BUS machen !
- Kurze Anleitung entlöten ? Mit Entlötkupfer und beide Seiten mit Lötkolben heiß machen …
- https://github.com/kejar31/VoronMods/tree/main/CB-C2
- https://github.com/HyperToxic33/EBB-Fan-Control
- https://github.com/Klipper3d/klipper/blob/master/docs/Multi_MCU_Homing.md
- https://klipper.discourse.group/t/canbus-communication-timeout-while-homing-z/3741/11
  - candump sample von Kevin
- https://maz0r.github.io/klipper_canbus/
- Schaltplan MCP2515 CAN Module TJA1050 https://robu.in/product/mcp2515-can-module-tja1050-receiver-spi-51-single-chip-program-routine-arduino/
- Spider Can Bus Expander https://de.aliexpress.com/item/1005004879963636.html

Einführung

CAN (Controller Area Network) wurde 1986 von Bosch entwickelt um Kabelbäume leichter und effizienter zu gestallten. Das Bussystem ist also prädestiniert dafür in einem 3D Drucker zur Verkabelung eingesetzt zu werden. Die Vorteile liegen klar auf der Hand:

- mehr Störsicherheit als bei serieller Übertragung (durch differentielle Signalleitungen)
- weniger Overhead als bei USB
- deutlich weniger Kabel - mit Zusatzplatine am Druckkopf reichen 4 Leitungen aus (inkl. Strom)
- mehrere Teilnehmer an einen Bus anschließbar (also auch das Drucker Board selber)
- Kommunikation schon über 2 Leitungen realisierbar

Im 3D Druck Bereich setzen sich vermehrt die Head Boards durch. Also kleine Platinen die alle Funktionen des Druckkopfs steuern und regeln können und direkt am Durckkopf angebracht sind. Diese Boards können fast alle mittels CAN angebunden werden. Im besten Fall reduziert dies die Kabel in der Schleppkette zum Kopf auf 4 (2x Strom, 2x CAN Bus).

Das folgende Dokument beschreibt die Einrichtung von CAN am Raspberry PI, zeigt verschiedene Anschlussvarianten auf und gibt zudem tiefergreifende Informationen (auch zur Fehlersuche).

> **Hinweis**: Klipper unterstützt CAN Bus derzeit nur auf STM32 Controllern. Und dann auch nur wenn die Controller CAN Support mitbringen.

Links

- https://www.klipper3d.org/CANBUS.html
- https://de.wikipedia.org/wiki/Controller_Area_Network

Quickstart

Die Doku enthält doch ziemlich viel Detail Informationen die evtl. nicht jeden interessieren. Darum hier ein Quickstart für Eilige ... Die Anleitung basiert auf folgender Hardware:

- Raspberry Pi
- Waveshare CAN HAT
- Spider Board (wobei das Board aber fast irrelevant ist)

Steps to do

- Der Raspberry Pi sollte Klipper installiert haben. Das geht entweder über MainSailOS oder ein Raspberry Pi Lite und KIAUTH.
  - Der Zugriff über Netzwerk / SSH sollte vorhanden sein (Tipp: MobaXTerm verwenden als SSH Client)
- Das Waveshare CAN HAT auf dem Raspberry Pi installieren (im stromlosen Zustand versteht sich)
- Den CAN Bus passend verkabeln. Das Spider Board braucht in meinem Fall einen extra 5V Transceiver. Grundsätzlich aber CAN H jeweils mit CAN H verbinden und das gleiche für CAN L. Eine explizite Masseverbindung braucht es nicht zwingend.
- Sicherstellen das der Bus an beiden Enden mit 120Ω terminiert ist. Das Waveshare Board hat das fest verbaut, das Spider Board auf dem Transceiver auch. Bei einem BTT Kopf Modul ist das jumperbar.
- Raspberry Pi starten und konfigurieren
  - System Update
    `sudo apt update && sudo apt -y upgrade && sudo apt install -y git can-utils ack-grep silversearcher-ag hexedit sudoku tcpdump iptraf mc wavemon`
  - SPI einschalten
    `sudo raspi-config` → 3 Interface Options → P4 SPI → Enable YES
  - MCP2515 Chip aktivieren

`sudo nano /boot/config.txt`
nach der Zeile `dtparam=spi=on` folgendes einfügen
`dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000`
Achtung : ocillator muss passen → siehe hier : https://www.waveshare.com/wiki/RS485_CAN_HAT

- Interface einrichten
  Achtung : die 500000er Bitrate muss auch in der Board Firmware eingestellt werden !
  `sudo nano /etc/network/interfaces.d/can0`
  folgendes eintragen und speichern

```
auto can0
iface can0 can static
    bitrate 500000
    up ifconfig $IFACE txqueuelen 128
```

- Raspberry Neustart
  `sudo reboot`

- Drucker Board flashen
  - `cd ~/klipper`
  - `make clean && make menuconfig`
  - Jetzt das Board einstellen und den richtigen Communication Interface aussuchen. Beim Spider Board ist das `Communication interface (CAN bus (on PD0/PD1))`. Und drauf achten das die Geschwindigkeit mit der Raspberry Config überein stimmt: `(500000) CAN bus speed`. Speichern ....
  - `make -j4`
  - neu compilierte Firmware `out/klipper.bin` auf das Board flashen
  - Nach dem Flashen am besten den ganzen Drucker mal neu starten oder zumindest das Drucker Board resetten.
- Kommunikation testen / Board suchen
  - `~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`
  -

    Wird kein Board gefunden (0 uuids found)? Dann besser mal weiter lesen 😉

    Wird ein Board gefunden (1 uuids found) → CAN Experte 😉
- Klipper Config anpassen
  - Bei der mcu keinen seriellen Port eintragen sondern eine canbus_uuid. Und zwar die aus dem vorherigen Schritt.

  - Save & Restart im MainSail Editor und das Board sollte zum Leben erwecken 🙂

  - Jetzt noch das Board konfigurieren mit seinen Anschlüssen und fertsch 🙂

Bus Struktur

Ein CAN Bus ist im Grunde immer gleich aufgebaut. Es gibt zwei Leitungen die zur Datenübertragung genutzt werden : CAN L(ow) und CAN H(igh). An die beiden Busleitungen werden dann die Knoten / Teilnehmer angeschlossen. Jeder der Knoten (und das könnte z.B. ein Drucker Board sein) hängt über einen Transceiver mit am CAN Bus. Am Anfang und am Ende des CAN Bus ist jeweils ein 120Ω Widerstand platziert. Diese sind für Signalterminierung zuständig und verhindern Reflektionen (und damit Störungen) auf dem CAN Bus.

Aus dem Schaubild ergeben sich daraus schon einige **Dinge die man beachten sollte**:

- Jeder Bus Teilnehmer muss korrekt angeschlossen werden (CAN L vom Teilnehmer an CAN L vom Bus, selbes für CAN H)
- Der CAN Bus muss am Ende jeweils mit 120Ω abgeschlossen sein. Darauf achten das nicht jeder Busteilnehmer 120Ω “mitbringt”. Die Widerstände sind häufig direkt auf den Boards verbaut und können über Jumper deaktiviert werden.
- Die CAN L und CAN H Leitung sollte im besten fall verdrillt sein.
- Jeder Bus Teilnehmer braucht einen Transceiver. Nicht alle Boards haben diesen verbaut - wie z.B. das FYSETC Spider Board.
- Alle Busteilnehmer müssen die gleiche Geschwindigkeit eingestellt haben!
- Um den Raspberry PI an CAN anzubinden braucht es einen extra Adapter (oder ein HAT).

Die meisten Themen werden aber noch im Detail erklärt.

