

Thunderbird



... ist eine customized firmware für den Yuneec Typhoon H (aka H480) auf Basis vom PX4 Autopilot, entwickelt von **Toni Rosendahl**. Einführung siehe hier:

<https://yuneecpilots.com/threads/typhoon-h-480-px4-v1-10-stability-issues.18205/page-3>

Die Firmware ist Open Source. Alle können mitmachen.

Das Projekt wird hier gepflegt: <https://github.com/tonirosendahl/Thunderbird>

PX4 Autopilot: <https://docs.px4.io/>

Wie flashe ich ein MCU-Board vom Typhoon H

Achtung: Wenn der Bootloader einmal aufgespielt wurde, gibt es kein zurück mehr. Der Typhoon H ist dann ein PX4 Thunderbird.

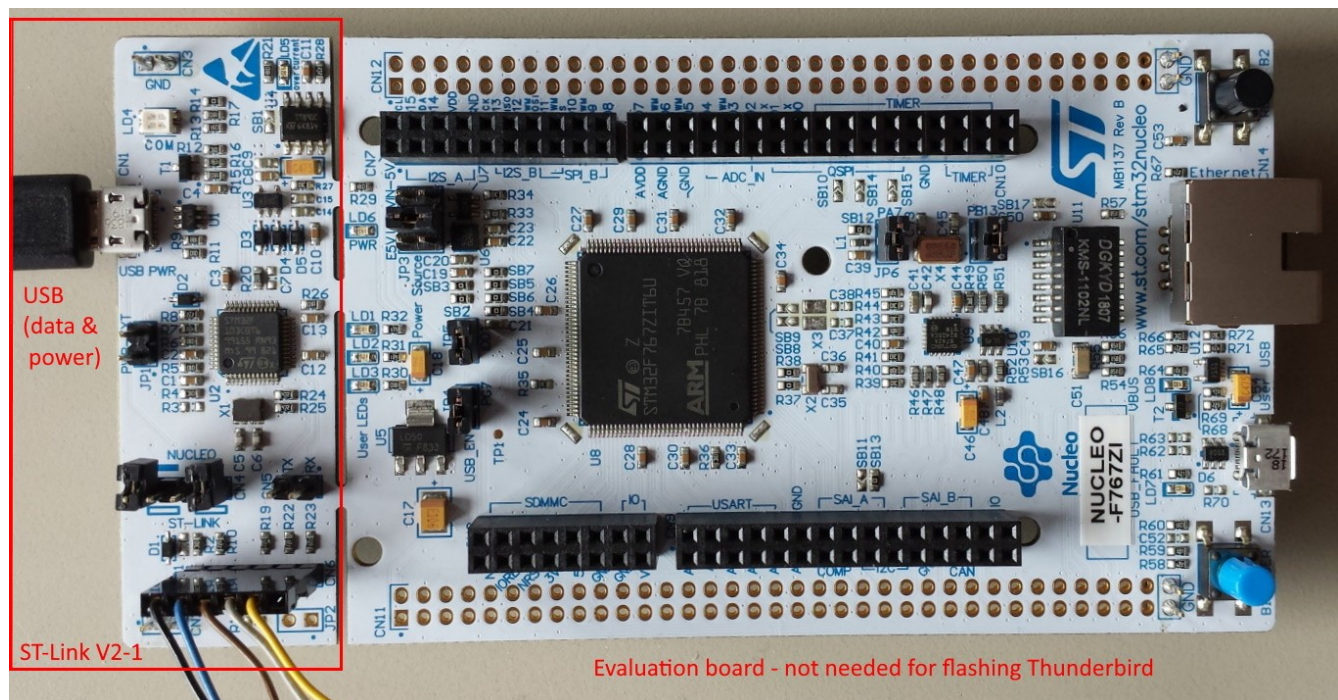
Was wir brauchen:

Binaries zum Flashen downloaden und auspacken:

https://github.com/tonirosendahl/Thunderbird/blob/Typhoon_H_480/Thunderbird_19122019_FT.zip

Einen Programmer für STMxx mit einem **ST-Link V2-1**. So etwas wie das:

<https://www.st.com/en/evaluation-tools/nucleo-f767zi.html>

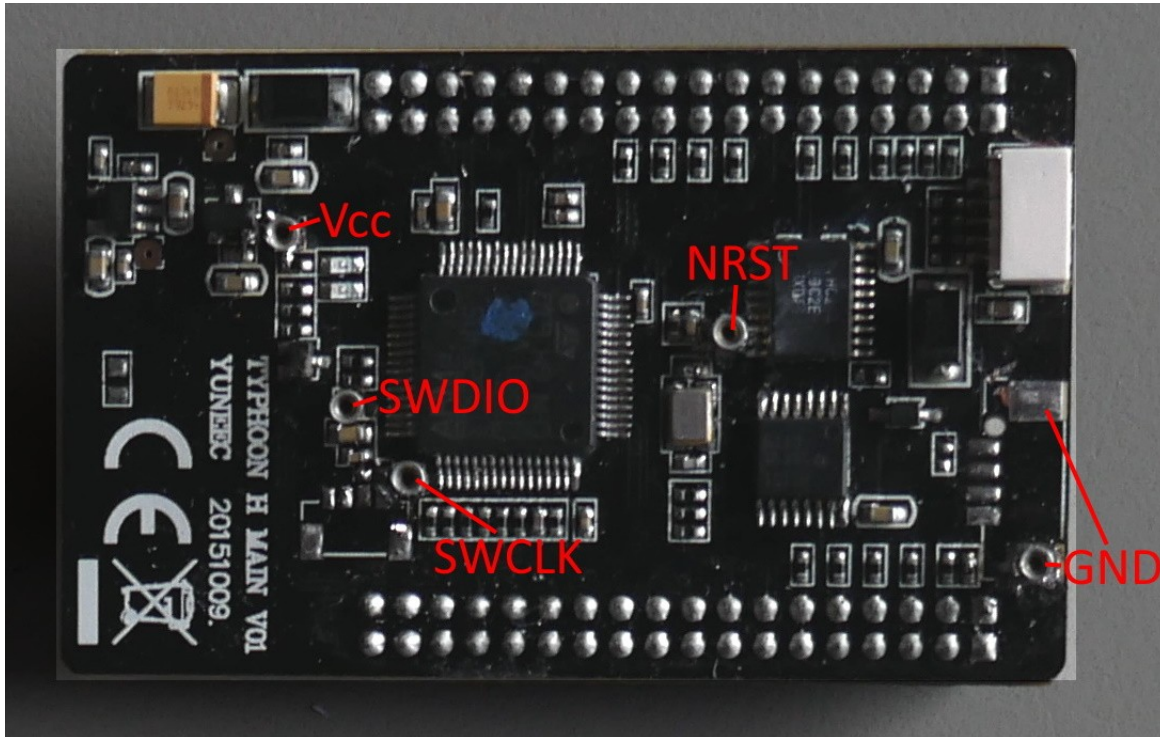


Dafür müssen wir noch das ST-Link utility downloaden und installieren.

<https://www.st.com/en/development-tools/stsw-link004.html>

Und natürlich brauchen wir ein MCU-Board vom Typhoon H. Am besten man hat ein Ersatz Board. Dann können wir nämlich durch umstecken eines normalen Boards auf den alten Typhoon H zurückfallen.

Test: Zuerst das Board auf Funktion prüfen. Beim Anschließen muss es hochlaufen, zu erkennen an den LEDs. Im Windows Geräte-Manager erscheint es als „STMicroelectronics Virtual COM Port (COM...)“.



Um die Parameter einzustellen und spätere Firmwareupdates zu machen müssen wir "QGroundControl" downloaden und installieren. Starten sollte man das Tool als Administrator (nötig jedenfalls bei Windows 10).

