Wellenvogel AvNav

AvNav Beschreibung

Erzeugt: 2.7.2023

Navigation im Browser

AvNav ist eine kostenlose Navigationssoftware für Sportbootfahrer. Wie andere Anwendungen in diesem Bereich kann man elektronische Seekarten laden und mit angeschlossenen GPS-Geräten schauen, wo sich das eigene Boot befindet. Es ist natürlich möglich, Marker zu setzen, Routen zu erstellen und AIS-Signale einzubinden. Besonderes Merkmal von AvNav ist das Serverkonzept: AvNav lässt sich auf einem Raspberry oder einem Windows-Gerät installieren und als Navigationszentrale betreiben, die alle relevanten Daten einsammelt und Seekarten bereit stellt. Der Zugriff auf den Server im Boots-Netz und damit die eigentliche Darstellung von Karten und Daten läuft auf einem Webbrowser und ist so unabhängig vom jeweiligen Betriebssystem der zugreifenden Geräte. AvNav ist konsequent für die Bedienung per Touchscreen ausgelegt Darüber hinaus bietet AvNav die Möglichkeit, für jedes Gerät, das man benutzen möchte, Layouts zu definieren, um so das Aussehen der App z.B. auf unterschiedliche Bildschirmgrößen anzupassen.

Karten

Einmal können Rasterkarten, die nicht herstellerverschlüsselt sind, verwendet werden. Vektorkarten Typs OESenc können ebenfalls genutzt werden. Sie sind allerdings nicht kostenlos und können über den O-Charts-Shop bezogen werden.

Varianten

Es gibt nicht nur die Server-Variante von AvNav: Ebenfalls zur Verfügung stehen ein „StandAlone-AvNav“ für den Raspberry (AvNav Touch) und eine Android-App. Eine Besonderheit ergibt sich beim Einsatz eines OpenPlotter-Image auf einem Raspberry: Die Installationsvariante „AvNav für Openplotter“ sorgt automatisch für die richtigen Verbindungen zwischen dem Herzstück von Openplotter, dem SignalK-Server und AvNav. Die gesamte Software steht zum Download unter einer Open Source Lizenz bereit.

Für einen schnellen Einstieg: Quickstart

15.12.2020

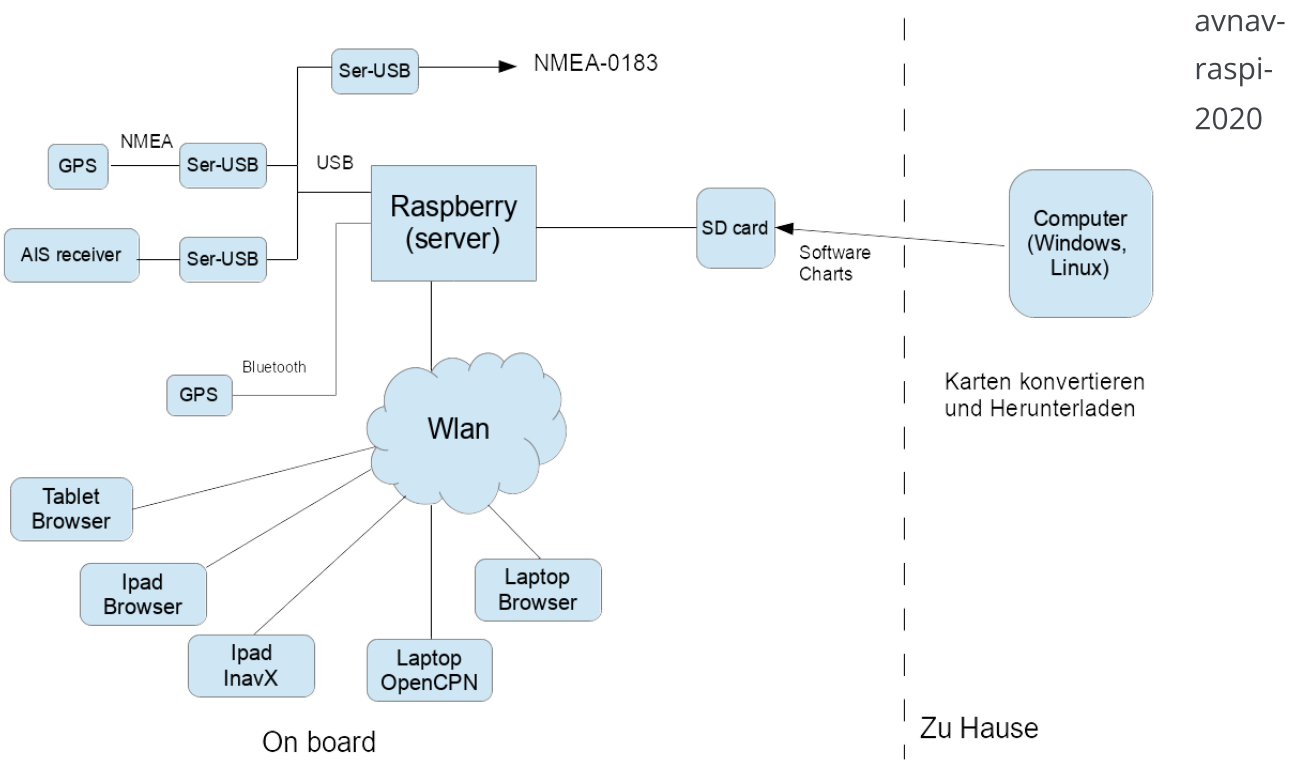
- YouTube Videos
- AvNav auf der Boot 2020 in Düsseldorf bei open-boat-projects
- Datenschutz: english,deutsch
- Thread im Segeln Forum

- Source Code auf GitHub
- Nutzer-Beschreibung
- Android App
- Karten herunterladen und umwandeln
- Installation
- Release-Notes
- Demo
- Foliensatz(PDF)

Ein Hinweis vorweg:

Ich kann keine Garantie für die Funktion der App übernehmen, insbesondere die Nutzung zu Navigationszwecken geschieht auf eigenes Risiko. In jedem Falle empfehle ich einen intensiven Test der Genauigkeit der Darstellung und des verwendeten Kartenmaterials.

Überblick



Wie im Bild zu sehen, besteht die gesamte Lösung aus mehreren Teilen:

- Raspberry Pi mit einer Server-Software, die die angeschlossenen Geräte abfragt, die Daten aufbereitet und per WLAN zur Verfügung stellt
- Software für Windows/OSx/Linux, die zum Vorbereiten und Konvertieren der Karten dient

Über ein WLAN, das der Raspberry Pi als Access Point bereitstellt, können verschiedene Geräte auf die Daten zugreifen. Dabei gibt es mehrere Varianten:

- Variante 1: Auf den Geräten (z.B. Ipad oder Laptop) kann eine Navigationssoftware laufen (getestet: InavX,OpenCPN), diese greift über TCP auf die NMEA-Daten zu. Navigationssoftware und Karten müssen natürlich auf den Geräten installiert sein.
- Variante 2: Auf den Geräten läuft nur ein Browser, die Navigation erfolgt per Java Script App, die vom Raspberry bereitgestellt wird. Dazu muss nur die entsprechende URL aufgerufen werden. In diesem Fall ist auf den Geräten keine Software installiert, nur ein aktueller Browser muss vorhanden sein (getestet: Chrome unter Windows, OSX, Safari, Android ab 4.x – Chrom/Stock/Boat Browser, IOS, Blackberry stockBrowser, WebBrowser mini).

"Unter der Haube"

Die Server Software auf dem Raspberry ist in Python geschrieben und über eine XML-Datei konfigurierbar - was im Normalfall aber nicht notwendig sein sollte. Neben dieser Software steht auch ein fertiges Image für den Raspberry zur Verfügung, das nur noch auf eine SD-Karte installiert werden muss (Empfehlung: mindestens 8GB, mehr ist besser...).

Die Web Applikation bietet eine Navigation mit Rasterkarten (gemf,mbtiles) oder OESenc Vektorkarten inklusive AIS-Darstellung, Wegpunkt-Navigation und Routing. Falls die Web-Applikation verwendet werden soll, müssen die Karten dafür auch auf dem Raspberry installiert werden.

OESenc Karten können im Shop von o-charts erworben werden.

Karten, die die Software nicht direkt verarbeiten kann (siehe Karten), müssen vorher auf dem PC (Windows, Osx, Linux) oder direkt auf dem Raspberry (innerhalb der App) in das gemf Format konvertiert werden. Im Wesentlichen können die folgenden Kartenquellen verarbeitet werden:

- Alle Kartentypen, die die GDAL-Software lesen kann (also insbesondere BSB-Karten)
- Mit Mobile Atlas Creator heruntergeladene Karten

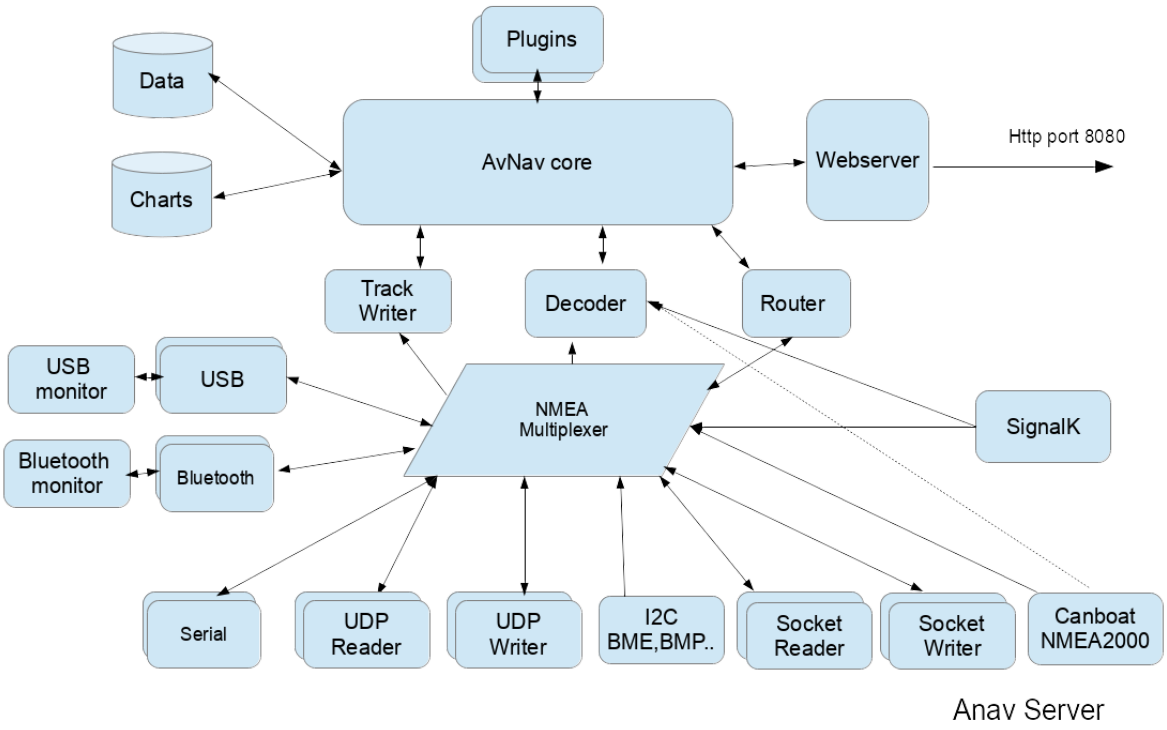
Daneben gibt es noch eine Android-App, die eine weitgehend identische Funktionalität bereitstellt: Der Server-Anteil ist hier nativ in Java geschrieben, die Anzeige-Funktionen sind identisch zur Raspberry-Variante.

In den folgenden Abschnitten gehe ich auf die Funktion der einzelnen Teile ein wenig genauer ein.

Die Server-Software (avnav_server.py)

Auf dem Raspberry Pi ist zunächst ein ganz normales Debian Image installiert (ca. 2GB). Dazu kommen einige Zusatzpakete (Liste siehe unten) und meine AvNav-Software.

Der Hauptbestandteil der Sofware auf dem Raspberry Pi ist ein in Python geschriebener Server. Im Folgenden beschreibe ich in groben Zügen, was dieser Server intern tut.



Der Server versucht alle am Raspberry angeschlossenen seriellen Geräte zu erkennen und deren NMEA-Daten zu lesen. Typisch werden die Geräte über Seriell-USB Wandler angeschlossen (z.B. PL2303). Man muss ein wenig aufpassen, dass man einen Wandler hat, der vom Raspberry auch sauber unterstützt wird - siehe z.B. hier. Da das Verwalten der seriellen Schnittstellen unter Linux etwas magisch ist, scannt der Server die angeschlossenen Geräte auf eine entsprechende serielle Klasse und ermittelt deren Schnittstelle (device). Anschliessend versucht er ein "auto bauding" zwischen 4800 und 34000 Baud und bemüht sich, NMEA Daten zu erkennen. Falls keine Daten empfangen werden, wird die Schnittstelle geschlossen und das Spiel beginnt von vorn. Damit „überlebt“ der Server auch das Anschliessen/Abstecken von Wandlern während des Betriebes oder das An- bzw. Abschalten von Geräten. Bei mir hängt ein RO4800 mit AIS-Decoder an einem Seriell-USB-Wandler, die GPS Daten werden durchgereicht. Alternativ versucht AvNav auch Kontakt zu seriellen Bluetooth-Geräten aufzunehmen. Falls die App per "discovery" Geräte findet, versucht sie von diesen ebenfalls NMEA-Daten zu lesen. Das wurde z.B. mit einer Holux GPS Slim236 getestet. In diesem Sinne arbeitet der AvNav-Server auch als NMEA-Multiplexer.

Alle GPS-Daten werden intern in eine Liste eingefügt und per TCP bereitgestellt. Verbundene TCP-Empfänger (z.B. OpenCPN) bekommen so jeden empfangenen Datensatz weitergereicht. Per Default "lauscht" der Server (intern:SocketWriter) auf Port 34567.

Daneben lassen sich Daten auch per TCP, UDP oder direkt über die seriellen Schnittstellen des Raspberry lesen und schreiben.

Anschliessend werden die NMEA Daten an den Decoder weitergereicht. Die dekodierten GPS- und AIS-Daten werden im Server abgelegt ("NMEA decoded data") und für den Zugriff per HTTP aus der WebApp bereitgestellt. Zusätzlich werden die dekodierten Daten auch benutzt, um Trackdateien zu schreiben.

Über den integrierten WebServer kann der Zugriff auf diese dekodierten Daten erfolgen (per HTTP GET, Antwort als json).

Der Route-Handler wertet eingestellte Routen (bzw. Wegpunkte) aus und berechnet daraus die Daten für eine Autopilot-Steuerung. Diese werden als RMB NMEA Datensätze wieder in die internen NMEA Daten eingespeist und stehen so an allen Schnittstellen zur Verfügung.

Falls gültige GPS Zeitinformationen empfangen werden, wird die Systemzeit des Raspberry entsprechend eingestellt.
Auf dem Raspberry gibt es noch einen Service, der den AvNav-Server beim Systemstart automatisch startet und es auch ermöglicht, ihn geordnet zu beenden.

Da der gesamte Server in Python geschrieben ist, kann er auch (vor allem zu Testzwecken) unter Windows, Osx (Mac) oder Linux laufen. Dazu muss Python ab 3.7 installiert sein (die Windows Installation bringt das selbst mit). Falls reale serielle Daten gelesen werden sollen, muss dazu noch pyserial installiert werden.

Der Server kann in weiten Grenzen durch eine XML-Datei konfiguriert werden, für die verschiedenen Nutzungsfälle liegen dokumentierte Beispiele vor. In den Versionen ab 20210322 können die meisten Einstellungen auch direkt in der App vorgenommen werden.

Die ausgelieferte avnav_server.xml Datei enthält Kommentare, so dass Anpassungen an die eigenen Bedürfnisse einfach möglich sein sollten.
Die Software ist auf github verfügbar - für die Installation sei die separate Beschreibung verwiesen..

Software auf dem Raspberry

Auf dem Raspberry ist die Software in der folgenden Verzeichnisstruktur installiert:

Verzeichnis	Inhalt
/usr/lib/avnav	die Software nach der Installation

/home/pi/avnav/data/	Basis für die Nutzer-Daten
.../data/charts	Verzeichnis für die Kartendateien -siehe Karten konvertieren.
.../data/log	logfiles
.../data/tracks	Die trackfiles (gpx). Sie werden in einem File pro Tag gespeichert, ausserdem werden Nmea-Logs aufgezeichnet.
.../data/routes	Routen - xxx.gpx und das aktuelle Segment (Leg) currentLeg.json
.../data/import	Hier abgelegte Karten werden konvertiert in das "gemf"-Format, sodass die WebApp sie verarbeiten kann

Bis auf die "systemd"-Scripte läuft die gesamte Software unter dem Nutzer pi (auf dem Raspberry) oder als beliebiger anderer Nutzer ("avnav" als default). Die Installation muss allerdings als "root" erfolgen.

Die Web App

Zur Navigation mit den auf dem Raspberry Pi vorhandenen Karten gibt es eine Web App. Diese ist mit ReactJs realisiert.
Die App kommuniziert mit dem in AvNav integrierten Webserver auf dem Pi. Die Einstiegsseite ist unter der url http://avnav.avnav.de/viewer/avnav_viewer.html erreichbar. Es ist eine sogenannte „single page app“, d.h. die weitere Kommunikation mit dem Server geschieht per Ajax durch den JavaScript-Anteil. Das Layout ist optimiert für die Darstellung auf einem 7 Zoll-Tablet oder größer, bei mir momentan im Einsatz: Nexus 7 am Navitisch, Blackberry Playbook draussen, sie läuft aber natürlich auch auf größeren Tablets (Ipad) oder auf einem Laptop/Desktop. Eine sinnvolle Nutzung ist ab etwa 900x540 Pixel möglich.

URL Parameter

Die WebApp unterstützt eine Reihe von URL Parametern mit denen man einige Funktionen steuern kann.

Parameter	Beschreibung

defaultLayout	Der Name eines existierenden Layouts das als initiales Layout genutzt wird.
defaultSettings	Der Name einer existierenden Einstellungsdatei (regex möglich). Diese wird genutzt als default, wenn AvNav das erste Mal in diesem Browser für diese URL startet. Beispiel: defaultSettings=.*localFirefox
fullscreen	Man kann einen Parameter in der Form "server:<command>" angeben. Command muss ein existierendes Kommando sein, das beim AVNCommandHandler konfiguriert wurde. Das wird ausgeführt, wenn der Fullscreen Button geklickt wird (anstelle der Fullscreen Funktion im Browser). Beispiel: fullscreen=server:fullCommand
dimm	Man kann einen Parameter in der Form "server:<command>" angeben. Command muss ein existierendes Kommando sein, das beim AVNCommandHandler konfiguriert wurde. Das wird ausgeführt, wenn der Dimm Button geklickt wird (anstelle der Nutzung einer Dimm Funktion in Java Script - wie bei AvNav auf Android) Beispiel: dimm=server:dimmCommand
noCloseDialog	Kein Dialog der warnt, wenn die AvNav Seite verlassen werden soll. Beispiel: noCloseDialog=true
splitMode	Starte AvNav im Split Mode. Beispiel: splitMode=yes
preventAlarms	Zeige keine Alarme. Beispiel: preventAlarms=true

Zur Beschreibung der WebApp hier.

Demo

Die Demo gibt einen kleinen Einblick in die Programm-Funktionen.
Der "connected" mode - also die Übertragung des Routing-Ziels von einem Browser zum anderen - funktioniert hier nicht.

Die Karten stammen (live) von http://kartor.eniro.se/ bzw. aus einem "meshup" vom BSH https://www.geoseaportal.de und von http://www.openseamap.org/ (siehe deren Copyright).
Die Daten vom BSH kommen sehr langsam - dadurch dauert der Karten-Aufbau etwas. Wenn man diese Karten verwenden möchte, dann kann man sie auch mit dem MobileAtlasCreator herunterladen - siehe Karten
Die GPS-Daten stammen von einer Tour im Sommer 2013 - Klintholm->Vitte. Damit kann man einige der Programm-Features sehen (inklusive der AIS-Anzeige).
Die Zeit ist etwas "komprimiert" - d.h. die GPS-Zeit läuft mit etwa doppelter Geschwindigkeit. Wenn die Demo "am Ende" ankommt, gibt es einen Sprung zurück in die Nähe von Klintholm - da sieht der Track dann etwas merkwürdig aus. In so einem Falle einfach das Demo-Fenster schließen und noch einmal neu starten.

Als Start sinnvoll auf den Boot-Button klicken und dann ein wenig zoomen...
Danach weiter spielen.
Die upload- und download-Funktionen werden potentiell nicht in allen Browsern korrekt im Demo Mode arbeiten. Mit Firefox und Chrome sollten sie aber testbar sein.

Neueste Version (Vorschau)

AvNav Quickstart

Ein Hinweis vorweg:

Ich kann keine Garantie für die Funktion der App übernehmen, insbesondere die Nutzung zu Navigationszwecken geschieht auf eigenes Risiko. In jedem Falle empfehle ich einen intensiven Test der Genauigkeit der Darstellung und des verwendeten Kartenmaterials.

Eine ausführliche Beschreibung der Konzepte findet man im Kapitel Einführung.

Generelles

AvNav ist primär ausgelegt für die Bedienung auf Touch-Geräten (auch mit relativ kleinen Bildschirmen). Es wurde versucht, die Bedienelemente und Anzeigen so zu gestalten, dass sie auch unter Bord-Bedingungen gut nutzbar sind.

Eine Bedienung über Maus (und Tastatur) ist natürlich ebenfalls möglich.

Installation und Inbetriebnahme

AvNav gibt es in 2 Varianten:

1. Client-Server Variante.
   Dabei wird der Server auf einem Linux- oder Windows-System installiert (z.B. einem Raspberry Pi). Als "Client" für die Bedienung dient ein beliebiger Browser z.B. auf einem Tablet.

2. Android App
   Hier ist die gesamte Funktionalität in einer App gebündelt. Auch hier können weitere Geräte per Browser zusätzlich zugreifen.

Client Server Variante

AvNav steht als Paket für verschiedene Linux-Distributionen (Debian Pakete, Rpm) sowie als Installer für Windows zur Verfügung.
Die Debian-Pakete sind in einem Repository verfügbar und können außerdem von der Release-Seite heruntergeladen werden.
Außerdem pflegen wir Images für den Raspberry Pi. Eine ausführliche Beschreibung findet sich im Kapitel Installation.

Nach der Installation und Start der App kann man sich mit dem Browser zu AvNav verbinden. Dadurch startet die WebApp.
Wenn man unsere Images nutzt, erzeugt der Raspberry ein WLAN (Name und Passwort können angepasst werden). Für Details, wie man sich nach der Installation mit dem Server verbindet, siehe die Image Beschreibung.
Wenn man über einen anderen Weg mit dem AvNav-Server verbunden ist, kann man http://avnav.local benutzen (oder die IP Adresse des Servers).
Für IOS- und Android-Geräte ist es empfehlenswert, einen Bonjour Browser zu nutzen. Dieser kann AvNav im lokalen Netz finden, so dass man ohne Adresseingabe sofort den Browser starten kann.

- IOS: 
- Android: 

Android App

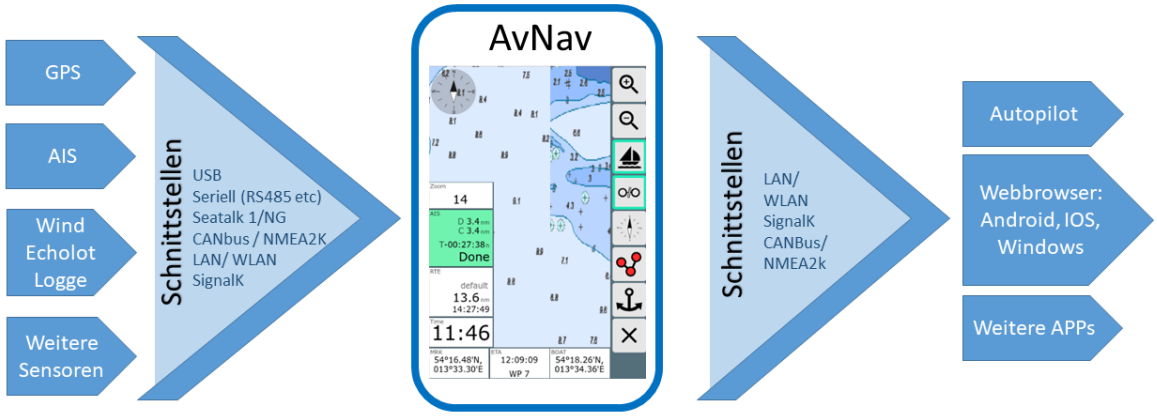
Die App steht im Play Store zur Verfügung 

Konfiguration

Nach der Installation oder der Nutzung eines Images wird AvNav in der Regel ohne weitere Konfiguration starten. An den USB-Anschlüsse werden serielle Schnittstellen mit ihren Baudraten automatisch ermittelt. Je nach gewählter Installation ist meist ein Empfänger für UDP-Daten auf Port 34667 aktiv.

Für weitergehende Anforderungen kann eine detaillierte Konfiguration des Servers über die Server/Status Seite vorgenommen werden. Eine Anpassung des Aussehens kann über die Einstellungen, über Anpassungen des Layouts und über nutzerdefinierten Code/CSS sowie Plugins vorgenommen werden.

NMEA-Daten



Client Server Variante

AvNav verarbeitet NMEA0183-Daten, die über angeschlossene USB-Geräte, serielle Schnittstellen, Bluetooth-Geräte, TCP (Client und Server) oder UDP empfangen werden können. Eine Multiplexer-Funktion erlaubt es, die Daten von allen angeschlossenen Schnittstellen zu empfangen und konfigurierbar an beliebige Schnittstellen wieder auszusenden.

Einen Teil der Daten dekodiert AvNav (Positionsdaten, AIS,...) und nutzt sie für die eigenen Anzeige-Funktionen.
Die folgenden NMEA-Datensätze werden von AvNav dekodiert:

- !AIVDM
- $xxGGA
- $xxGSV
- $xxGLL
- $xxVTG
- $xxRMC
- $xxMWV (ab 20220225 auch waterSpeed)
- $xxDPT
- $xxDBT
- $xxXDR (seit 20210114)
- $xxHDG (seit 210106xx)
- $xxHDT (seit 20210619)
- $xxHDM (seit 20210619)

- $xxVHW partiell (seit 20210619)
- $xxVWR (seit 20220225)
- $xxMTW (seit 20220225)
- $xxZDA (seit 20220421)

Je nach Konfiguration kann AvNav auch NMEA-Daten erzeugen:

- $GPRMB
- $GPAPB
- $AVXDR
- $AVMDA
- $AVMTA

In Zusammenarbeit mit Canboat and SignalK können auch NMEA2000- Daten empfangen und genutzt werden. Außerdem können alle Daten für das eigene Schiff, die in SignalK verfügbar sind, in AvNav angezeigt werden.
Ab Version 20220421 kann AvNav auch direkt seine Navigationsdaten von SignalK erhalten.

Android

Unter Android kann das interne GPS genutzt werden. Außerdem kann eine TCP- oder Bluetooth-Verbindung zum Empfang von GPS- oder AIS-Daten genutzt werden. Auf Geräten mit USB-Host-Funktion kann auch ein Seriell-USB-Adapter angeschlossen werden.

Es werden die folgenden NMEA-Daten dekodiert:

- !AIVDM
- $xxGGA
- $xxGSV
- $xxGLL
- $xxGSA
- $xxRMC
- $xxMWV (ab 20220225 auch waterSpeed)
- $xxDBT
- $xxXDR (seit 20210114)
- $xxHDG (seit 210106xx)
- $xxHDT (seit 20210619)
- $xxHDM (seit 20210619)
- $xxVHW partiell (seit 20210619)
- $xxVWR (seit 20220225)
- $xxMTW (seit 20220225)

Bei enstprechender Konfiguration kann AvNav $GPRMC und $GPRMB-Sätze (ab 20220225 auch $GPAPB) erzeugen und aussenden.