Kabel

Längen

Die CAN Bus Leitung verbindet alle Teilnehmer des CAN Bus miteinander. Bei der Länge ist das aus 3D Drucker Sicht unkritisch - denn selbst mit 500kBit/Sec kann die Buslänge immer noch ca. 100m betragen. Grundsätzlich kann man aber sagen - je länger das Kabel, desto geringer die Übertragungsgeschwindigkeit. Prinzipiell sollten sich also auch mehrere Drucker mit einem CAN Bus betreiben lassen. 🙂 Im Netz gibt es diverse Übersichten was die Maximallänge angeht. Aber ganz grob schaut das immer so aus:

Baud-Rate	Buslänge
1 MBit/s	< 20 m*
500 kBit/s	< 100 m
250 kBit/s	< 250 m
125 kBit/s	< 500 m
50 kBit/s	< 1000 m
20 kBit/s	< 2500 m
10 kBit/s	< 5000 m

Stichleitungen (Stub)

Angaben zum Thema Stub (siehe Schaubild oben) bzw. Stichleitungen findet man nur spärlich. Grundsätzlich sollte man solche **Stichleitungen vermeiden** denn sie führen zu Signalreflektionen (= Störungen). Ein grober Anhaltswert wären ca. 0,5m bei 500kBit/s. Aber wenn eben möglich sollte der

CAN Bus wie an einer Perlenkette aufgebaut werden, also von Gerät zu Gerät verbunden. Nur so ist eine saubere Terminierung sichergestellt.

Links

- Bitrate und Leitungslängen
  https://www.me-systeme.de/de/technik-zuerst/elektronik/can-bus-grundlagen
- Busrate
  https://wiki.ta.co.at/CAN-Bus#Busrate
- Buslänge
  https://infosys.beckhoff.com/index.php?content=../content/1031/cx805x_hw/2037601547.html&id=

Typen

Grundsätzlich sollten die Kabel bei CAN verdrillt sein. Bei guten Kabeln sind die Paare auch zusätzlich geschirmt. Schirmung ist bei den kurzen Distanzen im Drucker aber eher weniger notwendig. Der Querschnitt sollte ebenfalls nicht zu gering sein um den Leitungswiderstand klein zu halten.

Buslänge [m]	Widerstand [mΩ/m]	Querschnitt [mm²]
0 - 40	70	0,25 - 0,34
40 - 300	< 60	0,34 - 0,60
300	< 40	0,50 - 0,60
600 - 1000	< 26	0,75 - 0,80

Für einen 3D Drucker sollte also für die CAN Leitungen 0,25mm² Leitung reichen. Bei den stromführenden Leitungen sieht das etwas anders aus ...

Stromberechnung

Wer noch Strom zum Kopf führen will sollte das vorher mal grob überschlagen. Gehen wir mal von 24V aus und ca. 60W Heizpatrone. Dazu kommen vielleicht nochmal 0,5A an Strom für die Elektronik. Dann bedeutet das wir müssen ca. 60W/24V = 2,5A + 0,5A = ~3A zum Kopf transportieren. Gehen wir weiter davon aus das im Drucker ca. 3m Kabel lieben (und das wäre schon arg lang) dann hat ein Kabel mit 0,5mm² Querschnitt überschlagen ca. 0,11Ω an Leitungswiderstand. Damit lässt sich dann der Spannungsabfall berechnen ... U = R*I = 0,11Ω * 3,0A = 0,33V. Am Kopf kommen also von den 24V ca. 23,7V an. Das passt 🙂

→ http://www.e-formel.info/elektrotechnik/leitungswiderstand.html

Empfehlung

Für den Druckkopf sollte man 4*0,5mm² Leitung verwenden. Zudem sollte das Kabel Schleppkettentauglich sein sonst ist es nach kurzer Bewegung schnell hin. Da es die Igus Kabel nicht an jeder Ecke gibt kann man auch einfach etwas dickere nehmen. Bei Voelkner gibt es z.B. Igus-

CF9.10.04 - also 4×1,0mm². Das passt dann locker und ist auch nur ca. 6mm im Durchmesser.

Passende Leitungen gibt es von IGUS:
0.5mm² → https://www.igus.de/product/CF9?artnr=CF9.05.04
0.75mm² → https://www.igus.de/product/CF9?artnr=CF9.07.04
2 x 0,5mm² → https://www.voelkner.de/products/71738/Igus-CF9.05.02-Schleppkettenleitung-Chainflex-CF-2-x-0.50mm-Blau-Meterware.html
4 x 1mm² → https://www.voelkner.de/products/71853/Igus-CF9.10.04-Schleppkettenleitung-Chainflex-CF-4G-1mm-Blau-Meterware.html

Gleiches an Gleiches

Bei der Verkabelung vom CAN Bus ist darauf zu achten das die Anschlüsse richtig verbunden werden. Der BUS hat immer eine Leitung für CAN L(ow) und eine für CAN H(igh). Das ist auf den Boards oft mit H / L oder CH / CL oder eben CANL / CANH beschriftet. Dabei gilt ... Gleiches mit Gleichem verbinden 🙂
Beispiel mit dem Waveshare Board und dem BTT 42 Kopf Board:
H geht auf CAN-H und L geht auf CAN-L.

CAN Tx / CAN Rx

Es gibt Boards wo der CAN Anschluss mit Tx, Rx oder T, R gekennzeichnet ist. Manchmal findet sich auch die Bezeichnung CAN Tx und CAN Rx. Hier ist Vorsicht geboten denn diesen Anschlüssen fehlt oft der CAN Transceiver. Ein Paradebeispiel ist hier (leider) das Spider Board.
Wie man sieht werden die Pins mit RX und TX bezeichnet - das ist noch **kein CAN Anschluss**! Hier fehlt der Transceiver der die TX, RX Signale in CAN H und CAN L wandelt. Ein passender Adapter findet sich bei Ebay oder Amazon - einfach nach **CAN Transceiver** suchen.

Aber Obacht : Hier muss auf die Spannung geachtet werden. Das Spider Board liefert an seinem CAN Anschluss 5V. Viele CAN Transceiver bei Ebay nutzen aber den SN65HVD230 Chip und der arbeitet mit 3,3V ! Also genau drauf achten was man kauft. Faustregel: Suche nach **TJA1050 CAN Transceiver** liefert 5V Typen, Suche nach **SN65HVD230 CAN Transceiver** liefert 3,3V Typen 🙂

Der Anschluss der Transceiver ist dann trivial. 4 Leitungen gehen vom Board zum CAN Transceiver und der wird mit 2 Leitungen an den CAN Bus angeschlossen (auch hier auf L / H achten !).

Terminatoren

Wie im Schaubild oben zu sehen ist muss der CAN Bus am Ende mit jeweils einem 120Ω Widerstand terminiert werden. Wenn diese Endwiderstände nicht vorhanden sind kommt es auf dem Bus zu Signalreflektionen und damit früher oder später zu Störungen. Man muss allerdings aufpassen wo und

wie viele Widerstände “aktiv” am Bus hängen. Denn viele Boards bringen die Möglichkeit mit diese 120Ω zu aktivieren. Auf einigen Boards (wie z.B. die billigen eBay CAN Transceiver) lässt sich der Widerstand auch gar nicht deaktivieren (mittels Jumper).

Deshalb immer 2x hinschauen wo sich am BUS Kabel Widerstände befinden. Die Widerstände sollten immer am Ende des Kabels sein. Bei einem Drucker ist das oft direkt vorgegeben. Es gibt einen Buskoppler am Raspberry PI und ein Drucker Board z.B. am Druckkopf. Beide haben in der Regel einen 120Ω Widerstand direkt verbaut und der BUS ist damit sauber terminiert. Hängen aber mehrere Teilnehmer am CAN Bus kann das durchaus ein Thema werden.