<http://qgroundcontrol.com/downloads/>

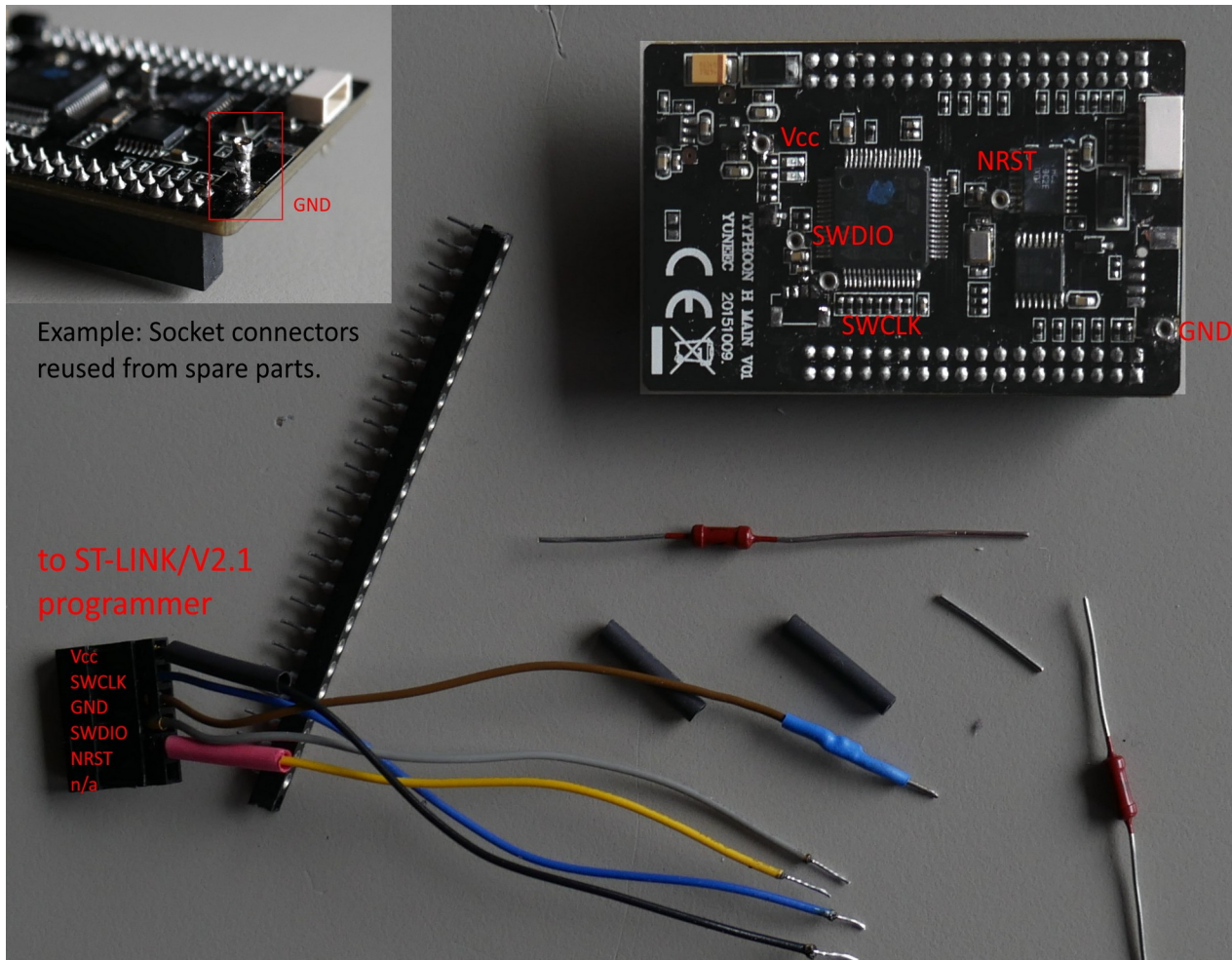
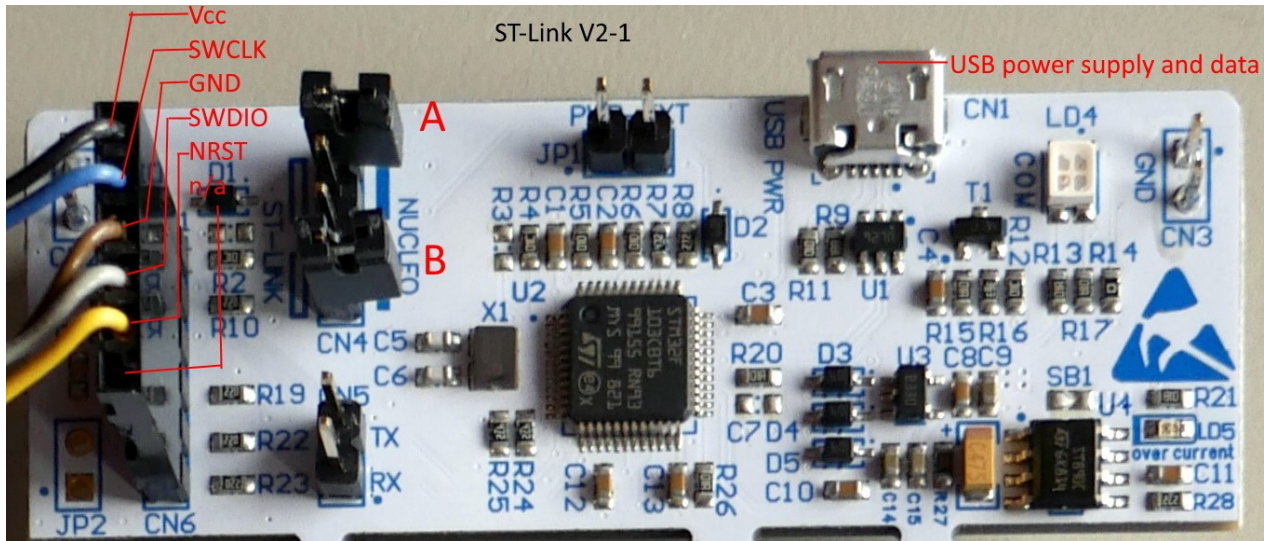
Achtung:

Bevor wir weitermachen, lesen wir erst die Dokumentationen für das Zeug oben.

Bitte auch die ESD-Regeln beachten. Alles erden, was geht.

Schritt 1: Den Programmer anschließen (der kritischste Schritt)

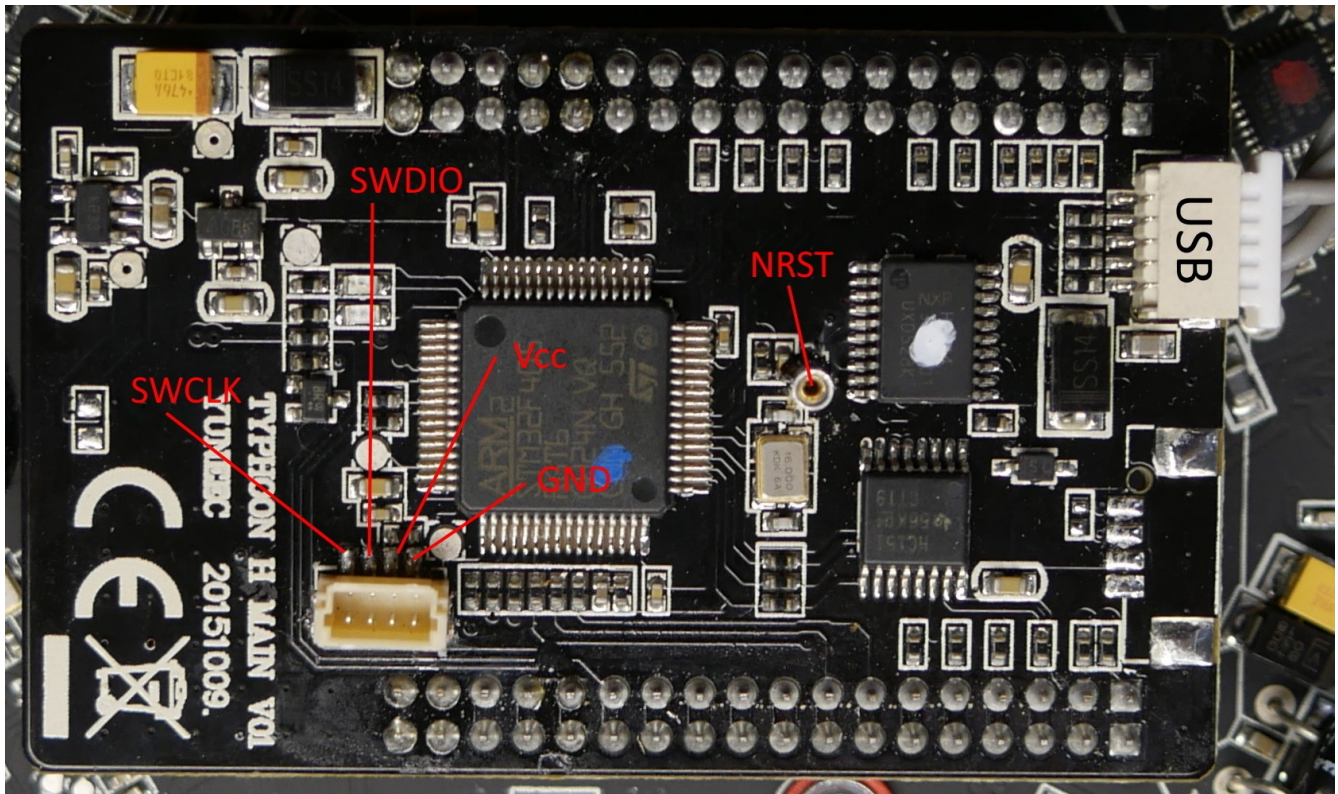
Wir brauchen fünf Anschlüsse: Ground (GND), +3.3V (Vcc), clock (SWCLK), data (SWDIO) und reset (NRST). Für alle diese Anschlüsse gibt es kleine, runde Meßpunkte auf dem MCU-Board. Nun ist es an euch, wie ihr diese mit dem SWD-Port des ST-Links verbindet. Im einfachsten Fall dünne Drähte anlöten oder, wie ich, kleine Buchsen aus alten IC-Fassungen.