Karten

AvNav verarbeitet grundsätzlich Karten im Rasterformat. Diese können aus verschiedenen Quellen im Netz heruntergeladen werden (z.B. OpenSeaMap or NOAA) oder z.B. mit dem MobileAtlasCreator oder SASPlanet aus Quellen im Netz erstellt werden.

Unmittelbar verarbeiten kann AvNav Karten im "gemf"- und "mbtiles"-Format. Daneben können Karten z.B. im BSB-Format (kap) in das gemf Format konvertiert werden, entweder direkt auf dem Raspberry - oder besser vorher auf einem Desktop-System.

Karten müssen zur Benutzung in AvNav "hochgeladen" werden - dies geschieht auf der Files/Download-Seite, dort den Reiter "Karten"  anklicken.

Außerdem kann AvNav o-charts- Karten verarbeiten, unter Android allerdings nicht. Der O-Charts-Shop bietet Karten für Open-Source-Software zu sehr günstigen Preisen an.

O-charts Karten werden über das o-charts-Plugin hochgeladen.

Mit dem mapproxy-plugin kann AvNav darüber hinaus verschiedene Online-Karten einbinden. Für die Offline-Nutzung einzelner Bereiche von Online-Karten kann das Plugin ebenfalls genutzt werden.

Weitere Details herzu finden sich im Kapitel Karten.

Anzeigen

Auf der Seite "Navigation" werden dargestellt: Die aktuelle Boots-Position, der Kurs, die Route zum nächsten Wegepunkt, die aktuelle Route, AIS Ziele und deren Kurse, Navigationskreise und definierte Overlays.

Weitere Navigationsdaten können dargestellt werden auf der Seite "Navigation", im Routen-Editor und auf bis zu 5 Dashboard-Seiten, die selbst konfiguriert werden können. Das schließt auch die in SignalKvorhandenen Werte ein.

Es können einfache Zahlenwerte, analoge Anzeigen oder auch Grafik-Anzeigen realisiert werden.

Die Darstellung der Navigationsdaten kann in weiten Grenzen an eigene Bedürfnisse angepasst werden. Dazu dient der Layout-Editor Eigene Anzeige-Layouts können mit ein wenig JavaScript-Code und mit CSS erstellt werden.

Diese Layouts beinhalten auch eine Anpassung an unterschiedliche Bildschirm-Ausrichtungen und -auflösungen.

Routen

In AvNav können sehr einfach Routen erstellt und bearbeitet werden. Das erfolgt in der Kartenansicht im Routen-Editor. Im Normalfall verschiebt man die Karte so, dass der Mittelpunkt auf dem gewünschten Wegepunkt liegt, und fügt diesen dann mit einem Klick der Route hinzu. Punkte der Route können verschoben, gelöscht und bearbeitet werden.

Wegepunkte oder andere Routen, die als Overlays angezeigt werden, können im Routen-Editor der aktuellen Route hinzugefügt werden.

Eine Route kann invertiert werden, alle Wegepunkte können auch gelöscht werden. Routen werden in AvNav als "gpx"-Dateien gespeichert. Sie können über die Files/Download-Seite importiert und exportiert werden. Aus dem Routen-Editor kann die Navigation direkt gestartet werden. Innerhalb einer Route erfolgt eine Alarmierung, wenn der nächste Wegepunkt erreicht wird bzw. sobald man sich innerhalb des "approach" genannten Umkreises befindet. Es erfolgt ein automatisches Weiterschalten zum nächsten Wegpunkt.

Ab Version 20220819 kann AvNav zwei verschiedene Routing Modi nutzen:

great circle

Hier wird eine Route so berechnet das man den kürzesten Weg zwischen Start und Ziel hat. Der Nachteil daran ist, das sich der Kurs im Verlauf der Route permanent ändert. Eine solche Route ist in der Kartendarstellung keine Strecke sondern eine Kurve.
In älteren Versionen hat AvNav immer great circle Routen berechnet, diese aber (fälschlich) als Strecken dargestellt.
Für kürzere Distanzen (< 100nm) spielt das aber praktisch keine Rolle.

rhumb line

Hier wird die Route so berechnet, das ein konstanter Kurs gesteuert werden kann. Die Kartendarstellung ist eine Strecke.
Die Umschaltung erfolgt im Router auf der Server/Status Seite. Für das Mess-Tool kann in den Einstellungen der Web App (unter Navigation/Measure Display RhumbLine) der Modus separat eingestellt werden. Damit können leicht die beiden Wege verglichen werden.

Wegepunkt Weiterschaltung

Für die automatische Weiterschaltung zum nächsten Wegepunkt in einer Route müssen immer zwei Bedingungen erfüllt sein:

1. Das Boot muss sich im Annäherungsbereich des Wegepunktes befinden (kann man in den Einstellungen der WebApp vor dem Starten einer Route definieren). Das wird ersichtlich durch einen ausgelösten Wegepunkt-Alarm und eine rote Darstellung im Route-Widget.
2. Je nach eingestelltem Mode (Router auf der Server/Status Seite: nextWpMode) - neu ab 20220819
   "late" (der default und in älteren Versionen): Die Entfernung zum aktuellen Wegepunkt nimmt nicht mehr ab aber die Entfernung zum nächsten Wegepunkt nimmt ab
   "90": Der Wegepunkt liegt "querab" (genauer: Das Boot hat eine Linie +/- 90° zum originalen Wegepunktkurs überquert)
   "early": Die Weiterschaltung erfolgt eine (einstellbare) Zeit nach dem Wegepunkt-Alarm ohne weitere Bedingungen

Es ist wichtig zu beachten, das die Weiterschaltung nur erfolgt, wenn beide Kriterien erfüllt sind. Falls eine manuelle Weiterschaltung gewünscht ist, kann man jederzeit durch Klick auf die Widgets links unten die Wegepunkt-Buttons anzeigen un den Button nutzen.

Tracks

AvNav zeichnet beständig den aktuellen Track auf und zeigt ihn auf der Karte. Dabei wird versucht, die Zahl der Trackpunkte zu begrenzen, indem nur bei größeren Änderungen oder nach einer bestimmten Zeit erneut ein Trackpunkt geschrieben wird (Siehe Konfiguration ANVTrackWriter). Die Tracks werden in regelmäßigen Abständen als "gpx"-Datei gespeichert und können auf der Files/Download-Seite exportiert und importiert werden (dort ist auch eine Anzeige ihrer Daten - also Strecke, Zeit, usw. möglich). Pro Tag wird eine separate "gpx"-Datei erzeugt. Außerdem können vorhandene Tracks auf der Karte als Overlay dargestellt werden.

Vorhandene Tracks können in Routen umgewandelt werden. Dabei wird über einen Reduktionsalgorithmus die Zahl der Wegepunkte automatisch reduziert.

AIS

In der Kartendarstellung werden AIS-Ziele in einem bestimmten Umkreis (default: 20nm) mit ihren Positionen und ihren Kursen dargestellt. Außerdem erfolgt eine Berechnung des nächsten Zieles und eine Warnung, falls eine einstellbare minimale Begegnungsentfernung unterschritten wird (CPA).

Die Symbole für die AIS-Ziele können durch eigene Symbole angepasst werden, bei Bedarf unterschiedlich je nach AIS-Schiffsklasse oder navigational status. Für das eigene Boot und für die AIS-Ziele wird ein Kurs-Vektor gezeichnet, dessen Ende die Position nach 10 Minuten (einstellbar) markiert.

Ab 20230614 werden auch AIS Atons dargestellt (dazu muss in den Einstellungen/AIS "only show moving targets" ausgeschaltet und "show other" eingeschaltet sein). Ausserdem ist es möglich eine geschätzte aktuelle Position des AIS Ziels abhängig vom Alter der Information, Kurs und Geschwindigkeit anzuzeigen (Einstellungen/AIS "show estimated position").
Es kann ausserdem gewählt werden, ob die Ausrichtung des AIS Zieles nach HDG erfolgt (sofern empfangen - Einstellungen/AIS "use heading for direction") - sonst nach COG. Der Course Vector eines AIS Zieles wird immer nach COG ausgerichtet.
Es gibt unterschiedliche Symbole für die AIS Ziele - für Details siehe unter "Nutzerdefinierte Icons".

Durch Klick auf ein AIS-Ziel (oder auf die Anzeige des nächsten Zieles in den Anzeige-Bereichen) erhält man alle Informationen zu diesem Ziel und kann zur Liste aller AIS-Ziele navigieren.

Alarme

AvNav kann Alarme erzeugen für:

- eine Ankerwache
  Sie kann auf den Dashboard-Seiten aktiviert werden. Abhängig vom Layout werden die Anzeigen angepasst. Bei Verlassen des vorgegebenen Bereiches (oder bei Ausfall des GPS-Signals) erfolgt eine Alarmierung
- bei Erreichen des nächsten Wegepunktes
- "Mensch über Bord" (MOB)
  Für das Auslösen eines MOB-Alarms ist in allen Ansichten von AvNav eine separate  MOB-Taste vorhanden. Durch diese wird die aktuelle Position zum neuen Zielpunkt, alle anderen Routings werden beendet. Außerdem wird der Alarm ausgelöst.

Alarme werden in AvNav auf dem Server erzeugt und verwaltet. Insbesondere z.B. für die Ankerwache können alle Anzeige-Geräte abgeschaltet werden, aber die Überwachung läuft weiter.

Es kann bei den Alarmen eine akustische Signalisierung sowohl auf dem Server, als auch in der Client-Anzeige erfolgen. Auf dem Server können auch weitere Aktionen ausgelöst werden (siehe Beispiel). Die Konfiguration erfolgt im configfile - AVNAlarmHandler.

Nachtmodus

AvNav kann auf der Hauptseite in einen Nachtmodus  geschaltet werden. Dann sind alle Anzeigen entsprechend angepasst.

Fernsteuerung

AvNav kann die Anzeige Funktionen auf einem Gerät von einem anderen Gerät oder vom Server aus steuern. Für Details siehe die Beschreibung dazu.

Anpassung

AvNav kann in unterschiedlicher Weise an eigene Bedürfnisse angepasst werden. Da die gesamte Darstellung im Browser abläuft, kann man das Aussehen mit CSS anpassen.

Die Konfiguration des Servers arfolgt über die Datei avnav_server.xml. Eine Änderung sollte im Normalfall jedoch nicht in der Datei direkt erfolgen, sondern über die Bearbeitungsmöglichkeiten auf der Server/Status-Seite.

Neben der bereits beschriebenen Anpassung der Layouts im Layout-Editor können ohne größeren Aufwand eigene Anzeigen mit ein wenig Java Script Code eingebunden werden.

Es ist auch möglich, weitere externe oder AvNav-interne Webseiten als "User Apps" einzubinden und anzuzeigen.

Die zur Darstellung genutzten Symbole können über eine Json-Datei angepasst werden, ebenso die Tastaturkürzel für Aktionen.

Mit Python, JavaScript und CSS können eigene Plugins geschrieben werden, die die Funktionalität von AvNav erweitern.

AvNav Installation

Software Versionen

Eine Beschreibung der Versionen und Links zu den Downloads finden sich im Release Dokument.

Für Eilige ist hier der Link zum aktuellen Image, zu den Entwickler-Versionen und zu den Release-Downloads.

Um den Start zu vereinfachen, gibt es fertige Images für den Raspberry Pi. Ab Version 20220421 unterstützen die Images sowohl den sogenannten "headless" Betrieb - d.h. es ist weder Tastatur noch Monitor am Pi angeschlossen als auch einen Betrieb mit einem angeschlossener (Touch-) Bildschirm (gerne auch optional Tastatur und Maus).

AvNav ist von der Bedienung für Touch Geräte optimiert - aber man kann es natürlich auch mit Bildschirm, Tastatur und Maus bedienen.

Wie man die Images nutzt, hängt also vom Anwendungsfall ab. Im "headless" Betrieb wird der Raspberry nur als Server eingesetzt, die Anzeige erfolgt dann z.B. auf Mobilgeräten. Für diesen Fall reicht ein Raspberry Pi 3B(+). Wenn ein Monitor und Peripherie wie Tastatur und Maus direkt an den Raspberry angeschlossen werden, sollte man einen Pi4 mit mindestens 2GB Speicher wählen.

Wenn man ein komplettes Desktop System mit vielen weiteren Anwendungen haben möchte, kann die OpenPlotter-Variante eine gute Basis sein. Dafür empfiehlt sich ein Pi4 mit 4GB Speicher. Auch 2GB Arbeitsspeicher wird ausreichen - dann bleibt aber nicht viel Raum für zukünftige Anforderungen.

Die früher vorhandenen speziellen AvNav Touch images werden leider nicht mehr weiter gepflegt.

AvNav Images(vormals HeadlessImages)

Diese Images werden von BlackSea gepflegt (vielen Dank...). Eine Beschreibung findet sich auf seiner Webseite.

Unter Windows/Linux/OSx lädt man das Image von free-x herunter und transferiert es wie unter http://www.raspberrypi.org/downloads (raw images) beschrieben auf eine SD Karte.

Diese Images enthalten

- avnav
- avnav-update-plugin
- avnav-ocharts-plugin
- avnav-mapproxy-plugin
- avnav-history-plugin
- SignalK
- Canboat
- Support for MCS
- optional einen X-Server mit openbox und firefox im Kiosk Modus

Die Images sind so vorkonfiguriert, dass NMEA0183-Daten von allen Interfaces zu AvNav und von dort zu SignalK geleitet werden. AvNav holt sich zusätzlich alle Daten von SignalK und kann diese anzeigen. Für Details zur SignalK Integration siehe die Beschreibung.
NMEA2000-Daten laufen über Canboat zu SignalK und zu AvNav.
Für Details zu canboat siehe CanBoatAndSignalK.

Image Vorbereitung

neu ab Version "20210322", erweitert ab Version "20220421"

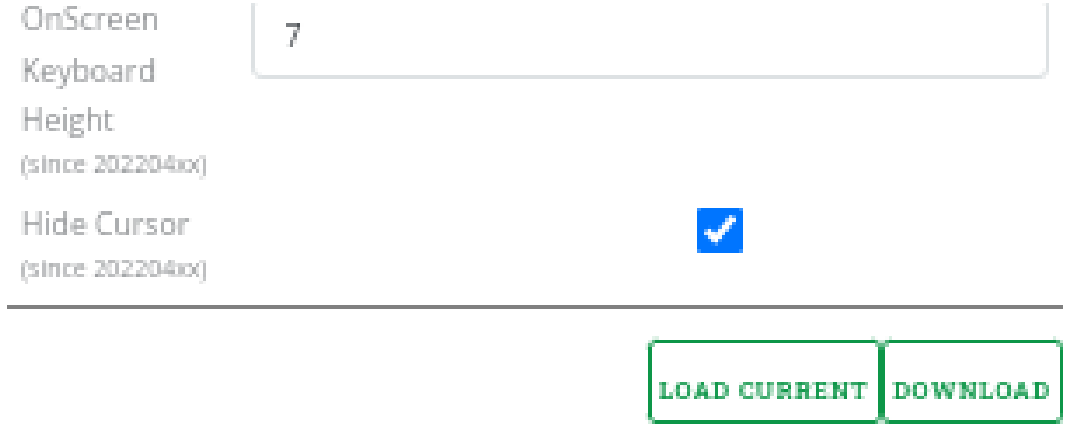
Bevor die fertig vorbereitete SD-Karte im Raspberry verwendet wird, sollte man einige Einstellungen anpassen. Das gilt vor allem für Passworte:
Die Images haben eine Konfigurationsdatei "avnav.conf". Sie findet sich in der ersten Partition der SD-Karte (Boot-Partition). Diese Datei kann mit einem Texteditor angepasst werden.
Dort kann auch eingestellt werden, ob ein lokaler Bildschirm genutzt werden soll ("Touch Variante")

Einfacher geht es mit einer kleinen Web-Oberfläche hier.

avnav.conf Generator

Hier kann eine avnav.conf Datei erzeugt werden, die dann in die erste Partition (boot) der SD Karte für den Pi gespeichert werden muss.
Die Daten werden nicht auf den Server geladen, sondern verbleiben komplett im Browser.
Auf jeden Fall sollten die Passworte geändert werden.

Config Sequence	1
Wifi SSID	avnav
Wifi Password	avnav-secret
User pi Password	raspberry
Hostname (since 20210429)	avnav
Base Board (since 202303xx)	NONE
HAT (since 202303xx)	NONE
Module RTL8188EU (since 202304xx)	☑
Module RTL8192EU (since 202304xx)	☐
TimeZone (since 20210429)	Europe/Berlin
Wifi Country (since 20210429)	Germany
Internal Wifi as Client (since 20210429)	☐
KeyboardLayout (since 20210429)	German
KeyboardType (since 20210429)	Generic 105-key PC (intl.)
TouchSupport (since 202204xx)	☐
DisplayDPI (since 202204xx)	96



Die Bedeutung der Felder:

Name	Default	Beschreibung
Wifi SSID	avnav	Der Name des WLAN-Netzwerks, das der Raspberry erzeugen soll. Die Images sind so vorbereitet, dass man durch Einstecken von WLAN-Adaptern auch weitere Netzwerke erzeugen kann. Daher wird eine einstellige Nummer an den Namen angefügt.
Wifi Password	avnav-secret	Das Passwort für das WLAN-Netzwerk. Das sollte in jedem Falle geändert werden. Jeder, der sich mit dem WLAN verbinden kann, kann damit auch die Navigation beeinflussen!
User pi password	raspberry	Das ist das Passwort für den Nutzer "pi". Dieser Standard-User wird genutzt, wenn man sich per SSH verbindet oder wenn man direkt per Monitor und Tastatur auf den Raspberry zugreift. Das Passwort für den User "pi"

		sollte ebenfalls unbedingt geändert werden.
Base Board	None	Hier kann man aus unterstützten Basis-Platinen wählen. • **MCS:** Wenn diese Optoin aktiviert ist, wird beim nächsten Bootvorgang die notwendige Software für den Marine Control Server von GeDad aktiviert. Die Änderung der Einstellung führt dann zu einem automatischen Reboot, wenn der Raspberry das erste mal mit dieser Einstellung startet. • **OBPPLOTTERV3:** Hiermit werden die Einstellungen für den Open Boat Projects Plotter (V3) gesetzt.
HAT	None	Hier kann man einen unterstützten Pi-HAT auswählen. AvNav wird die entsprechenden Einträge für die Overlays in /boot/config.txt machen und die CAN Netzwerk-Schnittstellen anlegen. • WAVESHAREB: waveshare RS485 CAN HAT (B) • WAVESHAREA8: waveshare RS485 CAN HAT (8Mhz) • WAVESHAREA12: waveshare RS485 CAN HAT (12 Mhz) • WAVESHARE2CH: waveshare 2CH CAN HAT • PICANM: PICAN-M

Module RTL8188EU	aus	Wenn eingeschaltet, wird der Kernel-Treiber für WLAN Adapter mit dem Chipsatz RTL8188EU per DKMS eingerichtet. Wenn der Kernel des Systems aktualisiert wird (Kommandozeile) wird der Treiber neu übersetzt.
Module RTL8192EU	aus	Wenn eingeschaltet, wird der Kernel Treiber für WLAN Adapter mit dem Chipsatz RTL8192EU per DKMS eingerichtet. Wenn der Kernel des Systems aktualisiert wird (Kommandozeile) wird der Treiber neu übersetzt.
TimeZone	Europe/Berlin	Die Zeitzone, die im Image genutzt werden soll.
WifiCountry	Germany	Das Land (muss für den Wifi Adapter aus legalen Gründen gesetzt werden)
InternalWifi as Client	aus	Wenn eingeschaltet, wird der interne Wifi Adapter des Pi nicht als Access Point definiert, sondern kann sich mit anderen Netzwerken verbinden. Achtung: Das erfordert eine andere Möglichkeit, um auf den Pi zugreifen zu können - siehe [Verbinden mit dem Raspberry].
KeyboardLayout	German	Layout für eine angeschlossene Tastatur

		(Kommandozeile und X)
KeyboardType	Generic 105-key PC(intl.)	Typ der angeschlossenen Tastatur
TouchSupport (ab 20220421)	aus	Wenn eingeschaltet, startet ein X-Server mit einem Firefox Browser im Kiosk Modus. Über einen Button in AvNav kann auf einen anderen "Bildschirm" gewechselt werden, über den File Manager, Terminal u.ä. verfügbar sind.
Display DPI (ab 20220421)	96	Nur für den lokalen Bilschirm. Die dots/inch für das angeschlossene Display. Beim Klick öffnet sich ein kleiner Rechner, in dem die Abmessungen des Bildschirmes in mm und Pixel angegeben werden können, daraus wird der DPI Wert berechnet. Basierend auf diesem Wert, werden einige Anzeige-Elemente skaliert.
OnScreen KeyboardHeight (ab 20220421)	7	Die Höhe einer Tastenzeile beim angezeigten OnScreen keyboard. Bei korrekter DPI Einstellung sollte dieser Wert ein guter Kompromiss sein. Wenn man den Wert sehr gross wählt, bleibt u.U. bei angezeigter Tastatur nicht mehr genug Bildschirmfläche...
HideCursor (ab 20220421)	an	Verbergen des Cursors auf dem lokalen Bildchschirm. Wenn mit einer Maus gearbeitet werden soll, muss

	dieser Schalter auf "aus" gesetzt werden.

Nach dem Eintragen der Werte kann man durch Klick auf den "download"-Button die "avnav.conf"-Datei herunterladen. Diese muss in die erste Partition der SD-Karte gespeichert werden. Eine eventuell dort vorhandene Beispieldatei muss überschrieben werden! Diese Partition muss dazu natürlich auf dem Computer sichtbar sein. Unter Windows wird man in der Regel nur die erste Partition sehen können. [Eventuell muss man dazu nach dem Schreiben des Images die SD-Karte noch einmal enfernen und wieder einstecken.] unklarer Satz, warum muss man das?

Es empfiehlt sich daher, die "avnav.conf"-Date noch einmal an einem sicheren Platz zu speichern, um sie ggf. beim Erzeugen einer neuen SD-Karte wiederverwenden zu können.

Nun kann man die SD-Karte in den Raspberry stecken und ihn starten. Der erste Boot kann einige Zeit dauern, da das gesamte Dateisystem auf der SD-Karte erzeugt werden muss. Je nach den Einstellungen in der Konfiguration wird der Raspberry noch ein weiteres Mal neu starten.

Wenn der Raspberry seine Systemeinrichtung endgültig abgeschlossen hat, kann man sich mit ihm verbinden.

Verbinden mit dem Raspberry Pi

Wenn das Image für einen lokalen Bildschirm konfiguriert wurde, kann man natürlich direkt mit einem angeschlossenen Bildschirm, ggf. noch Tastatur und Maus arbeiten.

Allerdings sollte man auch diesem Falle eine der hier im Folgenden beschriebenen Verbindungen vorbereiten - die braucht man eventuell in Fehlersituationen.

Prinzipiell kann man sich auf mehrere Arten mit dem Raspberry verbinden:

1. per Ethernet-Kabel
   Das geht entweder durch Anschluss einen einen Router oder Switch oder über eine einfache Verbindung z.B. direkt zu einem Laptop.
2. per internem WLAN
   Standardmässig macht der Pi einen Access-point mit der in der Konfiguration gewählten SSID auf. Er hängt dort jeweils noch eine nummer an (falls man weitere WLAN Adapter anschliesst, kann man auch mehrere Access Points erzeugen).
3. per USB von einem Android Gerät
   Moderne Android Geräte haben meist eine "USB-Tethering" Funktion, über die man das WLAN oder die Mobilfunk-Verbindung per USB weitergeben kann (leider meist nur Geräte mit Mobilfunk).
   Über diesen Weg kann man sich auch mit dem Pi verbinden.

4. Über ein anderes WLAN.
   Das erfodert aber zunächst eine der anderen Verbindungsmöglichkeiten, da man die Zugangsdaten einstellen muss. Ausserdem erfordert es einen zusätzlichen WLAN Adapter, der ein eine bestimmte USB Buchse gesteckt werden muss (ausser man hat in der Konfiguration "Internal Wifi as Client" gewählt).

Verbindung per Ethernet Kabel

Wenn man den Pi mit einem Router verbindet (z.B. im Heimnetz) dann erhält er von diesem eine IP Adresse. Über diese Adresse kann man sich mit dem Pi verbinden.
Da es oft mühsam ist, diese Adresse herauszufinden, macht sich der Pi im Netz per mDNS (Bonjour, Avahi) bekannt.
Auf diese Weise kann man sich z.B. mit einem Browser einfach zu AvNav verbinden:

```
http://xxxx.local:8080
```

xxx ist dabei der in der Image Konfiguration gewählte Hostname.
Auch ein Zugang per SSH (unter Windows z.B. per putty) íst auf diese Weise möglich - das Zugangs-Passwort für den Nutzer pi wurde ja in der Image Konfiguration gesetzt.

Falls man sich mit einem Netzwerkkabel direkt z.B. mit einem Laptop verbindet, wird der Pi nach einiger Zeit selbständig eine IP Adresse aufsetzen. Das kann 1...2 Minuten dauern. Diese gehört zum sogenannten Automatic Private IP Addressing Bereich 169.254.x.x. Die meisten Desktop Systeme unterstützen das ebenfalls (unter Linux muss man es ggf. explizit anschalten).
Wenn also der Laptop auch auf seinem Ethernet Interface eine solche Adresse aufgesetzt hat, sollte eine Verbindung wie beschrieben per xxxx.local funktionieren.

Falls der Zugriff über die xxx.local Adresse nicht funktionieren sollte, muss man versuchen, die IP Adresse des Pi zu ermitteln (z.B. in der Administration des heimischen Routers).

Verbindung über das eingebaute WLAN

Man kann das WLAN-Netzwerk verwenden, das der Raspberry erzeugt hat. Die SSID und das Passwort wurden wie oben beschrieben in der Datei "avnav.conf" definiert (mit noch einer angehängten Nummer).

Auch hier steht man vor dem Problem, zunächst die IP-Adresse des Pi herauszufinden. Wie schon beim Ethernet Zugang beschrieben, sollte auch hier der Zugriff per mDNS funktionieren.

```
http://xxx.local
```

Falls das nicht funktioniert, kann man es mit den festen IP Adressen 192.168.30.10, 192.168.40.10, 192.168.50.10, 192.168.60.10 versuchen:

```
http://192.168.30.10
```

Das sollte die Hauptseite von AvNav laden. Es sollte auch möglich seine, xxxx.local zu benutzen, wenn man sich mit dem Raspberry per SSH verbinden will (z.B. putty unter Windows).

Eine Einschränkung bleibt: Leider funktioniert xxx.local nicht auf Android-Geräten. Daher empfehle ich, dort ein Tool zu nutzen, das mDNS nutzen kann - einen BonjourBrowser . Für IOS gibt es ein vergleichbares Tool - auch wenn dort der Eintrag "xxx.local" im Browser funktioniert. Man wird seinen Raspberry mit dem AvNav-Image in den Browsern unter dem Namen "avnav-server" finden. Typischerweise wird man noch einen zweiten Eintrag "avnav" sehen - dahinter verbirgt sich der SignalK-Server auf dem Raspberry.
Wenn man seinen Raspberry im Bonjour-Browser sehen kann, der Aufruf der Seite dann aber fehlschlägt, kann es an einer Besonderheit von Android liegen, wenn zusätzlich z.B. per Mobilfunk eine Internet-Verbindung aktiv ist. In diesem Falle sollte man mobile Daten zeitweilig abschalten.

Ab der Version 1.12 unterstützt die Android BonjourBrowser app auch SSH. Das AvNav Image (ab 20220421) macht auch seinen SSH Zugang per mDNS bekannt. Wenn man unter Android dann noch einen passenden SSh client installiert (beispielsweise JuiceSSH) kann man sich auf diese Weise auch per SSH mit dem Pi verbinden. Das ist für ein normales Arbeiten meist nicht so komfortabel - aber für den Notfall kann man so ein paar Kommandos eingeben.

