Yuneec ZigBee Receiver SR24 am Raspberry Pi

Um die Funkfernsteuerungen von Yuneec-Koptern (ST10 oder ST16) auch anderweitig nutzen zu können, habe ich mich für den Raspberry Pi als Steuercomputer entschieden. Sehr gut eignet sich ein Raspberry Pi Zero, der sehr preiswert zu haben und universell einsetzbar ist.

Ich gehe davon aus, dass der Raspberry Pi mit dem üblichen Beriebssystem versehen ist.

Installation mit **Raspbian Pi OS Lite** - nur Terminal. Info: http://www.raspberrypi.org/

https://downloads.raspberrypi.org/raspios_lite_armhf_latest

Vorbereitung

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 downloadbaren kompilierten Programme sind für 32bit Betriebssysteme.

Die Software

Zur Programmierung gibt es 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.

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, ST12) und den Ausgängen des Raspberry Pi.

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

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

Die Settings

Die Settings dienen als Mittler zwischen den empfangenen Daten aus den Kanälen der Funkfernsteuerung ST16 (oder ST24, ST10) und den Ausgängen des Raspberry Pi. Per Settings kann man die Steuerung relativ frei konfigurieren, d. h. den verschiedenen Servos und Schaltausgängen Kanäle und Ausgabewerte zuordnen. Es handelt sich um eine einfache und lesbare Textdatei, die man mit einem normalen Texteditor bearbeiten kann.

Standard-Kanäle der ST16

Die Nummerierung der Kanäle startet hier immer mit Eins (nicht wie in der Telemetrie mit Null).

Channel 1 - Throttle THR (Default: 683=-100% bis 3412=+100%, Neutral ist 2048=0%)
Channel 2 - Roll AIL
Channel 3 - Pitch ELE
Channel 4 - Yaw RUD
Channel 5 - Flight mode
Channel 6 - RTH
Channel 7 - Camera tilt
Channel 8 - Camera pan*
Channel 9 - Gimbal tilt mode*
Channel 10 - Gimbal pan mode*
Channel 11 - Landing gear* (Werte für Schalter können zwischen 0 und 4095 liegen)
Channel 12 - Aux button*

Start/stop - Mixed mit Throttle = 0
* nicht bei ST10

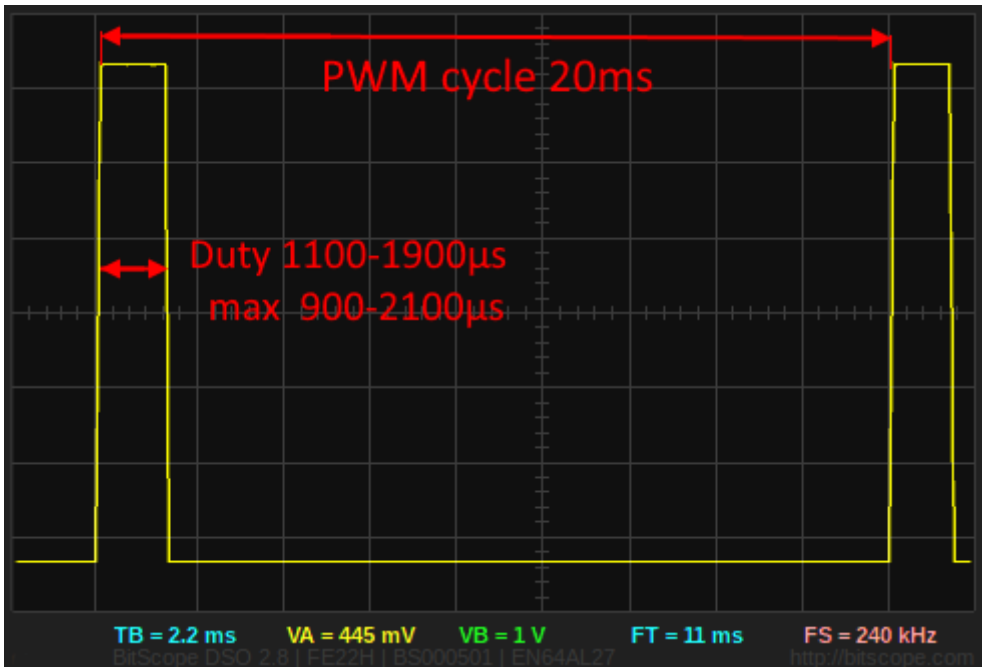
Die ST10 bedient zwar nur Channel 1 bis 7, kann aber genau wie die Sender ST12 und ST24 ebenso mit dem Empfänger SR24 genutzt werden.