Im Zweifel Messen

Wer sich unsicher bezüglich der korrekten Terminierung ist kann dies mit einem Digitalmultimeter (Ohm Messung) prüfen. Dazu alle Elektronik erstmal stromlos machen! Dann kann man mit dem Ohmmeter den Widerstand zwischen der L und H CAN Leitung messen. Dabei sollte das Messgerät ~60Ω anzeigen (Das Ergebnis von 120Ω parallel zu 120Ω). Ist das Ergebnis kleiner ist garantiert ein Widerstand zu viel aktiv am Bus. Misst man hingegen 120Ω fehlt ein Widerstand - bei unendlich Ω fehlen beide 😉

CAN Transceiver

Der CAN Transceiver stellt die Verbindung dar zwischen den CAN Daten die geschrieben und gelesen werden sollen und dem physikalischem CAN BUS. Er “übersetzt” sozusagen die Daten in passende Bussignale. Von diesen Transceivern (= Gerät was lesen und schreiben kann) gibt es eine ganze Menge, jedoch sind sie sich alle ziemlich ähnlich - meist kleine 8 pinnige ICs. Hier eine kleine Übersicht :

- SN65HVD23x (Vcc 3.3V)
  https://www.ti.com/lit/ds/symlink/sn65hvd230.pdf
  https://www.reichelt.de/high-speed-can-transceiver-1-mbit-s-3-3-v-so-8-sn-65hvd230d-p58427.html?search=SN65HVD230
- TCAN33x (Vcc 3.3V)
  https://www.ti.com/lit/ds/symlink/tcan330.pdf?ts=1655887915717
- XINLUDA XL1050 (Vcc 5.0V)
  https://www.micros.com.pl/mediaserver/UITJA1050t_XL_0001.pdf
- SN65HVD1050 (Vcc 5.0V)
  https://www.ti.com/lit/ds/symlink/sn65hvd1050.pdf
- MCP2542 (Vcc 5.0V)
  http://ww1.microchip.com/downloads/en/devicedoc/20005514a.pdf
- MCP2551 (Vcc 5.0V)
  http://ww1.microchip.com/downloads/en/devicedoc/21667e.pdf
- TJA1050 (Vcc 5.0V)
  https://www.nxp.com/docs/en/data-sheet/TJA1050.pdf

Versorgung

In einem Punkt unterscheiden sich die ICs aber deutlich und das ist die Versorgungsspannung. Viele der Transceiver brauchen 5V um zu funktionieren, wenige kommen mit 3,3V klar. Es hängt immer von der Schaltung ab wo ein CAN Transceiver eingesetzt wird. Bei einem Spider Board liefert das Board 5V als Versorgung. hier können also nur die Vcc 5,0V Typen verwendet werden. Am Raspberry Pi will man hingegen lieber einen 3,3V Typen verwenden. Denn der ganze Raspberry Pi arbeitet mit 3,3V Pegeln.

Gerade wenn man den Transceiver selber verbaut sollte man hier Obacht walten lassen

Bus Pegel

Die Bus Pegel bei einem CAN Bus sind - unabhängig von der Transceiver Versorgung - immer gleich.
Wenn sich der CAN-Bus im Ruhezustand befindet, führen beide Leitungen 2,5 V. Wenn Datenbits übertragen werden, geht die CAN High-Leitung auf 3,75V und die CAN-Low-Leitung auf 1,25V, wodurch eine 2,5V-Differenz zwischen den Leitungen entsteht.

Boards mit und ohne

Wie schon vorher erwähnt gibt es Boards mit und ohne CAN Transceiver. Ohne Transceiver funktioniert das Board am CAN Bus nicht! Hier hilft nur ein Blick in den Schalplan ob ein extra Transceiver benötigt wird oder nicht.

Beispiel Spider Board
Die Pins gehen direkt an den Controller und sind mit Tx / Rx beschriftet. Hier muss der Transceiver ergänzt werden!

Beispiel Octopus Board
Hier ist ein MCP2542 Transceiver verbaut und rechts sind die Anschlüsse auch mit CAN_H / CAN_L beschriftet. Hier ist alles vorhanden für CAN.

Faustregel
sind die Pins nicht mit CAN H / CAN L (oder H / L) beschriftet braucht es in der Regel einen extra Transceiver!

Geschwindigkeit

Bei einem CAN Bus ist es wichtig das alle Bus Teilnehmer die gleiche Geschwindigkeit benutzen. Sonst können Bus Teilnehmer nicht gefunden bzw. verwendet werden.

Also immer drauf achten das die Geschwindigkeit am **Raspberry Pi** ...
mit der in der **Klipper Firmware** übereinstimmt !

GND / Masse

Zu guter Letzt noch ein sehr kontrovers diskutiertes Thema beim CAN Bus. Muss ich die Masse der CAN Bus Teilnehmer miteinander verbinden? Grundsätzlich kann man sagen - nein. Auf dem CAN Bus werden die Daten differentiell übertragen woraus sich eine Spannungsdifferenz ergibt die ausgewertet werden kann. Als Beispiel dafür mein Testaufbau:

Der PI (links) ist über das Waveshare HAT nur mit 2 Strippen mit dem Drucker Board verbunden (über den Transceiver unten mittig im Bild). Es gibt keine direkte Masseverbindung bei den Boards.

Funktionieren tut das tadellos 🙂

ABER
In der realen Welt teilen sich PI und Drucker Board eh schon die gleiche Masse durch das Netzteil. Also ist es im 3D Drucker Umfeld irrelevant.
Prinzipiell sorgt eine CAN Masse dafür das die CAN Pegel zwischen den Boards nicht zu große Differenzen aufweisen. Das kann bei sehr störbehafteten Umgebungen zu Problemen führen. Aber wie schon erwähnt haben wir im Drucker eh eine gemeinsame Masse und das Thema ist damit abgehakt.

Links

- https://electronics.stackexchange.com/questions/198864/is-a-ground-common-needed-for-proper-can-bus-communication
- https://www.edn.com/does-the-can-bus-need-a-common-ground/

Raspberry PI einrichten

Letztlich gibt es da nicht sehr viel zu tun...

- System Updaten und ein paar Tools installieren
  `sudo apt update && sudo apt -y upgrade && sudo apt install -y git ack-grep silversearcher-ag tcpdump can-utils python3-can`
- SPI einschalten
  `sudo raspi-config` → 3 Interface Options → P4 SPI → Enable YES
- Interface einrichten
  Achtung : die 500000er Bitrate muss auch in der Board Firmware eingestellt werden !
  `sudo nano /etc/network/interfaces.d/can0`
  folgendes eintragen und speichern

```
auto can0
iface can0 can static
    bitrate 500000
    up ifconfig $IFACE txqueuelen 128
```

Alle weiteren nötigen Schritte werden bei den Buskopplern erklärt. Wenn auch die eingerichtet sind empfiehlt sich ein Reboot `sudo reboot`

> **Hinweis** : Werden keine CAN Boards gefunden sollte man die Bitrate hier mit 250000 versuchen. Dann müssen natürlich auch alle Firmware Versionen neu kompiliert werden mit der Geschwindigkeit. Hier kann schlechtes Kabel einem schwer in die Suppe spucken …

Buskoppler

Also Buskoppler werden hier die Geräte gemeint welche den Raspberry PI (oder jeden anderen SBC) in die Lage versetzen mit einem CAN Bus zu kommunizieren.

Grundsätzlich gibt es da 3 Typen:

- Buskoppler mit USB Anschluss
- Buskoppler mit SPI Anschluss
- Drucker Board im Bridge Mode (Experimentell)

Die USB Typen sollen (laut Klipper Doku) stabiler laufen vor allem wenn mehrere CAN Boards angeschlossen werden. Die SPI Typen erzeugen wohl höhere CPU Last am PI. Auf der anderen Seite sind SPI Typen günstiger, leichter zu beschaffen und einfacher in der Handhabung als USB Typen. Die Klipper Entwicklern nutzen wohl auch eher das Waveshare HAT was ein SPI Busklopplerist. Ganz so verkehrt kann das also nicht sein 😉 Hier muss man sicher noch etwas probieren und testen …

Der Bridge Mode ist derzeit in der experimentellen Phase. Ob er in den Hauptzweig von Klipper aufgenommen wird muss sich noch zeigen.