Example: Socket connectors reused from spare parts.

to ST-LINK/V2.1
programmer

Man kann auch die vorgeleisteten Anschlüsse für einen Steckverbinder nutzen. NRST muss extra noch angeschlossen werden.



Der Steckverbinder sollte mit einem Tropfen Sekundenkleber fixiert werden. Bitte immer Vorsicht beim Stecken und besonders beim Ziehen der Kabel.

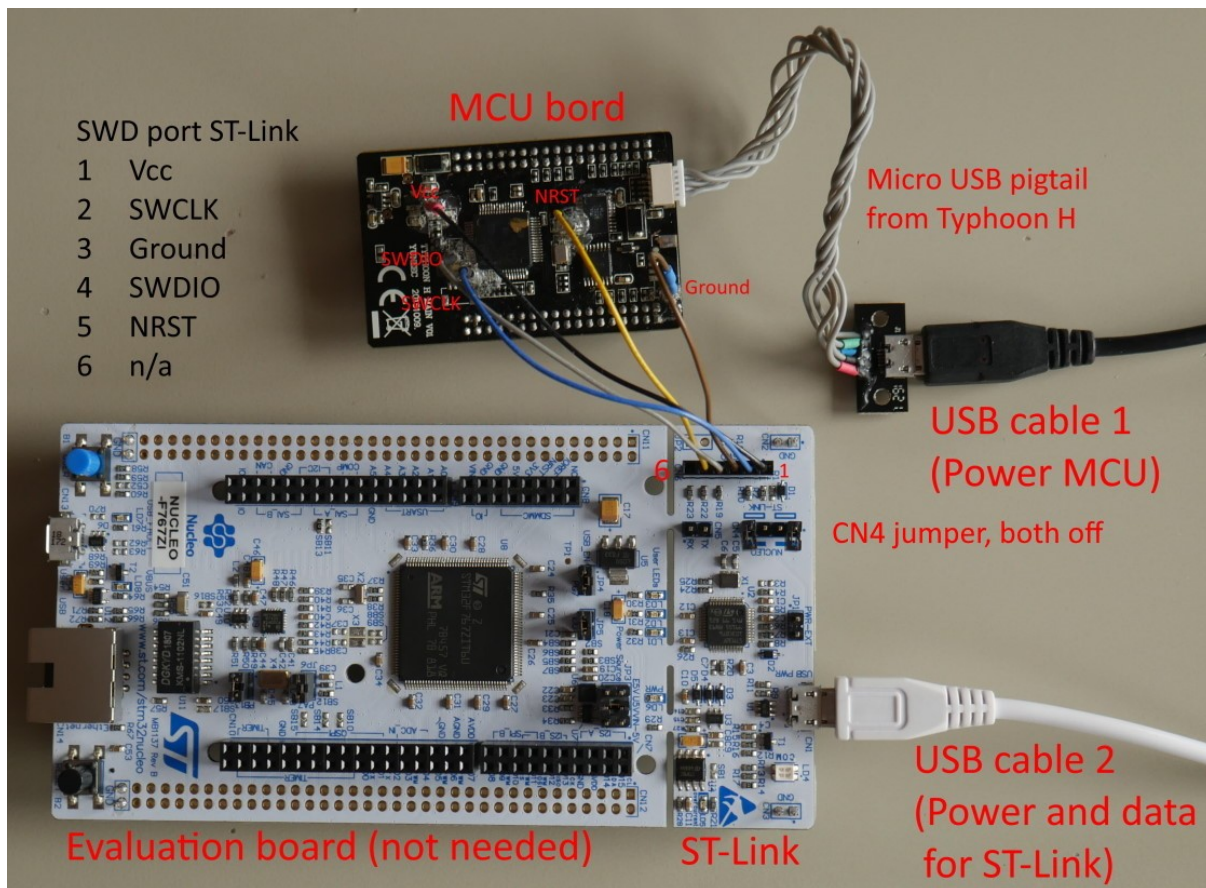
Test: Nach dem Löten muss das Board erneut auf Funktion prüfen. Beim Anschließen muss es hochlaufen, zu erkennen an den LEDs. Im Windows Geräte-Manager erscheint es immer noch als „STMicroelectronics Virtual COM Port (COM...)“.

Schritt 2: Den Bootloader mit dem PX4 Bootloader ersetzen (ab jetzt gibt es kein zurück!)

Beide CN4 Jumper A und B müssen offen sein (Brücken entfernt), um den SWD-Port für externe Prozessoren freizugeben.

Verbinde das MCU-Board via USB mit dem PC, um die Stromversorgung sicherzustellen, auch wenn das MCU-Board auf der ESC-Platine steckt.

Verbinde den SWD-Port mit den Messpunkten auf dem MCU-Board.



Wenn alles richtig wie oben gezeigt verbunden ist, starte das STM32 ST-Link Utility.

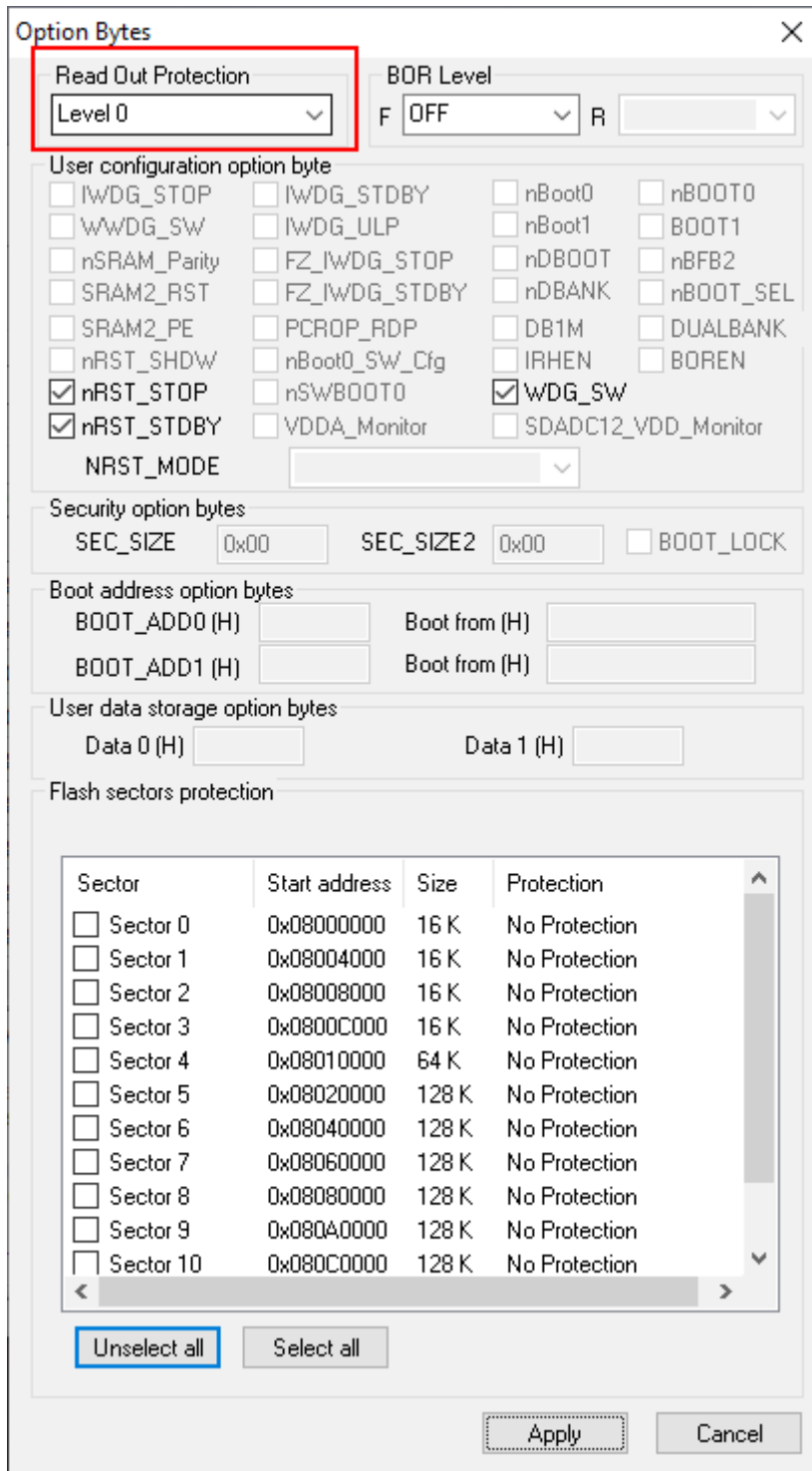
Menü Target > Connect, um festzustellen, ob das MCU-Board korrekt verbunden ist. Wenn nicht, dann Verkabelung prüfen. Siehe Bilder oben.