Achtung: Bei der ST16 können durch Channel Settings und Mode-Wechsel diese Zuordnungen auch anders sein. Für die Steuerung eines eigenen Modells sollte man ein neues Modell auf der ST16 anlegen. Damit hat man automatisch die Standardeinstellungen.

Die Kanäle der ST16 (oder kompatible Typen wie ST10/12) kann man nun durch Editieren der Settings-Textdatei "rc_settings.set" den Ausgängen des Raspberry Pi zuordnen und die Servowege einstellen.

Es stehen zwei Hardware-PWM Kanäle für Servos oder Motorsteuerung zur Verfügung:
GPIO18 (pin12) PWM0
GPIO13 (pin33) PWM1

Servo output – PWM wird üblicherweise zwischen 1100-1900µs genutzt. Maximal ist es bei Analogservos 900-2100µs. Die PWM Impulse werden alle 20ms gesendet.



Von der ST16 kommen folgende Werte:

untere Stellung der Sticks:	683=-100%
Mittelstellung:	2048=0%
obere Stellung der Sticks:	3412=+100%

Für Schalter gibt es verschiedene Werte zwischen 0 und 4095 (entspricht -150% bis +150%). Was wie bei der ST10/16 zugeordnet ist, kann man im HW-Monitor oder im Output-Monitor auf der Fernsteuerung sehen.

Für Schalter stehen die folgenden GPIO-Pins zur Verfügung:

GPIO 5	Pin 29	
GPIO 6	Pin 31	
GPIO 12	Pin 32	
GPIO 16	Pin 36	
GPIO 17	Pin 11	
GPIO 22	Pin 15	Reserviert für „Voltage warning 1“ (Input)
GPIO 23	Pin 16	
GPIO 24	Pin 18	
GPIO 25	Pin 22	
GPIO 26	Pin 37	
GPIO 27	Pin 13	Reserviert für Shutdown key (Input)

Die GPIO-Pins können Schaltern oder auch Sticks zugeordnent werden. Werden Sticks digitalen Ausgängen zugeordnet werden, dann schalten sie bei etwa einem Drittel Servoweg um.
Die anderen GPIO pins sollten für spezielle Anwendungen freibleiben.

Gemeinsam genutzte Settings: Hier gibt es zur Zeit nur die Impulsfrequenz für die beiden Hardware-PWM Kanäle. Diese beträgt 20000µs, das sind 20ms und das (noch nicht realisierte) Logging. Logging bedeutet, dass auf der SD-Karte des Raspberry Pi Logdaten im gleichen Format wie bein den Yuneec-Koptern geschrieben werden. Diese können dann mit den Flightlog auf der St16 verglichen werden.
0 … Default – kein Logging
1 … Nur Telemetry. Die Spalten sind mit Default-Werten aufgefüllt. GPS-Daten stammen von der ST16.
2 … Telemetry und Remote. Hier sehen wir, was auf den Kanälen angekommen ist.
3 … Telemetry, Remote und RemoteGPS. In RemoteGPS werden die GPS-daten der ST16 gespiegelt.

Warn1 und Warn2 sind frei zuordenbare GPIO-Eingänge. Hier sollten Schmitt-Trigger mit hardwaremäßigen Schwellwerten für Voltage Warning 1 und 2 angeschlossen sein. 88 heißt: Nicht genutzt. Standardmäßig sind GPIO23 und 24 dafür vorgesehen.