In dieser Klipper Doku finden sich dazu weitere Informationen:
The MCP2515 is a very common SPI connected CAN bus chip. It is a pretty bad options since it has very small buffers on chip and creates a lot of CPU load on the Raspberry Pi, and has a tendency to drop packet. It is not recomended if you use more than 1 or 2 boards or if you plan to use the accelerometer. You have to run the CAN bus at 250kbits/s or possibly 500kbits/s. An older Pi, or a Pi Zero will not work reliably, it has to be a Pi3 or Pi4 or better.

Hier finden sich auch noch weitere Infos zu dem Performance Problem →
https://www.beyondlogic.org/adding-can-controller-area-network-to-the-raspberry-pi/

Waveshare CAN HAT (SPI)

Im Grunde ein schönes Board. Es hat einen SN65HVD230 Transceiver verbaut und ist somit schon mal 3,3V kompatibel mit dem Raspberry Pi. Aber leider hat es auch einen Nachteil - es hat zusätzlich noch einen RS485 Anschluss. Und dieser belegt auch noch den seriellen Port des Raspberry Pi. Das ist arg unpraktisch wenn das Haupt Drucker Board seriell mit dem PI verbunden ist.

Links

- https://www.waveshare.com/rs485-can-hat.htm
- https://www.waveshare.com/wiki/RS485_CAN_HAT

- Manual
  https://www.waveshare.com/wiki/File:RS485-CAN-HAT-user-manuakl-en.pdf
- Schaltplan
  https://www.waveshare.com/w/upload/1/1d/RS485_CAN_HAT_Schematic.pdf

Einrichtung am PI

- MCP2515 Chip aktivieren
  `sudo nano /boot/config.txt`
  nach der Zeile `dtparam=spi=on` folgendes einfügen
  `dtoverlay=mcp2515-can0,oscillator=12000000,interrupt=25,spimaxfrequency=2000000`
  Achtung : ocillator muss passen → siehe hier :
  https://www.waveshare.com/wiki/RS485_CAN_HAT

RS485 entfernen

Dieser Umbau ist nur dann nötig wenn man den seriellen Port am Raspberry PI verwenden möchte. Wer das Drucker Board über USB betreibt müsste hier nichts tun. Grundsätzlich brauche ich den RS485 Part aber so oder so nicht. Also kann auch alles vom Board runter was unnötig Strom frisst ....
Nach dem “Umbau” ...

Läuft tadellos, aber die Garantieansprüche sind dann auch dahin 😎

Tests

Die Tests funktionieren nur wenn der PI eingerichtet wurde, das Waveshare HAT eingerichtet ist und der Raspberry Rebootet wurde !

- `dmesg |grep '251\|can\|spi'` sollte folgende Ausgabe liefert:
- `ip a` liefert ein can0 Interface mit dem Status UP:

2 Channel CAN HAT (SPI)

Ein Board was zwei isolierte CAN Interfaces für den Raspberry PI bereit stellt.

Links

- https://www.waveshare.com/2-CH-CAN-HAT.htm

- https://www.waveshare.com/wiki/2-CH_CAN_HAT
- Schaltplan
  https://www.waveshare.com/wiki/File:2-CH-CAN-HAT-Schematic.pdf

Einrichtung am PI

Siehe Wiki → https://www.waveshare.com/wiki/2-CH_CAN_HAT

Tests

Siehe Wiki → https://www.waveshare.com/wiki/2-CH_CAN_HAT → Testing

EBay Adapter (SPI)

Bei Ebay, Amazon und Co. gibt es für wenige Euro ebenfalls Buskoppler. Diese sind aber nicht als HAT konzipiert sondern einfach als kleine Extraplatine. Die Boards haben nur einen Nachteil wenn man sie mit einem Raspberry PI verwenden will - 5V Versorgung. Der verbaute Transceiver braucht in der Regel 5V Versorgungsspannung. Und das macht ihn ohne weiteres nicht verwendbar für den

Raspberry PI. Der gleich folgende Hack beschreibt wie das dennoch klappt 😉

Links

- https://www.ebay.de/itm/253369172743
- Schaltplan
  https://www.electronicshub.org/arduino-mcp2515-can-bus-tutorial/
  https://robu.in/product/mcp2515-can-module-tja1050-receiver-spi-51-single-chip-program-routine-arduino/

Einrichtung am PI

- MCP2515 Chip aktivieren
  `sudo nano /boot/config.txt`
  nach der Zeile `dtparam=spi=on` folgendes einfügen
  `dtoverlay=mcp2515-can0,`**`oscillator=12000000`**`,interrupt=25,spimaxfrequency=2000000`

Achtung : ocillator muss passen !
Auf dem Board ist ein Bauteil mit silberner Kappe. Das ist der Quarz für den MCP2515 Chip. Da ist eine Nummer aufgedruckt, entweder 8.000 (oscillator=8000000) oder 12.000 (oscillator=12000000). Je nachdem muss bei oscillator der passende Wert eingetragen werden!

Transceiver 3,3V Hack

Tja leider hat die Platine den falschen Transceiver verbaut. Also bleibt nix anderes als den Transceiver zu tauschen. So einen 8 Pinner kann man noch ganz gut von Hand auslöten. Bei Ebay sucht man sich eine Platine mit SN65HVD230 Chip. Den lötet man dann einfach um und fertig. Die Chips sind pinkompatibel.
Alternativ kann man sich auch den Chip bei Reichelt oder Aliexpress bestellen.

Siehe dazu auch hier →
https://www.beyondlogic.org/adding-can-controller-area-network-to-the-raspberry-pi/

Tests

Die Tests funktionieren nur wenn der PI eingerichtet wurde, das Waveshare HAT eingerichtet ist und der Raspberry Rebootet wurde !

- `dmesg |grep '251\|can\|spi'` sollte folgende Ausgabe liefert:
- `ip a` liefert ein can0 Interface mit dem Status UP:

candleLight Adapter (USB)

candleLight ist ein Projekt auf GitHub und stellt eine Firmware für diverse USB <> CAN Adapter bereit. Die Firmware “übersetzt” also den CAN Bus in ein USB Gerät. Mangels Hardware konnte das noch nicht getestet werden.

Wer eine Alternative zum BTT Adapter sucht sollte auf Aliexpress nach “CANable” suchen. Da finden sich für ~25€ passende USB Adapter die mit candleLight funktionieren.

Adapter

- CANable: USB to CAN Adapter
  https://openlightlabs.com/products/canable-0-4
- Innomaker USB CAN
  https://www.amazon.com/Innomaker-Converter-Module-Raspberry-Zero/dp/B07P9JGXXB?th=1
- Aliexpress
  Bsp: https://de.aliexpress.com/item/1005004004826408.html

Links

- candleLight Firmware
  https://github.com/candle-usb/candleLight_fw
- Weitere Infos zu USB Adaptern
  https://github.com/bondus/KlipperToolboard/blob/master/doc/canbus.md

BTT U2C Modul (USB)

Dann gibt es da noch die U2C Boards von BIGTREETECH in zwei Versionen. Die V1.1 hat lediglich noch ein paar mehr Anschlussmöglichkeiten als die V1.0 Version. Technisch sind die aber identisch. Auch diese Boards nutzen die candleLight Firmware. Ansonsten werden die sehr ähnlich eingerichtet wie die SPI Buskoppler. Der Unterschied besteht lediglich darin das kein extra Interface erzeugt werden muss.