Read Out Protection auf Level 0 setzen:

Target > Option Bytes...

"Level 0" auswählen > Apply.

Jetzt wird der alte Bootloader entfernt.



The image shows a software window titled "Option Bytes" with a close button (X) in the top right corner. The window is divided into several sections for configuring device options.

Read Out Protection: A dropdown menu is set to "Level 0". This section is highlighted with a red rectangle.

BOR Level: Two dropdown menus are present. The first is labeled "F" and is set to "OFF". The second is labeled "R" and is currently empty.

User configuration option byte: A grid of checkboxes for various configuration options. The checked options are `nRST_STOP`, `nRST_STDBY`, and `WDG_SW`. Other options include `IWDG_STOP`, `IWDG_STDBY`, `IWDG_ULP`, `nBoot0`, `nBOOT0`, `nBoot1`, `BOOT1`, `nSRAM_Parity`, `FZ_IWDG_STOP`, `nDBOOT`, `nBFB2`, `SRAM2_RST`, `FZ_IWDG_STDBY`, `nDBANK`, `nBOOT_SEL`, `SRAM2_PE`, `PCROP_RDP`, `DB1M`, `DUALBANK`, `nRST_SHDW`, `nBoot0_Sw_Cfg`, `IRHEN`, `BOREN`, `nSwBOOT0`, `VDDA_Monitor`, and `SDADC12_VDD_Monitor`. Below this grid is a dropdown menu for `NRST_MODE`.

Security option bytes: Includes input fields for `SEC_SIZE` (0x00) and `SEC_SIZE2` (0x00), and a checkbox for `BOOT_LOCK`.

Boot address option bytes: Includes input fields for `BOOT_ADD0 (H)` and `BOOT_ADD1 (H)`, each followed by a "Boot from (H)" label and an empty input field.

User data storage option bytes: Includes input fields for `Data 0 (H)` and `Data 1 (H)`.

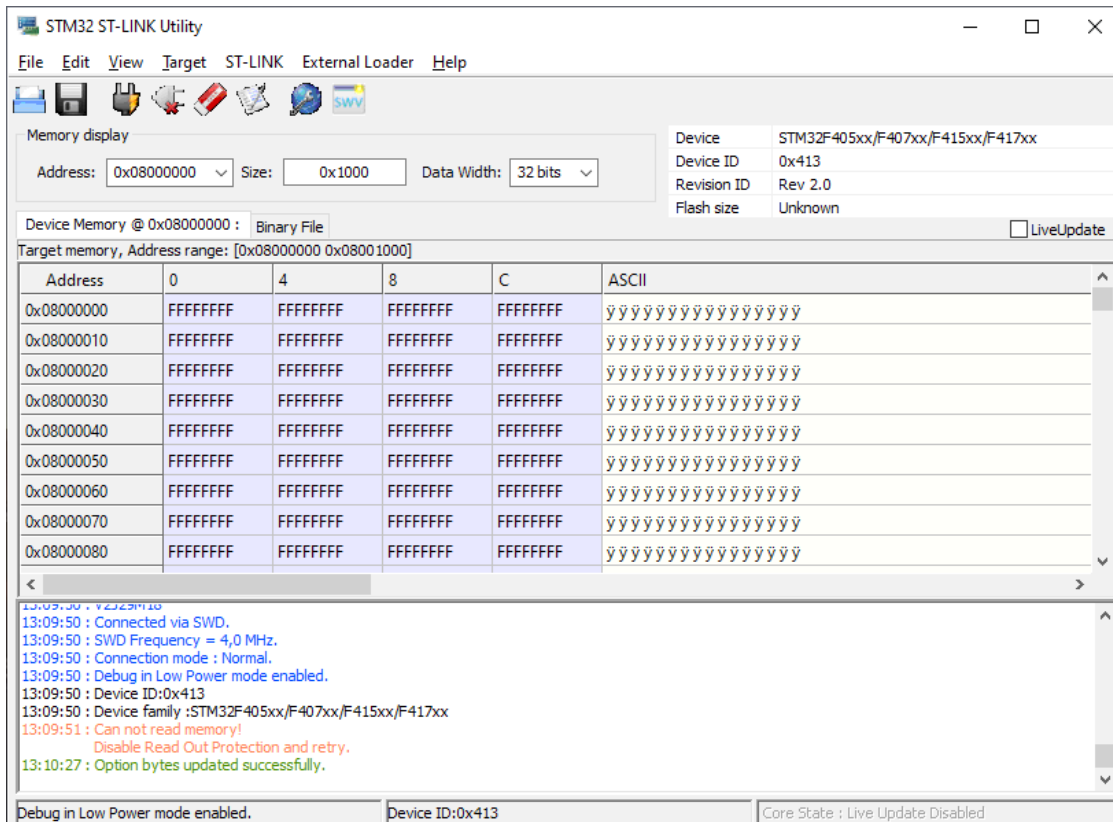
Flash sectors protection: A table listing 11 sectors with their start addresses, sizes, and protection status. All sectors are currently set to "No Protection".

Sector	Start address	Size	Protection
<input type="checkbox"/> Sector 0	0x08000000	16 K	No Protection
<input type="checkbox"/> Sector 1	0x08004000	16 K	No Protection
<input type="checkbox"/> Sector 2	0x08008000	16 K	No Protection
<input type="checkbox"/> Sector 3	0x0800C000	16 K	No Protection
<input type="checkbox"/> Sector 4	0x08010000	64 K	No Protection
<input type="checkbox"/> Sector 5	0x08020000	128 K	No Protection
<input type="checkbox"/> Sector 6	0x08040000	128 K	No Protection
<input type="checkbox"/> Sector 7	0x08060000	128 K	No Protection
<input type="checkbox"/> Sector 8	0x08080000	128 K	No Protection
<input type="checkbox"/> Sector 9	0x080A0000	128 K	No Protection
<input type="checkbox"/> Sector 10	0x080C0000	128 K	No Protection