Shutdown ordnet einen Eingang für einen Shutdown Schalter zu. Dieser Schalter muss in der HW vorhanden sein. Der Eingang (GPIO pin) muss mit einem Pull-up Widerstand 10kOhm beschaltet sein, wenn er benutzt werden soll.
Wenn kein Empfänger verbunden ist, setzt der Shutdown Schalter den Empfänger in den Bindemodus. Nach ungefähr 3s oder sofort wenn der Empfänger gebunden ist, wirkt er wieder als Shutdown Schalter.

```
PWM cycle = 20000
Logging = 0
Warn1 = 22
Warn2 = 88
Shutdown = 88
```

Korrekturfaktoren: Korrokturfaktoren sind Ganzzahlen zum Umrechnen von analogen Werten aus Analog-Digital-Wandlern (ADC). Hier kann man Spannung, Ströme. Temperaturen usw. messen und übergeben. ADC-Auslesen ist noch nicht realisiert. Die Werte sind Platzhalter.

```
Voltage = 1
Speed = 1
Aux 5 = 1000
```

Servos: Dann kommt in der Textdatei die Zuordnung der Servos und Schalter zu den Kanälen der ST16 auf der Eingangsseite und den Raspberry Pi GPIO-Pins auf der Ausgangsseite. Wir haben zu vergeben:

Eingänge:
Channel Nummer der ST16: 1 … 12

Servo 1 .. 6 sind die vier Sticks und die Tilt und Pan Steller.

Servowege in µs: Min- Neutral – Max. Die Werte von der ST16 werden proportional umgerechnet. Neutral muss nicht in der Mitte sein, es können auch 'Verzerrungen' eingegeben werden (z.B. spezielle Mittelstellungen bei Motorsteuerung).

Ausgänge:
Hardware PWM0: 0
Hardware PWM1: 1
GPIO Portnummer (BCMxy): xy (siehe Liste oben)
Keine Zuordnung (Platzhalter): 88

PWM Zyklen können invertiert werden. Dies scheint aber keine Funktion bei meinen Analogservos zu haben.
Wenn die Servos nicht den beiden Hardware-PWM-Kanälen zugeordnet sind, können sie als Schwellwertschalter genutzt werden. Dann sollte an dieser Stelle GPIOnr 2 stehen.

Beispiel für Throttle-Stick and HW-PWM Kanal 0:

```
[Servo 1]
Channel = 1
Minimum = 900
Neutral = 1500
Maximum = 2100
GPIOnum = 0
PWM reverse = False
```

Beispiel für Yaw-Stick als 3-Wege-Schalter:

```
[Servo 6]
Channel = 4
Minimum = 1100
Neutral = 1520
Maximum = 1900
GPIOnum = 23
GPIOnum2 = 24
```

Werden die Servos digitalen GPIO Ports zugeordnet erfolgt die Umschaltung bei etwa einem Drittel des Servowegs.

Switches: In der nächsten Sektion erfolgt die Zuordnung der Schalter. GPIOnum 2 ist ein zweiter Port für 3-Wegeschalter. Er ist der unteren Position des Schalters zugeornet.

Beispiel Flightmode-Schalter:

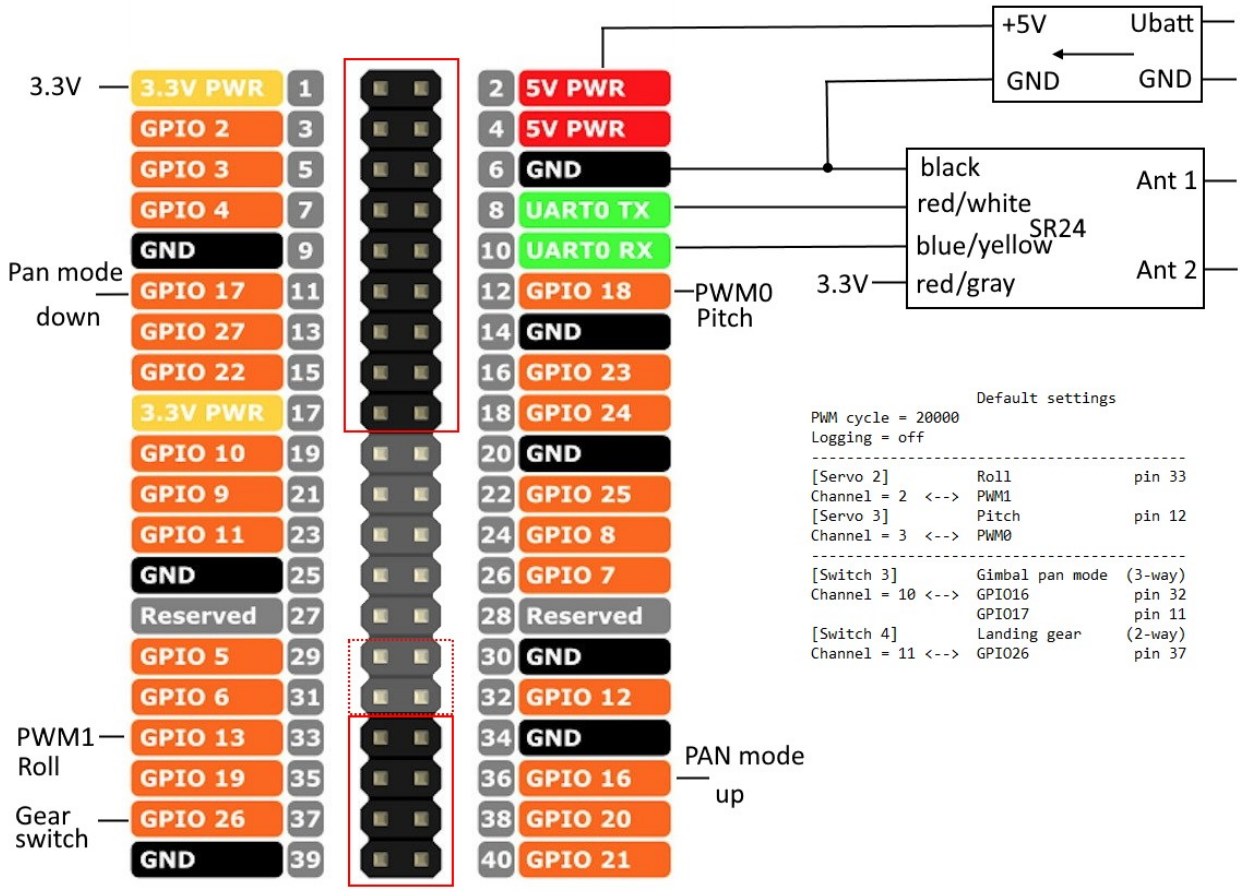
```
[Switch 1]
Channel = 5
Upper  position = 3412
Middle position = 2048
Lower  position = 683
GPIOnum = 16
GPIOnum 2 = 17
```

Beispiel Fahrwerk-Schalter:

```
[Switch 4]
Channel = 11
Upper  position = 4095
Middle position = 4095
Lower  position = 0
GPIOnum = 26
GPIOnum 2 = 88
```

Defaulteinstellungen

Wenn die Hardware so beschaltet ist, wie in den Defaulteinstellungen beschrieben, dann braucht man gar keine Settings-Datei "rc_settings.set". Alles was nicht gefunden wird oder fehlt wird duch die Defaulteinstellungen ersetzt. In den Defaulteinstellungen sind keine Eingänge belegt.



In der Settings-Datei braucht man also nur die Einträge aufführen, die von den Defaulteinstellungen abweichen. Das betrifft zusätzliche oder auch wegfallende GPIO pin Zuordnungen.
Beispiel: Nur Voltage Warning 1 soll zusätzlich beschaltet werden. Die Settingsdatei sieht dann so aus:

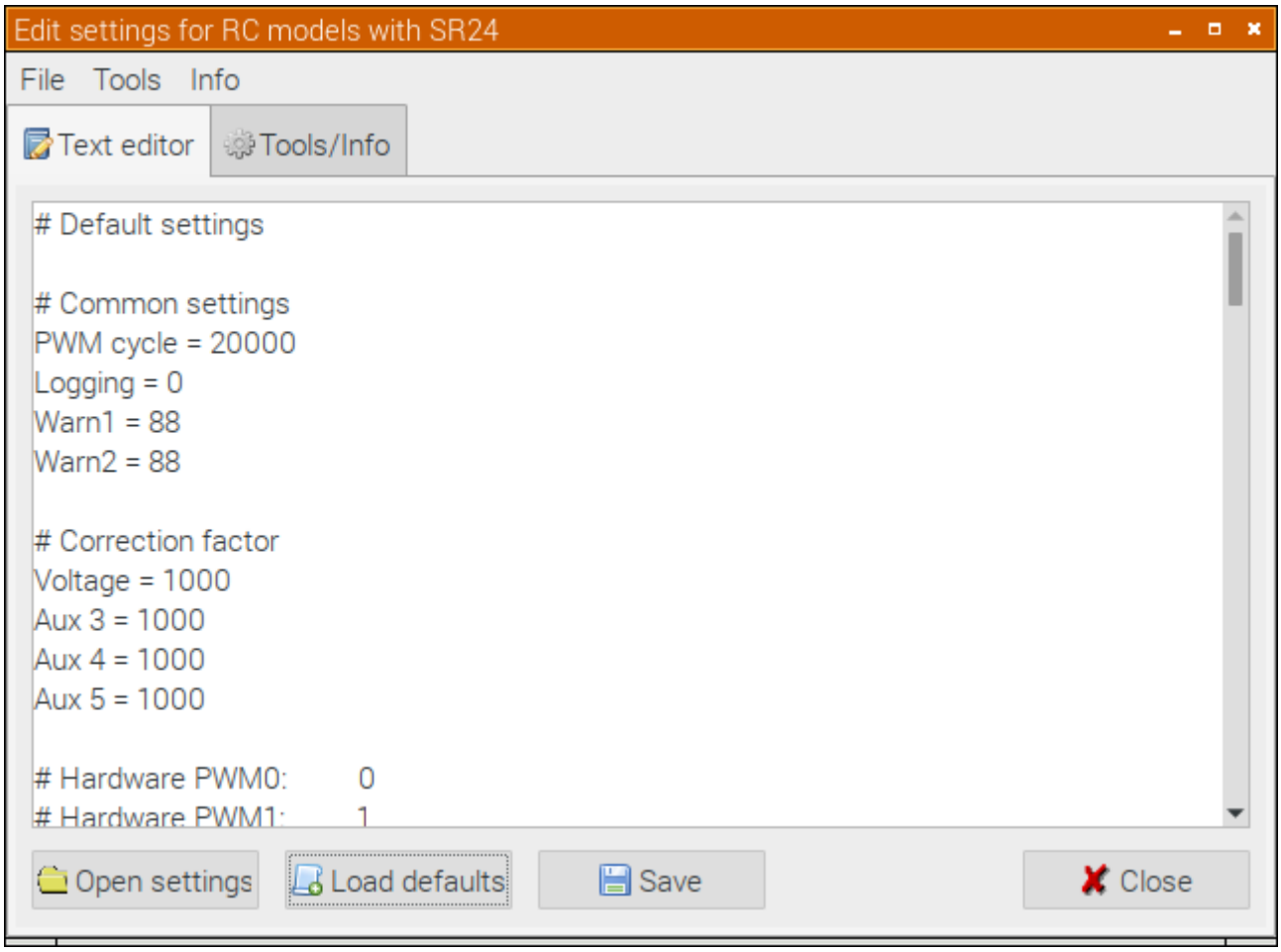
```
# Add voltage warning to GPIO pin 22

Warn1 = 22
Warn2 = 88
```

Mehr braucht eigentlich nicht eingetragen werden.

Hilfsprogramm SR24wizard

Das Wizard-Programm dient dazu, eine Settings-Textdatei zu erzeugen und die Settings zu verwalten. Es ist eigentlich auch nur ein Texteditor, bietet aber auch eine Konsistenzprüfung der Settings-Datei an. Außerdem kann man als Grundlage die Default-Einstellungen laden und diese dann nach Wunsch (bzw. Hardware-Beschaltung) verändern.



Auf der zweiten Seite "Tools/Info" sind der Binde-Button und ein paar wichtige Informationen über die Settings zu finden.

Das Programm verwendet folgende Freepascal-Units :

SR24_chsets: Diese Unit verarbeitet die Konfiguration und vermittelt zwischen Eingang (Kanäle) und Ausgang des Systems. Es wertet die Settings-Datei aus.
SR24_dec: Diese Unit stellt alle Funktionen zum Kontrollieren der Seriellen Schnittstelle und zur Auswertung der empfangenen Daten zur Verfügung. Dies wird nur für das Binden benötigte und funktioniert nicht unter Windows.