Das Board wird später noch genauer getestet - ist aber noch in der Post

Links

- https://de.aliexpress.com/item/1005003746105255.html
- https://de.aliexpress.com/item/1005004263354507.html
- GitHub
  https://github.com/bigtreetech/U2C
- Handbuch
  https://github.com/bigtreetech/U2C/blob/master/BIGTREETECH%20U2C%20V1.0%26V1.1%20User%20Manual.pdf

Drucker Boards

Inzwischen gibt es eine ganze Reihe an Drucker Boards die mit CAN betrieben werden können. Die folgende Liste ist auch sicher unvollständig, sollte aber ein paar nützliche Hinweise liefern für das ein oder andere Board

FYSETC Spider Vx.x

Das FYSETC Spider Board ist in allen Versionen CAN kompatibel. es hat allerdings keinen CAN Transceiver mit auf der Platine. Auf dem Board findet sich nur ein 4poliger Anschluss für CAN:

Von Links nach Rechts sind die Anschlüsse **Masse, Rx, Tx, 5V**. Der CAN Bus liegt auch auf dem EXP1 Anschluss und teilt sich die Pins mit dem Display. Wer also ein Display betreibt ist hier bei CAN leider aus dem Rennen.

Transceiver nachrüsten

Damit das Board überhaupt am CAN Bus funktioniert muss ein Transceiver nachgerüstet werden. Auf jeden Fall einen 5V Typen verwenden!

So wäre der dann zu verkabeln. Ganz Links am Transceiver würde dann CAN L und CAN H

angeschlossen. Und auch hier Obacht bei den Endwiderständen des CAN Bus! Bei dem Adapter im Bild ist der fest verlötet.

Firmware kompilieren

Folgende Einstellungen sind zu treffen:

Wenn ein Bootloader verwendet wird muss der natürlich hier berücksichtigt werden. Die CAN bus speed muss auch zu den Einstellungen am Raspberry Pi passen.

Firmware flashen

Zum Flashen der Firmware nutzt man am einfachsten das Tool dfu-util in der Linux Konsole.

- Jumper beim Spider Board setzen:
- Das Board mittels USB Kabel mit dem PI verbinden
- Das Board resetten (Taster neben dem SD Slot)
- Mit folgendem Kommando flashen
  `dfu-util -R -a 0 -s 0x08000000:leave -D out/klipper.bin`
  Achtung : Dabei wird der Bootloader vom Spider Board gelöscht. Muss man ggf. wieder aufspielen ...

BTT Octopus (Pro)

Das BTT Octopus (Pro) Board hat den Vorteil das bereit ein CAN Transceiver verbaut ist. Genaugenommen ein MCP2542 Transceiver. Der CAN Bus kann also direkt an die Pins CAN_H und CAN_L angeschlossen werden. CAN wird beim Octopus über eine RJ45 Buchse verbunden:

Der CAN Bus ist auf die mittleren beiden Pins der RJ45 Buchse gelegt.

Firmware kompilieren

Folgende Einstellungen sind zu treffen:

Wenn ein Bootloader verwendet wird muss der natürlich hier berücksichtigt werden. Die CAN bus speed muss auch zu den Einstellungen am Raspberry Pi passen.

Firmware flashen

Zum Flashen der Firmware nutzt man am einfachsten das Tool dfu-util in der Linux Konsole.

- Jumper beim Octopus Board setzen:
- Das Board mittels USB Kabel mit dem PI verbinden

- Das Board resetten (Taster neben dem SD Slot)
- Mit folgendem Kommando flashen
  `dfu-util -R -a 0 -s 0x08000000:leave -D out/klipper.bin`
  Achtung : Dabei wird der Bootloader vom Octopus Board gelöscht. Muss man ggf. wieder aufspielen ...

BTT EBB36 & 42 CAN V1.0

Die EBB Boards sind Erweiterungen für den Druckkopf. Sie werden an den Extrudermotor geschraubt und können alle Kopffunktionen bis hin zum Licht übernehmen. EBB36 ist die Variante für NEMA14 Stepper, EBB42 die für NEMA17. Die Boards sind bei Ebay und Aliexpress für ~20-50€ zu bekommen - je nach Ausstattung.

Handbuch, Doku und Schaltplan → https://github.com/bigtreetech/EBB

Firmware kompilieren

Folgende Einstellungen sind zu treffen:

Einen Bootloader braucht man hier nicht installieren da kein SD Slot vorhanden ist.

Firmware flashen

Zum Flashen der Firmware nutzt man am einfachsten das Tool dfu-util in der Linux Konsole.

- Das Board mittels USB Kabel mit dem PI verbinden
- Die Boot Taste am Board gedrückt halten, dann Reset Taste drücken, Reset loslassen, Boot loslassen.
  Die Tasten befinden sich auf der Rückseite des Boards.
- Mit folgendem Kommando flashen
  `dfu-util -R -a 0 -s 0x08000000:leave -D out/klipper.bin`

Shop

- https://de.aliexpress.com/item/1005004243374113.html

BTT EBB36 & 42 CAN V1.1

Hier gibt das Gleiche wie bei Version V1.0. Der Unterschied ist nur ein anderer Prozessor.

> **Achtung**: Wenn man das Board in den Boot Modus (DFU) versetzt um eine neue Firmware zu flashen wird kurzzeitig das Hotend geheizt! Das liegt an der Art und Weise der Controller Initialisierung. Im Zweifel das Hotend kurz abklemmen für ein Update!

Firmware kompilieren

Folgende Einstellungen sind zu treffen:

Einen Bootloader braucht man hier nicht installieren da kein SD Slot vorhanden ist.

Firmware flashen

Zum Flashen der Firmware nutzt man am einfachsten das Tool dfu-util in der Linux Konsole.

- Das Board mittels USB Kabel mit dem PI verbinden
- Die Boot Taste am Board gedrückt halten, dann Reset Taste drücken, Reset loslassen, Boot loslassen.
  Die Tasten befinden sich auf der Rückseite des Boards.
- Mit folgendem Kommando flashen
  `dfu-util -R -a 0 -s 0x08000000:leave -D out/klipper.bin`

Mellow Fly-SHT Boards

Das scheinen mir doch ziemliche Clone von den BTT Boards EBB36 / EBB42 zu sein. Insofern gilt bei denen auch das Gleiche bezüglich Firmware und Anschluss.

Hier gibt es noch weitere Infos:
http://mellow.klipper.cn/?spm=a2g0o.detail.1000023.17.31063265qOYbBR#/board/fly_sht36_42/

Shop

- https://de.aliexpress.com/item/1005004048980837.html

Huvud

Das Huvud Board war wohl eines der ersten CAN Boards für den Druck Kopf. Es gibt inzwischen diverse Versionen und Stände von diesem Board. Das größte Problem ist die Verfügbarkeit - es ist kaum mal irgendwo zu bekommen.

Links

- https://github.com/bondus/KlipperToolboard
- https://hackaday.io/project/174429-huvud-a-3d-printer-tool-head-controller-board
- https://www.tindie.com/products/huvud/huvud-3d-printer-hotend-control-board/
- Halterung
  https://github.com/VoronDesign/VoronUsers/tree/master/printer_mods/120decibell/huvud_chain_mount

Firmware kompilieren

Folgende Einstellungen sind zu treffen:
Weitere Infos zum Flashen finden sich hier:
https://github.com/bondus/KlipperToolboard/blob/master/doc/klipper.md

PandaCAN Extruder

Ein weiteres Board mit CAN. Allerdings noch nie in der Hand gehabt oder gesehen. Deswegen hier nur der Verweis auf die Webseite für weitere Infos.

https://www.pandapi3d.com/product-page/pandacan-extruder

Eigenbau

Auch Board Eigenbauten sind mit Klipper möglich. Hier können alle Controller verwendet werden die einen CAN Bus mitbringen. Zu erkennen ist das immer daran das unter dem Cummunication Interface CAN ausgewählt werden kann.
In diesem Fall ist das ein STM32F103 der u.a. auch auf dem Blue Pill Board verbaut wird.

