Yuneec ZigBee Receiver SR24 an Raspberry Pi

Quellen

PX4 Autopilot: https://github.com/PX4/PX4-Autopilot/blob/master/src/lib/rc/st24.h

Scheint veraltet zu sein und war im Hinblick auf die ST24 gemacht worden. Mich wundert, dass es da keine neuere Version gibt (bzw. ich habe sie nicht gefunden), da ja H Plus und H520 auch auf PX4 Autopilot beruht.
In den Definitionen z.B. fehlt der Packet Type 03.

Untersuchungen und Programmierung von @dylanfm:
https://www.rcgroups.com/forums/showthread.php?2973916-Yuneec-Receiver-protocol
https://yuneecpilots.com/threads/advice-for-using-hacking-st16-for-fathom-rov.20671/post-231170

Besonders die Steuerung des Modellautos mit der ST16 hat mich stark motiviert, das auch zu versuchen. Außerdem fand ich hier viel Hintergrundwissen.

Ein Testprogramm, welches in Vorarbeit zur Thunderbird FW von @Pöllö gemacht wurde:
https://yuneecpilots.com/threads/typhoon-h-480-px4-v1-10-stability-issues.18205/post-206189

Da braucht man wohl nichts dazu zu sagen. Der Thunderbird fliegt perfekt mit der ST16 und zeigt die Telemetrie an.

Mein Testtool

Trotzdem habe ich mich (endlich) mal aufgemacht, den SR24 an einen Raspberry Pi anzuschließen. Ziel war es die Kommunikation zwischen ST16 und SR24 kennenzulernen und eine Unit in FreePascal zu schreiben, die die gesamte Kommunikation kapselt. Dies ist dann die Grundlage für weitere Projekte mit dem Raspberry als Steuerrechner für beliebige Modelle, zu steuern mit meiner ST10 oder ST16.