Wenn man sich per SSH verbindet, ist der Nutzername "pi". Das Nutzer-Passwort wurde in der Datei "avnav.conf" (hoffentlich) gesetzt. .
Wenn das in der Konfiguration gesetzte Passwort nicht funktioniert, kann man noch einmal das Default-Passwort versuchen. Es lautet "raspberry". Eventuell wurde die avnav.conf zuvor nicht korrekt gespeichert.
Eine Root-Shell kann man mit sudo -i erhalten.

Verbindung von Android über USB

Dazu benötigt man ein Android Gerät, das USB Tethering unterstützt (meist bei den Verbindungseinstellungen). Nachdem man das Gerät per USB mit dem Pi verbunden hat, muss man das USB Tethering einschalten (wird meist automatisch wieder ausgeschaltet, wenn man die Verbindung trennt).
Neben der Möglichkeit, den Pi so mit dem Internet zu verbinden, kann man auch auf den Pi mit dem Browser oder per SSH zugreifen. Da auch wieder die Ermittlung der IP Adresse

erfolgen muss, empfehle ich wieder die Bonjour Browser App zu installieren - siehe unter WLAN. Für SSH Zugriffe ebenfalls wieder JuiceSSH.

Über diesen Weg kann man auch auf den Pi zugreifen, falls z.B. das WLAN nicht funktioniert. Im BonjourBrowser wird man 2 http: Adressen finden (Port 8080 für AvNav und Port 3000 für SignalK). Dazu (ab 20220421) noch einen SSH Zugang.

Verbindung über ein anderes WLAN

Wenn man wie unten beschrieben eine WLAN Verbindung zu einem anderen Netzwerk eingerichtet hat (erfordert einen WLAN Stick oder Umschaltung des internen WLANs auf Client), kann man den Zugriff auf den Pi über dieses Netzwerk freigeben ("external access" beim Aufsetzen).

Das sollte man aber nur in einem geschützten Netzwerk tun (z.B. das Netz eines eigenen LTE Routers). **Auf keinen Fall sollte man das in einem öffentlichen WLAN erlauben - der Zugriff ist nicht geschützt und prinzipiell kann jeder aus dem Netz auf den Pi zugreifen**.

Wenn man mit dem client Netzwerk verbunden ist, kann man wieder wie unter WLAN beschrieben auf den Pi zugreifen.

Pi mit dem Internet verbinden

Für einige Funktionen (z.B. update von Software) benötigt der Pi eine Internet Verbindung. Diese wird natürlich nicht für die grundlegenden Navigationsfunktionen benötigt.

Vor der Image Version 20220421 ist dabei zu beachten, das der Pi nicht automatisch seine Zeit einstellt, wenn kein GPS angeschlossen ist. Das kann bei vielen Internet Zugriffen zu Problemen führen.
Ab der Version 20220421 synchronisiert der Pi nach einer Wartezeit automatisch seine Zeit mit dem Netz (ntp).

Für die Verbindung zum Internet gibt es die folgenden Möglichkeiten:

1. Ethernet Verbindung zu einem Router
2. Verbindung über ein anderes WLAN
3. Verbindung über ein per USB angeschlossenes Android Gerät

Der Pi stellt seine Internet-Verbindung grundsätzlich über sein eigenes WLAN auch verbundenen Geräten zur Verfügung.

Verbindung über Ethernet Kabel

Hier wird der Pi über ein Ethernet Kabel an einen Router angeschlossen.
Dazu ist auf dem Pi nichts weiter einzurichten, das sollte automatisch gehen.
Auf einigen Pi3 kann es vorkommen, das ein Netzwerkkabel was erst nach dem boot

angeschlossen wird, nicht richtig erkannt wird. In diesem Falle den Pi mit angeschlossenem Netzwerk neu starten.

Verbindung über ein anderes WLAN

Dazu wird ein weiterer WLAN Adapter (USB Adapter) benötigt. Bitte vorher die Kompatibilität mit dem Pi prüfen - z.B. hier.

Der Stick muss wie im Bild gesteck sein (auf dem Pi4 die blaue USB Buchse an der Platinen-Seite).
Der interne Name des Netzwerk-Interfaces ist wlan-av1.

Alternativ kann in der Image-Konfiguration "InternalWifi as Client" gesetzt werden, damit wird der interne WLAN Adapter für die verbindung zu anderen Netzten verfügbar. Dann benötigt man aber einen anderen Zugriff zum Verbinden mit dem Pi da er keinen Access Point mehr aufmacht.

Man kann die Verbindung zu einem WLAN in der App konfigurieren.
Bei jedem WLAN, mit dem man sich verbindet, kann man auswählen, ob ein Zugriff auf den Pi von aussen möglich sein soll ("external access"). Wenn das nicht ausgewählt ist, kann über dieses WLAN nicht auf AvNav zugegriffen werden. Bitte die Hinweise zum Zugriff beachten.

Verbindung über ein per USB angeschlossenes Android Gerät

Wie bereits beim Zugriff beschrieben, kann man ein Android Gerät mit USB Tethering verbinden. Intern ensteht ein Netzwerkinterface usb0.
Darüber kann der Pi ebenfalls auf das Internet zugreifen.

Das kann eine einfache Möglichkeit sein, wenn man den Zugriff nur temporär braucht und keinen zusätzlichen WLAN Adapter zur Verfügung hat.

Falls man vorher anders mit dem Internet verbunden war, kann es sein, das der Pi die USb Verbindung erst nach einem Neustart wirklich nutzt (Achtung: USB Tethering auf dem Android Gerät wieder einschalten, wird beim Pi-Neustart normalerweise ausgeschaltet).

Technische Details

Der Raspberry wird ein (oder mehrere) WLAN-Netzwerke aufsetzen, eines mit dem internen Adapter und weitere mit potentiell gesteckten WLAN-Sticks. Diese Netzwerke haben die Adressen:192.168.20.0/24, 192.168.30.0/24, 192.168.40.0/24, 192.168.50.0/24. Der Raspberry selbst hat dabei jeweils die Adresse 192.168.x.10.

Auf dem Raspberry wird dazu ein DHCP-Server und ein DNS-Server eingerichtet (dnsmasqd).

Wenn der Raspberry über ein Ethernet-Kabel verbunden wird, versucht er per DHCP eine Adresse aus dem Netzwerk zu erhalten. Er setzt dann eine NAT-Weiterleitung aus seinem WLAN-Netz zum Ethernet auf. So kann z.B. eine Internetverbindung aufgebaut werden, während man in das WLAN des Raspberry eingewählt ist.

Für die meisten Aktionen sollte ein Kommandozeilen-Zugang jedoch nicht erforderlich sein. Für Updates nutzt man das bereits vorinstallierte Update-Plugin. Die Server-Konfiguration kann innerhalb der App auf der Server/Status-Seite vorgenommen werden.

Image mit Bildschirm

In früheren Versionen gab es ein eigenes AvNav Touch Image.
Dieses wird jedoch nicht mehr weiter gepflegt.
Ab der Versiion 20220421 wurde daher der Support für einen angeschlossenen Bildschirm (mit dem Schwerpunkt touch) in die AvNav images integriert.

Wie unter Vorbereitung beschrieben, kann man die Unterstützung für einen Bildschirm dort aktivieren.
Wenn das dort eingeschaltet wurde, startet ein service "avnav-startx". Dieser erzeugt einen lokalen X Server, eine Nutzer-Sitzung für den Nutzer pi mit openbox als Fenster-Manager und Firefox im Kiosk Mode.

Als Bildschirm-Tastatur (On Screen Keyboard) wird onboard verwendet.

Auf der AvNav Hauptseite (und auf einigen anderen Seiten) wird ein "Raspberry" Button angezeigt, mit diesem wechselt man auf einen zweiten virtuellen Bildschirm, auf dem man einen Dateimanager, ein Terminal und verschiedene weitere Tools findet.
Das System ist ganz bewusst nicht als ein komplettes Desktop-System ausgelegt, um möglichst Resourcen schonend zu arbeiten.

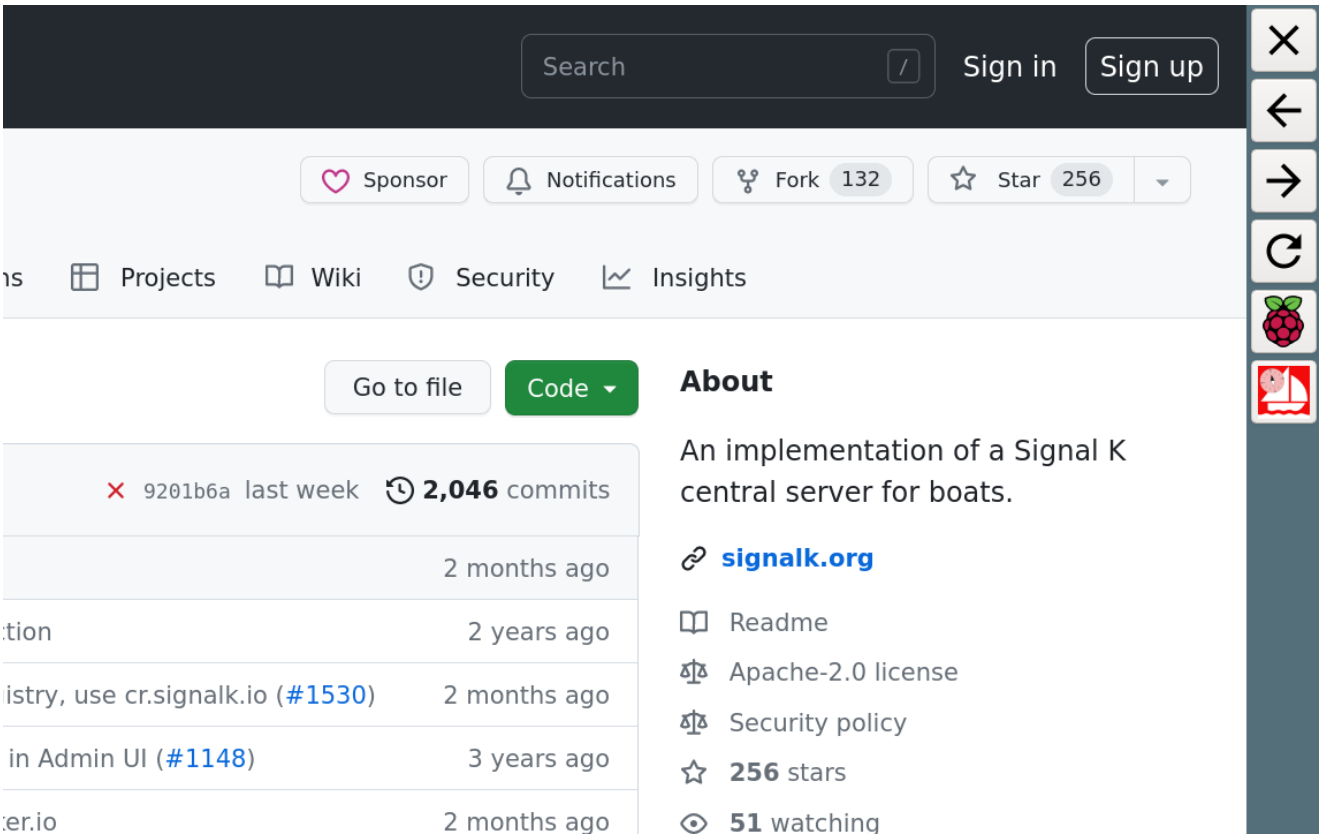
Da man an die Systemtools nur über den Button in der AvNav app herankommt, ist es sinnvoll, sich einen weiteren Zugang zum Pi wie weiter oben beschrieben zuzulegen.
Damit kann man im Fehlerfall auf das System zugreifen.
Ein Restart der Nutzeroberfläche von der Kommandozeile kann mit

```
sudo systemctl restart avnav-startx
```

erfolgen.
Falls Firefox einmal nicht mehr richtig starten möchte, kann man das Nutzerprofil entfernen. Das wird beim nächsten Start neu angelegt.
Achtung: AvNav Einstellungen, die nicht auf dem Server gespeichert wurden, gehen dabei verloren.

```
sudo systemctl stop avnav-startx
rm -rf /home/pi/.mozilla/firefox/avnav
sudo systemctl start avnav-startx
```

Ab Version 20230614 wird auf dem Hauptbildschirm immer dann, wen AvNav nicht (oder nicht komplett) aktiv ist, ein zusätzliches Panel angezeigt.



Über diese Panel können einige Navigationsfunktionen in firefox gesteuert werden, es kann zum 2. Bildschirm (system) gewechselt werden - und man kann die (oben beschriebene) Reset-Funktion für das firefox Nutzerprofil ausführen ( ).

Damit ist eine Bedienung des Systems auch möglich, wenn einmal AvNav nicht komplett startet. Die Reset Funktion findet sich auch auf dem System-Bildschirm (allerdings nur für komplette Neu-Installationen).

Paket Installation

Dank Oleg gibt es fertige Paket-Repositories, die man in sein Debian-Linux einbinden kann. Das geht auf dem Raspberry Pi - aber auch auf jeder anderen Debian-Variante (z.B. Ubuntu). Informationen dazu findet man wieder in seiner Beschreibung.
Die Paketquellen bindet man wie folgt ein. Das ist nur nötig, wenn man nicht das AvNav-Image nutzt.

Debian Buster (amd64,armhf,arm64)

```
wget https://www.free-x.de/debian/oss.boating.gpg.key
sudo apt-key add oss.boating.gpg.key
wget https://www.free-x.de/debian/boating.list
sudo cp boating.list /etc/apt/sources.list.d/
```

Debian Bullseye (amd64,armhf,arm64)

```
wget -O - https://www.free-x.de/debian/oss.boating.gpg.key | gpg --
dearmor | sudo tee /usr/share/keyrings/oss.boating.gpg
echo "deb [signed-by=/usr/share/keyrings/oss.boating.gpg]
https://www.free-x.de/debian bullseye main contrib non-free" | sudo
tee -a /etc/apt/sources.list.d/boating.list
```

Ubuntu Jammy (amd64)

```
wget -O - https://www.free-x.de/ubuntu/oss.boating.gpg.key | gpg --
dearmor | sudo tee /usr/share/keyrings/oss.boating.gpg
echo "deb [signed-by=/usr/share/keyrings/oss.boating.gpg]
https://www.free-x.de/ubuntu jammy main" | sudo tee -a
/etc/apt/sources.list.d/boating.list
```

Für die Installation auf einem Linux System muss man nach Einbindung der Paketquellen die folgenden Schritte ausführen:

```
sudo apt update
sudo apt install avnav
```

Danach kann man als beliebiger Nutzer mit dem Kommando

```
avnav
```

den Server starten.
Mit

```
sudo systemctl enable avnav
sudo systemctl start avnav
```

kann man AvNav mit dem Benutzer "avnav" automatisch beim Systemstart aktivieren.
Alternativ kann man auch die Debian-Pakete direkt von der Download-Seite herunterladen:

- Releases
- Tägliche Builds

Nach dem Herunterladen kann man die Pakete auf einem raspberry pi mit

```
sudo apt install ./avnav_xxxxxxxx_all,deb
```

installieren.
Falls man sein raspberry pi System vergleichbar zu unseren Images aufsetzen möchte, kann man dazu noch das "avnav-raspi"-Paket installieren.
Das ändert die Netzwerk-Konfiguration so , wie AvNav das möchte, sorgt dafür, das AvNav unter dem Nutzer "pi" startet und aktiviert das Einstellen der Systemzeit sowie das Verwalten von WLAN client Netzwerken.

Ich würde in jedem Fall empfehlen, das AvNav Update-Plugin zu installieren - aus dem Paket Repository mit

```
sudo apt-get install avnav-update-plugin
```

oder mittels Download von GitHub.

Wenn man nicht das "avnav-raspi"-Paket installiert, man aber AvNav mit einem anderen Nutzer als "avnav" - also z.B. dem Nutzer pi starten möchte, braucht man einige zusätzliche Schritte.

Man kann dann als Nutzer "pi" AvNav einfach von der Kommandozeile starten lassen.
Wenn man AvNav als "systemd service" laufen lassen möchte, sollte man das Verzeichnis:
/usr/lib/systemd/system/avnav.service.d
anlegen und dort die Datei raspberry.conf hinein kopieren.
Die Zeile 5 in der Datei muss dann noch etwas abgeändert werden, da das dort angegebene Template für die avnav_server.xml nicht existiert.
Also muss geändert werden in:

```
ExecStart=/usr/bin/avnav -q -b /home/pi/avnav/data -t /usr/lib/avnav/avnav_template.xml
```

Danach kann man mit den Kommandos

```
sudo systemctl daemon-reload
sudo systemctl enable avnav
sudo systemctl start avnv
```

Avnav als Systemdienst starten. Wenn man diese Datei nicht anlegt/kopiert, wird AvNav nicht mit den Nutzer "pi", sondern mit dem Nutzer "avnav" arbeiten.

Wenn man auch die Karten-Konvertierung auf dem Linuxrechner mit einer kleinen GUI machen möchte, muss zusätzlich das Paket "python3-wxgtk3.0" installiert werden. Dann muss man AvNav mit

```
avnav -g
```

starten. Das sollte im Normalfall aber nicht nötig sein, man kann auch direkt in der App die zu konverierenden Karten hochladen.

OpenPlotter

Für OpenPlotter gibt es eine komplette Integration von AvNav (Dank an e-sailing). Im Repository https://www.free-x.de/deb4op/ , das bereits standardmäßig mit OpenPlotter 2 (und 3) kommt, sind die notwendigen Pakete bereits vorhanden. Somit kann man sie einfach installieren:

```
sudo apt update
sudo apt install openplotter-avnav
```

Seit 2021/03 ist AvNav offiziell in OpenPlotter verfügbar. So sollte nach einem Update von OpenPlotter "openplotter-avnav" bereits verfügbar sein.

Das Paket "avnav-raspi_xxx.deb" sollte man auf OpenPlotter nicht installieren, weil es sich nicht mit den Netzwerkeinstellungen von OpenPlotter verträgt. Innerhalb der OpenPlotter-AvNav-Konfiguration kann man den HTTP-Port für AvNav ändern, wenn es Probleme mit anderen Apps geben sollte. Die Defaultwerte sind: :8080 für den Browserzugriff, :8082 für ocharts.

Wenn man AvNav mit der OpenPlotter-App installiert, empfängt AvNav alle NMEA-Daten von SignalK und sucht nicht selbst nach USB Geräten. Alle Geräte-Konfigurationen oder Schnittstellen-Einrichtungen können so direkt in OpenPlotter und SignalK vorgenommen werden.

Windows

Für Windows gibt es einen Installer. Dieser wird nicht bei jedem Release neu ausgeliefert. Die aktuelle Version findet man hier.
Es empfiehlt sicht, den "AvNavNetSetup-jjjj-mm-dd-0.exe"-Installer zu verwenden, da dieser die eigentliche Software nachlädt und somit eine Möglichkeit bietet, jeweils die neueste AvNav-Version zu installieren, ohne den Installer selbst neu herunterladen zu müssen.
Die Windows-Version ist primär auf die Karten-Konvertierung ausgelegt, bietet aber die volle Funktionalität und kann somit auch zur Navigation unter Windows verwendet werden. Wie bei allen anderen Versionen bietet auch die Windwos-Version den Zugriff per Browser von anderen Geräten aus über das Netzwerk..

Avnav Releases

Das Verzeichnis mit allen Releases findet man hier.

Hinweise

Die Installationsanleitung enthält eine detaillierte Beschreibung der Installationsoptionen. Auf dieser Seite finden sich Informationen zu den einzelnen Release-Versionen.
Nicht mit jedem Update werden neue Images bereitgestellt. Zwischen-Updates sind entweder Entwickler-Versionen oder Releases.
Um ein solches Update zu installieren, benötigt man eine Kommandozeile auf dem pi (oder das AvNav Update Plugin). Von Windows aus geht das z.B. mit putty . Dazu die IP Adresse vom pi ermitteln und dann mit putty dorthin verbinden. Nutzername: pi, Passwort: raspberry (bzw. so, wie man das für das Image angepasst hat).
Wie man die Pakete installiert, ist ebenfalls in der Installationsanleitung beschrieben.
Anschliessend muss man (Hinweise auf dieser Seite beachten) eventuell noch die Konfigurationsdatei

```
/home/pi/avnav/data/avnav_server.xml
```

anpassen. Dass kann man z.B. so machen:

```
sudo systemctl stop avnav
nano /home/pi/avnav/data/avnav_server.xml # dann ändern und speichern
sudo systemctl start avnav
```

Wenn man das AvNav Update Plugin installiert hat, kann man die Konfiguration auch von dort bearbeiten ohne die Kommandozeile nutzen zu müssen.

Auf dieser Seite sind nur Hinweise zu den release-Versionen enthalten, für tägliche Builds kann man die commits auf GitHub prüfen.

Versionen

20230702 link

Fokus: AvNav Android :

- Freigabe für Android SDK 33, neu im Play Store (enthält alle Änderungen von 20230426)
- Quellen Prioritäten für Input Kanaäle (wie in der Linux Variante)

- #284: Anzeige der Versions-Information

AvNav base (avnav) :

- Verbessertes Handling für das Verschieben der Karte im "course up" Modus
- Verbesserte Schrift auf der aisinfo-Seite
- #283: Korrektes Handling für Button-Verkleinerung bei "2 button columns"
- #282: Korrektes Handling für das automatische Verbergen der Buttons mit "2 button columns"
- #275: Installation auf Bookworm scheitert
- #286: Höhere Baudrate. z.B. Moitessier Hat

20230614 link

AvNav Images (avnav-raspi):

- Verbessertes Handling für Images mit Touch (auch OBP Plotter)
  - Neues Panel auf der Startseite, wenn AvNav nicht komplett startet (siehe Dokumentation)
  - Reset für AvNav GUI (firefox Profil)
  - **Wichtiger Hinweis für OBP Plotter**: Bitte auch avnav-obp-plotter-v3-plugin mindestens auf 20230601 updaten und nach allen Updates neu starten!

- #274 update RPI4 (CM) Versionen in uart_control (GPS auf obp plotter v3)
- Update auf canboat >= 4.12 um einen Fehler im n2kd zu korrigieren (neuer Prozess alle 30s, kann alle Systemressourcen verbrauchen)

AvNav base (avnav):

- #272: Verbessertes AIS Handling (Dokumentation)
  - verschiedene Icons je nach navigational status
  - nutze HDG zur Anzeige, wenn verfügbar
  - Anzeige von Atons
  - zeige eine geschätzte aktuelle Position

- #276: Karte verschieben im Lock Modus(Dokumentation)
- #275: Installation auf bookworm (kein pip im postinstall)
- #277: requirements für fedora
- Installation auf OpenSuse
- zeige die formatter Parameter im EditWidget Dialog für Widgets mit flexiblen Formattern
- ermögliche die Nutzung von online (AIS) streams die nach IEC 62320-1 arbeiten: Neuer Parameter stripLeading für Reader

- #279:Korrigiere Zugriffsrechte für den avnav System Nutzer für serielle/USB Geräte
- Weniger strenge checks für kml Overlays
- Keine Abfrage der Sound-Berechtigung, wenn Sound deaktiviert ist

AvNav Android:

- #273: AIS auf Android für südliche Breiten

20230426 link

Unterstützung für dem OpenBoatProjects 10 Zoll Plotter V3 - siehe Projekt Dokumentation
- Neues Plugin obp-plotter-v3
- Unterstützung in avnav.conf und in der Image Vorbereitung

Konfiguration für einige Raspberry Pi HATS
- für AvNav Images können wir jetzt die notwendigen Einstellungen für einige HATs automatisch vornehmen.
  Siehe die Beschreibung bei der Image Vorbereitung.

Erweiterungen in der Plugin API(Server)
- Plugins (auf dem Raspberry Pi) können jetzt Scripte haben, um Systemkonfigurationen zu erzeugen basierend auf Einträgen in der avnav.conf.
- Neue API Funktionen sendRemoteCommand, registerSettingsFile, registerCommand

Einbettung einiger zusätzlicher Kernel Treiber (Raspberry)
- ein neues Paket wird auf AvNav images installiert, das Treiber für RTL8188EU und RTL8192EU WLAN Chipsätze enthält.

Kleinere Verbesserungen und Fehlerkorrekturen
Alle:
- #249: Nutzung einer Event-Übversetzungsliste um Event-Name für User Widgets korrekt durchzureichen.
- #251: Erlaube das Deaktivieren von Tastenzuordnungen in Nutzer- key.json Datei
- #252: Korrekte Behandlung von Tastenzuordnungen zu Buttons - auch wenn diese momentan nicht sichtbar (oder in der 2. Spalte) sind.
- #261: Verhindere einen Fehler bei inkonsistenten Daten im store (value und dict), Limitiere einzelne logs auf ca. 110MB
- Vermeide Error-Fluten im Log bei Bluetooth Problemen
- Tastenzuordnung z zum Umschalten (Toggle) des Dimm Modus
- Zusätzlicher Button "reload page" auf der Einstellungsseite
- Einige neue URL Parameter

- Halte die Scrollposition auf der AIS und WPA Seite (Lange Listen von AIS-Zielen oder WLAN Netzwerken)

Raspberry:

- #270: Kein Stop von wpa_supplicant (Client WLAN) beim Restart von AvNav (z.B. vom Updater)
- #266: Vermeide eine Blockierung beim Setzen der Systemzeit
  verbesserte Behandlung für das initiale Setzen der Systemzeit ohne GPS, korrekte Umschaltung zwischen GPS Zeit und NTP Zeit
- Kein Aufruf des startup-check (Einlesen der avnav.conf) während der Installation von avnav-raspi (Verhindert einen reboot während des Updates)
- Zusätzliches Template für die avnav_server.xml in /etc
- Änderung aller WLAN access point Konfigurationen auf "manual" um Konflikte mit dem avahi-autoipd zu vermeiden (manchmal kein Korrektes Aufsetzen des Access Points)
- Setze restart-ms für alle CAN Interfaces (siehe Forum) um die Robustheit des NMEA2000 Adapters zu verbessern
- Korrigierte country codes für die Image Vorbereitung
- Entfernen des nicht mehr vorhandenen spi-bcm2835-overlay
- Update der Installationsbeschreibung für Pakete (bullseye)
- Neustart von wlan-av1 bei Änderung der Systemzeit (wpa_supplicant funktioniert potentiell nicht mehr nach Zeit-Umstellungen)
- Korrektur für das Wpa Firewall Kommand (external access für WLAN Netzwerke) , besseres Logging in diesem Teil der Software
- uart_control Script jetzt in /usr/lib/avnav/raspberry
- korrekte Konfiguration für CANboat (n2kd)

Android:

- einige erste Implementierungen für ein plugin-Handling

20220819 link

Split Screen

- Wechsel zu und vom Split Screen per button
- Partiell separate Settings für jeden Tab, Details siehe Beschreibung

Verschiedene Routing Modes: RhumbLine, GreatCircle

- default: GreatCircle
- umschaltbar im RoutingHandler
- korrekte Anzeige von great circle Kursen
- Beschreibung

Verbesserungen

- nicht vorhandene Daten zu Wegepunkt u.ä. werden auf undefined gesetzt (Anzeige: ---)
- #224: AIS Verbesserungen length, beam, draught
- #213: JS API zum Zeichnen auf die Karte (SailSteer Widget)
- Überarbeitung der Bezeichnungen an den Wind Widgets, Wahlmöglichkeit der Werte (true, apparent,...)
- #237, #238: Nutzung der ersten kml Datei in kmz Dateien, wenn keine doc.kml vorhanden ist
- Bei Overlay Änderungen wird nur das Overlay neu gezeichnet, nicht mehr die gesamte Karte
- Scale Parameter jetzt auch für kml und geojson Overlays
- Info Dialog bei geänderter Server-Version
- JS Code (inklusive Widgets) für disabled plugins wird nicht mehr geladen
- Default Icon für GPX overlays, Farbe und Größe für Punkte setzbar
- Zeige Zeit, Kurs, Geschwindigkeit beim Klick auf einen Track (Feature Info)
- Restart Leg Button (Restart der XTE Berechnung)
- Das JS LatLon Modul wird jetzt am API exportiert
- #243, #217: Zeige remote control Buttons nur, wenn das enabled ist
- #170: Verschiedene Modi der Wegepunkt Weiterschaltung: early,90,late

Fehlerkorrekturen

- #232: korrekte Umrechnung für AIS SOG in m/s auf Android
- Umwandlung von Tracks in Routen auf der Download-Seite funktioniert wieder
- #222: Neu-Laden der angezeigten Liste auf der Download Seite nach Löschen von Elementen
- #228: $HOST nicht mehr beim Editieren einer User App ersetzen
- #228: Die Einstellung "new window" wurde bisher nicht über einen AvNav Restart hinaus gespeichert
- #225: Kartenkonvertierung auf Windows geht wieder (gdal2 auf Windows)
- #223: Korrektes Zusammenführen der storeKeys Einträge aus einer Widget Definition und aus den editierbaren Parametern
- #221: translateFunction funkltioniert jetzt auch für einfache user widgets
- #219: Vermeidung von fortlaufenden Log Nachrichten der SignalK Time Verbindung
- #220: Vermeidung einer Endlosschleife im NMEA Dekoder bei fehlerhaften XDR Sätzen
- Ersatz des Paketes gir1.2-appindicator3-0.1 durch gir1.2-ayatanaappindicator3-0.1 für die Touch Screen Einrichtung
- Korrekte Auslieferung der default user.js
- Android: Track wurde beim Beenden der App geleert
- Richtige Berechnung der Annäherung and einen Wegepunkt in der App
- #231: Track GPX OpenCPN Kompatibilität

- #244: falsche XTE Richtung

20220421 link

Starke Erweiterung der SignalK Integration

- AvNav kann jetzt direkt seine Navigationsdaten (inklusive AIS) von SignalK empfangen
- Wegepunkt-Daten können an SignalK geschickt werden
- Alarme können zu SignalK geschickt und von dort empfangen werden
- Das SignalK Handling wurde von einem Plugin in den "core" verlagert
- Für Details siehe die Beschreibung

Images mit Touch Support

- Die AvNav Images haben jetzt auch Support für einen lokalen Monitor (Schwerpunkt: Touch)
- Das ersetzt die nicht mehr gepflegten AvNav-Touch images.
- Details in der Installationsbeschreibung

Setzen der Systemzeit ohne GPS

- die AvNav Images haben jetzt ein fallback-Handling zum Setzen der Systemzeit
- wenn für eine gewisse Zeit kein GPS Signal empfangen wird, wird versucht über NTP die Zeit zu ermitteln
- das sollte Probleme beim Internet Zugriff vermeiden (SignalK, mapproxy,...)