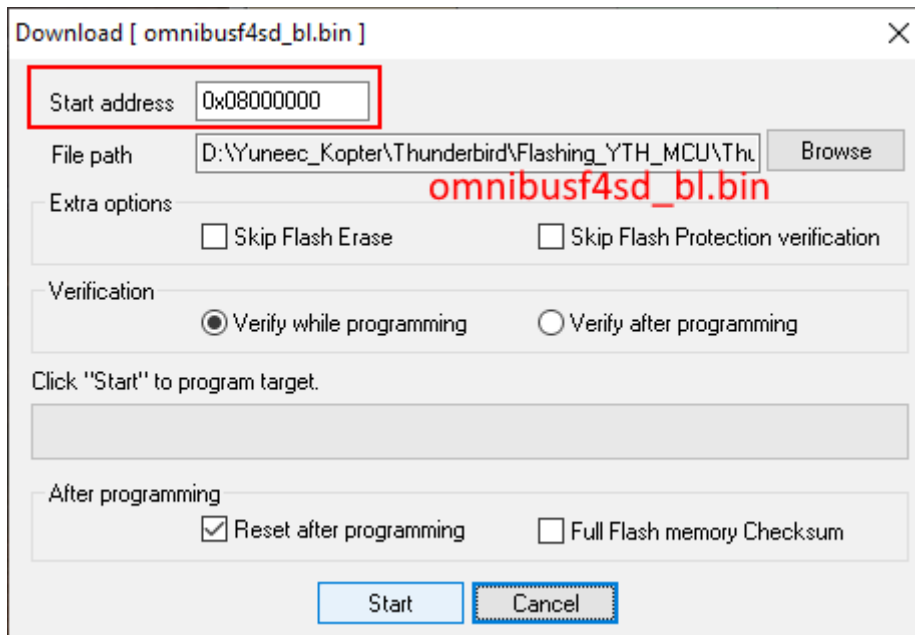
Below the table are two buttons: "Unselect all" (highlighted with a blue border) and "Select all".

At the bottom right of the window are two buttons: "Apply" and "Cancel".

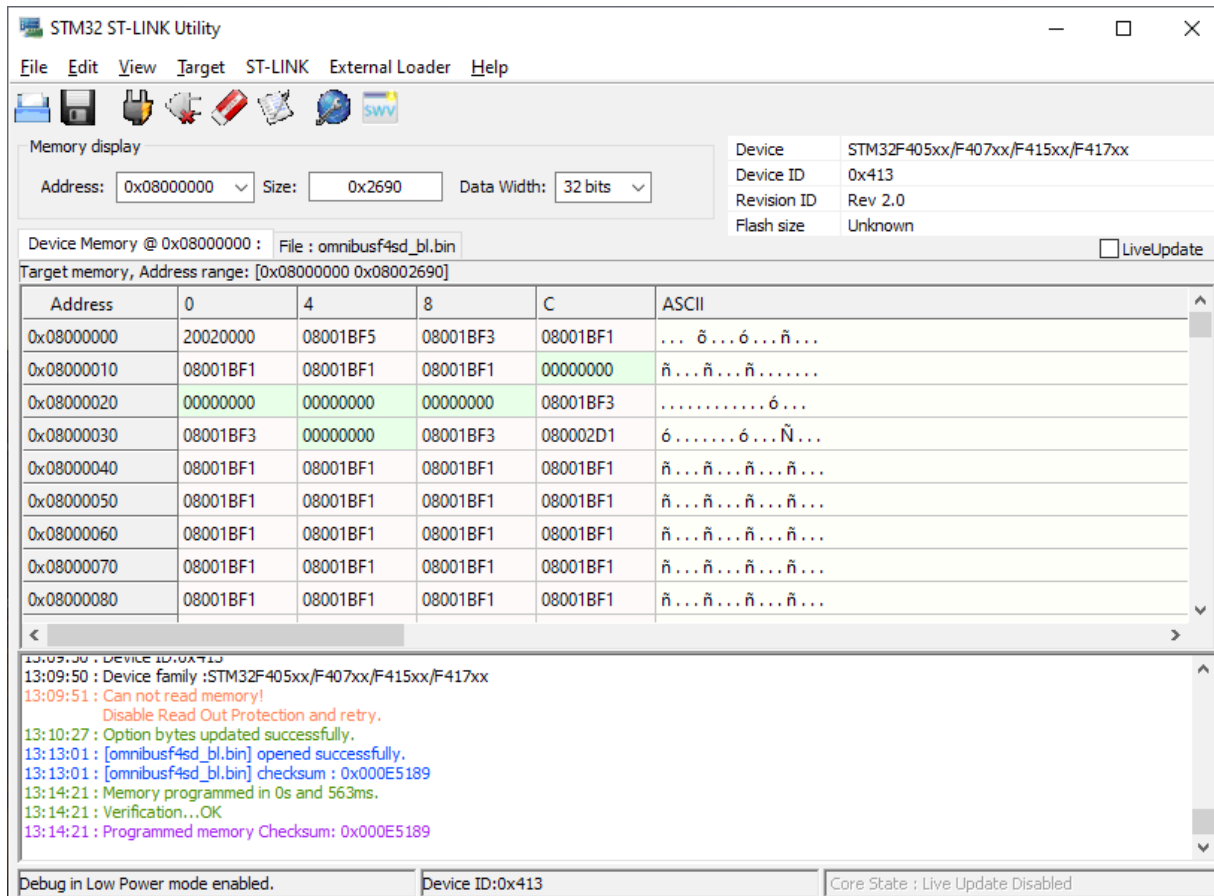
Das sieht dann so aus:



Nun flashen wir den neuen Bootloader auf Adresse 0x08000000. Menü Target > Program & Verify. Die Datei "**omnibusf4sd_bl.bin**" aus den Binaries auswählen > Start.



Nun sollte es so aussehen:



Das MCU-Board hat nun den PX4 Bootloader am Anfang des Flashbereichs, 0x8000000. Das ist die Adresse, wo der Bootprozess startet. Man kann hier nichts kaputt machen. Was immer hier ist wird vom Bootloader gestartet, der fest im ROM eingebrannt ist. Dem ist es egal, was sich hier befindet. Ist etwas falsch gelaufen, kann man einfach erneut flashen.

Die derzeitige PX4 Firmware ist so aufgebaut, dass sie immer von Adresse 0x8008000 startet. Damit ist genug Platz, um Parameter zwischen Bootloader und PX4 Hauptanwendung zu speichern. Solange wir keine SD-Karte angeschlossen haben, brauchen wir den Platz. Man könnte die PX 4 Anwendung auch gleich auf Adresse 0x8000000 starten und bräuchten den Bootlader gar nicht, aber dann müssen die Parameter auf einer (hier nicht vorhandenen) SD-Karte gespeichert werden.

Bis jetzt werden wir weiter mit dem Bootloader arbeiten und nichts ändern.

Schritt 3: Die PX 4 Hauptanwendung flashen

Es gibt zwei Möglichkeiten., die PX4 Hauptanwendung zu flashen:

A) Mit Hilfe des Bootloaders: Der gerade aufgespielte PX4 Bootloader kann die Hauptanwendung via Typhoon H USB Buchse flashen und von Adresse 0x8008000 starten. Wir brauchen den ST-Link gar nicht mehr und könnten den Typhoon H zusammenbauen.

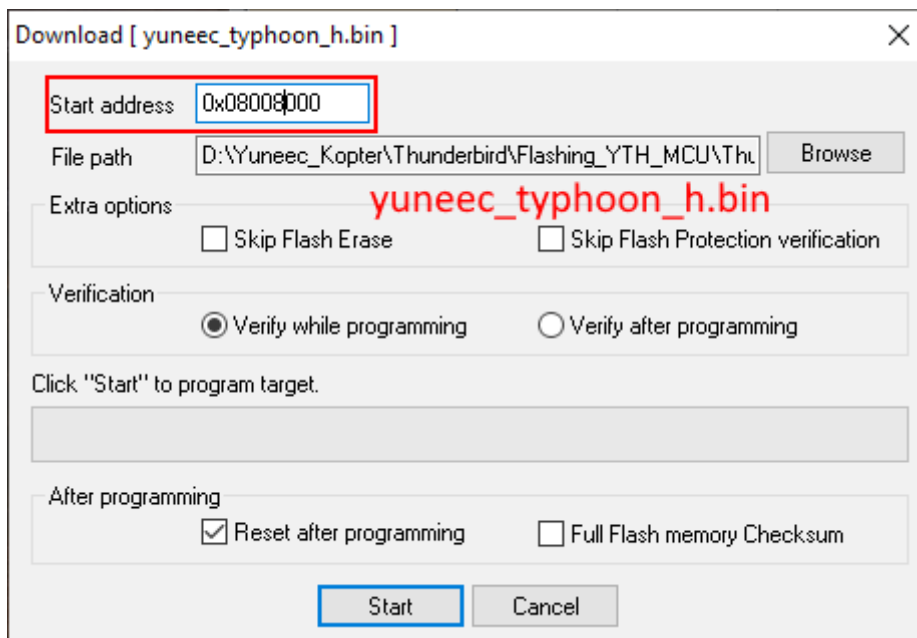
Der Bootloader braucht ein paar Sekunden zum flashen. Das Flashen wird mit einem etwas abgewandeltem px_uploader.py Script gemacht. Dieses Script erfordert, die .bin Datei in eine .fw zu konvertieren. Schau dir das "flash_typhoon_bootloader.py" Script und trage dort den COM-Port ein, der der USB-Verbindung zum Typhoon H zugeordnet ist.