Stufe 1 – Alle von der ST16 empfangen Daten aufzeichnen

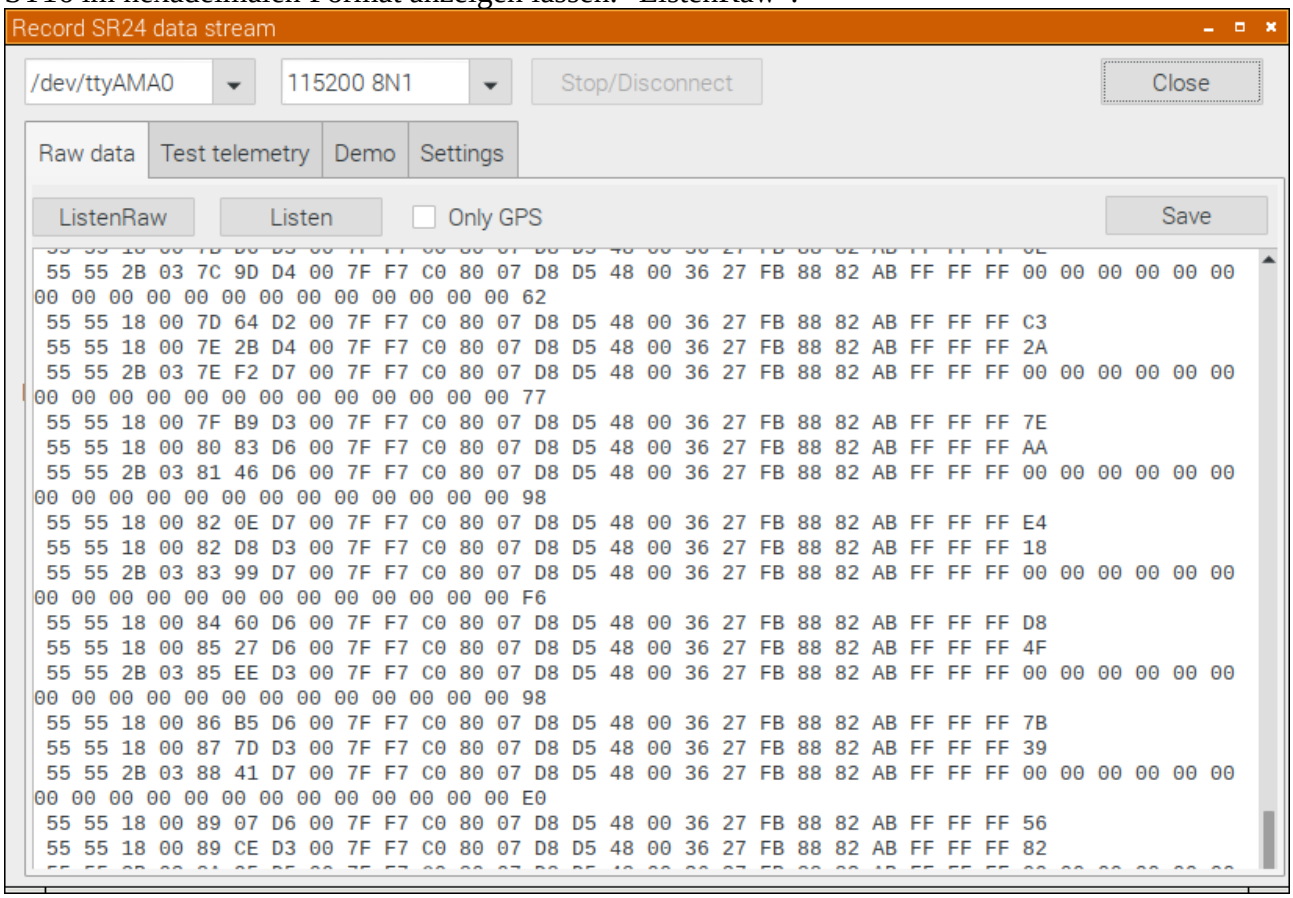
Das Programm lauscht auf der seriellen Schnittstelle und zeichnet alle empfangenen Bytes auf. Diese werden im hexadezimalen Format ausgegeben und diese Ausgabe kann gespeichert werden. Ziel war es, auch nicht beschriebene, unbekannte Datenpakete zu erkennen und zu untersuchen. Es gibt scheinbar bei der ST16 keine, außer den folgenden Paket Typen:

Typ 0: Die Daten von ersten zwölf Kanälen Ch0 – Ch11.
Typ 3: Die Daten von den zwölf Kanälen und die GPS-Daten.
Typ 4: Das Paket, um den Empfänger in den Bindemodus zu bringen.
Der wird jedoch nicht empfangen, sondern gesendet.