Weitere neue Funkionen

- Trennung von true und apparent Wind in den internen Daten - die Anzeige der Wind-Widgets kann jetzt fest einem von beiden zugeordnet werden
- Beim Parsen von RMC wird jetzt korrekt geprüft, ob Kurs und Geschwindigkeit vorhanden sind
- #169: Anzeige für Map Scale (in den Settings unter Layer)
- #87: Alarm und rote Titelzeite, wenn die Verbindung zum Server abbricht
- #206: Setzen von Ankerwache mit Keyboard Kommandos
- #200: Verschiedene Boot-Symbole für COG und HDM/HDT sowie Stillstand, bessere Berechnung des Kurs-Mittelwertes
- #202: Parsen von ZDA
- #188: Umschalten/Ausschalten des Remote Channels direkt auf Navigations- und Dashboard Seiten
- Pitch und Roll für BME280, skPitch,skRoll Widgets können auch Input in deg verarbeiten
- Kleinere interne Korrekturen

Fehlerkorrekturen

- korrekte Mittelwertbildung für Kurs-Werte
- besseres Fehlerhandling bei avahi
- restart button manchmal nicht sichtbar auf der Status-Seite
- android: Korrektes Arbeiten der overlay-change-detection
- #212: Speed Gauge: Setzen von maxValue < 10 führt zum crash
- Karten-Konvertierung auch unter bullseye (GDAL3)

Migrationshinweis

- falls nicht besondere Alarm-Kommandos definiert wurden, sollten Einträge für den AVNAlarmHandler aus der avnav_server.xml entfernt werden.
  Dann können die Alarm-Sounds über die Settings-Seite eingestellt werden und es können eigene Alarm-Sounds genutzt werden.

20220306 link

- Fehlerkorrektur #196: Android AvNav empfängt keine Positionsdaten wenn es im Hintergrund ist
- Fehlerkorrektur #195: Beim Laden von Settings mit einem anderen Layout als system.default beim initialen Dialog wird das Layout nicht geladen
- Kleinere interne Korrekturen

20220227 link

Nur Windows !

- Fehler in 20220225 korrigiert, der den Server Start verhindert

20220225 link

Speichern und Laden von Einstellungen

- Die Einstellungen auf einem Gerät können auf dem Server gespeichert und auf diesem oder anderen Geräten wieder geladen werden.
- Das plugin interface wurde erweitert um settings in einem Plugin mitzubringen und zu registrieren.
- Beim erstmaligen Start von AvNav auf einem Gerät wird eine Auswahl der gespeicherten Einstellungsdateien angezeigt, falls es mehr ale eine gibt. Der Nutzer kann entscheiden, welche er nutzen möchte. Später können andere Dateien über die Einstellungsseite geladen werden.
- Auf der Einstellungsseite werden geänderte Werte durch unterschiedliche Schriftstile dargestellt.

- Die Möglichkeit Einstellungen innerhalb einer Layoutdatei zu speichern, wurde entfernt (man kann aber umgekehrt in einer Settings-Datei auch den Namen des zu nutzenden Layouts angeben)

AIS Verbesserungen

- #185: zeige sowohl AIS Ziele im eingestellen Umkreis um die Bootsposition als auch um den Kartenmittelpunkt.
- #187: In den Einstellungen kann gewählt werden, welche AIS Ziele angezeigt werden sollen (class A, class B,other).
- Optimierung der Anzeige-Performance in der AIS Liste
- Die pro Ziel in der AIS Liste angezeigten Informationen können reduziert werden
- Class B Ziele können in der Karte kleiner angezeigt werden
- Das momentan "verfolgte" AIS Ziel wird in einer separaten Farbe dargestellt.
- Das Verfolgen beginnt durch Klick auf den Center button  auf der AIS Seite.
- #149: Die Werte, die bei einem AIS Ziel angezeigt werden, können ausgewählt werden (bis zu 3)
- #145: AIS Ziele, die in Ruhe sind, können verborgen werden.

Andere neue Funktionen:

- #135: Entfernungen können durch das Setzen einer Markierung auf der Kartenmitte (button ) gemessen werden. Die Entfernung und der Kurs zur Kartenmitte werden im "Center Display Widget" und neben der Kartenmitte angezeigt.
- Der Dialog für das Bearbeiten einer Route wurde klarer gestaltet
- #141: Mehr Möglichkeiten für das Aktivieren der Ankerwache. Man kann die aktuelle Position oder die Kartenmitte als Ankerposition nutzen sowie einen Offset angeben. Siehe die Beschreibung.
- Wenn eine neue TCP Verbindung zu AvNav erfolgt, werden NMEA Daten nur ab dem aktuellen Zeitpunkt gesendet (vorher wurden bis zu 10 ältere Sätze gesendet).
- #146: Die Karten auf der Hauptseite sind alphabetisch sortiert. Ausserdem wurde die Dialogabfrage "center map now" entfernt.
- #148: AvNav erkennt automatisch, wenn sich Overlays ändern und zeichnet die Karte neu. Das gilt auch für Routen.
- #151: UserApps können jetzt in einem eigenen Browser-Fenster/einem eigenen Tab geöffnet werden. Das geht allerdings nur für externe URLs.
- #155: USB Monitoring, Bluetooth reader und NMEA logger können über die Server/Status Seite ausgeschaltet werden.
- #160: Dekodierung der Wasser-Temperatur von MTW und der Geschwindigkeit durchs Wasser von VHW. Zusätzliche widgets für STW und water temp
- #163: Sie Socket Reader können jetzt auch NMEA Daten senden
- AvNav kann jetzt auch gestartet werden, wenn auf dem System kein python-bluez verfügbar ist. (Ein enstprechender Status wird angezeigt)

- Unter Android wird jetzt auch APB gesendet
- Dekoder für VWR NMEA0183
- #164 Höhere Bauraten als 115200
- #180: Android: Es wird eine Warnung ausgegeben, wenn die Android Power Savee Einstellungen das GPS im Hintergrund abschalten, da dann keine Track-Aufzeichnung oder das Aussenden von NMEA Daten erfolgen kann.
- Es gibt 2 neue Widgets für SignalK Daten: skpitch/skroll (danke an Tom)
- Die Seiten-Navigation zwischen der AIS-Liste und der AIS Detail-Seite wurde verbessert (Hin- und Zurücknavigation möglich)

Änderungen für den Server Start:

- Das logfile enthält jetzt auch alle potentiellen Fehler beim Start
- Man kann beim Aufruf von der Kommandozeile jetzt den log level mitgebenlogging
- Im Normalfall wird mit dem Log Level "error" gestartet (Option -q)

Fehlerkorrekturem:

- #192, #193: korrektes Übergeben der Parameter für Formatierer
- #191: Entfernung der nihct genutzten "caption" für das Combined Widget
- #190: Hinzufügen der erforderlichen minValue/maxValue für radialGauge
- #189: Korrektur in canvas gauges
- Anpassung der Grössen für wind widget und depth widget
- #186: Verhinderung eines sporadischen crashes
- #184: Richtiges Handling für store keys (value) und Formatter Parameter im Layout Editor
- registrierte Layouts und settings werden entfern wenn ein Plugin disabled wird
- #182: Korrektes Handling für default Formatter Parameter
- Dokumentation korrigiert für nutzerdefinierte Formatter
- Verhinderung von mehrfacher USer App Registrierung durch plugins
- Erhalt der eingestellten Farben beim Editieren von Overlaysnumber
- #161: Rückfall handling bei nicht ladbaren Konfigurationen
- #178: Die "translate" ist jetzt für alle Widgets verfügbar
- Korrektur des erzeugten APB Satzes
- #157: Weiterschaltung zum nächsten Wegepunkt einer Route, wenn kein Client verbunden ist
- #154: Update für die Dokumentation der Installation mit Paketen
- #150: Dokumentation: Direkter Link zu mbtiles von OpenSeaMap
- #153: Overlay Dialog für Mobilgeräte korrigiert
- #152: Löschen von Routen auf Android funktioniert wieder
- Korrektur für den Konverter wenn utf-8 in KAP Dateien vorhanden ist
- Android: Verhinderung eines Absturzes, falls kein Dateimanager installiert ist
- #143: Korrektes Herunterfahren des Servers aus der App heraus

Interne Änderungen:

- Build Werkzeuge und Bibliotheken auf aktuelle Stände gebracht
- Explizite definition der nodejs Version (16) beim Bauen per gradle
- Optimierung des Builds für die Web-App

20210619 link(Android Version: 20210618)

Neue Funktionen

- zeige den Nachtmodus Button auch im Layout Editor
- #88: dekodiere HDG,HDT,HDM,VHW (teilweise) auf dem Server und unter Android
- HDT,HDM widgets
- ermögliche die Nutzung von HDT oder HDM für die Richtungsanzeige des Boot-Symbols (Einstellungen/navigation/boat direction)
- #138 ermögliche das Boot auf jeder beliebigen Position auf dem Bildschirm zu fixieren(Einstellungen/map/lock boat mode)
- #139 zeige den Nachtmodus Button auch auf der Navigationsseite
- Erlaube einen "schwebenden" Mode der Buttons und Anzeigen über der Karte (Einstellungen/map/float map behind buttons)
- Verberge die Buttons auf der Navigationsseite und auf den Dashboard Seiten nach einer einstellbaren Zeit (Einstellungen/buttons/auto hide buttons...)
- #132 zeige +/- 180 im Wind Widget (einstellbar im Layout Editor), kleinere Verbesserungen am Wind Graphics Widget
- setze den Layout-Namen als CSS Klasse an der Applikation um CSS Stile für verschiedene Layouts zu ermöglichen
- Fernsteuerung
  - Steuere ein Display von einem anderen oder vom Server
  - 5 Kanäle, jedes Display kann auf einem davon senden oder/und empfangen
  - UDP Port für den Empfang von Fernsteuerkommandos
  - Plugin für die Zusammenarbeit mit der obp-rc-remote Fernbedienung

- Interface Wichtungen für eine einfache Umschaltung zwischen ethernet und WLAN für den Internet-Zugang (avnav-raspi)

Fehlerkorrekturen:

- Einfrieren bei der Nutzung von Linear Gauges
- Falsche Button-Sortierung auf alten Browsern
- #140: Nachtmodus für XTE Widget, Wind Graphics, einige Gauges
- Umschaltung zwischen Access Point und WLAN client über die Config GUI funktionierte nicht
- Wenn gleichzeitig apparent und true Wind vorhanden sind, springt die Anzeige zwischen beiden. Jetzt wird "apparent" bevorzugt und nur wenn das nicht empfangen wird, wird

"true" genutzt. Es erfolgt eine Anzeige welcher Wert genutzt wird.
- Korrektur das Filter Handlings für Komma separierte Filter am Plugin API
- Korrektur des read filter Handling am SocketWriter

20210502 link

Neue Funktionen

- Erweiterung der Konfiguration für die Headless Images (Hostname, Tastatur, Umschaltung der WLAN Nutzung)
- Bessere Verbindung zu anderen WLAN Netzen (IP Adresse wird schneller ermittelt)

Fehlerkorrekturen:

- Korrekte Anzeige der Button Icons im Lite Mode vom Chrome auf Android (#128)
- einige MCS Module für one wire wurden nicht korrekt geladen (nur für headless images)
- Die Auflösung für NMEA0183 services erfolgte nur einmal (jetzt korrekt permanent wiederholt im Fehlerfall)

20210424 link

Focus: Android App

- Die Android App hat jetzt einen kompletten NMEA0183 Multiplexer - so wie die Server Variante
- Damit kann man:
  - verschiedene TCP/UDP NMEA0183 Datenquellen anschliessen
  - NMEA zu TCP/UDP Zielen senden
  - Mehrere USB-Seriell-Wandler nutzen um NMEA Daten zu senden und zu empfangen
  - NMEA Daten per Bluetooth senden und empfangen
  - Die Daten des eigenen GPS aussenden
  - Den Multiplexer im Hintergrund laufen lassen und die Daten für anderen Navigations-Apps bereitstellen
  - Mdns (Bonjour/Ahavi) nutzen um leicht eine Verbindung zu NMEA Datenquellen herzustellen - auch in sich verändernden Netzwerken
  - Eigene NMEA Ausgänge per MDNS für andere Apps oder Systeme bekannt machen
  - Die NMEA Daten einkommend und ausgehend mit Filter-Ausdrücken und Blacklisten einschränken
  - Ein NMEA Log in eine Datei schreiben.
  - Den Status aller Verbindungen auf der Status/Server Seite anzeigen

- Android integriert jetzt den Webserver in den "normalen" Modus (wurde vorher "external browser mode" genannt)
  - Der Zugriff kann vom gleichen Gerät oder von anderen Geräten per Browser erfolgen
  - Der default Port für den Webserver ist jetzt 8080

- Erzeugung von RMB Datensätzen, wenn ein Routing aktiv ist (kann ausgesendet werden, um z.B. einen Autopiloten zu steuern)
- Die Konfiguration der Android App ist jetzt weitgehend identisch zur Server Version und erfolgt auf der Status/Server Seite
  - HINWEIS: einige Konfigurationen werden von einer älteren Version übernommen - jedoch nicht alle

Für die Raspberry Version (Headless Image):

- Unterstützung von USB Tethering von einem Android Gerät. Damit kann man den Raspberry über das angeschlossene Gerät ins Internet bringen - oder auch die Verbindung zum AvNav Server über USB herstellen (einfach, wenn man unter Android den BonjourBrowser nutzt)

Für alle Server Versionen (Headless Image, OpenPlotter,...):

- Bekanntmachen von TCP NMEA Ausgängen (AVNSocketWriter) per Avahi (mdns/Bonjour) - andere AvNav server oder die Android App können sich so einfach verbinden.
- AVNNmea0183ServiceReader zur Verbindung mit einem Avahi (mdns/Bonjour) NMEA Service (wie z.B. SignalK)
- Erweiterung im plugin API - die initFunction erhält nun auch die WidgetParameter beim Aufruf

Fehler Korrekturen:

- #123: bluetooth Verbindungsabbrüche
- Anpassungen an neuere gdal Versionen für den Importer
- Files immer mit utf-8 encoding zum Lesen oder Schreiben öffnen (Chart Konverter)
- CSS Anpassung für bessere Kompatibilität mit alten Browsern
- Mehr Robustheit gegen NMEA Nachrichten mit leeren Elementen
- Manchmal wird im Layout Editor auch ein anderes Widget mit geändert, wenn man ein kopiertes Widget ändert

20210323 link

Fehlerkorrekturen für 20210322

- Windows: AvNav startet nicht nach update

- Windows: SocketWriter lässt sich nicht umkonfigurieren
- #119: Crash in der App bei linear Gauges
- #120: Crash in der App wenn ein Formatter nicht (mehr) existiert
- Defaultwerte für Widgets werden im Layout-Editor nicht richtig gesetzt (z.B. Farben bei Gauges)
- Crash in der App bei Date/Time formattern wenn keine oder falsche Daten vorhanden sind

neuer Windows Installer

20210322 link

Die größte Änderung ist die Möglichkeit, die Konfiguration des Servers direkt in AvNav zu ändern (z.B. das Hinzufügen neuer Interfaces). Das erfordert auch ein Update für alle plugins auf ihre neuesten Versionen.

AvNav Core:

- Man kann jetzt die Konfiguration direkt in AvNav ändern (Server/Status Seite), die Änderungen werden ohne Restart sofort wirksam
- Wenn man Karten mit leeren zoom layern nutzt (viele mbtiles Dateien) ist AvNav jetzt in der Lage geringer aufgelöste Kacheln vergrößert anzuzeigen. Das vermeidet leere Bereiche auf der Karte in bestimmten Zoomstufen.
- Man kann die Kartendarstellung jetzt skalieren(Settings/Map). Das kann die Sichtbarkeit auf hochauflösenden Displays verbessern.
- Es gibt eine experimentelle Vorschau auf einen split creen Modus, erreichbar mit http://address:port/viewer/viewer_split.html, siehe demo
- Die Anpassung der Symbolgrössen an hochauflösende Displays sollte jetzt korrekt sein.
- Direkte Registrierung von AvNav bei MDNS (Bonjour), man kann den Namen anpassen, der registriert wird.
- Erweiterungen im plugin API

Raspberry Pi (Images)

- Unterstützung für den MCS von GeDad, siehe Image Beschreibung
- Anpassung der Images vor der Benutzung mit einer Konfigurationsdatei und einer Web-Oberfläche für die Erzeugung
- Sowohl AvNav als auch SignalK sind jetzt per MDNS/Bonjour sichtbar
- Bessere WLAN Stabilität durch Abschaltung des power save managements
- Automatisches Aufsetzen einer IP Adresse am Ethernet Interface wenn keine Verbidnung zu einem DHCP Server besteht
- Weiterleitung von Port 80 zu AvNav (8080), erspart die Eingabe von :8080 in den URLs

Kleinere Fehlerkorrekturen (engl.)

- handle utf-8 in download names correctly
- keep status on status page until we get server error, correctly show and hide server error
- increase network timeout
- work around strange height bug on ios 12,13 safari
- always use shutdown when closing sockets, re-enable timeout for socket reader
- #113: merge pr, correctly handle null values in SK
- #117: prevent browser 2 finger zoom
- restart avahi when changing the system time
- avoid hiding second buttons on status page
- decrease default font sizes
- create an error gemf for failed conversions, add importer log display on download page, timeout importer status items
- allow opensuse install
- show plugin info in status if startup fails
- restart server
- log display
- server download log
- cleanup name handling and thread id for logging
- better timeout handling for serial (avoid reopen after 1s)
- rename AVNGpsdFeeder to AVNFeeder, completely remove gpsd related code
- #106: avoid writing errors to the log if empty positions lead to decode errors
- merge #105
- adapt to new handling of com port names on windows
- correct windows download links

20210115 link

- #73: Umstellung des Servers auf python3 (das bisherige python2.7 wird nicht mehr gepflegt)
- Plugin API Java Script Erweiterung für featureFormatter
- Dekodierung von XDR Datensätzen
- Formatter für Druck und Temperatur
- Erweiterung des Java Script API - Registrierung von Formattern
- besseres Handling für formatterParameters im Layout Editor
- NMEA filter für AVNSocketReader,AVNBluetoothReader und AVNUdpReader
- das avnav history plugin ist in den Paketquellen enthalten und kann mit

  ```
  sudo apt-get install avnav-history-plugin
  ```

  installiert werden.
- Fehlerkorrekturen:
  - #97: NMEA Prüfsummen mit kleinen Buchstaben
  - Korrektes Handling des Dimmfaktors für die Karte im Nachtmodus

- **Installationshinweise**:
  Bei der Installation mit Paketen (apt) müssen neue Abhängigkeiten installiert werden.

  ```
  sudo apt-get update
  sudo apt-get upgrade
  ```

  Bei den headless Paketen muss vorher die Zeit gesetzt werden.

  ```
  sudo date -s "2021/01/15 15:00"
  ```

  Es ist wichtig, dass auch vom ocharts-plugin die aktuelle Version 20210115 installiert wird.
  Bei der Windows Installation sollte ein neuer Installer genutzt werden.
  Falls mit dem vorhandenen Installer ein Update ausgeführt wird, muss es 2 mal nacheinander durchgeführt werden, da beim ersten Mal die neue Python Version nicht installiert wird.

20201227 link

- Plugin API Erweiterung für USB Geräte
- Fehlerkorrekturen
  - **RMB funktionierte nicht in den letzten Versionen, repariert**
  - kleinere Fehler im widget handling für plugins
  - kleinere Fehler bei den angezeigten Buttons
  - einige verbesserte Logs am Server

20201226 link

- Fullscreen Button
  Falls vom verwendeten Browser unterstützt, gibt es auf der Hauptseite, Navigationsseite und auf den Dashboard Seiten
  einen ⛶ Button, um Fullscreen ein- und auszuschalten
- Separate Einstellungs-Kategorie für Buttons
- Fehlerkorrekturen
  - OSM online Karten
  - bessere Kompatibilität für ältere Browser
  - Android Dim Handling

20201219 link

- Karten Overlays
  - Man kann gpx,kml,kmz und geojson Dateien über die Karten legen
  - Mehrere Karten übereinander legen
  - Anzeige von vorhandenen Tracks und Routen
  - Nutzerdefinierte Symbole und links mit HTML Seiten

- Anzeige von Informationen zu einem Punkt beim Klick

- Anzeige von Objekt-Informationen (Feature Query) bei o-charts (erfordert neue avnav-ocharts Version)
- Erweiterungen beim Track Handling
  - Anzeige von Informationen (Länge, Zeit, Geschwindigkeit)
  - #67: Anzeige von Tracks auf der Karte (als Overlay)
  - Konvertierung von Tracks zu Routen
  - Import von gpx Tracks

- Verbesserungen im Routen Handling
  - Neuer Dialog für Umbenennen, Leeren, Kopieren, Löschen
  - Verbinden von Routen
  - #9 Nutze einen Wegepunkt aus einer Gpx Datei zum Hinzufügen zu Routen
  - Routen können jetzt auch "rückwärts" gebaut werden ("insert before")

- Verbessertes Fehlerhandling
  - Zeige einen Fehlerdialog mit der Möglichkeit den Fehler zu speichern
  - Anzeige von Fehlern in der user.js

- Erweiterungen in den Settings
  - "increase fonts on hires" - Vergrößerung verschiedener Anzeige-Elemente bei hochauflösenden Displays
  - "Overlay Info on Click" - Anzeige von Overlay Informationen wenn man auf die Karte klickt (default: ein)
    das steuert auch die Anzeige von Objekt-Informationen bei o-charts
  - "Always Info on Chart Click" - Anzeige einer Information in Bereichen ohne Overlays (default: aus)

- Verbesserte Dokumentation
  - Tastaturunterstützung beschrieben
  - Index

- #79: Aufrufparameter noCloseDialog um die Abfrage beim Verlassen der Seite zu verhindern
- Fehlerkorrekturen:
  - Zurückspringen zur Startseite nach Klick auf ein AIS symbol
  - "cancelTop" funktioniert jetzt auch unter firefox
  - Korrektes Senden von MTA NMEA Sätzen ($ am Anfang fehlte)

20201202 link

Fehlerkorrekturversion für 20201105

- Bug: Die SignalK Verbindung mit WebSockets (neu in 20201105) funktioniert nicht komplett. Nicht geänderte Daten verschwinden nach 30s. Man konnte auch die Nutzung von WebSockets nicht abschalten
- Bug: In der Dokumentation war für signalK useWebsockets falsch angegeben

20201105 link

- Feature: Button für Power Save (Android und BonjourBrowser) #69
- Feature: Geschwindigkeitsabhängige Kurs-Vektoren für eigenes Boot und AIS #59, siehe Navigationsseite (dazu einige neue Einstellungen)
- Feature: Einstellungen für die AIS Symbolgröße und einen Rand #58
- Feature: Eigene Symbole für das Boot und AIS Ziele (unterschiedlich pro Typ) #53, siehe Beschreibung
- Feature: Overflow Button wenn mehr als 8 Buttons nicht passen, Möglichkeit für 2. Button Spalte #68, siehe Navigationsseite
- Feature: Einstellmöglichkeit um den Zurück-Button immer ganz oben zu haben
- Feature: Package Abhängigkeit zu gpsd entfernt
- Feature: Möglichkeit für Plugin Widgets mit dem Server zu kommunizieren (event handler) #75, siehe Plugin Beschreibung und User Java Script
- Feature: Nutzung von Karten, die unter SignalK installiert sind #76, siehe SignalK Karten
- Bug: N2K (canboat) arbeitet weiter, auch wenn lange keine Daten vorhanden sind
- Bug: Unerwarteter Wechsel von einer Seite zur anderen (doppeltes Klick Handling in Listen)
- Bug: Gehe im Routen Editor zum vorletzten Punkt, wenn der letzte gelöscht wird
- Bug: Robusteres Dekodieren von teilweise leeren DPT und DBT Sätzen #60
- Bug: korrekte Behandlung von Filtern in avnav_server.xml die nur eine Blacklist enthalten
- Bug: Robustere Behandlung von verkürzten Class 5 AIS Nachrichten (keine Namen, Callsign,...)
- Bug: Neuladen einer Karte im Browser, wenn sie auf dem Server modifiziert wurde (mbtiles, ocharts)
- Bug: (Android) akzeptiere unbekannte NMEA talker ids
- Bug: Center to Ais target funktioniert nicht #74
- Bug: Richtige Anzeige der anchor watch distance in m #62
- Bug: Darstellung kaputt für lange URL in User App Dialog #66
- Bug: Korrektur für das Handling des Schemas bei mbtiles (xyz,tms) #63, siehe Beschreibung Karten
- **Hinweis**: Nach Installation per Hand mit dpkg (erzeugt einen Fehler) müssen ggf mit

```
sudo apt-get install -f
```

die neuen Abhängigkeiten nachinstalliert werden.