Um eine Eigenkonstruktion an CAN zu betreiben muss natürlich ein Transceiver her. Hier ist wieder auf die 3,3V bzw. 5V Variante zu achten - was natürlich von dem verwendeten Controller abhängt.

So ein Board könnte man z.B. zur LED Beleuchtung hernehmen. Aber grundsätzlich kann man mit so einem Board alles steuern und Regeln was Klipper so anzubieten hat. Wer da Inspirationen braucht sollte sich mal den Schaltpan der BTT Kopfboards ansehen. Der sitzt der Controller auch drauf 🙂

Links

- https://www.dailyduino.com/index.php/2020/06/01/stm32-can-bus/

Klipper Konfig

Die Konfiguration von Klipper mit einem CAN Board unterscheidet sich nur in einem Punkt - die **canbus_uuid**. Boards die nicht über CAN betrieben werden nutzten einen seriellen Port für die Kommunikation mit dem Raspberry PI:

```
[mcu]
restart_method          : command
serial                  : /dev/ttyAMA0
```

Wir CAN verwendet ändert sich nur die serial Zeile zu canbus_uuid:

```
[mcu]
restart_method           : command
canbus_uuid              : 5b5a812a7283
```

Bleibt noch die Frage wie man an die UUID ran kommt. Dafür gibt es Klipper Python Script

- `sudo systemctl stop klipper.service`
- `~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`
  Wenn ein Board gefunden wird, dann sollte folgende Ausgabe erscheinen:
  
- `sudo systemctl start klipper.service`

Kommt hier die Meldung `Total 0 uuids found` wurde kein Board am CAN Bus gefunden. Das kann leider eine ganze Menge an Gründen haben:

- Reset Taste an den Drucker Boards drücken und nochmal versuchen
- Firmware falsch oder nicht korrekt aufgespielt (oder gar nicht aufgespielt)
- Firmware ohne CAN Support kompiliert
- CAN Bus falsch verdrahtet
- CAN Bus ohne die 120Ω Widerstände am Ende der Busleitung
- Kein Transceiver auf dem Board bzw. keiner extra angeschlossen
- CAN Anschluss am PI läuft nicht sauber
- CAN Bus Speed ist zu schnell für das Kabel
- Das `canbus_query.py` Skript ist auch eher von launischer Natur. Es findet die Boards teilweise nicht obwohl sie mit Klipper funktionieren. Also vor canbus_query.py immer erst den Klipper Dienst stoppen die Boards resetten!

Wer hier hängt sollte sich das Kapitel Fehlersuche mal genauer durchlesen.

Beispiel 3 CAN Geräte

Nachdem am Anfang schon ein “einfaches” Beispiel mit einem CAN Bus Board war kommt hier ein Beispiel mit 2 CAN Boards. Das Beispiel setzt praktisch das Schaubild vom Anfang um. Wir haben also folgende Komponenten:

- Raspberry PI mit Waveshare CAN HAT
- Spider Board V1.1 mit extra angeschlossenem Transceiver
- BTT EBB42 CAN V1.0 Board für den Druckkopf

Betrachtet wird nur der Teil bis die Boards in und mit Klipper funktionieren. Die Konfiguration ist hier nicht Bestandteil - sollte aber normalerweise auch kein Problem darstellen.

Raspberry Pi einrichten

Wie der Raspberry PI eingerichtet werden muss steht in diesem Kapitel.

Firmware Spider Board

Wie ein Spider Board mit Firmware geflasht wird steht in diesem Kapitel.

Firmware BTT Board

Wie ein BTT Board mit Firmware geflasht wird steht in diesem Kapitel.

Bus verkabeln

Wie schon oben im Schaubild zu sehen ist werden alle 3 BUS Teilnehmer wie an einer Perlenschnur miteinander verbunden.

(1) ist der CAN Bus. Das schlimmste Kabel was man verwenden kann, aber so findet man Fehler
Es ist aber wie beschrieben von CAN Device zu CAN Device einfach durchgeschliffen.
(2) ist der Transceiver für das Spider Board.

Viel mehr ist es nicht. Man sollte nur peinlichst drauf achten das jeweils CAN_L an CAN_L ist und gleiches für CAN_H.

Busteilnehmer suchen

Die Busteilnehmer muss man suchen, weil man die UUIDs für die Konfiguration in Klipper braucht. Das Kommando dazu lautet folgendermaßen:
`~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`
Als Ergebnis müsste dann eine Ausgabe wie diese kommen:

Welches Board nun zu welcher UUID gehört kann man mit dieser Ausgabe nicht ermitteln. Hier hilft nur raten oder die Boards einzeln am CAN Bus anschließen und auslesen.

Problematisch ist das `canbus_query.py` Skript bei der Benutzung. Es gibt Situationen wo die UUIDs nicht angezeigt werden und das obwohl die Boards definitiv funktionieren. Deswegen folgendermaßen vorgehen:

- Klipper Dienst stoppen
  `sudo systemctl stop klipper.service`
- Jetzt alle Drucker Boards resetten. Dazu haben eigentlich alle Boards immer eine extra Taste verbaut.
- Erst danach den bus scannen
  `~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`

Wer hier hängt sollte sich das Kapitel Fehlersuche mal genauer durchlesen.

printer.cfg canbus_uuid

Bleibt noch die printer.cfg mit passenden Einträgen zu versehen. Hier ist für einen ersten Test nur die Minimal Konfig abgebildet:
Wie schon vorher beschrieben gibt es hier keinen Eintrag für serial.

Nach einem Restart von MainSail sollte dann das Ergebnis so aussehen:
(1) ist das Spider Board (zu erkennen an den 180MHz)
(2) ist das BTT EBB42 Kopf Board

CAN Bus Bridged Mode

Es gibt seit kurzer Zeit einen neuen Modus für CAN fähige STM32 Boards - den Bridge Mode. Dabei kann man sich am Raspberry PI das extra CAN HAT ersparen. Die Aufgabe übernimmt ein Stückchen Software in der Drucker Board Klipperfirmware. Das schaut dann in etwa so aus:

Auch wenn dieser neue Modus erstmal sehr interessant klingt, bringt er doch ein paar Einschränkungen mit sich:

- Der Bridge Mode ist nur auf “potenten” STM32 Controllern verfügbar. Der Controller muss dafür USB und CAN Bus parallel betreiben können. Kleiner Controller - wie der oft verbaute STM32F103 - sind da leider raus.
- Der Controller der im Bridge Mode läuft erscheint nicht als CAN Bus Gerät. Er kann zudem auch nicht direkt mit anderen CAN BUS Teilnehmern kommunizieren - das geht immer über den Raspberry PI und den USB Port.
- Die can0 Schnittstelle muss trotzdem konfiguriert werden. Timings werden aber ignoriert und nur die Timings von `make menuconfig` sind relevant.
- Immer wenn der Bridge Controller zurückgesetzt wird, deaktiviert sich gleichzeitig auch die can0 Schnittstelle in Linux. Man kann das über `sudo ip link set up can0` wieder reaktivieren, jedoch gibt es dafür derzeit noch keinen Automatismus.

Wer das zur Zeit testen möchte muss den git Branch “work-usbcan-20220608” verwenden. Den bekommt man so:

- `cd ~/klipper`
- `git checkout work-usbcan-20220608`
- Jetzt ist der neue Bridge Mode bei `make menuconfig` verfügbar:
- Zurück zum normalen Klipper Branch kommt man danach wieder mittels `git checkout master`

Könnte einen spannende Sache werden, denn es erspart das CAN Interface am Raspberry PI. Allerdings sollte man warten bis diese Funktion in den Hauptzweig von Klipper eingefügt wurde.