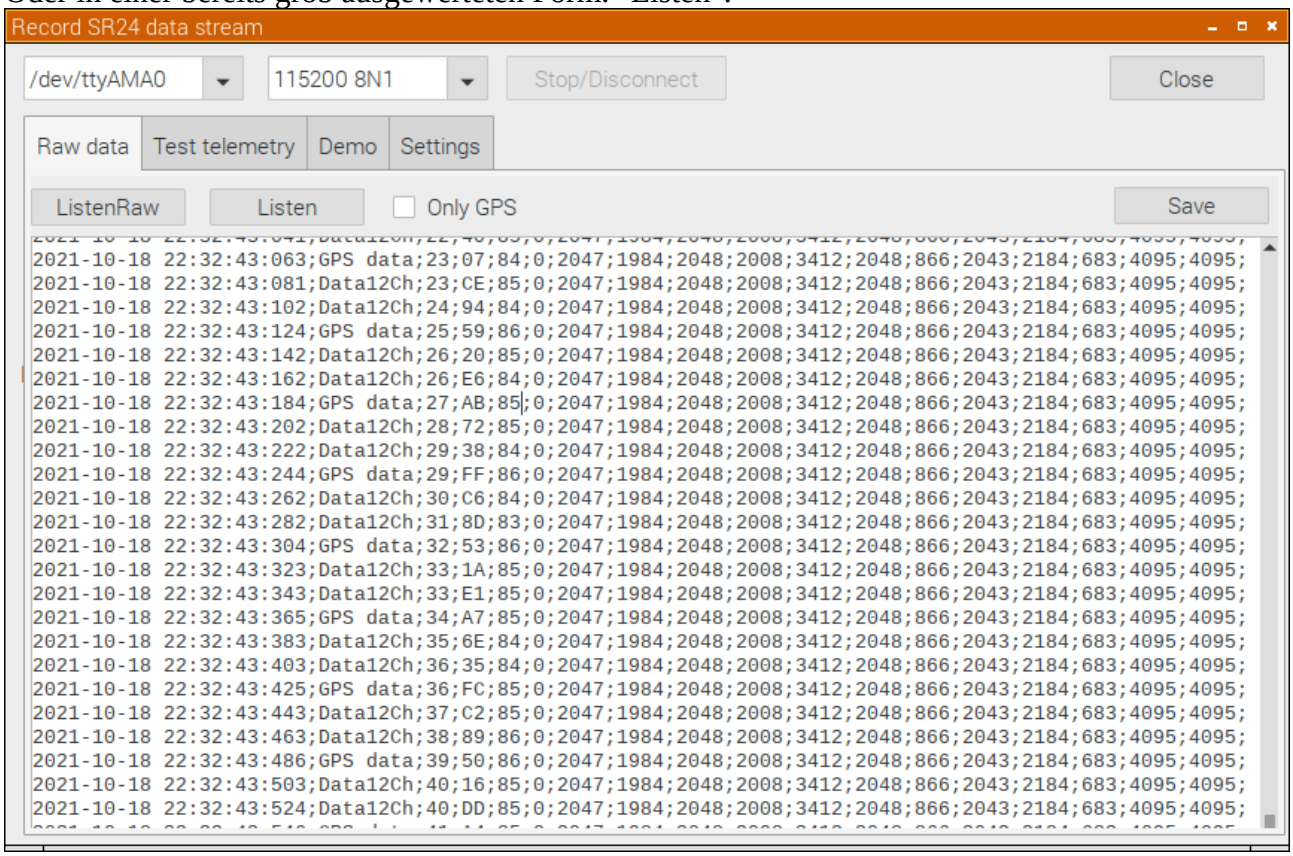
Im Paket Typ 0 sind genau die Daten drin, welche in den FlightLogs in den Dateien "Remote_xxxxx.csv" aufgezeichnet werde.

Im Typ 3 sind zusätzlich genau die Daten drin, welche in den FlightLogs in die Dateien "RemoteGPS_xxxxx.csv" gespeichert werden.

Um diese Aufgabe zu erfüllen gibt es die Seite "Raw data". Hier kann man sich die Daten von der ST16 im hexadeimalen Format anzeigen lassen. "ListenRaw":



Oder in einer bereits grob ausgewerteten Form. "Listen":



Beide Ausgaben können mit LibreOffice Calc weiter verarbeitet werden. Das wurde zur weiteren Dokumentation verwendet.

Hier zum Beispiel das Paket, um den SR24 in den Bindemodus zu versetzen:

	Header1	Header2	Length	MsgType	MsgCounter		B	I	N	D	CRC8
Bind mode message	55	55	8	4	0	0	42	49	4E	44	B0