20200609 link

- korrektes Handling von Karten mit Leerzeichen im Namen
- Android Korrekturen (Version 20200605) - korrekter external Browser mode in älteren Android Versionen

20200515 link

- GPS Status Anzeige auf der Hauptseite (Danke free-x)
- Wind Grafik auf der Nav Seite wird nicht mehr kleiner
- Tiefenanzeige funktioniert wieder
- CSS Klasse für Widgets im Layout Editor funktioniert
- NMEA logger funktioniert wieder richtig
- Kein mehrfaches Laden von Plugins mehr
- eniro aus den Demo Sourcen entfernt

20200401 link

- Verbindung mit freien externen WLANs funktioniert wieder

20200325 link

- Layout Editor zur Anpassung des Layouts an eigene Vorstellungen
- Karten im mbtiles Format
- Einbindung von grafischen Anzeigen (mittels canvas-gauges)
- Verwalten von Nutzerdateien und "user apps" um eigene Web Seiten einzubinden
- Erweiterung und Anpassung von AvNav mit java script und css
- Überarbeitete Dokumentation (auch in Englisch)

20200204 link

- Support für NME2000 (via canboat) und SignalK - siehe Beschreibung
- Anpassungen am Layout

20200126 link

- Fehlerkorrektur Android ständiger Notification-Ton
- Fehlerkorrektur Routen-Länge auf der Detail-Seite.
- MOB Handling. Bei Klick auf die MOB Taste wird ein aktuelles Routing abgebrochen, die aktuelle Position wird als Wegepunkt "MOB" gesetzt und es wird ein Alarm ausgelöst. Die

MOB Taste bekommt einen roten Rand. Beendet wird das MOB über die MOB Taste oder über das Stoppen der Navigation.
Wenn in avnav_config.xml ein AlarmHandler konfiguriert wurde, benötigt diese Funktion einen neuen Eintrag dort.
Wenn nur die Standard-Sounds für die Alarme verwendet werden sollen, kann der gesamte AVNAlarmHandler Eintrag entfernt werden.

```
<AVNAlarmHandler>
       <Alarm name="waypoint" command="sound"
parameter="$BASEDIR/../sounds/waypointAlarm.mp3" repeat="1"/>
       <Alarm name="ais" command="sound"
parameter="$BASEDIR/../sounds/aisAlarm.mp3" repeat="1"/>
       <Alarm name="anchor" command="sound"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="20000"/>
       <Alarm name="gps" command="sound"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="20000"/>
       <Alarm name="mob" command="sound"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="2"/>
</AVNAlarmHandler>
```

20191224 link

- Größerer interner Umbau
- Keyboard Support. Die aktuellen Bindings findet man auf GitHub. Eine Anpassung ist im Layout über einen parameter "keys" möglich. Siehe Beispiel auf GitHub. Eine detaillierte Beschreibung folgt. Auch unter Android.
- Einführung von Layouts: Man kann über eine Json Datei die Widgets auf der Navigationsseite und auf den Dashboard Seiten anpassen. Dazu unter Settings->Layout eine andere Layout-Datei wählen. Zum ändern die vorhandene system.default Datei herunterladen (Download Seite), Anpassen, ggf. umbenennen und wieder hochladen. Eine detaillierte Beschreibung folgt.
  Das ist auch unter Android möglich.
- Anpassung des Aussehens per css. Über eine Datei /home/pi/avnav/data/user/viewer/user.css können eigene css-Regeln eingebracht werden. Das ist auch unter Android möglich.
- Anpassen des Verhaltens über eigene Widgets. Mit einer Datei /home/pi/avnav/user/viewer/user.js können eigene Widgets definiert und später im Layout genutzt werden. Ein Beispiel findet sich auf GitHub. Das ist auch unter Android möglich.
- Plugin Konzept. Mit Python code und Js-Code kann avnav jetzt erweitert werden. Mit den Python Anteilen können z.B. weitere NMEA Sätze dekodiert werden, NMEA Daten gelesen und geschrieben werden oder auch Daten in den internen Speicher (und damit zur

Anzeige) geschickt werden. Ein Beispiel ist auf GitHub vorhanden. Eine detaillierte Beschreibung folgt. Nicht unter Android.

20190429 link

- Im neuesten Raspbian gibt es Fehler, die verhindern das bei bestimmten REALTEK basierten WLAN-Adaptern ein Neu-Verbinden mit einem Netz funktioniert. Es betrifft Adapter die das Kernel Module 8192cu nutzen. Man sieht im log dann Fehler dieser Art:

  ```
  wlan-av1: Association request to the driver failed
  ```

  Als Workaround habe ich eine Überwachung eingebaut, die wenn dieser Fehler ein paar mal auftritt, das Kernel module neu lädt. Damit arbeitet der WLAN Zugang zu externen Netzen wieder. Es kommt nur beim Neu-Verbinden immer zu einer kurzzeitigen Nicht-Verfügbarkeit des WLANs - sieht man im Status. Schwierig wird es nur dann, wenn man mehrere WLAN-Module mit diesem Chipsatz verwendet und eines davon auch als access point. Hier würde das WLAN auch kurzzeitig unterbrochen - Mobile Geräte verbinden sich dann gerne mal mit einem anderen. Solange man den Access Point (default) über das interne WLAN abwickelt, hat man keine Probleme. Da es nur ein workaround ist, wird der nicht standarmäßig aktiviert - nach der Installation des Updates muss man noch den entsprechenden service aktivieren und starten:

  ```
  sudo systemctl enable avnav-check-wlan
  sudo systemctl start avnav-check-wlan
  ```

  Ich hoffe, das es irgendwann eine Korrektur gibt, die den workaround wieder überflüssig macht.
- Standardmässig ist der Zugang von aussen (Web-Oberfläche oder ssh) über das wlan-av1 (d.h. den WLAN Adapter, der eine Verbindung nach draussen macht) gesperrt, damit aus einem öffentlichen WLAN niemand auf den raspberry zugreifen kann. Für manche Szenarien wäre es aber wünschenswert, wenn man das kann. Ich habe z.B. einen mobilen Hotspot und möchte den raspi gerne mit diesem verbinden. Wenn dann bei mir Geräte direkt mit dem Hotspot verbunden sind (und nicht mit dem avnav WLAN) möchte ich trotzdem auf den raspi zugreifen können. Daher gibt es jetzt eine Möglichkeit, diesen Zugriff von aussen pro WLAN zu gestatten. Man sollte hier sorfältig sein, und das auf keinen Fall in einem öffentlichen WLAN (z.B. Hafen) zulassen!
  Um die Funktion zu ermöglichen, muss nach der Installation ein Eintrag in der avnav_server.xml vorgenommen werden:

  ```
  <AVNWpaHandler>
   firewallCommand="sudo -n $BASEDIR/../raspberry/iptables-ext.sh
  wlan-av1"
   wpaSocket="/var/run/wpa_supplicant/wlan-av1">
  </AVNWpaHandler>
  ```

Wenn dieser Eintrag vorhanden ist (danach neu starten), wird bei WLANs, die für diesen Zugriff freigeschaltet sind, "ext access" angezeigt, beim Verbinden kann das ausgewählt werden.
Im Status für das Interface wird ebenfalls angezeigt, ob der externe Zugriff erlaubt ist und ob das firewallCommand erfolgreich war. Wenn es nicht erfolgreich war, sollte der raspi neu gestartet werden, damit die firewall wieder korrekt funktioniert.

20190415 link

- Neues Image basierend auf 2019-04-08-raspbian-stretch-lite
- Support für Raspi3b+
- Alle Korrekturen bis 2019/04/15 (Udp Writer thx to BlackSea)
- ahavi config für avnav (mit Bonjour-Browser zu finden)
  - für IOS Bounjour Search
  - für Android Bonjour Browser

- Möglichkeit in der Start-Page andere URLs einzubinden
- Größerer interner Umbau - Plugin-Interface (Doku folgt...)
- Dieses Image ist das letzte direkt von mir bereitgestellte Image - neuere finden sich wie unter Installation beschrieben.

20180313 link

- Ausschalten aller Alarme mittels Taster an GPIO Pin. Dazu muss in der avnav_server.xml für den AlarmHandler ein Attribut stopAlarmPin angegeben werden. Beispiel:

  ```
  <AVNAlarmHandler stopAlarmPin="7">
  ```

  Die Pin Nummer ist dabei die Nummer am Pi connector - siehe howto - im Beispiel Pin 7 = GPIO4.
  Zum Ausschalten der Alarme muss dieser Pin mit Masse verbunden werden (einfacher Taster).

Korrekturen:

- Konverter für die Karten auf dem Pi funktioniert wieder mit dem neuesten Image (neue gdal Version)

20180306 link

Korrekturen:

- gps Alarm aktiv
- korrektes Icon für Android "add to homescreen"

20180218 link

kleinere Fehlerkorrekturen zu 20180215

- Status auf Hauptseite vertauscht für NMEA/AIS
- Versions-Info auf Hauptseite falsch
- kleinere Android Korrekturen

20180215 link

Anker Alarm
Achtung: Dazu muss die avnav_server.xml erweitert werden. (Wenn man neu aufsetzt, wird das automatisch erzeugt, dann muss man nichts tun).
Am Ende bitte einfügen:

```
<AVNCommandHandler >
    <Command name="shutdown" command="sudo shutdown -P"/>
    <Command name="sound" command="/bin/sh
$BASEDIR/../raspberry/sound.sh 90%" repeat="1"/>
</AVNCommandHandler>
<AVNAlarmHandler>
     <Alarm name="waypoint" command="sound"
parameter="$BASEDIR/../sounds/waypointAlarm.mp3" repeat="1"/>
     <Alarm name="anchor" command="sound"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="20000"/>
     <Alarm name="gps" command="sound"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="20000"/>
</AVNAlarmHandler>
```

Das Kommando, das den Alarm-Sound ausgibt, kann frei angepasst werden ("sound" command oder ein anderes) - man kann auch z.B. ein gpio Pin schalten.
Falls man ein anderes Kommando nutzen möchte (z.B. eigenes Shell-Script) muss man das als neues Kommando unter AVNCommandHandler eintragen, ihm einen Namen geben und dann bei den Alarmen diesen neuen Namen nutzen.
Man kann natürlich auch das oben angegebene "sound" Kommando durch etwas anderes ersetzen.
Der unter "Alarm" angegebene Parameter wird an das entsprechende Kommando weitergereicht.
In jedem Falls sollte eine Alarm-Ausgabe am pi direkt vorgesehen werden, es kann ja sein, das gerade kein Tablet an ist.
Wenn beim Alarm als Parameter eine sound-Datei angegeben ist, kann der sound (zusätzlich) auch auf dem Tablet erzeugt werden (dort in den settings prüfen, ob er eingeschaltet ist).

Fehlerkorrekturen/Verbesserungen:

- https://github.com/wellenvogel/avnav/issues/27
- https://github.com/wellenvogel/avnav/issues/30
- SenseHat Support https://github.com/wellenvogel/avnav/pull/31
- besseres Handling von mehreren WLAN interfaces
- support für BMP180,BME820 (Danke Oleg!)
- https://github.com/wellenvogel/avnav/issues/33
- erlaube Verbindungen zu offenen WLANs (kein Passwort)
- router output jetzt auch APB (nicht nur RMB)
- navipack Konverter (Danke Oleg!)
- Datenschutzerklärung
- NMEA checksum error https://github.com/wellenvogel/avnav/pull/37 (Danke Oleg!)
- workaround für IOS Geräte (keine Anzeige der AIS-Seite)
- Layout tuning

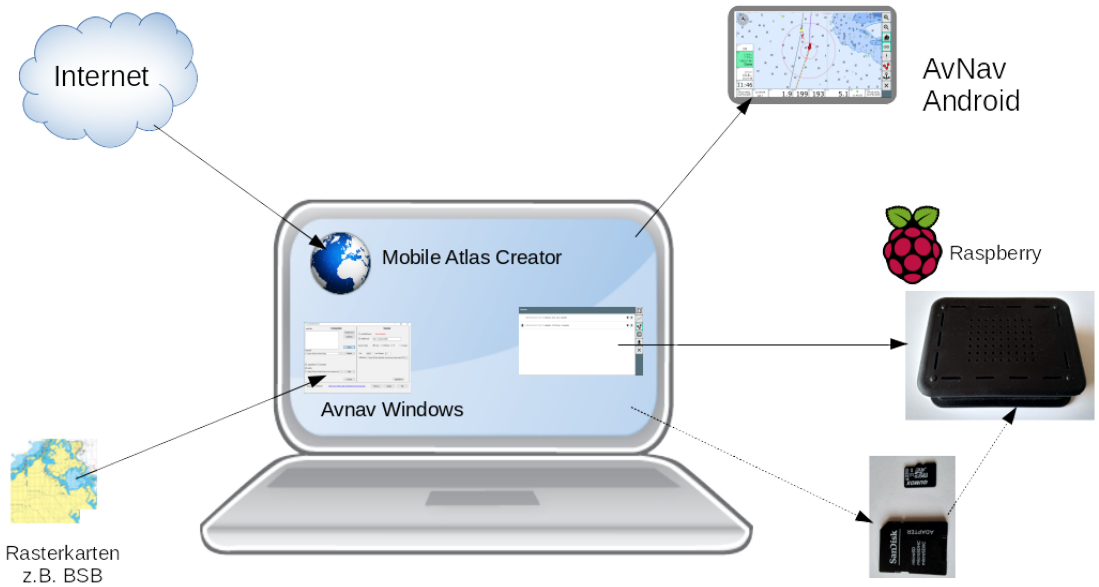
20170410 link

Fehlerkorrekturen:

- track display
- route handling wenn nicht verbunden
- erhalte wlan config files bei updates
- #29: Radius der Kreise
- modus ohne gpsd funktioniert wieder
- korrekte Ermittlung der Layer-Grenzen für gemf Karten
- Android Korrekturen

Avnav Karten und Overlays

Einführung



Damit Karten in der WebApp verwendet werden können, müssen sie in einem „Kachelformat“ vorliegen. Das ist das Format, das durch Dienste wie OpenStreetMaps oder GoogleMaps benutzt wird. Eine Kartenkachel ist 256x256 Pixel gross. Die Welt wird dabei auf eine ebene Fläche projiziert (das kann man sich wie einen Papierzylinder vorstellen, der senkrecht steht

und am Äquator um die Erde gewickelt wird). Jeder Punkt mit seinen Koordinaten (Länge/Breite) wird dann auf diesen Zylinder projiziert. Wie man das macht, welche Einheiten in der Projektion verwendet werden, und ob die Erde als Kugel oder Ellipsoid mit verschiedenen Parametern modelliert wird, beschreiben die verschiedenen Projektionen. Die WebApp benutzt die sogenannte Google-Mercator Projektion (die Erde wird dabei als Kugel betrachtet) - mit dem EPSG code 900913. Die Einheiten auf dem Papier sind dabei Meter (die man natürlich in die entsprechenden Koordinaten umrechnen kann). Karten in einem anderen Format (z.B. WGS84 – Erde als Ellipsoid, immer in Grad) müssen daher ggf. reprojiziert werden.

Die gesamte Projektionsfläche wird bei der Google Projektion in Kacheln unterteilt. Der Zoom Level gibt an, in wieviele Kacheln die Fläche unterteilt wird. Zoom Level 0 bedeutet: die gesamte Erde (von -85° bis +85° Breite – darüber ist die Projektion nicht definiert) auf einer Kachel von 256x256 Pixel. Mit jedem weiteren Zoom Level wird feiner unterteilt: Zoom Level 1: 2x2 Kacheln, 2: 4x4 Kacheln usw. Für uns reichen die interessanten Zoom Level von ca. 7 bis 18..19. Das bedeutet (Level 19) 2^19x2^19 Kacheln.

Zur Darstellung wird die Library openlayers verwendet. Diese lädt die entsprechenden Kartenkacheln je nach Zoom Level vom Raspberry und zeigt sie an. OpenStreetMaps verwendet typischerweise diese Library.

Man kann sich leicht vorstellen, dass bei hohen Zoom Levels schnell große Datenmengen zusammenkommen. Daher müssen wir für unsere Kartenkacheln ähnlich vorgehen, wie es auch bei den Papierkarten ist: für Übersichten ein kleinerer Zoom Level, Detailkarten größer und z.B. Hafenpläne dann mit Level 18 oder 19 (60cm/pixel bzw. 30cm/pixel). Um damit arbeiten zu können, werden die verschiedenen Detailgrade dann in Layern (Schichten) übereinandergelegt. Wenn es für ein Gebiet einen Layer mit besserem (grösserem) Zoom Level gibt, wird dieser angezeigt - wenn nicht, der mit der geringeren Auflösung (ggf. noch vergrössert). Um unsere Anzeigegeräte nicht zu überlasten, kann man typisch mit 3-5 Kartenlayern arbeiten (je nach Gerät...).

Damit besteht für die Kartenkonvertierung die Aufgabe darin, vorhandene Karten in solche Layer einzusortieren, sie ggf zu reprojizieren und dann die Kartenkacheln (sowie eine Beschreibungsdatei) zu erzeugen. Das erfordert meist eine ganze Menge Rechenleistung (und ggf. Zeit), daher ist das etwas, das auf einem PC oder Laptop laufen sollte (der Pi braucht potentiell sehr lange dafür). Das ist aber nicht schlimm – man muss das ja nur einmal machen. Belohnt wird man dafür mit einer sogenannten „blattschnittfreien“ Darstellung.

Für den Download gilt Vergleichbares - hier muss man geeignet die Zoom Level und Bereiche auswählen.

Das Verfahren dazu hängt von der Quelle ab. Im Moment habe ich Support für 3 „Wege“ der Kartenerstellung eingebaut:

- Konvertierung von Karten mittels GDAL (z.B. BSB) – hier kann alles gelesen werden, was GDAL kann

- Nutzung von online Karten (und deren Download) mit Hilfe von MapProxy - über das MapProxy Plugin
- Nutzung von downloads mit dem Mobile Atlas Creator.

Kartenformate

AvNav nutzt als primäres Kartenformat das GEMF Format - das ist ein kompaktes Format - anstelle von sehr vielen einzelnen Dateien sind diese in einem großen File zusammengefaßt. Mit diesem Format können auch spezielle Funktionen wie eine automatische Anpassung des Zoom Levels ermöglicht werden.
Ab der Version 20200325 kann AvNav auch direkt das mbtiles Format lesen. Bei diesem Format muss ggf. noch die richtige interne Anordnung der Kacheln gewählt werden - der Standard is "xyz", es gibt aber auch Dateien, die im "tms" Format vorliegen. Eine Umschaltung kann auf der Files/Download Seite erfolgen. MbTiles können z.B. direkt von OpenSeamap heruntergeladen werden.
Hinweis: Bis Version 202011xx hat AvNav einen Eintrag "scheme" in den Metadaten von mbtiles Files ausgewertet. Leider ist die Benutzung nicht wirklich klar geregelt und verschiedene Quellen nutzen das unterschiedlich (siehe auch Issue #63). AvNav hat auch intern xyz und tms genau invers verwendet. Mit der Version 202011xx ignoriert jetzt AvNav diesen Wert und nimmt tms an. Falls er in den Metadaten vorhanden ist, wird die Karte in rot angezeigt, über die Download-Seite kann dann gewählt werden, welche interne Kodierung genutzt werden soll.

Zusätzlich können (ab 20200325) auch Online Kartenquellen eingebunden werden, wenn sie ein entsprechendes URL Format unterstützen. Die Konfiguration erfolgt über XML Dateien. Ein Beispiel ist die mitgelieferte Quelle für OpenSeaMap.

Ab Version 20200515 können mit einem Plugin auch Karten von o-charts verwendet werden.

Ab Version 202011xx können (wenn die signalK Integration aktiv ist) auch Karten genutzt werden, die im signalK chart provider vorhanden sind.

Mit dem MapProxy Plugin können online Karten von verschiedenen Quellen eingebunden werden.

Installation von Karten

Nach der Installation ist in AvNav zunächst nur eine/einige Online-Demo-Karten vorhanden. Zur realen Nutzung müssen die Karten zunächst bei AvNav installiert werden.
Das kann durch direkte Kopie in das Karten-Verzeichnis (/home/pi/avnav/data/charts) bzw. in

das externe Kartenverzeichnis unter Android erfolgen. Alternativ können die Karten auch direkt in der WebApp hochgeladen werden (einige erst ab 20200325). Das Hochladen erfolgt auf der Files/Download Seite.
Für Android ist zu beachten, dass mbtiles Karten nur gelesen werden können, wenn sie hochgeladen wurden, nicht aus dem externen Karten Verzeichnis.
Für mbtiles bitte auch diesen Hinweis auf der Files/Download Seite beachten.

In den normalen Versionen (nicht Android) können ab 20200325 auch Karten, die erst noch konvertiert werden müssen direkt hochgeladen werden. AvNav enthält immer auch den unter Konvertierung beschriebenen Konverter. Der liest Dateien aus seinem Import Verzeichnis (/home/pi/avnav/data/import) bzw. aus Unterverzeichnissen. Wenn beim Hochladen eine Karte mit einem Dateinamen für den Konverter gewählt wird (z.B. .kap), dann wird gefragt, ob diese direkt dem Konverter übergeben werden soll. Der Arbeitszustand des Konverters kann auf der Status Seite beobachtet werden. Es sollte dabei beachtet werden, das die Konvertierung ein rechenintensiver Prozeß sein kann, der auf einem Raspberry Pi viele Stunden dauern kann. Das sollte daher ggf. auf einem Desktop Rechner erfolgen.

Wenn das MapProxy plugin installiert ist, werden dessen Karten in AvNav sofort sichtbar und müssen nicht getrennt installiert werden.

Overlays

Ab Version 20201219 kann AvNav über (und unter) den eigentlichen Karten noch weitere Informationen anzeigen bzw. Karten können kombiniert werden.

Für Details siehe Overlays.

Konvertierung von Karten

Die Konvertierung kann direkt auf dem Raspi erfolgen. Da hier die Resourcen jedoch ziemlich begrenzt sind, muß mit längeren Laufzeiten gerechnet werden (Beispiel: Auf einem Raspi 2 dauerte das Konvertieren eines älteren NV Satzes (Ostsee 4) ca. 1 Stunde).

Einfacher geht die Konvertierung auf einem Windows (oder Linux/Mac) Rechner. Für die nötigen Installationen siehe:

- Linux
- Windows

Der Hauptteil der Konvertierung wird durch tiler_tools durchgeführt (https://code.google.com/p/tilers-tools/) diese sind Bestandteil der Software und müssen nicht extra heruntergeladen werden..
Die Karten werden per Default im Verzeichnis <UserHome>/AvNavCharts/out erzeugt (also z.B. c:\Users\Andreas\AvNavCharts\out). Zumindest in der Windows GUI kann das Verzeichnis

gewählt werden. Standarmäßig ensteht für jeden Konverter-Lauf eine xxx.gemf Datei, xxx ist dabei der Dateiname der ersten Datei oder der Name des ersten gewählten Verzeichnisses.

Im AvNavCharts Verzeichnis gibt es auch noch ein work-Verzeichnis, dieses sollte nicht gelöscht werden, da dann bei einem Update (z.B. werden Berichtigungen eingepflegt) nur die geänderten Karten neu erzeugt werden müssen.

Die Konvertierung verläuft in 2 Schritten:

- Sortierung der Karten in Layer (und ggf. soweit nötig Konvertierung der karten).
- Erzeugung der Kacheln
- Erzeugung der gemf Datei

Wenn die Datei xxx.gemf erzeugt wurde, diese auf den raspberry installieren.
Man kann auch die Karte direkt auf dem Desktop Computer testen - nach Abschluss der Konvertierung wird der AvNav Server gestartet und man kann sich mit einem Browser verbinden.

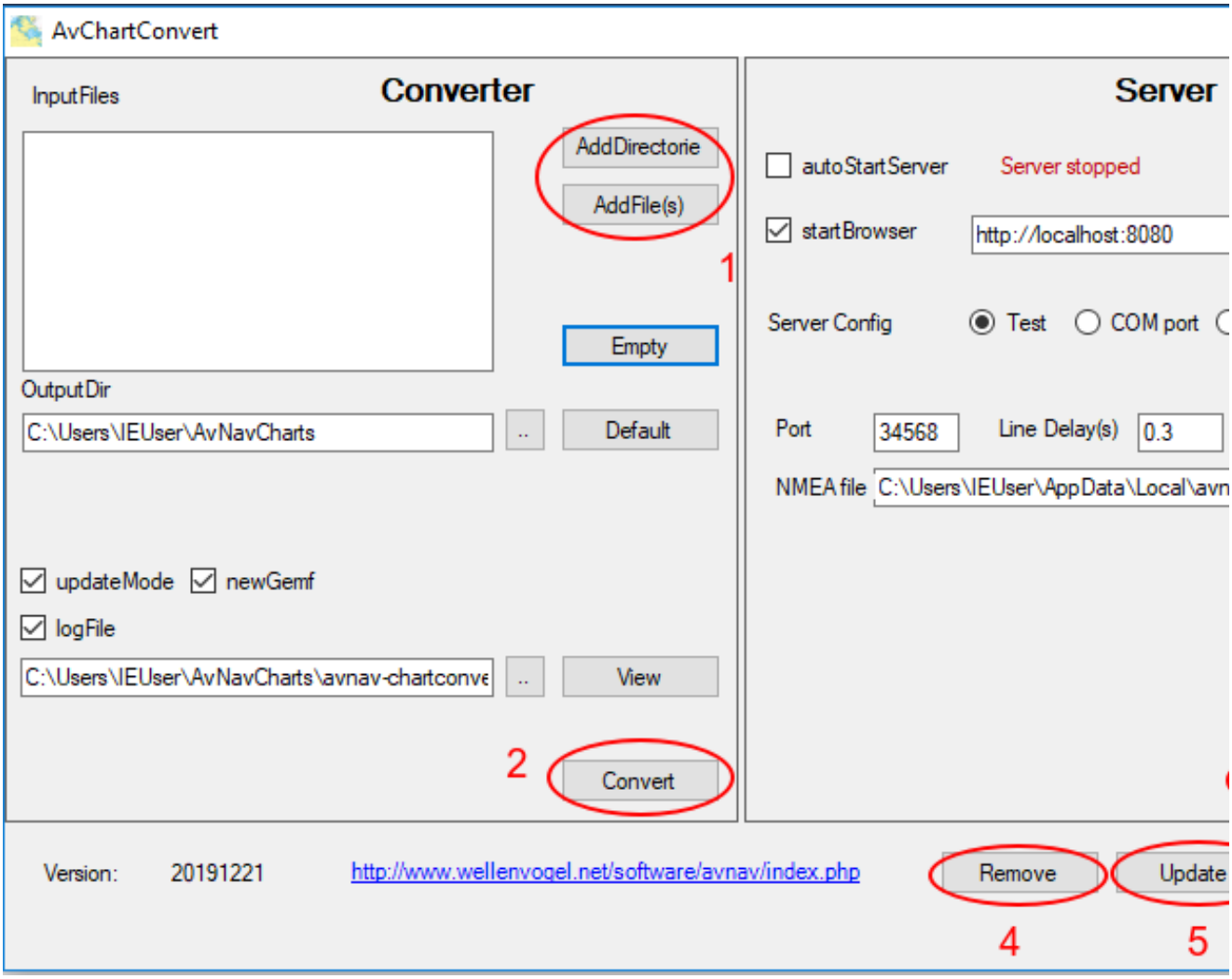
Windows

Der Net-Installer (AvNavNet...) für Windows installiert neben der eigentlichen Konverter-Software auch die nötigen Programme:

- python 2.7.10 (32 Bit)
- GDAL 1.11 für python 2.7 (32 Bit)
- Pillow 2.7.0
- pyserial 2.7

Der Installer wird unter ProgramFiles\AvNav installiert. Wenn AvNav deinstalliert wird, werden nur diese Anteile mit entfernt.
Die nachgeladenen Teile werden im Home-Verzeichnis des Nutzers installiert und müssen aus der UI deinstalliert werden.
Nach der Installation kann man mit AvChartConvert die Applikation starten.

Die Applikation hat nur einen Bildschirm:



Bei (1) können die zu konvertierenden Karten gewählt werden, bei (2) startet die Konvertierung. Es wird eine Log Datei geschrieben, die man mit "view" anschauen kann (das hätte ich auch gerne, wenn etwas nicht funktioniert).

Mit den Default-Einstellungen startet danach sofort der AvNav Server (also die Software, die sonst auf dem Raspberry läuft) und ein Browser wird geöffnet, der die App anzeigt.

Wenn die nötige Software noch nicht installiert wurde, öffnet sich ein Dialog, der die Installation anbietet und die nötigen Teile aus dem Netz nachlädt. Mit (5) kann das für ein Update wiederholt werden (die aktuell vorhandene Version wird angezeigt). Mit (4) kann die nachinstallierte Software entfernt werden.