Alle genannten Dateien sind in den Binaries enthalten. Diese Prozedur ist derzeit nur unter Ubuntu LINUX getestet. Bei Windows muss man entsprechend anpassen. Aber es gibt ja noch Option B).

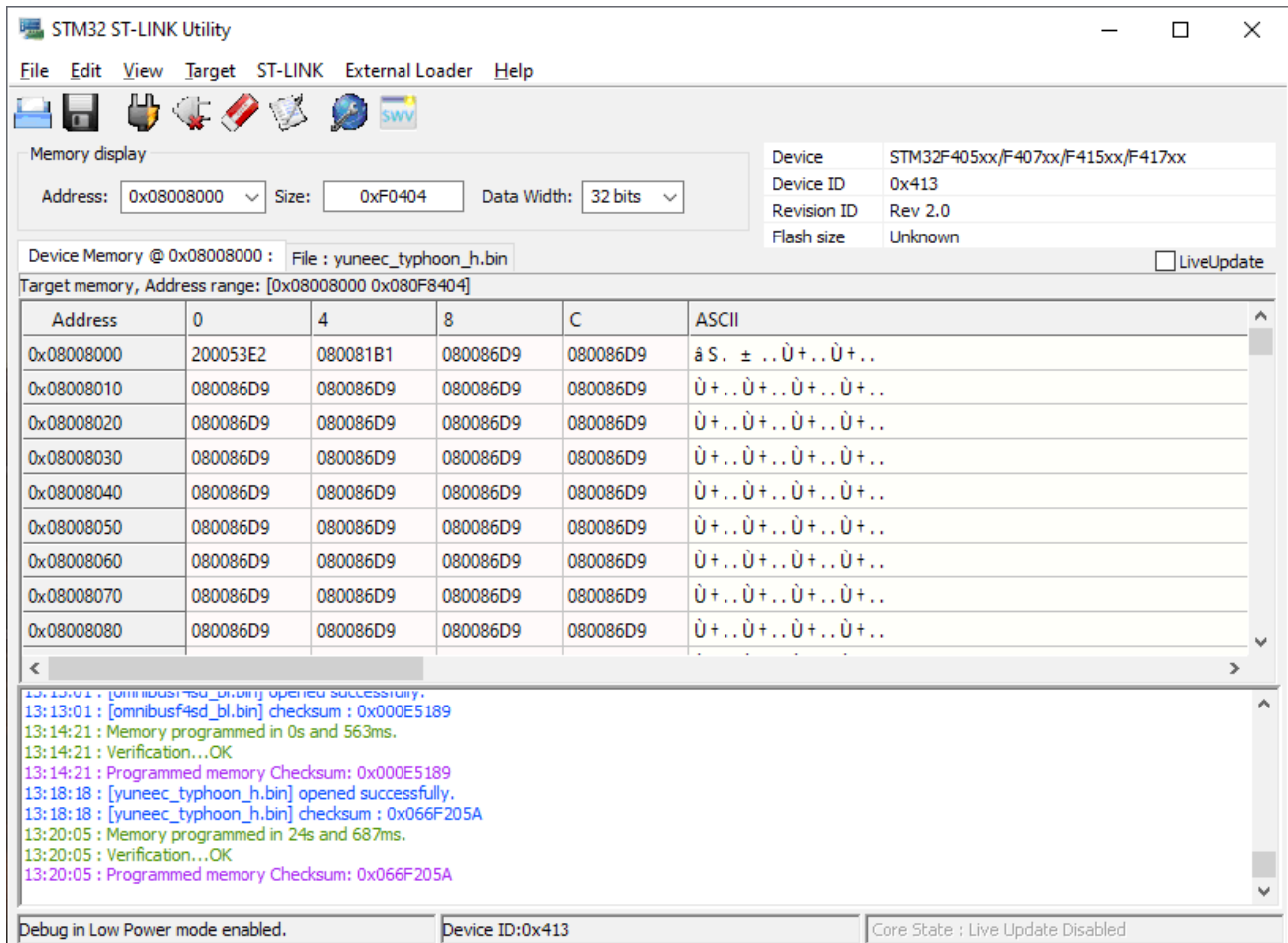
B) Weil wir unser MCU-Board noch so schön angeschlossen haben, nehmen wir ST-Link Flasher (aka "my python scripts suck" -way). Einfach die Datei "yuneec_typhoon_h.bin" genau wie den Bootloader vorhin auf das MCU-Board flashen, aber dieses Mal ab Adresse 0x8008000.

Option A) ist für spätere FW-Updates. Option B) ist genau richtig für jetzt.

Menü Target > Program & Verify. Datei "yuneec_typhoon_h.bin" aus den Binaries auswählen.
Nach dem Auswählen der Datei, Start address auf 0x8008000 setzen und Start drücken.



Nun sollte es so aussehen:



Erledigt!

Alle USB Verbindungen trennen und abschalten. Die Programmer-Verkabelung zum MCU-Board trennen. ST-Link wird nicht mehr gebraucht, es sei denn aus irgendwelchen Gründen wird ein neuer Bootloader gebraucht. Das sollte nicht sein, kann aber in so einer frühen Phase der Projektentwicklung nötig sein.

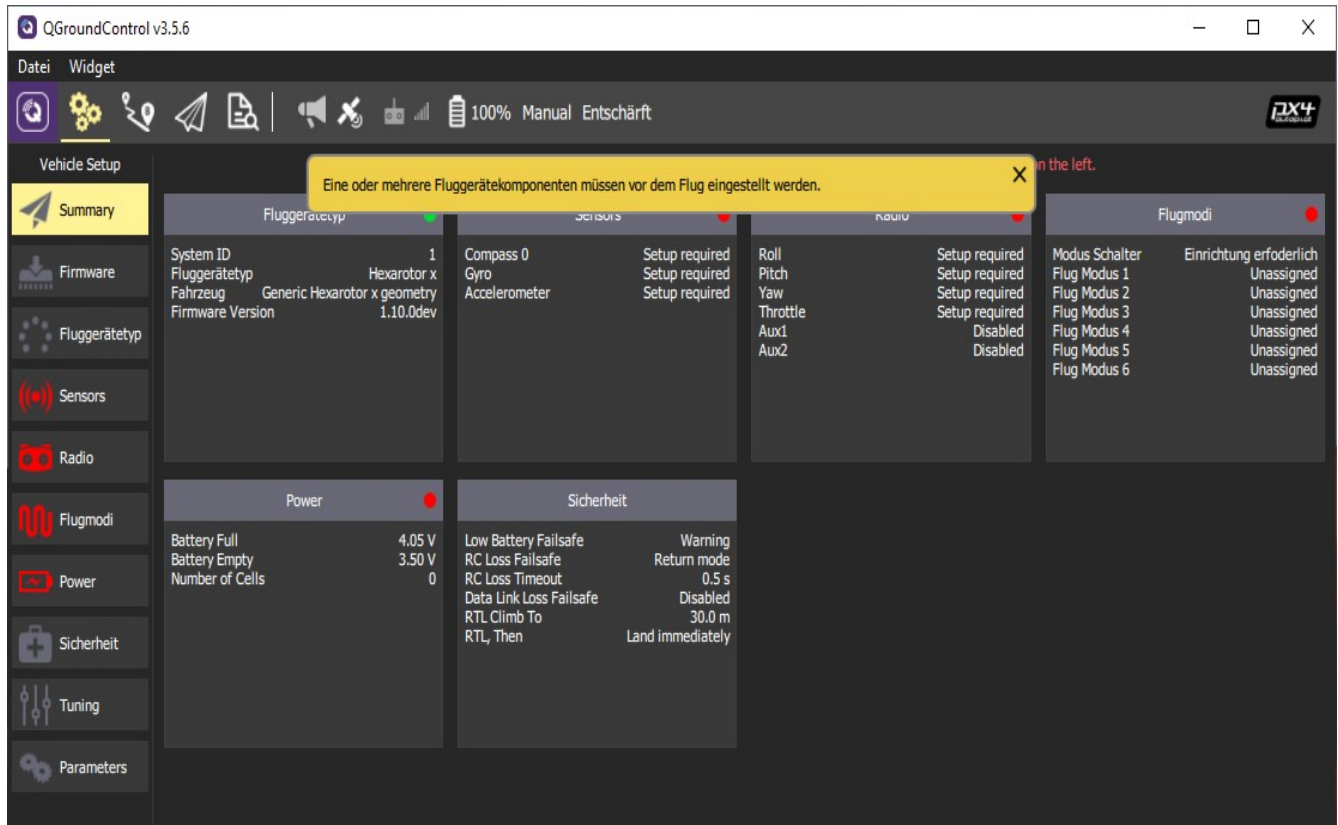
Test:

MCU-Board wieder über USB mit dem PC verbinden. Die LEDs auf dem MCU-Board haben keine Funktion mehr, aber im Windows Geräte Manager > Anschlüsse (COM & LPT) muss jetzt ein Gerät "Legacy FMU (COM...)" auftauchen. Wenn ds so ist, ist alles OK.

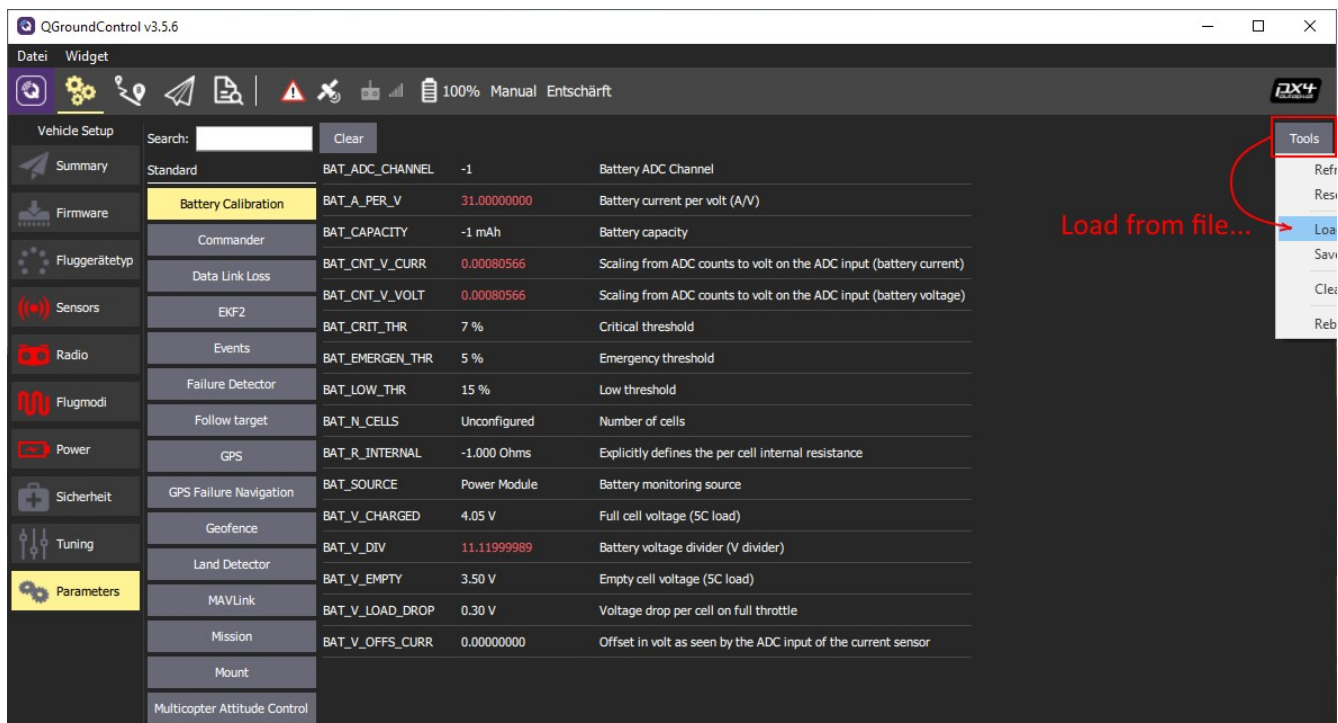
Den Thunderbird zusammenbauen und neu starten. Der Powerbutton hat jetzt leider eine Verzögerung von rund 8 Sekunden. Der Powerbutton muss gedrückt gehalten werden bis die Lichter angehen. Ausschalten geht nur durch Entfernen der Batterie.

Schritt 4: Parameter einstellen

Wir müssen QGroundControl (QGC) installieren und als Administrator starten. Kopter einschalten und über USB mit dem PC verbinden.



QGC erkennt, dass Parameter fehlen und eingestellt werden müssen. Am besten wir laden sie aus der mitgelieferten Datei **"typhoon_h_parameters.params"**. QGC > Parameters > Tools > Load from File...



Schritt 5: Hardware einstellen und Sensoren kalibrieren

Für Typhoon H gibt es zwei unterschiedliche Kompass Chips. Ältere GPS Module haben einen HMC5883 verbaut, neuere Module haben einen IST8310 als Kompass Chip. Um heraus zufinden, welchen du hast, schließe den Kopter an QGC an und gehe zur Mavlink Konsole:

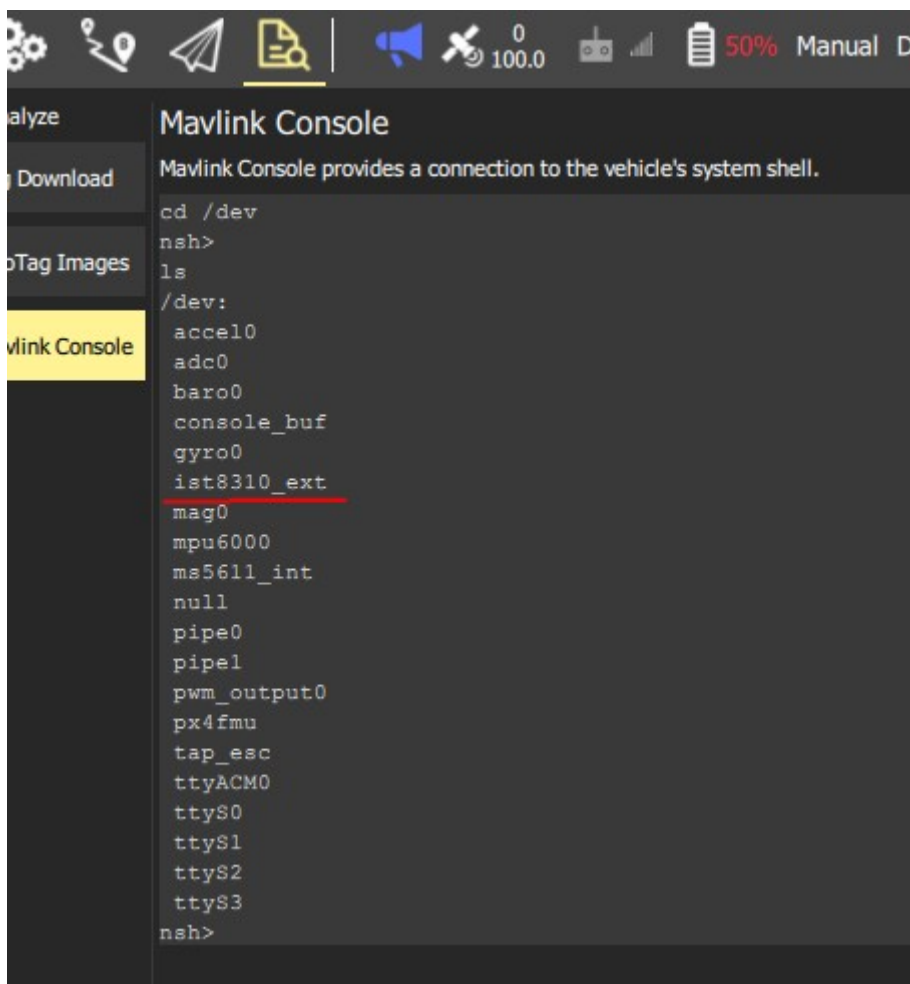
Eingabe:

```
cd /dev
```

```
ls
```

und eine Liste der Hardware-Treiber erscheint.