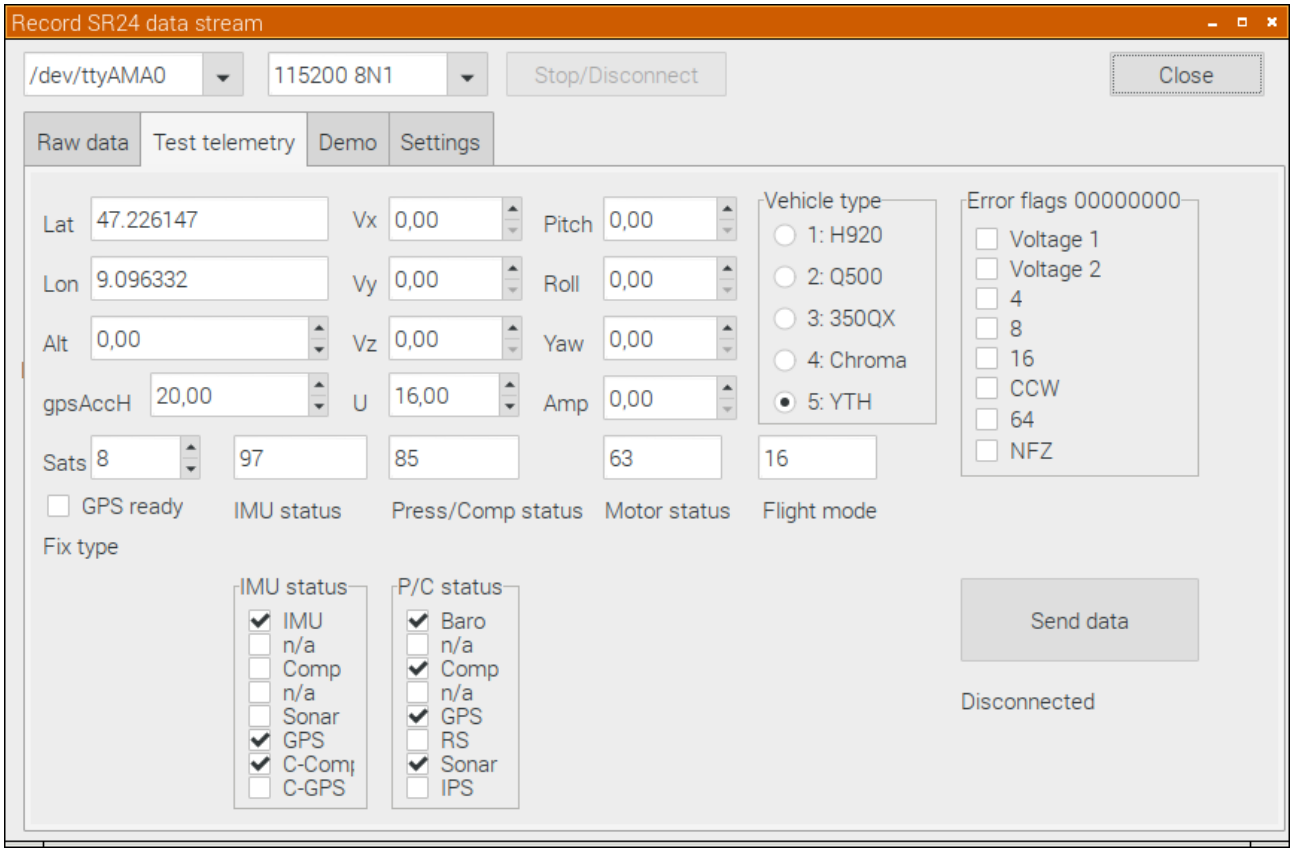
Stufe 2 – Austesten, was man zur ST16 schicken kann und was das bewirkt

Nachdem das Lesen und Schreiben funktioniert und die Datenstruktur im Westentlichen klar ist, wollte ich sehen, was als Telemetrie an die ST16 geschickt werden kann und was dies bewirkt. Dazu gibt es nun die Seite "Test telemetry":

Man kann ausprobieren, ob es funktioniert, wie die Datenformate sind und welche Daten notwendig sind, um der ST16 vorzuspiegeln, dass alles in Ordnung ist, auch wenn man ein ganz anderes Modell bedienen will.