Wie im Bild zu sehen, liefere ich eine Datei mit NMEA Daten mit (vom Greifswalder Bodden). Es kann natürlich zum Test auch jede andere eigene NMEA-log Datei gewählt werden.

Alternativ kann der Server auch direkt die NMEA Daten von einem angeschlossenen seriellen Gerät oder Bluetooth lesen (COM port) oder auch z.B. von einem Raspberry (IP). Wenn spezielle Setups erforderlich sind, kann man über 'custom' auch die Server-Konfiguration per Hand bearbeiten, eine Vorlage mit Kommentaren wird mitgeliefert.

Mit diesem Setup kann die Software damit also auch komplett z.B. and Bord auf einem Windows-Laptop benutzt werden. Die Funktionen sind die gleichen wie mit dem Raspberry.

Wenn man nur den Server starten möchte - (3) "startServer". Das muß man auch nach jeder Änderung tun.

Linux

Wie unter Installation beschrieben, startet man avnav -g und wählt auch hier die zu konvertierenden Karten.

Details

Im Folgenden werden die Konvertierungsschritte beschrieben - normalerweise muss man sich darum nicht kümmern - nur wenn die default-Einstellungen der GUI nicht gut genug sind...

> *Der erste Schritt geht relativ schnell. Alle Kartendateien werden gelesen, Auflösung und Abdeckung werden ermittelt (falls nötig wird konvertiert). Im Ergebnis entsteht im workdir/<name> Verzeichnis eine Datei chartlist.xml. Der Aufruf dazu lautet:*
>
> ```
> read_charts.py -g -o name -m chartlist inputdir [inputdir...]
> ```
>
> *Anschliessend sollte die chartlist.xml noch einmal mit einem Texteditor überprüft werden, manchmal macht es Sinn, Kartendateien noch einem anderen Layer zuzuordnen. Das kann einfach durch Verschieben des entsprechenden XML Elements erfolgen. Man kann sich dazu an den Namen der Karten orientieren - meist mach es Sinn Karten vergleichbaren Detailgrades in einen Layer zu verschieben.*
>
> *Der zweite Schritt ist etwas langwieriger, hier erfolgt die eigentliche Erzeugung der Kartenkacheln. Der Aufruf:*
>
> ```
> read_charts.py -g -o name -m generate inputdir [inputdir...]
> ```
>
> *Unter workdir/<name> muss bereits eine chartlist.xml existieren. Die Erzeugung läuft multi-threaded, auf einem Dual Core 2x2Ghz ca. 20 min für einen Kartensatz mit ca. 20 Karten.*
>
> *Zum Schluss muss man noch die gemf Datei erzeugen*
>
> ```
> read_charts.py -g -o name -m gemf inputdir [inputdir...]
> ```
>
> *Man kann auch alle Schritte kombinieren – dazu einfach -m all noch vor den anderen Parametern bei Schritt 1 angeben:*
>
> > ```
> > read_charts.py -g -m all [-o name] inputdir
> > ```

MapProxy plugin

Es gibt ein Plugin, das MapProxy einbindet und damit Zugriff auf viele Online Kartenquellen erlaubt. Das Plugin unterstützt bei der Konfiguration von MapProxy und erlaubt das

komfortable Herunterladen von Karten. Für Details siehe die plugin Beschreibung.

Es bietet viele Funktionen, die dem Mobile Atlas Creator ähneln, und kann diesen damit weitgehend ersetzen.

Download von Karten mit dem Mobile Atlas Creator

Für die Nutzung des Mobile Atlas Creators ist ausser Java und dem MOBAC selbst keine weitere Software auf dem PC/Laptop nötig. Man muss beim Download der Karten allerdings ein gewisses Schema einhalten, damit die Karten in das oben beschriebene Layer-Konzept passen und die Datenmengen überschaubar bleiben.

Dazu sollte man (je nach Kartenquelle) z.B. 3 Layer vorsehen: Übersicht(Zoom Level 7-10) Navigation (level 10-15), Details (Level 16-18). Anschliessend sollte man im MOBAC layerweise vorgehen. Dazu jeweils als gewünschte Zoom Level die zum Layer gehörigen anklicken (links oben), danach alle Teilbereiche jeweils markieren und unter einem beliebigen Namen dem Atlas hinzufügen. Das jeweils für die anderen Layer wiederholen (dabei sinnvolle Auswahlen treffen). Anschliessend sollte man die Atlas-Konfiguration unter einem beliebigen Namen speichern - die kann man ggf. noch für weitere Versuche brauchen. Als output Format OsmDroid GEMF (File->convertAtlasFormat) wählen und die Atlas-Erzeugung starten. Im output Verzeichnis entsteht eine xxx.gemf Datei. Diese auf den raspberry installieren.
Auf der Mapsources Seite sammle ich chart sources für den mobac, die für uns nützlich sein könnten.

Avnav Ocharts

=== nicht für Android ===

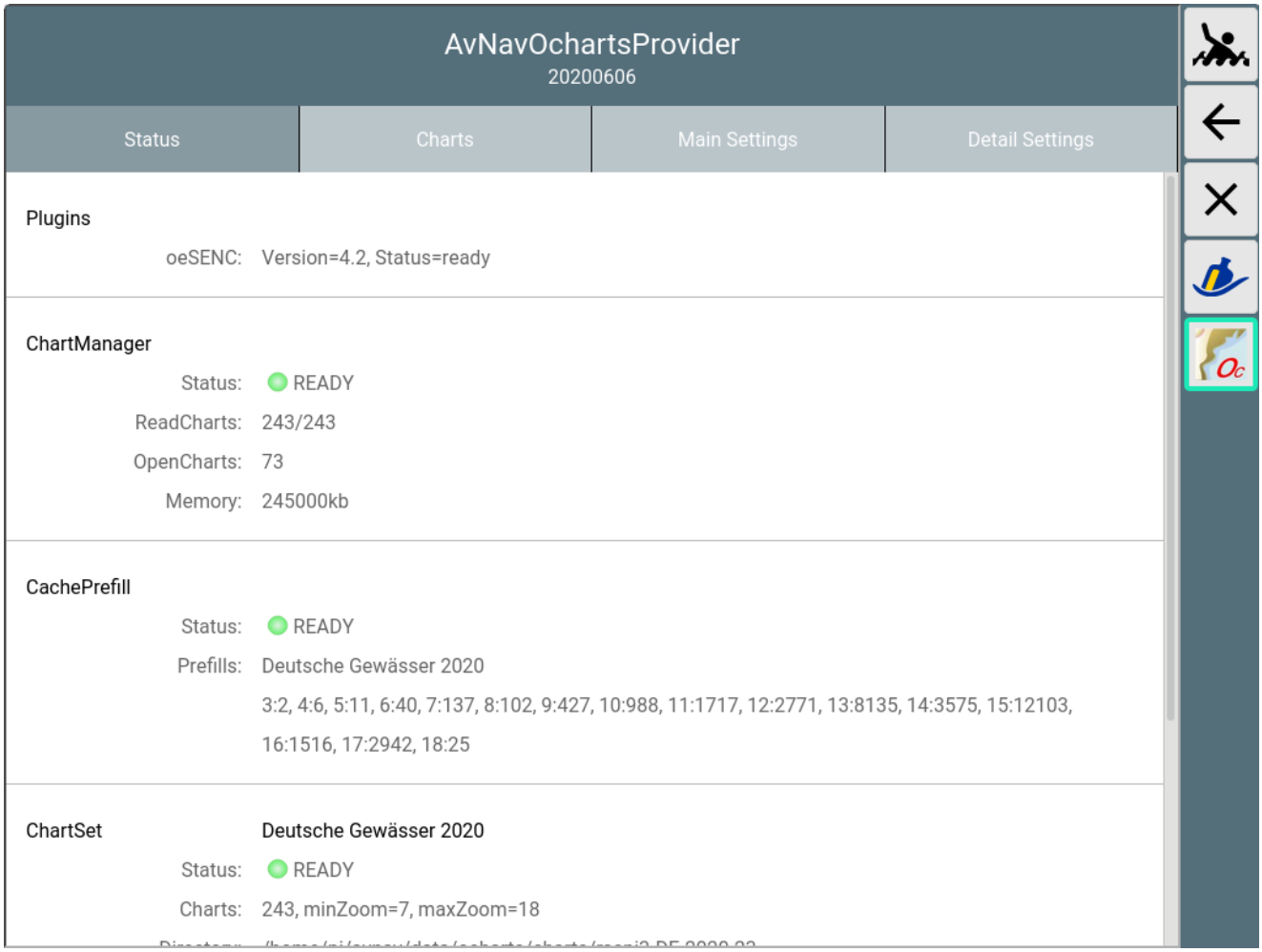
Inhalt

AvNav kann Karten in den verschiedenen Raster-Formaten verarbeiten. Bisher war es aber nicht in der Lage, offiziell verfügbare Seekarten zu lesen und anzuzeigen. Seit einiger Zeit gibt es die Firma o-charts, die Karten für viele Gebiete der Erde für OpenCPN bereitstellt.

Nach einigen Absprachen mit der Firma können diese Karten nun (ab Version 20200515 mit einem Plugin - s.u.) auch für AvNav genutzt werden. Bisher können die oesenc Vektor-Karten genutzt werden und ab Version 20220225 (und den zugehörigen Änderungen im o-charts shop - siehe unter releases) können die oeRNC Raster Karten genutzt werden.

Um die Karten in AvNav darzustellen, müssen sie zunächst einmal in Raster Karten umgewandelt werden. Das erledigt ein neues Plugin für AvNav (avnav-ocharts). Die Umwandlung erfolgt dabei im laufenden Betrieb immer dann, wenn die Karten dargestellt werden sollen (teilweise initial in einen cache). Damit kann mit diesen Karten weitgehend normal gearbeitet werden ohne dass man sich um diesen Prozess kümmern muss.

Das Handling der Karten erfolgt dabei vollständig durch das Plugin - das betrifft auch die Installation (die Karten können nicht über die normale Download-Seite hochgeladen werden). Das Plugin hat dazu eine eigene GUI, die von der Hauptseite über den Button (User Apps) und dort über den Button Ocharts-Provider erreichbar ist.



Erwerb und Installation der Karten

Wichtiger Hinweis: Wenn man keinen Dongle von o-charts hat, sind die Karten an das System gebunden. Wenn es daher beim Einbringen oder bei der Nutzung der Karten Probleme gibt, bitte **nicht** das System neu aufsetzen - sondern eine Reparatur versuchen. Wenn man das System neu aufsetzt, werden die Lizenzen ungültig. Ich helfe gerne bei Problemen - Kontakt z.B. per email.

Nur bei Nutzung der AvNav Images ist auch ein update der Images möglich. Bitte im Zweifel vorher testen.

Um bei ocharts Karten kaufen zu können, muss dort zunächst ein Konto angelegt werden.

Anschliessend muss man seine Systeme, auf denen man die Karten nutzen möchte auf der Seite von o-charts registrieren. AvNav benutzt dabei den "Offline" Prozess .

Dieser besteht aus den folgenden Schritten:

1. Erzeugung eines "Fingerprints" für das System, auf dem die Karten genutzt werden sollen. Dieser kann in AvNav über die Bedienoberfläche des Plugins erzeugt und heruntergeladen werden.
2. Hochladen dieses Fingerprints zu o-charts und Anlegen eines Systems (im Wesentlichen Vergabe eines Namens)

3. Kaufen von Karten
4. Zuordnen zu dem angelegten System
5. Nach kurzer Zeit gibt es eine Mail von o-charts mit einem Download-Link für die Karten (Zip-File)
6. Hochladen der Karten in AvNav (wieder über die Bedienoberfläche des Plugins).

Für Updates werden die Schritte 4,5 und 6 wiederholt (bei 4 nur Anforderung des Updates).

Für weitere Kartensätze Schritte 3-6.

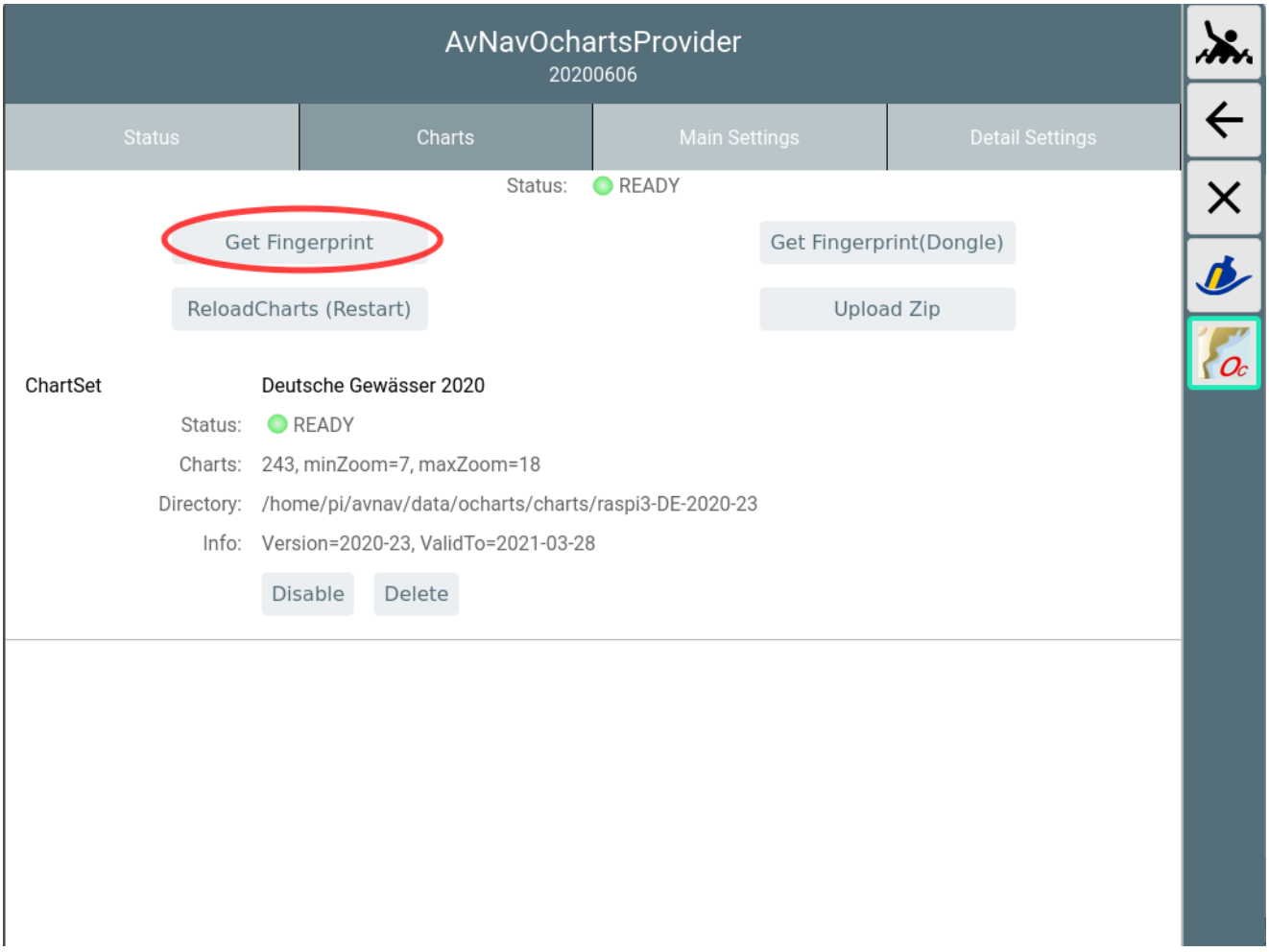
Für die Schritte 2,3,4 und 5 wird natürlich ein System mit Internet-Verbindung benötigt. Das kann z.B. ein Laptop oder auch ein Android Tablet sein.

Für den Ablauf des Prozesses habe ich ein Video gemacht, um ihn zu verdeutlichen. Hier noch einmal eine kurze Beschreibung dazu.

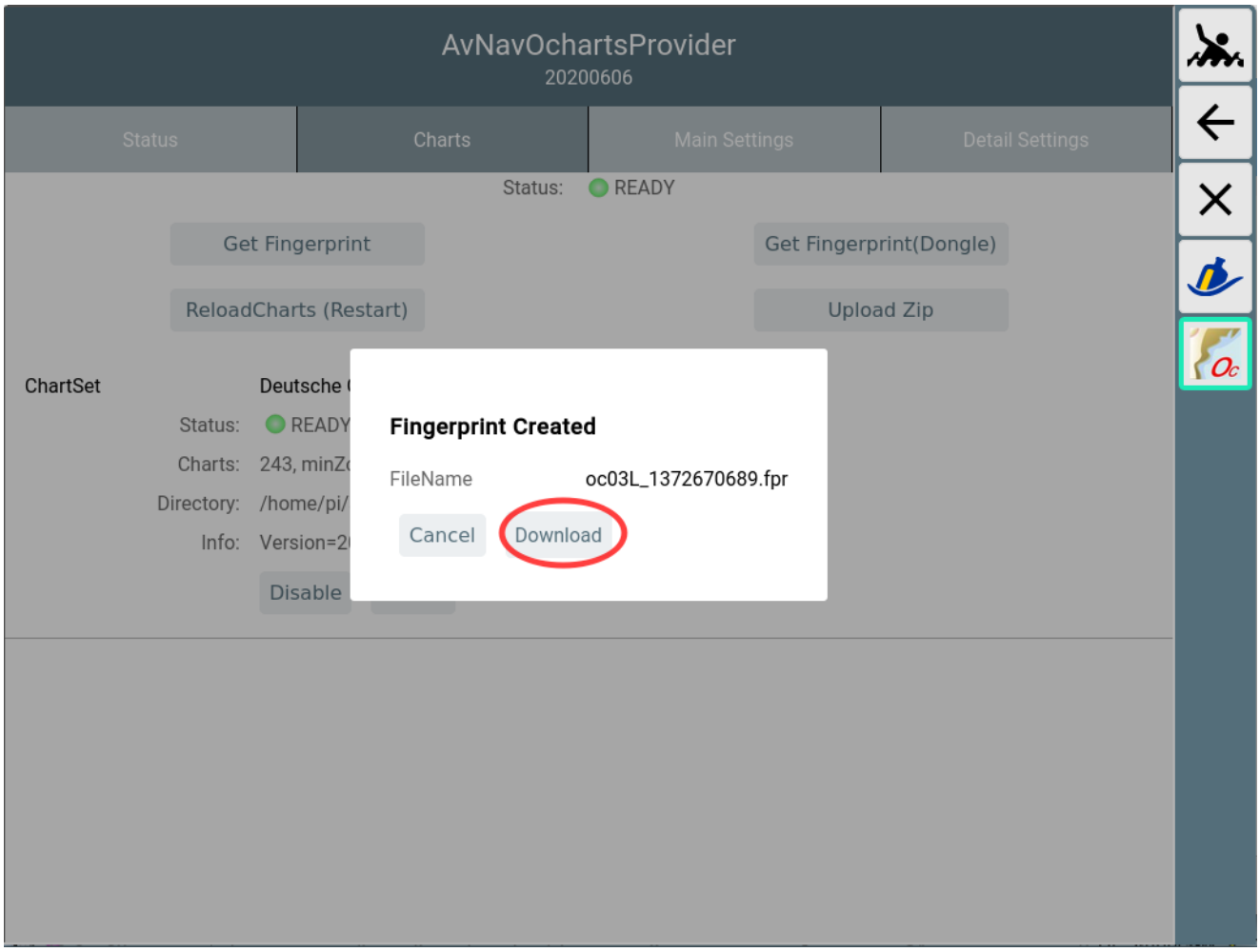
Hinweis: Wenn die Karten auf dem gleichen System bereits für OpenCPN registriert sind, dann kann man direkt mit Schritt 6 starten. Als Alternative der Zugriff auf die OpenCPN Kartenverzeichniss mit o-charts Karten auch im Plugin konfiguriert werden.

1. Erzeugung des Fingerprints

Über ⠿ ->  auf die Oberfläche des Plugins gehen, dort den Tab "Charts" auswählen.



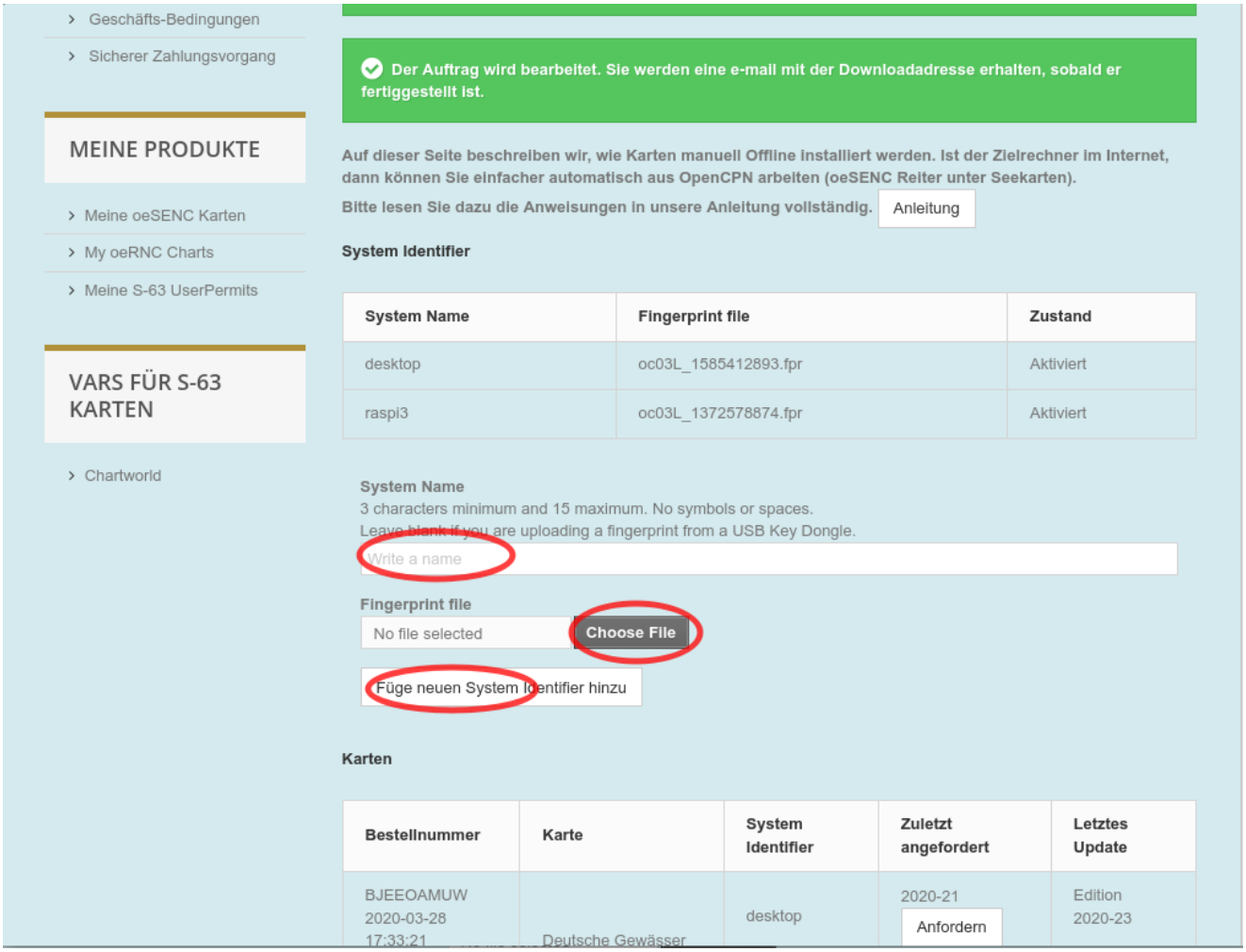
Mit "Get Fingerprint" die Erzeugung des Fingerprints anstossen. Falls man einen Dongle von o-charts benutzt, den Fingerprint über "Get Fingerprint(Dongle)" erzeugen.



Im Dialog die erzeugte Datei auf meinem Gerät speichern.

2. Hochladen des Fingerprints zu o-charts

Auf die o-charts Seite gehen und den Fingerprint hochladen.

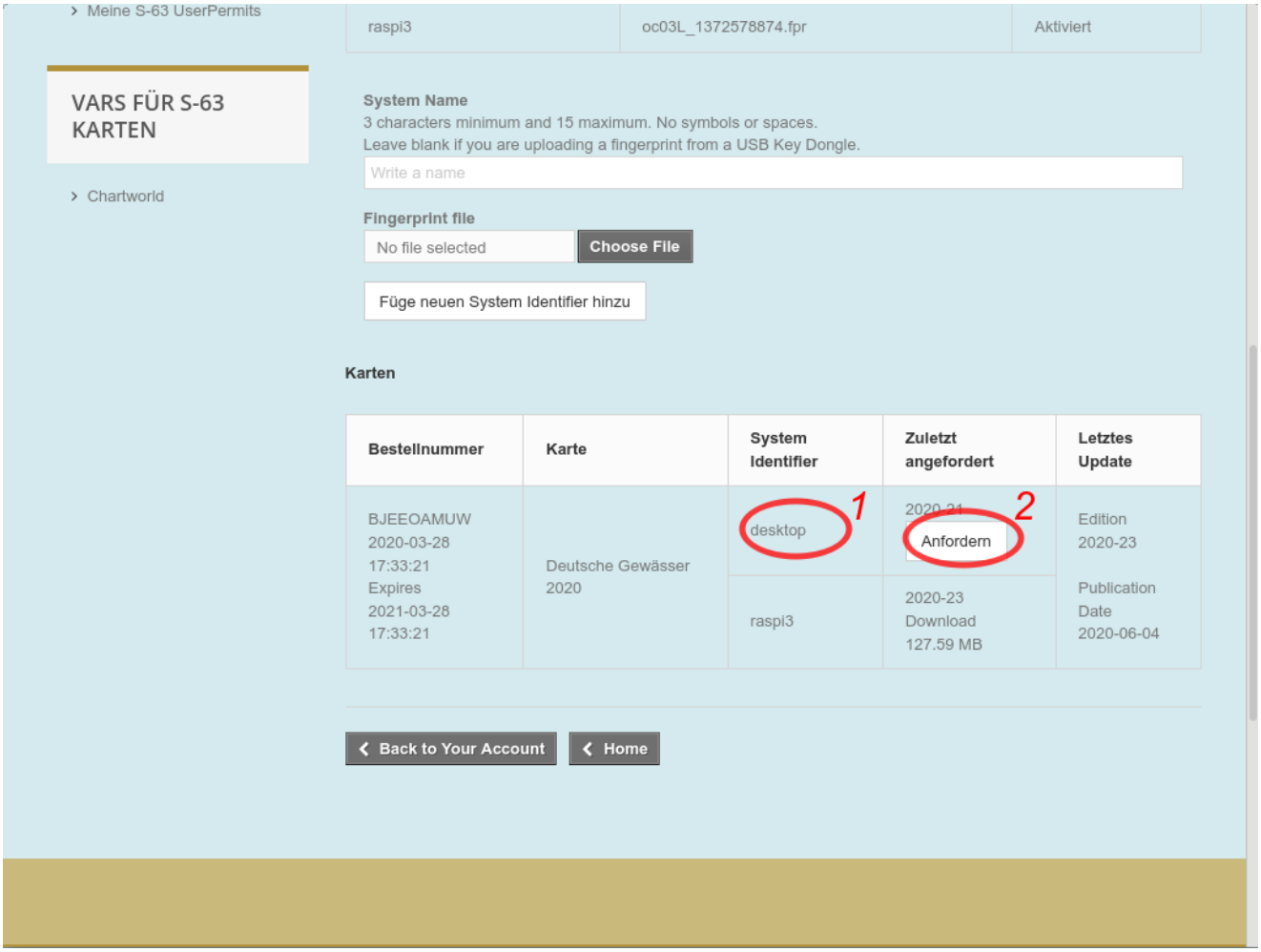


Mit Choose File die unter 2. gespeicherte Datei wählen. Dazu einen sinnvollen Namen vergeben (dieser findet sich später in den Mails mit den Download Links).