Schön, was wir hier alles sehen können! Für den Kompass gibt es zwei Möglichkeiten: "hmc5883_ext" oder "ist8310_ext".



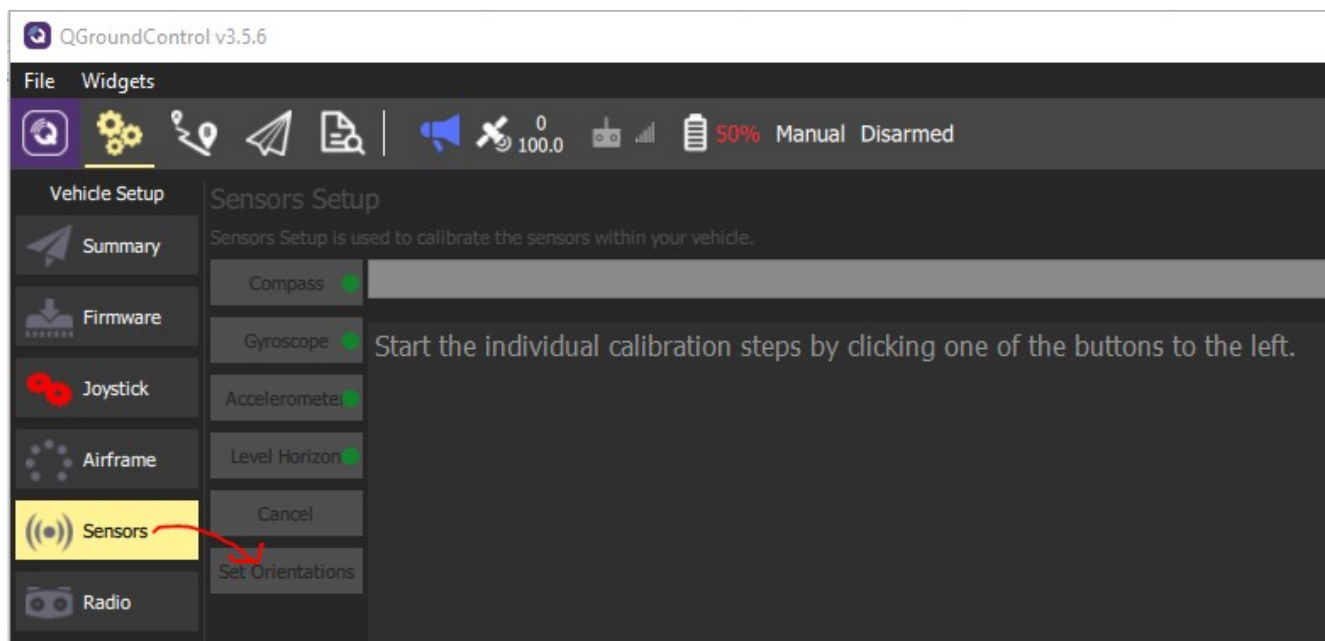
```
cd /dev
nsh>
ls
/dev:
 accel0
  adc0
 baro0
console_buf
 gyro0
ist8310_ext
  mag0
 mpu6000
ms5611_int
  null
 pipe0
 pipel
pwm_output0
 px4fmu
 tap_esc
ttyACM0
ttyS0
ttyS1
ttyS2
ttyS3
nsh>
```

Wichtig: Wenn man "hmc5883_ext" hat, muss man die External Compass Orientation auf "ROTATION_YAW_270" einstellen.

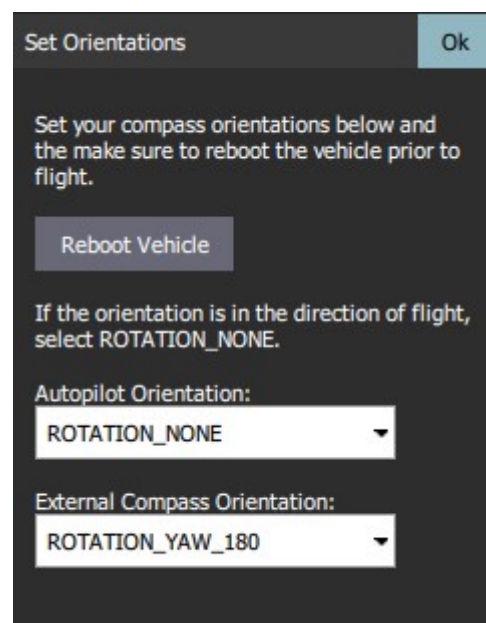
"ist8310_ext" muss " ROTATION_YAW_180" haben.

Niemals die Autopilot Orientation ändern. **Hier muss immer "ROTATION_NONE" stehen.**

Gehe zu Settings > Sensors > Set Orientation



Dann bei External Compass Orientation den richtigen Wert abhängig vom Kompass-Chip auswählen. Danach neu booten.



Nun müssen noch alle Sensoren kalibriert werden, wie bei einem völlig neuen PX4 Kopter üblich. Siehe dazu PX4 Basic Configuration · PX4 v1.9.0 User Guide. **Niemals die ESCs kalibrieren. Dies ist hier nicht nötig und geht auch nicht.**

<https://docs.px4.io/master/en/config/>

Abschließend, prüfe Parametereinstellungen, Sensoren und Kalibrierungen.

Handling

Jetzt haben wir einen komplett neuen Kopter mit neuen Flugeigenschaften. Wir müssen erst einmal lernen, den zu fliegen. Dazu bitte genug Freiraum einplanen, weg von Bäumen, Mensch und Tier.

In den mitgelieferten Parametereinstellungen ist die obere Position des Flightmode-Schalters "Manual/Stabilized" (Typhoon ohne GPS), mittlere Stellung ist "Position" (Typhoon with the GPS, Angle Mode) und die untere ist RTH. Acro/Rattitude ist nicht vorgeleistet. Man kann es, wie so vieles Anderes, einstellen, aber sollte damit vorsichtig sein.

Der rote Startknopf geht nicht mehr. Bitte findet selbst heraus, wie alles bei PX4 funktioniert. Dies soll euch anregen, die Dokumentation zu lesen, denn ihr habt ja jetzt einen völlig neuen Kopter.

<https://docs.px4.io/master/en/flying/>

Eines sollte man jedoch noch wissen: Motor-Start/Stop geht nur, wenn der Geschwindigkeitsregler an der rechten Seite auf Hase (Schnell) steht.

Die ST16 bleibt erst einmal unverändert. Die Werkseinstellungen sind OK.

Fehlerbehandlung

- Nach dem Löten der Verbindungen auf den Messpunkten am MCB-Board und vor dem Flashen prüfen, ob die Firmware noch hoch läuft bzw. Kopter fehlerfrei bootet. Wenn nicht, auweia! Bitte mit der Lupe auf Lötbrücken prüfen.

- Wenn der Kopter nach dem Flashen nicht als USB-Gerät erscheint, dann ist der Bootloader beschädigt. In dem Falle muss alles von Anfang an neu geflashed werden.

- Wenn der Kopter als USB-Gerät erscheint, aber nicht hoch läuft, auch wenn der Startbutton länger als 8 Sekunden gedrückt gehalten wurde, dann ist die Hauptanwendung beschädigt. Bitte diese ab Adresse 0x8008000 neu flashen (Schritt 3). Achtung! Die Adresse muss **nach** dem Auswählen der Firmware-Datei eingestellt werden, sie springt sonst wieder zurück.

- Prüfen, ob das MCU-Board richtig steckt. Es kann sein, dass es um einen Pin verschoben aufgesteckt ist. Außerdem neigen die Pins zum Verbiegen. Auch hilft eine Lupe.

- Nach jeder HW-Änderung oder Reparatur muss Schritt 5: Hardware einstellen und Sensoren kalibrieren erneut durchgeführt werden.