Derzeit ist es eher ein experimentelles Feature mit Potential

Links

- https://github.com/Klipper3d/klipper/pull/5583
- https://klipper.discourse.group/t/experimental-usb-to-canbus-bridge-mode/3136
- https://github.com/Klipper3d/klipper/blob/work-usbcan-20220608/docs/CANBUS.md#usb-to-can-bus-bridge-mode

Flashen per CAN

Es gibt seit einiger Zeit ein sehr interessantes Projekt namens “CanBoot”. CanBoot ist ein alternativer Bootloader für STM32 Controller. Er ermöglicht das Flashen von Klipper über den CAN Bus. Es muss also weder ein USB- oder serielles Kabel angeschlossen werden, noch muss ein Jumper gesetzt oder eine SD Karte eingesteckt werden. Finden kann man dieses Wunderwerk der Entwicklung hier: https://github.com/Arksine/CanBoot

> **Hinweis**: In den vorherigen Kapiteln wurde immer beim Flashen auf einen Bootloader verzichtet. Damit CanBoot aber funktionieren kann sind ein paar zusätzliche Schritte nötig. Der Bootloader muss zuerst geflasht werden wie jede andere Firmware auch. Erst danach kommt man in den Genuss die Klipper Firmware über CAN aufzuspielen.

Download

Um CanBoot zu nutzen muss zunächst einmal der Code geladen werden:

- `cd ~`
- `git clone https://github.com/Arksine/CanBoot`
- `cd CanBoot`

Controller Konfig

Dann muss der Controller konfiguriert werden - ähnlich wie beim Klipper kompilierern:

- `make menuconfig`
- Für ein EBB46 V1.0 Board schaut das dann so aus
- Abschließend dann den Bootloader compilieren mit `make`

> **Hinweis**: Nur es nochmals klar darzustellen - auch wenn es so aussieht ist das nicht Klipper! Es ist ein Bootloader um später Klipper über CAN flashen zu können.
>
> **Wichtig**: Es ist zwingend erforderlich sich den Punkt “Application start offset” zu merken. Wenn später Klipper kompiliert wird muss das als **Bootloader Offset** angegeben werden! Sonst würde nämlich der frisch installierte CanBoot Bootloader wieder überschrieben.

CanBoot flashen

Der nächste Schrott besteht dann darin den frisch kompilierten Bootloader auf das Drucker Board zu

flashen. Für diesen Test ist das ein EBB46 V1.0 Board. Aber das sollte mit allen anderen STM32 Controllern ganz genauso funktionieren.

- Das Board per USB mit dem Raspberry PI verbinden
- Das Board in den DFU Modus bringen.
  - Beim BTT Board geht das über die folgende Button Reihenfolge: Boot gedrückt halten, Reset drücken und loslassen, Boot loslassen.
  - Bei anderen STM32 Boards muss der Jumper bei BOOT0 oder auch BT0 gesetzt werden.
  - **Tipp**: Ob der Controller im DFU Modus ist lässt sich mit diesem Befehl ermitteln: `dmesg` . Am Ende sollte in etwa folgende Anzeige kommen:
    BOOTLOADER meint in diesem Fall aber den in Hardware realisierten STM32 Bootloader - nicht CanBoot
- Dann den Bootloader flashen
  `dfu-util -R -a 0 -s 0x08000000:mass-erase:force -D out/canboot.bin`
- Nach dem Flashen des CanBoot Bootloader ist es immer eine gute Idee mal den Reset Button vom Board zu drücken.

Board suchen

Nachdem das Board nun den CanBoot Bootloader installiert hat muss es am CAN Bus auch gefunden werden. Dazu gibt es ein extra Python Script im CanBoot Verzeichnis `flash_can.py`.

- Boards am CAN Bus suchen die CanBoot installiert haben
  `~/CanBoot/scripts/flash_can.py -i can0 -q`
- Die UUID - in diesem Fall 539892be834d - brauchen wir für das Flashen von Klipper später

Alternativ geht die Board suche auch mittels `canbus_query.py`

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

Board flashen

Bleibt noch der letzte Schritt - Klipper über CAN zu flashen.

- Das Board wird wie gehabt konfiguriert
  - `cd ~/klipper/scripts/`
  - `make menuconfig`
    Achtung: Hier muss jetzt der **Bootloader offset** eingestellt werden so wie er bei CanBoot eingestellt wurde !
  - Abschließend ein `make` zum Kompilieren der klipper Firmware
- Dann das Board mittels CAN flashen
  - `cd ~/CanBoot`
  - `python3 flash_can.py -i can0 -f ~/klipper/out/klipper.bin -u <uuid>`

In diesem Beispiel wäre das dann
`python3 flash_can.py -i can0 -f ~/klipper/out/klipper.bin -u 539892be834d`
- Das Ergebnis:

- Kleine Überprüfung ob alles geklappt hat ...
  - `~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0` liefert als Ergebnis danach dann
    `Found canbus_uuid=539892be834d, Application: **Klipper**`
  - Passt also

Das tolle an diesem Bootloader ist, dass man am Drucker Board nichts mehr machen muss wenn er einmal aufgespielt ist. Also keinen Taster drücken für DFU Mode, keine SD Karte, nichts. Einfach nur erneut das Flash Skript aufrufen mit der neuen Klipper Firmware und alles geht wie von Zauberhand.

Eine enorme Vereinfachung für den Flashvorgang

Fehlersuche

Wie schon etwas weiter oben im Text aufgezeigt, kann es eine ganze Reihe an Gründen / Problemen geben warum der CAN Bus nicht sauber funktioniert. Dafür im Folgenden ein paar praktische Tipps zur Fehlersuche ...

https://github.com/Esoterical/voron_canbus/tree/main/troubleshooting

Hardware

CAN Bus falsch verdrahtet

Auch wenn der CAN Bus nur 2 Leitungen benötigt muss auf die Verdrahtung geachtet werden. CAN L gehört immer an CAN L, CAN H immer an CAN H. Das gilt für **alle** Geräte am CAN Bus!

CAN Bus ohne die 120Ω Widerstände am Ende der Busleitung

Am Ende des CAN Bus muss jeweils ein 120Ω Widerstand für die Terminierung vorhanden sein. Diese beiden Widerstände hängen am *äußersten Ende der Busleitung* jeweils zwischen CAN L und CAN H. Prüfen kann man die Widerstände mit einem Ohmmeter. Wenn die Hardware (= der Drucker) komplett stromfrei ist, einfach zwischen CAN L und CAN H den Widerstand messen:

- ~60Ω und alles ist im Lot
- ~120Ω und es fehlt ein Abschlusswiderstand
- unendlich Ω und es ist gar kein Widerstand vorhanden
- alles kleiner 60Ω (also genauer so ab 40Ω und tiefer) deutet darauf hin das zu viele Abschlusswiderstände aktiv sind

CAN Bus Speed ist zu schnell für das Kabel / schlechtes Kabel

Für den CAN Bus sollte im Bestfall gedrilltes Kabel verwendet werden. Auch die Dicke der Adern sollte nicht zu gering sein. Sind diese Kriterien nicht erfüllt kann es vorkommen das der CAN Bus nicht sauber funktioniert. Geräte werden dann z.B. nicht sauber oder nur sporadisch erkannt. In einem solchen Fall sollte man die Busgeschwindigkeit herunter setzen auf 250000 Bit/s (oder auch noch

tiefer wenn nötig - siehe dazu Tabelle am Textanfang).

Einstellen kann man die Bus Geschwindigkeit in der Datei `/etc/network/interfaces.d/can0` → bitrate

> **Achtung**: Wenn die Busgeschwindigkeit angepasst wird muss auch jder Busteilnehmer mit einer neuen Firmware geflasht werden (CAN bus speed)!