3. Kaufen von Karten

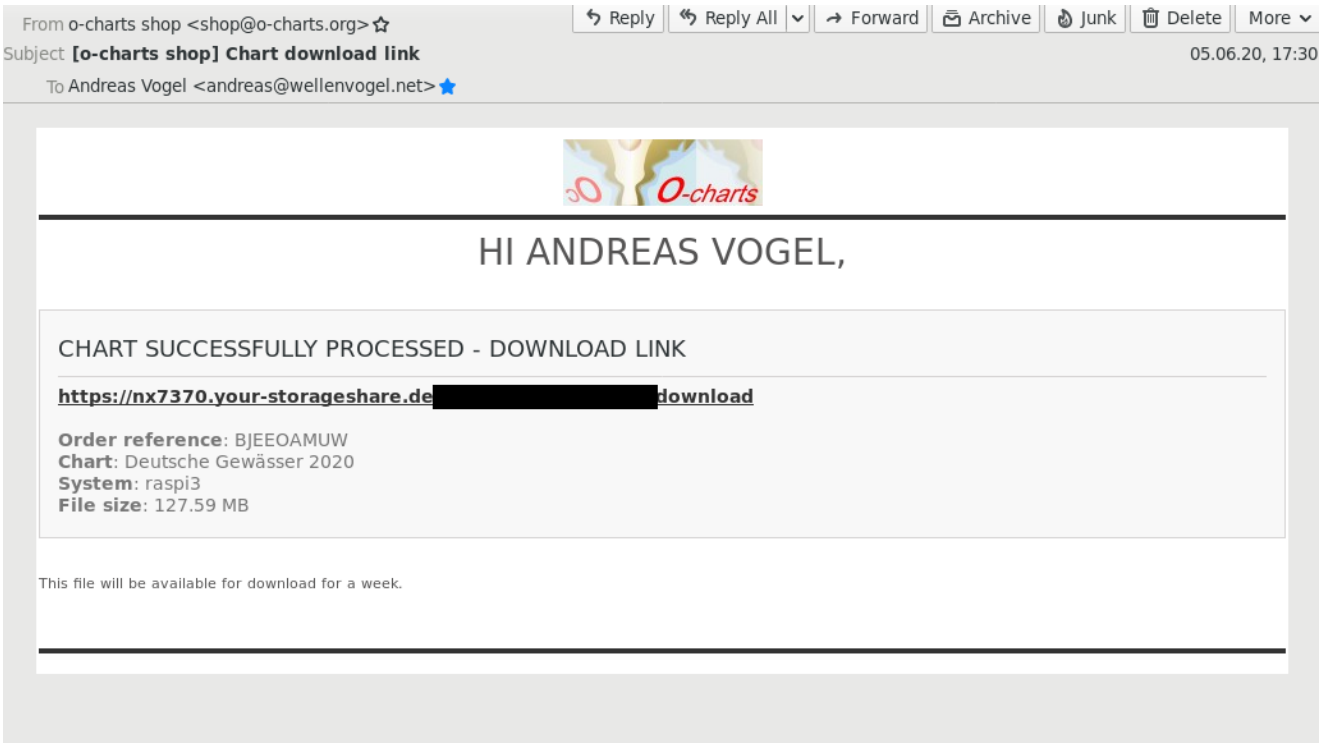
Bei o-charts aus den verfügbaren oeSENC Karten die gewünschten kaufen.

4. Zuordnen zum angelegten System



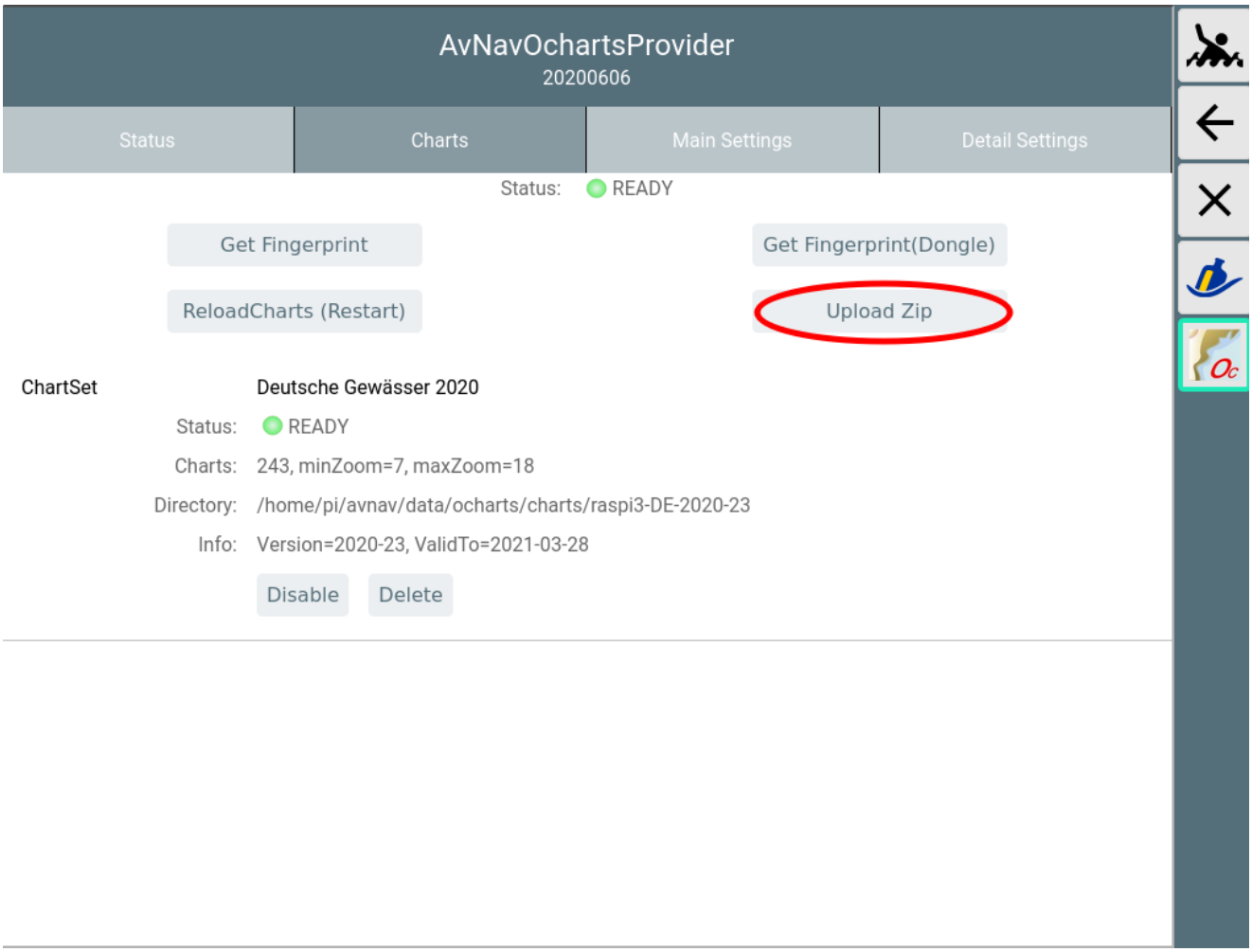
Bei 1 kann die Zuordnung zu einem angelegten System erfolgen (hier nicht mehr möglich, weil die Karten bereits zu den 2 maximal verfügbaren Systemen zugeordnet sind). Bei 2 wird dann die Mail mit dem Download-Link angefordert (das erfolgt auch bei Updates - hier im Bild zu sehen: Die letzte abgerufene Version ist 21, verfügbar ist 23).

5. Herunterladen der Karten



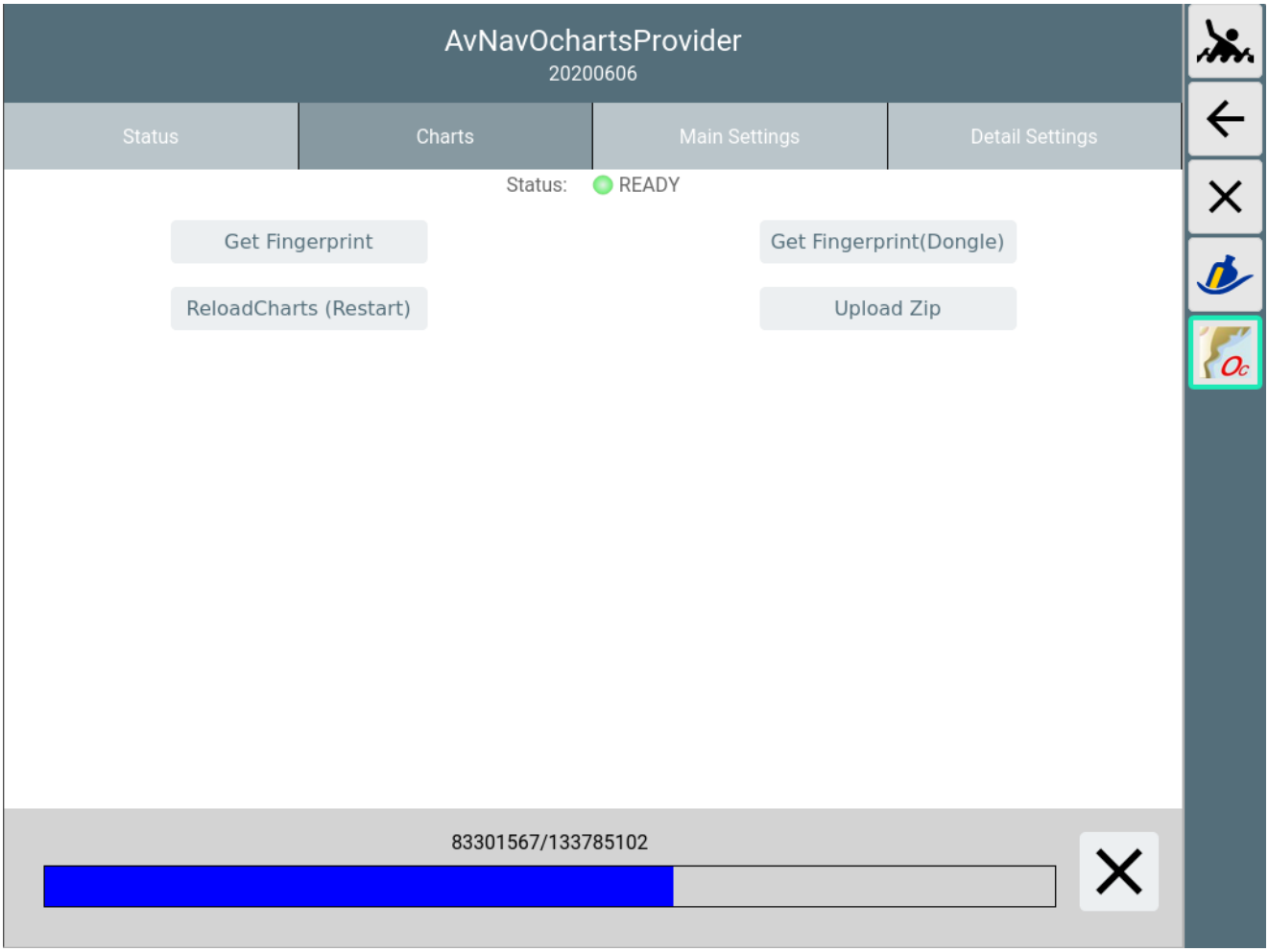
Nach kurzer Zeit erhält man von o-charts eine Mail mit dem Download-Link. Diese Datei (zip) herunterladen.

6. Hochladen der Zip-Datei zu AvNav.



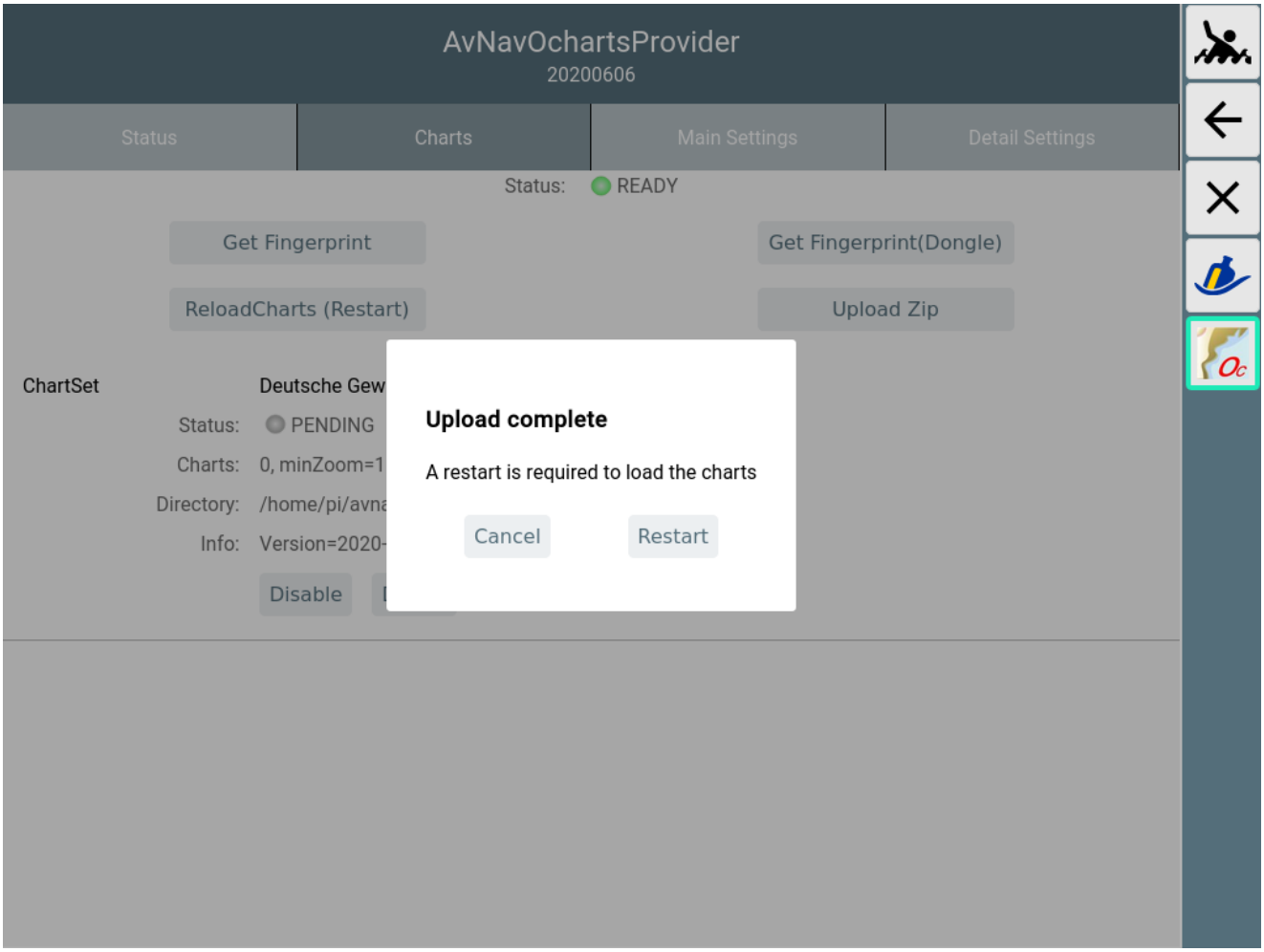
In der GUI des o-chart plugins über "Upload Zip" die im Schritt 5. heruntergeladene Datei zu AvNav hochladen.

Während des Uploads wird ein Fortschritt angezeigt.



Am Ende des Uploads wird auf dem AvNav Server die Datei ausgepackt, und es werden einige Prüfungen durchgeführt (das kann einige Sekunden dauern).

Die Karten werden (falls nicht anders konfiguriert) in das Verzeichnis /home/pi/avnav/data/ocharts/charts hochgeladen.

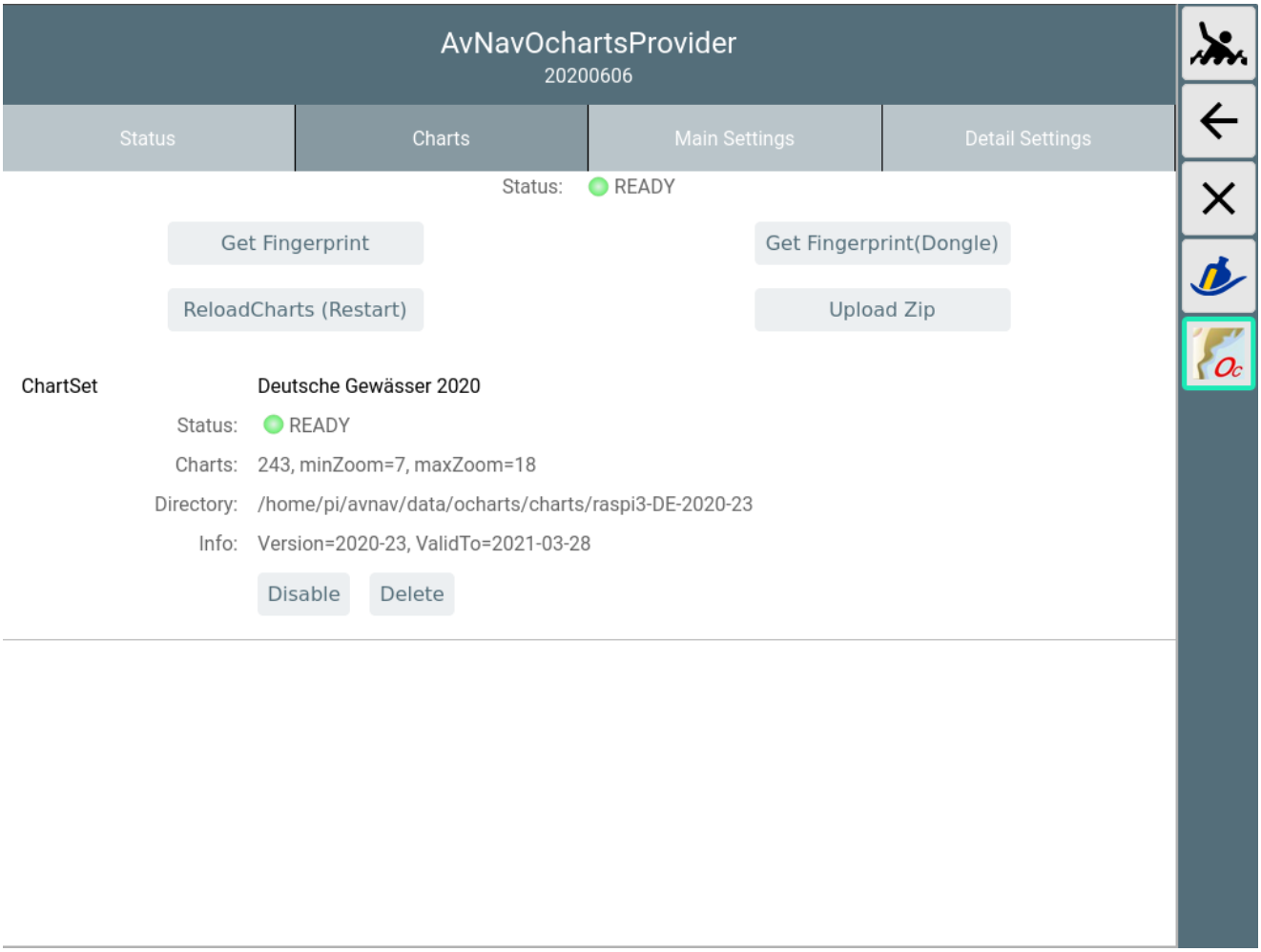


Wenn alle Prüfungen erfolgreich waren, wird angeboten sofort das Plugin zu restarten um die Karten nutzen zu können.

Falls es sich bei den neu hochgeladenen Karten um ein Update schon vorhandener Karten handelt, wírd der vorhandene Satz deaktiviert und der neue Satz aktiviert (beim Restart). Das kann in der GUI später geändert werden. Nicht mehr benötigte Sätze können hier gelöscht werden.

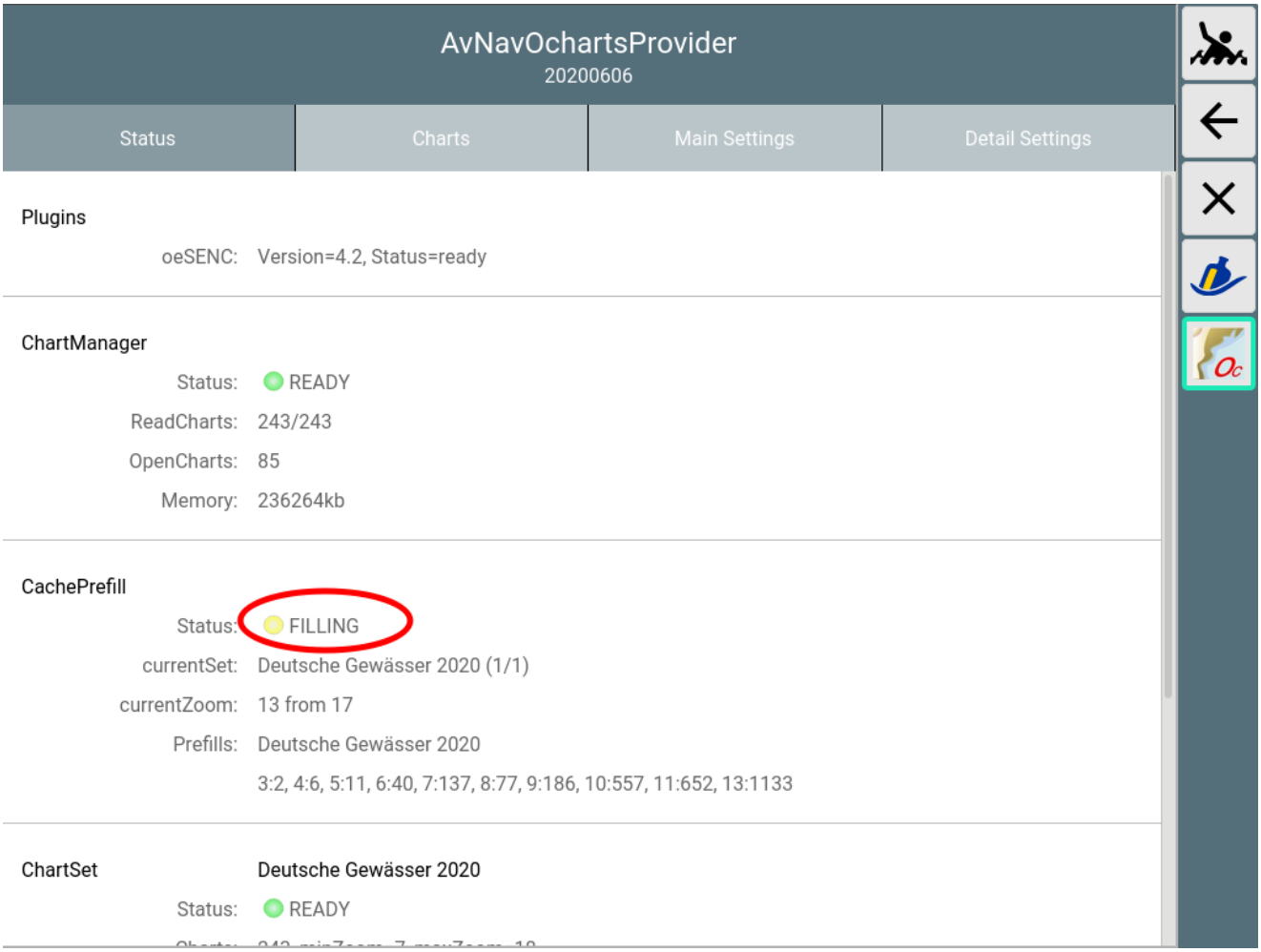
Beim Restart kommt es kurzzeitig zu einigen Fehlermeldungen, aber nach max. 30s sollte der Status zumindest wieder gelb sein (das Plugin liest jetzt alle vorhandenen Karten).

Wenn die Karten erfolgreich geladen werden konnten, sollte am Ende der Status für die Karten auf grün (READY) gehen.



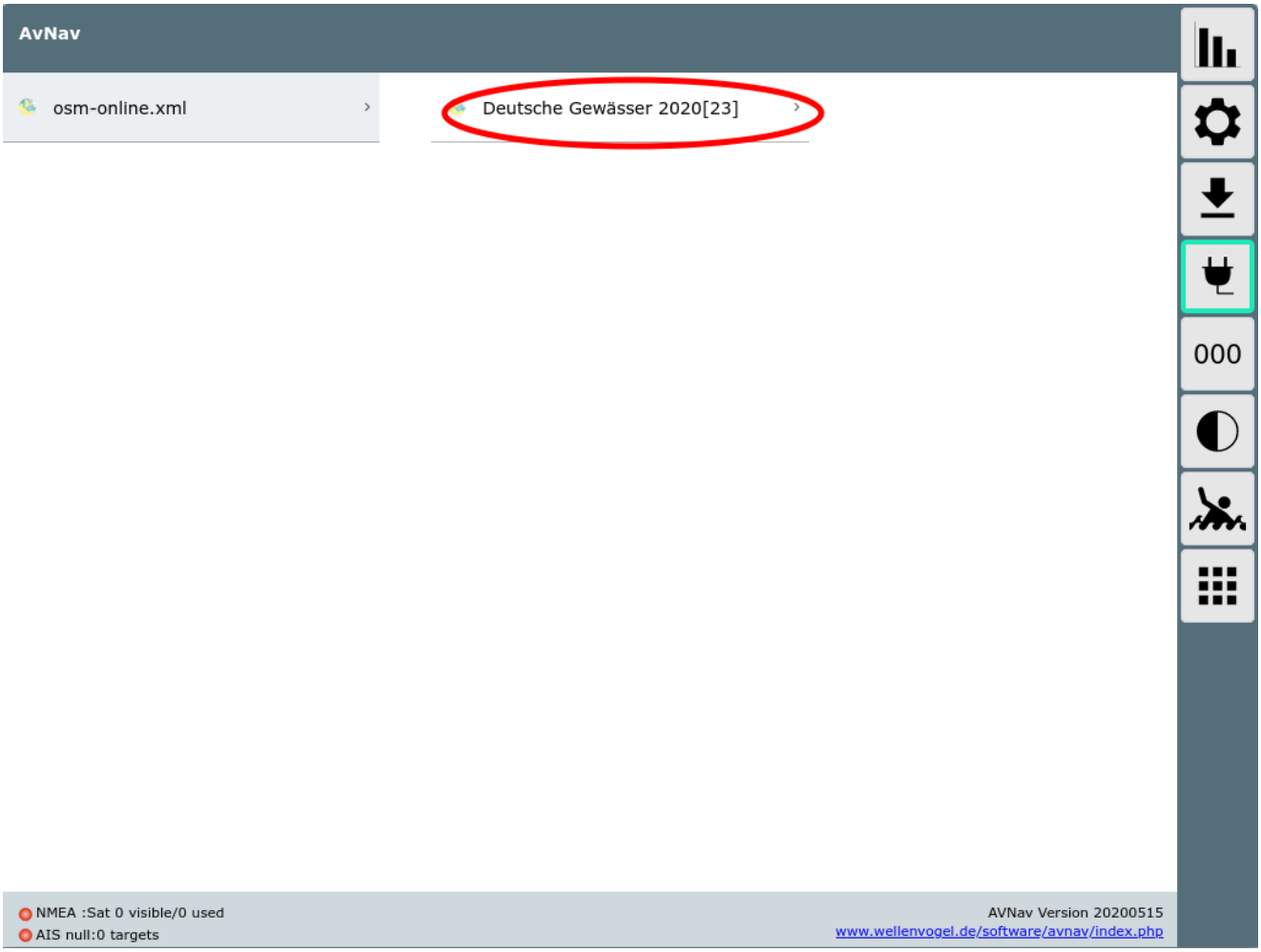
Falls der Status zu "ERROR" (rot) wird, wurde u.U. ein Zip-File hochgeladen, was für ein anderes System zugeordnet war. Details kann man im Log File (/home/pi/avnav/data/ocharts/provider.log) sehen.