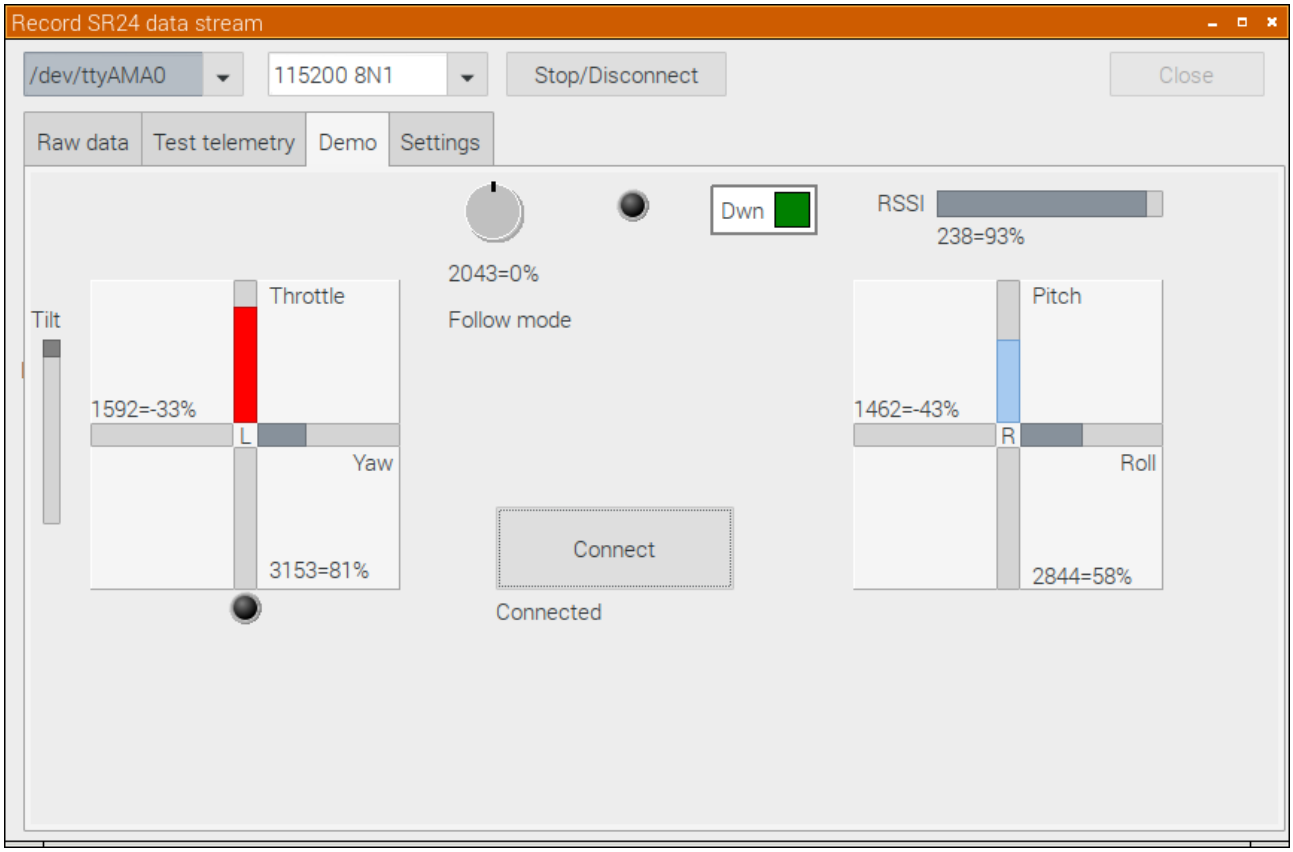
Außerdem kann man mit bestimmten Werten experimentieren und dann mit den Einträgen in den FlightLogs vergleichen. Das gibt ein weites Forschungsfeld und dieses Tool hilft dabei deutlich. Man kann bei den Versuchen noch einigen Mysterien der ST16 auf die Spur kommen.

Damit werde ich wohl in der nächsten Zeit viel herumexperimentieren.