Kein Transceiver auf dem Board bzw. keiner extra angeschlossen
Einige Boards haben keinen Transceiver verbaut - wie z.B. das Spider Board. Hier muss ein extra Transceiver angeschlossen werden der auch zur Spannungsversorgung des Boards passen muss. Hier sollte man ggf. im Schaltplan des Boards nachsehen ob ein Transceiver verbaut ist.

> **Faustregel**: Sind die CAN Pins auf dem Board nicht mit CAN H, CAN L sondern eher mit Tx, Rx beschriftet fehlt in der Regel der Transceiver!

SPI nicht aktiviert
Damit die SPI basierten Buskoppler funktionieren muss auf dem Raspberry PI die SPI Schnittstelle eingeschaltet werden. Das geht über `sudo raspi-config` → 3 interface Options → P4 SPI. Nach dem einschalten den PI einmal durchbooten. Äußern kann sich das dann in der Meldung `mcp251x spi0.0 **can0: bus-off**` wenn man den Befehl `dmesg |grep '251\|can\|spi'` absetzt.

SPI mcp251x funktioniert nicht
Fast alle SPI basierten Buskoppler basieren auf dem Chip mcp251x. Damit dieser sauber funktioniert muss er in der Konfiguration vom PI eingerichtet werden. Siehe dazu die Buskoppler Infos. Wenn der Chip sauber läuft liefert der Befehl `dmesg |grep '251\|can\|spi'` in etwa folgende Ausgabe:

Software

Firmware falsch oder nicht korrekt aufgespielt (oder gar nicht aufgespielt)
Es sollte natürlich sichergestellt sein das die richtige Firmware für das Board eingerichtet und kompiliert wurde. Tückisch kann dabei ein vorhandener Bootloader sein (Stichwort Bootloader offset) oder eine falsch eingestellte CPU und / oder Clock Reference. Hier also genau auf die richtigen Einstellungen achten.

Manchmal kann es auch helfen vor dem Schreiben der Firmware den Flash Speicher zu löschen. Dann ist der Aufruf von dfu-util leicht anders. Anstatt wie sonst üblich mit leave
`dfu-util -R -a 0 -s 0x08000000:leave -D <FIRMWARE FILE>`
wird jetzt mass-erase:force verwendet
`dfu-util -R -a 0 -s 0x08000000:mass-erase:force -D <FIRMWARE FILE>`

Firmware ohne CAN Support kompiliert
Essentiell beim Kompilieren neuer Klipper Firmware ist natürlich der CAn Support. Dazu das richtige Communication interface auswählen. Hier besonders auf die verwendeten Controller Pins achten. Zur Not im Schaltplan des Boards nachsehen welche Pins genau für CAN reserviert wurden.

Tools finden keine CAN Geräte
Es kommt vor das **canbus_query.py** wie auch **flash_can.py** keine CAN Geräte am Bus finden. Hier ist es mitunter sinnvoll mal die Reset Taste der Druckerboards zu drücken und die Suche zu

wiederholen. Zudem ist es ratsam vor der Nutzung dieser Tools den Klipper Dienst zu stoppen:
`sudo systemctl stop klipper.service`
Das stellt sicher das keine Software die Kommunikation mit dem Drucker Board stört. Abschließend den Klipper dienst natürlich wieder starten:
`sudo systemctl start klipper.service`

CAN Anschluss am PI läuft nicht sauber
Wenn der CAN Bus am PI nicht verfügbar ist, oder nicht korrekt läuft liegt das zu 99% an einer falschen Konfiguration:

- entweder ist die Datei `/etc/network/interfaces.d/can0` fehlerhaft oder nicht vorhanden
- oder aber SPI ist nicht eingeschaltet und / oder der mcp251x funktioniert nicht

Prüfen kann man das mittels `dmesg |grep '251\|can\|spi'` (siehe oben unter Fehlersuche → Hardware)

Falsche txqueuelen in /etc/network/interfaces.d/can0
Es wird in manchen Dokus empfohlen die txqueuelen in der Datei `/etc/network/interfaces.d/can0` zu erhöhen (teilweise auf 1024). Das hat bei meinen Tests nicht immer geklappt. Teilweise war das can0 Interface im Raspberry PI dann down und somit der CAN Bus nicht funktional. Standard ist jedenfalls 128.

Die txqueuelen kann auch im Betrieb (bis zum nächsten Reboot) neu gesetzt werden:
`sudo ifconfig can0 up txqueuelen 1000` oder alternativ
`sudo ip link set txqueuelen 1000 can0`
Könnte für Tests manchmal ganz hilfreich sein …

can0 Interface down
Es ist immer eine gute Idee den Status des can0 interfaces zu prüfen. Das geht recht einfach mittels `ip a`:
In diesem Fall ist das can0 Interface UP = betriebsbereit. Und hier funktioniert es auch mit 1024 für die txqueuelen.

Sollte das can0 Interface mal auf DOWN stehen kann man versuchen es mittels
`sudo ip link set up can0`
wiederzubeleben. Möchte man das Interface (wozu auch immer) abstellen geht das mittels
`sudo ip link set down can0`

Interface auf UP setzen mit bitrate
`sudo ip link set can0 up type can bitrate 500000`

Bitrate testweise umsetzen
`sudo ip link set can0 type can bitrate 125000`

Tools

Es gibt unter Linux (und Klipper) eine ganze reihe an Tools, um den CAN Bus zu untersuchen.

Klipper - canbus_query.py
Das Skript canbus_query.py listet alle Klipper und CanBoot Geräte am CAN Bus auf. Das ist dann von

Nöten wenn die UUIDs für die Klipper Konfig ermittelt werden müssen. Ganz nebenbei zeigt es aber auch recht simple ob die Geräte am Bus funktionell sind.
`~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`
Die Ausgabe ist dann folgendermaßen:

> **Achtung**: canbus_query.py ist ein eher “launisches” Tool. Manchmal werden keine Boards gefunden wenngleich sie funktional am Bus hängen.

Beste Vorgehensweise

- Klipper Dienst stoppen
  `sudo systemctl stop klipper.service`
- Alle Boards am Bus einmal resetten über die Hardware Taste am Board
- Dann die Suche ausführen
  `~/klippy-env/bin/python ~/klipper/scripts/canbus_query.py can0`
- Klipper Dienst wieder starten
  `sudo systemctl start klipper.service`

Klipper - console.py
ein weiteres Klipper Skript ist console.py. Es ist nur schwer in Dokumentationen zu finden, aber im Grunde sollte es jeder kennen. Es stellt eine Verbindung zum Drucker Board her und zeigt die Kommunikation an. Damit kann man nicht nur testen ob das Board sauber mit der Klipper Firmware arbeitet. Damit lässt sich auch schnell die Firmware Version auslesen. Anmerkung am Rande … Das Tool funktioniert auch mit seriellen Verbindungen.

Beste Vorgehensweise

- Klipper Dienst stoppen
  `sudo systemctl stop klipper.service`
- evtl. das Baord einmal resetten über die Hardware Taste
- `~/klippy-env/bin/python ~/klipper/klippy/console.py -c can0 <CAN_UUID>`
  - Hinten steht die CAN UUID die man mit canbus_query.py ausgelesen hat
- Klipper Dienst wieder starten
  `sudo systemctl start klipper.service`

CanBoot - flash_can.py
Das nächste interessante Skript stammt vom CanBoot Projekt. Es kann ebenfalls alle CAN Geräte am Bus finden.

Beste Vorgehensweise

- Klipper Dienst stoppen
  `sudo systemctl stop klipper.service`
- Alle Boards am Bus einmal resetten über die Hardware Taste am Board
- `~/CanBoot/scripts/flash_can.py -i can0 -q`
- Klipper Dienst wieder starten
  `sudo systemctl start klipper.service`

Linux - can-utils

```
sudo apt-get install can-utils
sudo cansniffer can0
```

Linux - tcpdump

```
sudo apt-get install tcpdump
sudo tcpdump -i can0
```