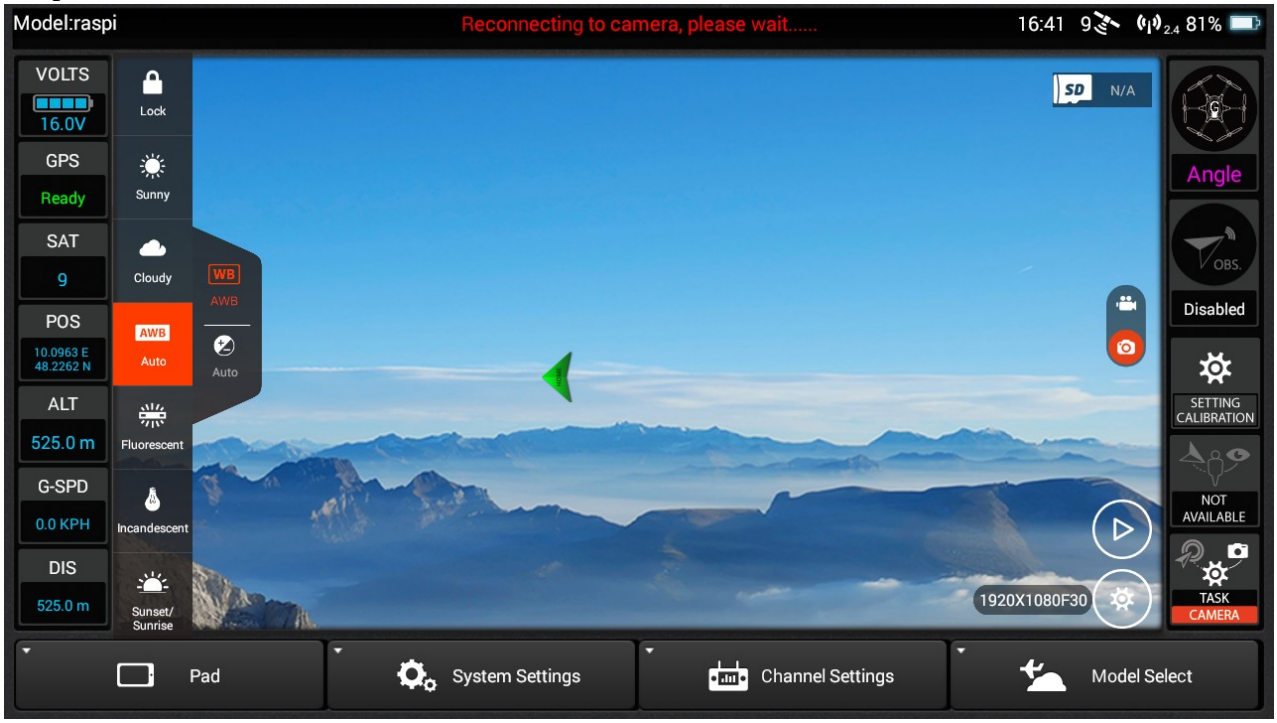


Stufe 3 – Eine kleine Demo

Um zu zeigen, dass es funktioniert, ist auf der Seite "Mirroring" eine Demo zu sehen. Mirroring heißt Spiegeln und genau das macht diese Funktion. Es empfängt die GPS Daten aus der ST16 und gibt diese als Telemetrie zurück. Man sieht auf der ST16 also Position und Höhe der ST16 anstatt eines fliegenden Modells mit GPS. Genauso werden FlightMode Schalter und GPS Status gespiegelt, so dass man GPS Status und Flight Mode auf der ST16 sieht, als würde der Kopter antworten.



Der passende ST16 Screen. Man sieht z.B. die Altitude ASL, wie es die ST16 liefert.



Vorbereitung

Quellen:
https://www.ics.com/blog/gpio-programming-using-sysfs-interface

https://www.raspberry-pi-geek.de/ausgaben/rpg/2020/04/grundlagen-der-pulsweitenmodulation/

Hardware PWM Unterstützung einschalten und Bluetooth abschalten:

```
sudo nano bootconfig.txt
```

Und hier folgende Zeilen eintragen:

```
[Switch-on PWM]
dtoverlay=pwm-2chan,pin=18,func=2,pin2=13,func2=4

[Switch-off bluetooth to get serial]
dtoverlay=pi3-disable-bt
```

Nach dem Reboot stehen die beiden PWM Kanäle an den GPIO pins 18 (PWM0) und 13 (PWM1) zur Verfügung. Bluetooth abzuschalten ist natürlich nur nötig, wenn der Raspberry Pi überhaupt Bluetooth hat.

Außerdem müssen wir noch die Konsoleneingabe über UART abschalten:

`sudo raspi-config` > Interface Options > Serial port >
"Would you like a login shell to be accessible over serial?" --> No
"Would you like the serial port hardware to be enabled?" --> Yes

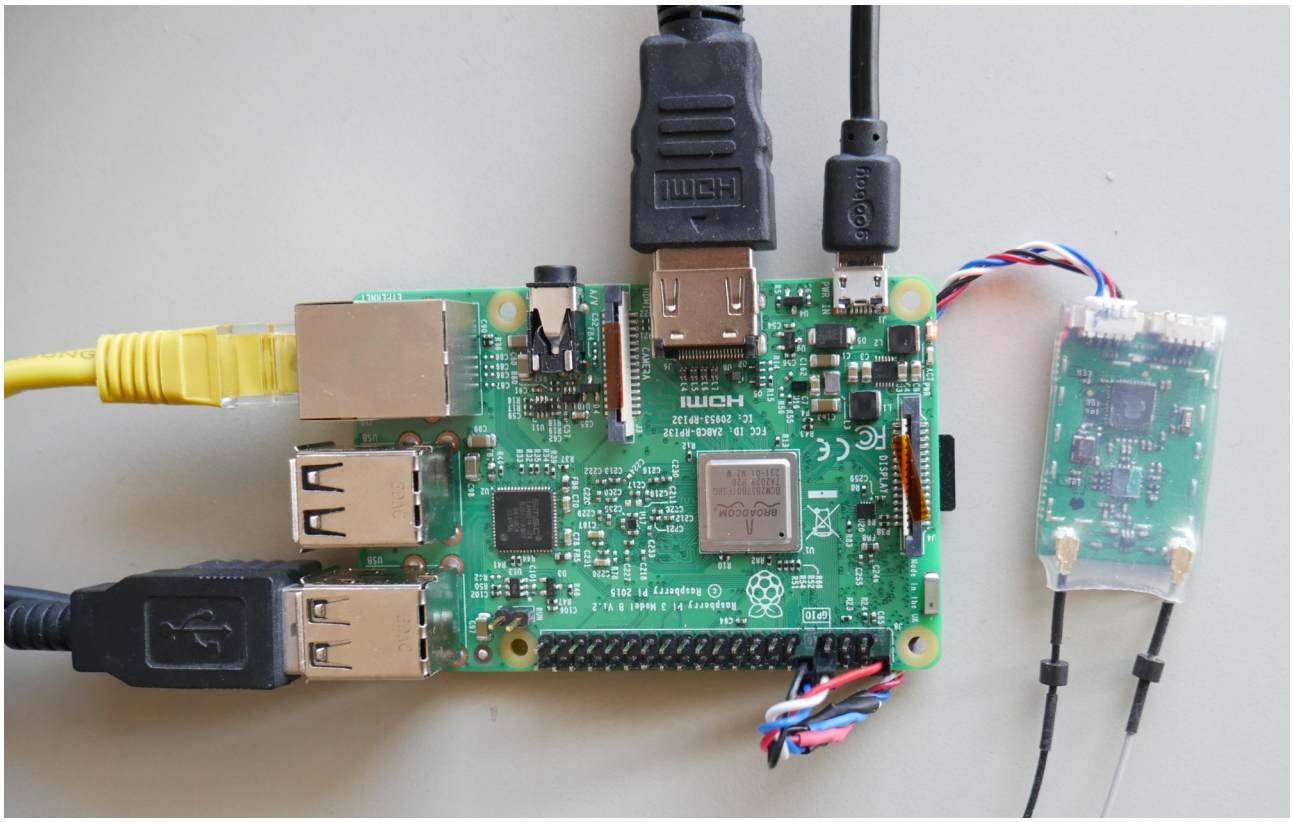
Feststellen ob 32 bit oder 64bit OS:

Um herauszufinden, ob Sie ein 32-Bit- oder 64-Bit-System installiert haben, geben Sie in der Konsole / Terminal folgenden Befehl ein:

```
getconf LONG_BIT
```

Anschließend gibt Ihnen die Konsole den entsprechenden Wert zurück. (32 oder 64).
Die kompilierten Programme sind für 32bit Betriebssysteme.

Der Versuchsaufbau



Der SR24 muss an folgende Pins des GPIO Ports angeschlossen werden:

- 3.3V an Pin 1
- GND an Pin 6 oder 9 (schwarzes Kabel am SR24)
- Rx an Tx, Pin 8 (grau oder weiss, je nach SR24 Version)
- Tx an Rx, Pin 10 (ge oder blau)

2	4	6: GND	8: Tx	10: Rx	12: PWM0
GPIO Header			gr - ws	ge - bl	
1: 3,3V	3	5	7	9: GND	11

Bevor man die serielle Schittstelle ungestört benutzen kann muss man die Bootkonsole über serielle Schnittstelle abschalten. Das geht mit raspi-config im LX Terminal.
sudo raspi-config > 3 Interface Options > P6 Serial Port > Die erste präsentierte Option mit <Nein> beantworten und die zweite mit <Ja> und mit <OK> bestätigen.
Dann kann es losgehen.

Die LED am SR24 blinkt langsam, wenn auf Verbindung gewartet wird. Dauerleuchten bedeutet: Verbunden. Schnelles Blinken zeigt an, dass der Receiver im Binde-Modus ist.

"SR24_decode" starten und damit herumspielen. Wenn das programm nicht startet, dann muss man die Datei ausführbar machen, also die Berechtigung "Executable" für das Programm vergeben.

Fazit

Ein schönes Projekt, welches mir viel Spaß gemacht hat und welches noch Luft nach oben bietet.
Zu finden auf GitHub: https://github.com/h-elsner/SR24_decode

Die Software

Ein Ergebnis der Programmierung des Testtools sind drei Freepascal-Units, die als Legobausteine für Anwendungen dienen, die ein Fahrzeug, ein Boot oder was auch immer steuern können.

SR24_dec: Diese Unit stellt alle Funktionen zum Kontrollieren der Seriellen Schnittstelle und zur Auswertung der empfangenen Daten zur Verfügung. Dies ist sozusagen der Eingang des Steuersystems.

SR24_chsets: Diese Unit verarbeitet die Konfiguration und vermittelt zwischen Eingang (Kanäle) und Ausgang des Systems. Es wertet die Settings-Datei aus.

SR24_ctrl: Diese Unit bietet die notwendigen Funktionen zum Steuern der Ausgänge des Raspberry Pi GPIO Ports an. Das ist also der Ausgang zur Hardware des zu steuerenden Modells.

SR24_log: Diese Unit bietet Funktionen zum Zum Generieren von Logdateien im legacy Yuneec Format, welche mit Q500log2kml ausgewertet werden können. Dazu gehören Funktionen, wie Daten als String präsentiert werden.

Für mehr Informationen über die Prozeduren und Funktionen bitte die Kommentare in den Quelltexten der vier Units lesen.

Die kompilierten Beispielprogramme:

st16cars: Ein Beispielprogramm ohne GUI zum Steuern eines Modells, welches die oben genannten Prozeduren und Funktionen benutzt. Per Telemetrie-Rückmeldung wird der ST16 vorgespiegelt, dass alles in Ordnung ist (HW-Status, GPS-Daten).

st16bind: Ein Konsolenprogramm, welches den Empfänger SR24 in den Bindemodus bringt.

SR24_decode: Das im ersten Teil beschriebene Testtool zum Testen der Funktionen, sowie zum Ausprobieren, auf welche Daten die ST16 (oder ST10/12/24) wie reagiert. Dies lässt zusammen mit den Flightlogs auf der Funkfernsteuerung Einblicke auf die Funktionsweise des Yuneec Flugsteuerungssystems zu.

SR24wizard: Ein Hilfsprogramm zum Editieren der Settings (Konfigurationsdatei für die Hardware). Die Settings dienen als Mittler zwischen den empfangenen Daten aus den Kanälen der Funkfernsteuerung ST16 (oder ST24, ST10) und den Ausgängen des Raspberry Pi.

Das Projekt (Software, Quelltexte und Dokumentation) findet ihr hier:

https://github.com/h-elsner/SR24_decode