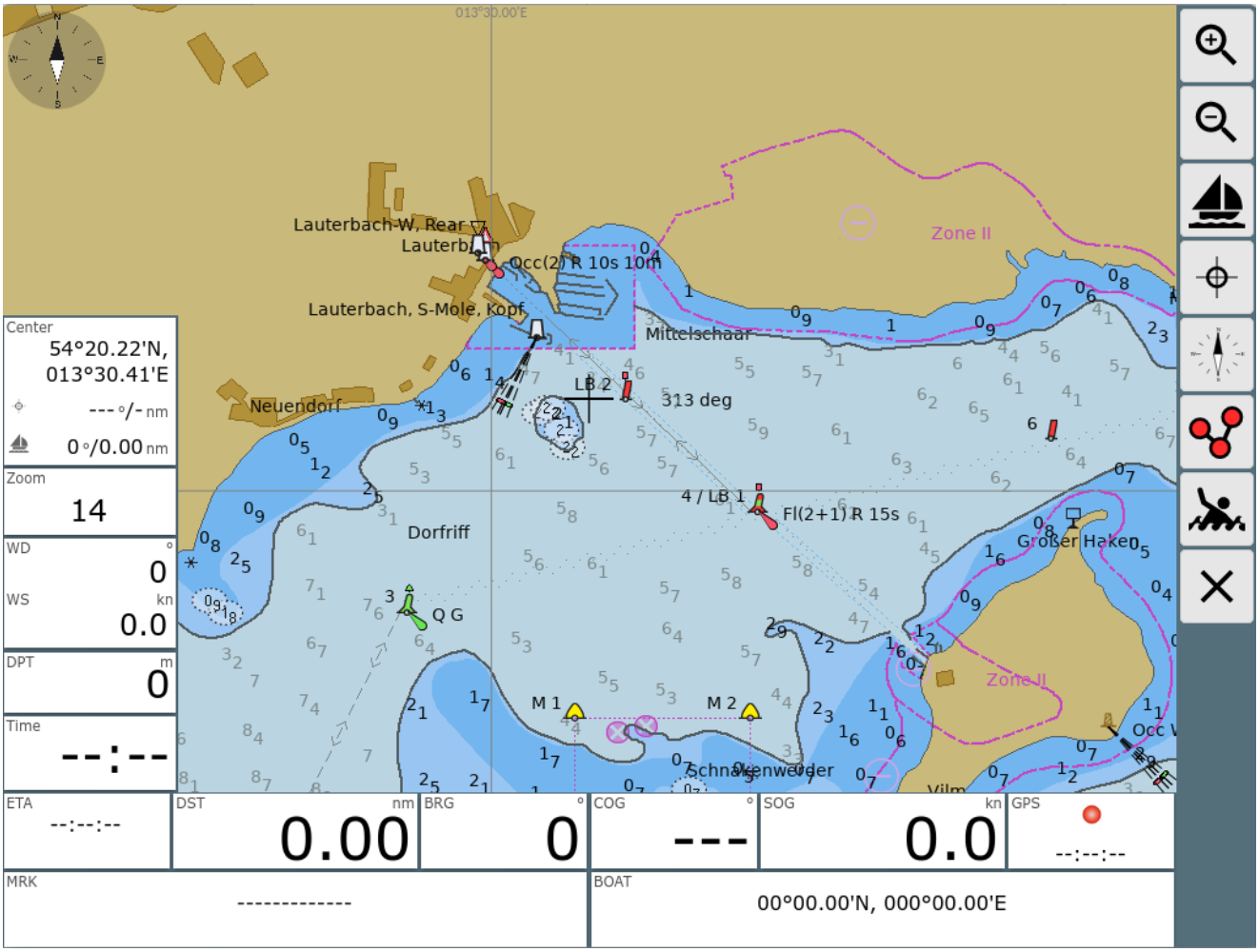
Nun sind die Karten verfügbar und können genutzt werden.
Unter dem Tab Status kann man etwas mehr Details erhalten.



Im Bild ist zu sehen, das nach dem Hochladen jetzt automatisch bereits eine Erzeugung von Kartenkacheln angelaufen ist ("Cache Prefill"). Das sorgt dafür, das die Verzögerungen, die sonst bei der Nutzung durch die begrenzten Resourcen des Raspberry Pi entstehen können, verringert werden. Dazu wird nach einem bestimmten Verfahren ein Teil der Kacheln im Bereich des Kartensatzes vorab erzeugt und in einem Cache File gespeichert.

Trotzdem können die Karten auch bereits unmittelbar genutzt werden.


AvNav
osm-online.xml
Deutsche Gewässer 2020[23]
000
NMEA :Sat 0 visible/0 used
AIS null:0 targets
AVNav Version 20200515
www.wellenvogel.de/software/avnav/index.php


Lauterbach-W, Rear
Occ(2) R 10s 10m
Lauterbach, S-Mole, Kopf
Mittelschaar
Zone II
LB 2
313 deg
Neuendorf
4 / LB 1
Fl(2+1) R 15s
Dorfriff
Großer Haken
Q G
M 1
M 2
Zone JJ
Schnaase
Vilm
Center
54°20.22'N,
013°30.41'E
---°/- nm
0°/0.00 nm
Zoom
14
WD
0
WS kn
0.0
DPT m
0
Time
--:--
ETA
--:--:--
DST nm
0.00
BRG °
0
COG °

SOG kn
0.0
GPS
--:--:--
MRK

BOAT
00°00.00'N, 000°00.00'E

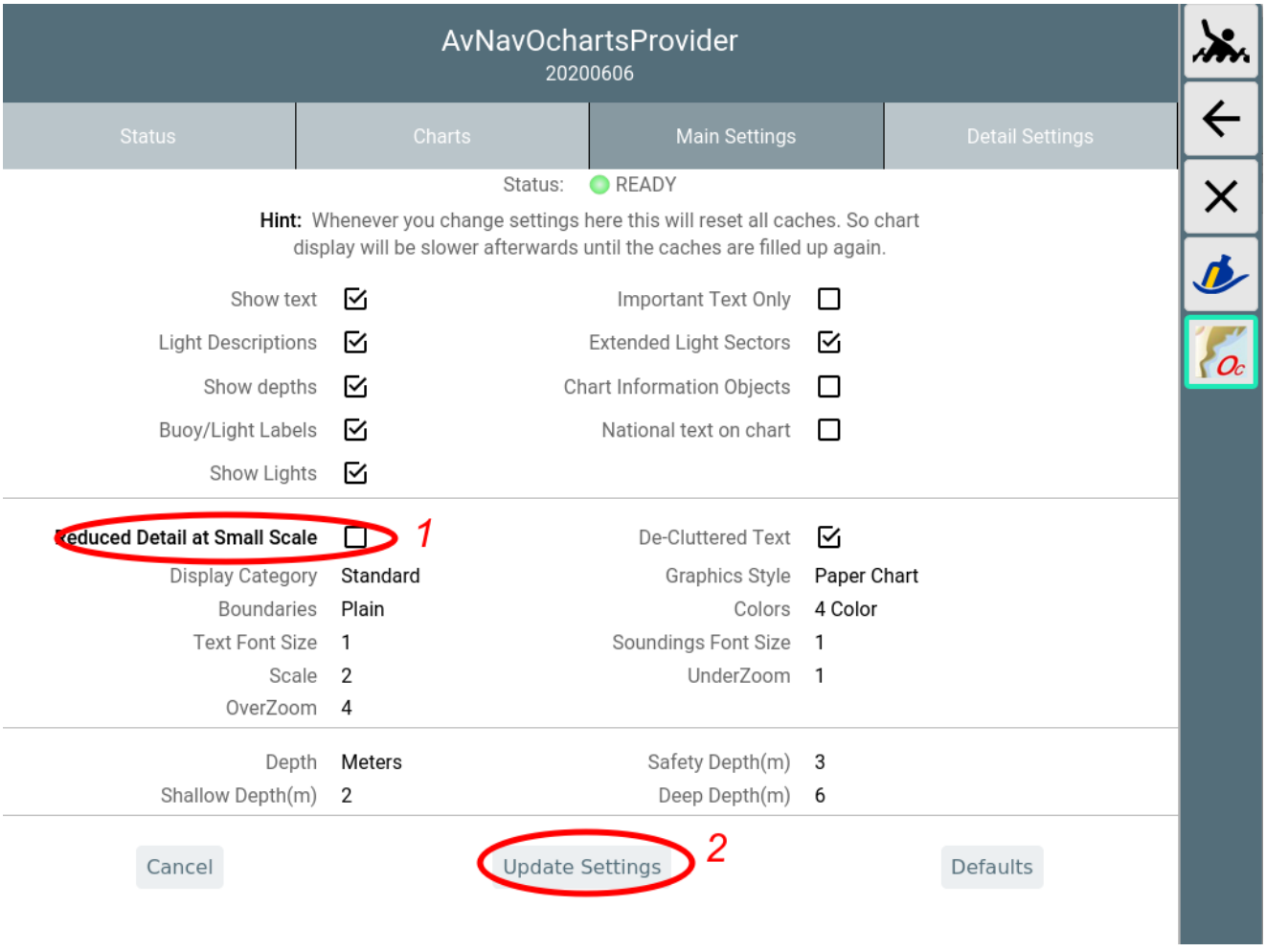
Wie bereits erwähnt, kann es bei der erstmaligen Nutzung in einem bestimmten Bereich zu Verzögerungen kommen (insbesondere auf den kleineren/älteren Pi's) - nach der erstmaligen Nutzung sind die Kacheln für den Bereich aber im Cache-Speicher und die Verzögerungen sind minimal.

Anpassung des Aussehens

Da die O-charts Karten zunächst als Vektorkarten vorhanden sind, kann in weiten Bereichen das Aussehen der Karten angepasst werden. Dabei sind allerdings einige Einschränkungen zu beachten:

1. Die Anpassung erfolgt auf der Server-Seite und ist damit für alle verbundenen Displays gleichartig wirksam
2. Wenn das Aussehen geändert wird, müssen alle bereits im Cache vorhandenen Daten gelöscht werden und alle Karten-Kacheln müssen neu berechnet werden. Das kann auf langsamen Systemen wieder zu Verzögerungen führen. Es läuft sofort wieder der automatische Prefill-Prozess an, um möglichst viele Kacheln schon vorberechnet zur Verfügung zu haben.

Die Veränderung der Parameter erfolgt über die GUI des Plugins (->), Tab "Main Settings".



Wenn eine Einstellung geändert wird (1) wird der Parameter fett dargestellt. Die Änderungen werden erst wirksam, wenn "Update Settings"(2) angeklickt wird.

Mit Cancel können die Änderungen zurückgenommen werden, Defaults setzt die Einstellungen auf Default-Werte. Die Parameter entsprechen weitgehend den bei OpenCPN vorhandenen Settings.

Die folgende Tabelle listet die Einstellungen.

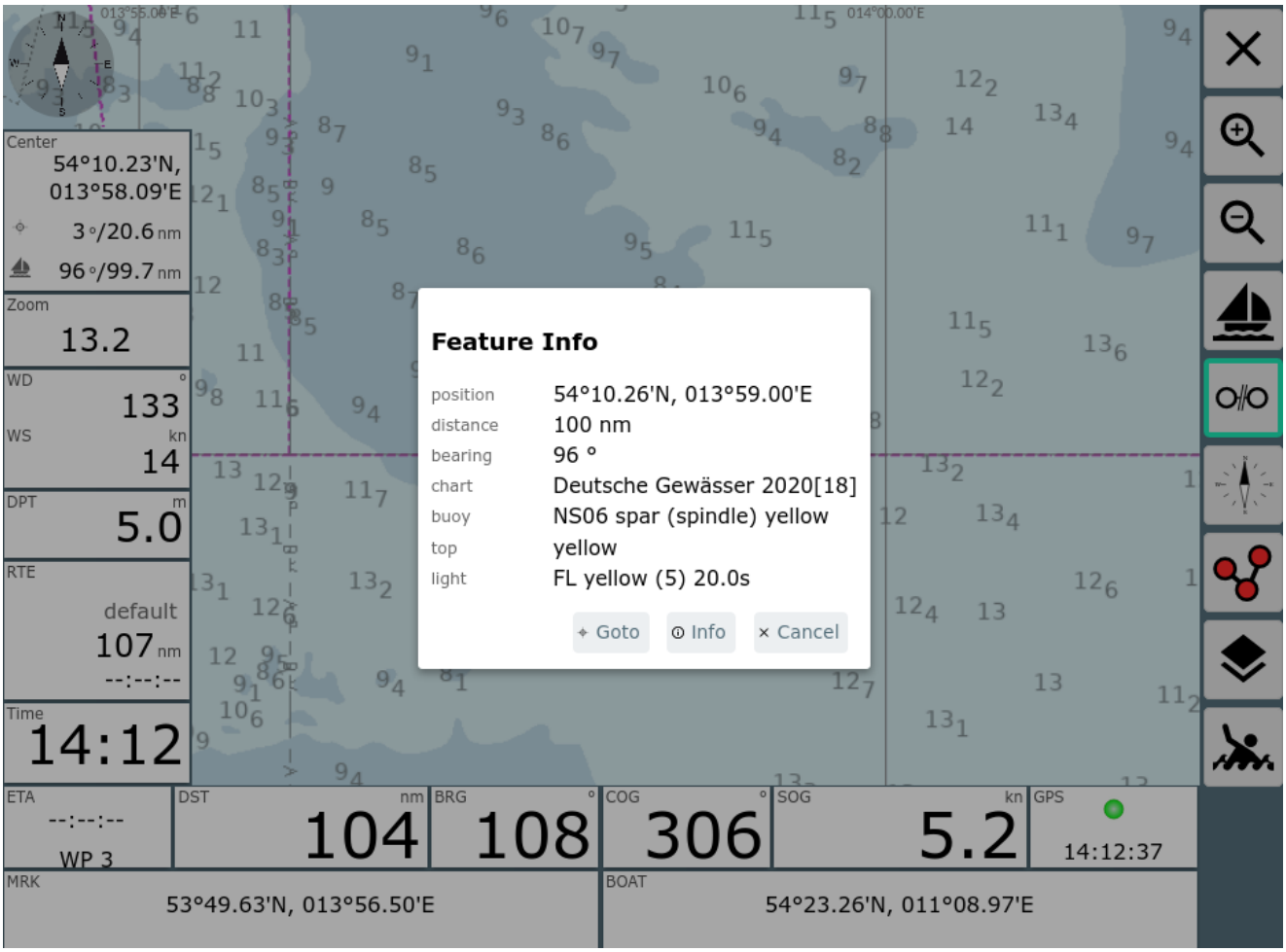
Name	Bedeutung	Default
Show Text	Zeige Texte zu den Objekten auf der Karte	true
Important Text Only	Verberge weniger wichtige Texte	false
Light Descriptions	Beschreibungen für Feuer	true
Extended Light Sectors	Sektoren für Feuer	true
Show Depth	Zeige Tiefen Werte	true
Chart Information Objects	spezielle Objekte auf der Karte	true
Buoy/Light Labels	Bezeichnungen für Feuer/Tonnen	true
National text on chart	Nationale Texte	true
Show Lights	Zeige Feuer	true
Reduced Detail at Small Scale	Reduziere die Details auf geringeren Zoom-Leveln	true
De-Cluttered Text	Bessere Anordnung der Texte	true

Display Category	Art der Darstellung (Base,Standard,All,User Standard)	All
Graphics Style	Grafische Darstellung (Paper Chart, Simplified)	Paper Chart
Boundaries	Art der Begrenzungen (Plain, Symbolized)	Plain
Colors	Farben (4 Color, 2 Color)	4 Color
Text Font Size	Skalierung für die Text-Grösse	1 (ca. 12px)
Soundings Font Size	Skalierung für die Tiefen-Angaben (ab oesenc-pi 4.2.x)	1 (ca. 12px)
Scale	Basis Skalierung. Höhere Werte sorgen für mehr Details auf kleineren Zoom-Stufen	2
UnderZoom	Anzahl der Zoom Stufen, die eine höher aufgelöste Karte verkleinert wird, wenn auf dem gewünschten Level keine Karte vorhanden ist	1
OverZoom	Anzahl der Zoom Stufen, die eine niedriger aufgelöste Karte vergrößert wird,wenn es keine besser aufgelöste Karte gibt. **Hinweis**: Scale,UnderZoom und OverZoom bestimmen massgeblich, wie aufwendig der Render-Vorgang ist, d.h. wieviele Karten an der Erzeugung einer Kachel beteiligt werden müssen. Kleinere Werte führen zu weniger Karten (schneller) können aber in bestimmten Bereichen zu weissen Flächen zwischen Karten-Teilen führen. Die Defaults sollten ein guter Kompromiss sein.	4
Depth	Einheit für die Tiefen-Angaben (Meters, Feet, Fathoms)	Meters
Shallow Depth	Tiefe für Flachwasser	2
Safety Depth	Tiefe für Sicherheits-Kontur	3
Deep Depth	Tiefe für Tiefwasser	6

Unter dem Tab "Detail Settings" können gezielt einzelne Karten-Features an- oder abgeschaltet werden.

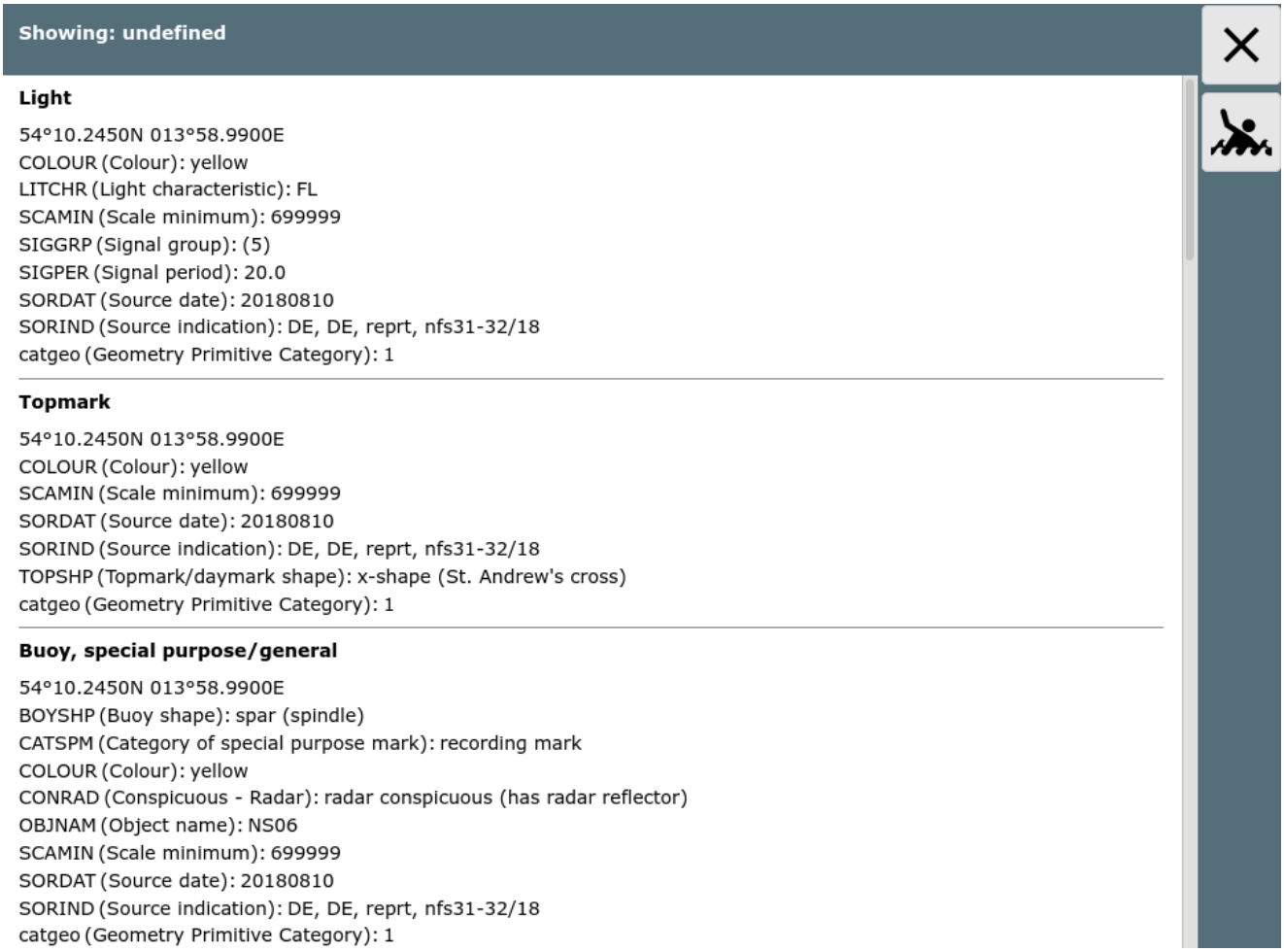
Feature Info (Object Query)

Ab der Version 20201219 (erfordert entsprechende Version von AvNav und vom plugin) gibt es eine Information zu den Objekt Eigenschaften bei Klick auf die Karte.



Es wird in dieser Darstellung zunächst die komprimierte Information zu einem Objekt angezeigt. Diese ist jedoch nur für Lichter, Tonnen und einige andere ausgewählte Klassen so verfügbar.

Über "Info" können die Roh-Informationen der Karten angezeigt werden.



Installation

Die Installation kann normal als Paket in den AvNav Images erfolgen. Für die Headless Images sind die nötigen Pakete bereits installiert. Zusätzlich sind die Pakete auch im Repository vorhanden. Es müssen die Pakete

- avnav-ocharts-plugin
- avnav-oesenc

installiert werden. Für das Paket avnav-ocharts-provider ist mindestens die Version 20200606 nötig. Das Paket avnav-oesenc ist das oesenc-pi plugin - allerdings so verpackt, das es nach /usr/lib/avnav/plugins/ocharts installiert wird, um Konflikte mit einer OpenCPN Installation zu vermeiden.

```
sudo apt-get update
sudo apt-get install avnav-ocharts-plugin avnav-oesenc
sudo systemctl restart avnav
```

Falls auf anderen Images gearbeitet wird, sollte das Repository von free-x hinzugefügt werden.

```
deb https://www.free-x.de/debian buster main contrib non-free
```

Die Pakete sind auch in der Release Liste hier darunter zu finden. Um ein solches Paket zu installieren (falls es noch nicht im Repo ist - oder um ein älteres zu verwenden) - das Paket herunterladen und installieren (die Version durch die jeweils aktuelle ersetzen).

```
cd /home/pi/avnav
wget -O avnav-ocharts-plugin_20200606-raspbian-buster_armhf.deb
https://www.wellenvogel.net/software/avnav/downloads/release-
ochartsplugin/20200606/avnav-ocharts-plugin_20200606-raspbian-
buster_armhf.deb
sudo dpkg -i /home/pi/avnav/avnav-ocharts-plugin_20200606-raspbian-
buster_armhf.deb
sudo systemctl restart avnav
```

Alternativ kann man das Paket natürlich auch auf einem PC herunterladen und dann z.B. per scp/WinScp auf den pi kopieren und dann dort installieren.

Releases

Alle Releases und auch zwischenzeitlich eventuell gebaute Entwickler-Versionen (daily builds) findet man unter:

- Releases
- Daily Builds

Release Versionen

- 20230702 packages
  - Fehlerkorrektur #41: Korrektes Handling von SENC overlays
  - Fehlerkorrektur #47: Korrektes Handling von obsoleten Karten (open error 3)
  - Verbesserung: Erlaube das Setzen eines Render-Timeouts in den Plugin-Einstellungen

- 20220605 packages
  - Fehlerkorrektur #36: Probleme unter OpenPlotter mit OpenCPN auf flatpak

- 20220421 packages
  - Fehlerkorrektur: Fehler beim restart des providers
  - dependencies to avnav-ocharts

- 20220307 packages

- Fehlerkorrektur #31: Checkbox für OpenCPN Integration nicht angezeigt
- Fehlerkorrektur: falsche Version angezeigt
- besseres Fehlerhandling beim Nutzen der Karten, Anzeige solcher Fehler im Status
- korrekter Bereich für "memPercent" bei der Konfiguration

- 20220225 packages
  - Neuer O-charts shop
    In den nächsten Tagen wird im o-charts Shop ein neues Verschlüsselungsschema eingeführt.
    Das erfordert diese neue Version für das AvNav plugin. **Ausserdem muss man das neue avnav-ocharts Paket installieren** - mindestens in der Version 0.9.0.72 (sinnvoll nutzt man dafür das avnav update plugin). Das (alte) Paket avnav-oesenc ist jetzt überflüssig und kann deinstalliert werden. Es kann aber auch problemlos auf dem System verbleiben.
    Die Benutzung der Karten und der Kaufprozess haben sich nicht geändert. Ebenso funktionieren alle vorhandenen Karten weiter. Wenn die Änderung im Shop vollzogen ist, können allerdings keine Karten mehr heruntergeladen und mit dem alten plugin genutzt werden.
    Mit dem neuen plugin(und dem neuen Shop) **ist AvNav jetzt auch in der Lage oeRNC Karten zu nutzen** (wie z.B. die Imray Karten für das Mittelmeer).
  - Es gibt jetzt auch Pakete für die raspberry debian bullseye OS Versionen (sowohl 32 bit als auch 64 bit)
  - Der Schalter "reduce details on lower zoom levels" funktioniert jetzt korrekt
  - Die "under zoom" Einstellung arbeitet jetzt effitzineter, so das der erlaubte Bereich vergrössert und der default auf 4 gesetzt wurde. Das vermeidet potentielle weisse Flächen, wenn es keine Karten direkt für den gewählten zoom level gibt.
  - Man kann jetzt direkt die Nutzung der ggf. unter OpenCPN auf dem gleichen System installierten Karten direkt in den AvNav plugin Einstellungen aktivieren (man muss nicht mehr die Konfig-Datei bearbeiten). Das funktioniert aber nur für Karten, die mit dem neuen ocharts_pi plugin bei OpenCPN installiert wurden.

- 20210711 Pakete
  - Fehlerkorrektur: Caches werden nicht neu gebaut nach Parameter-Änderung

- 20210328 Pakete
  - Parameter handling in AvNav (erfordert AvNav ebenfalls ab 20210322)
  - Limit für Zip-Grösse auf 3GB erweitert
  - Fehler beim Restart behoben

- 20210115 Paket
  - Umstellung auf python3 (erfordert AvNav ebenfalls ab 20210115)

- besseres Handling von Fehlern beim Laden der Karten

- 20201219 Paket
  - Anzeige von Objekt-Informationen (erfordert AvNav ebenfalls ab 20201219)

- 2020115 Paket
  - Verbessertes Memory Handling: Der Xvfb wird überwacht und restartet, wenn er > 120 MB Speicher braucht
  - Schnellerer Start: Die Karteninformationen werden in einem Cache gehalten, es wird nur neu gelesen, wenn sich an den Karten etwas ändert
  - Speicherleak im OpenCPN plugin beseitigt (workaround) - damit wächst der Speicherverbrauch vom Xvfb nicht mehr so stark
  - Hinweis: Nach Installation per Hand mit dpkg (erzeugt einen Fehler) müssen ggf mit

    ```
    sudo apt-get install -f
    ```

    die neuen Abhängigkeiten nachinstalliert werden.

- 20200710 Paket
  - Das Verzeichnis für die Karten kann separat eingestellt werden - siehe Details
  - Beim Hochladen von Karten wird jetzt sofort geprüft, ob die Karten gelesen werden können, sonst wird das Hochladen verweigert
  - Keine Fehler mehr in der GUI beim Restart des Providers
  - Funktioniert auch auf einer vfat Partition wie sie von avnav-touch genutzt wird

- 20200705 Paket
  - Korrektur für ein Problem im OpenCPN plugin das potentiell dafür sorgt, das nach längerer Laufzeit keine Karten mehr dekodiert werden können.

- 20200606 Paket
  - erste Version

Lizenzhinweisehttps://www.wellenvogel.net/softwa ochartsplugin/20220307/?dir=software/avnav/downloads/release-ochartsplugin//20220307&up=1

Die Nutzung der Karten für AvNav mit dem oesenc-pi Plugin ist so mit o-charts diskutiert und abgestimmt worden und ist damit legal im Sinne der Lizenzen.

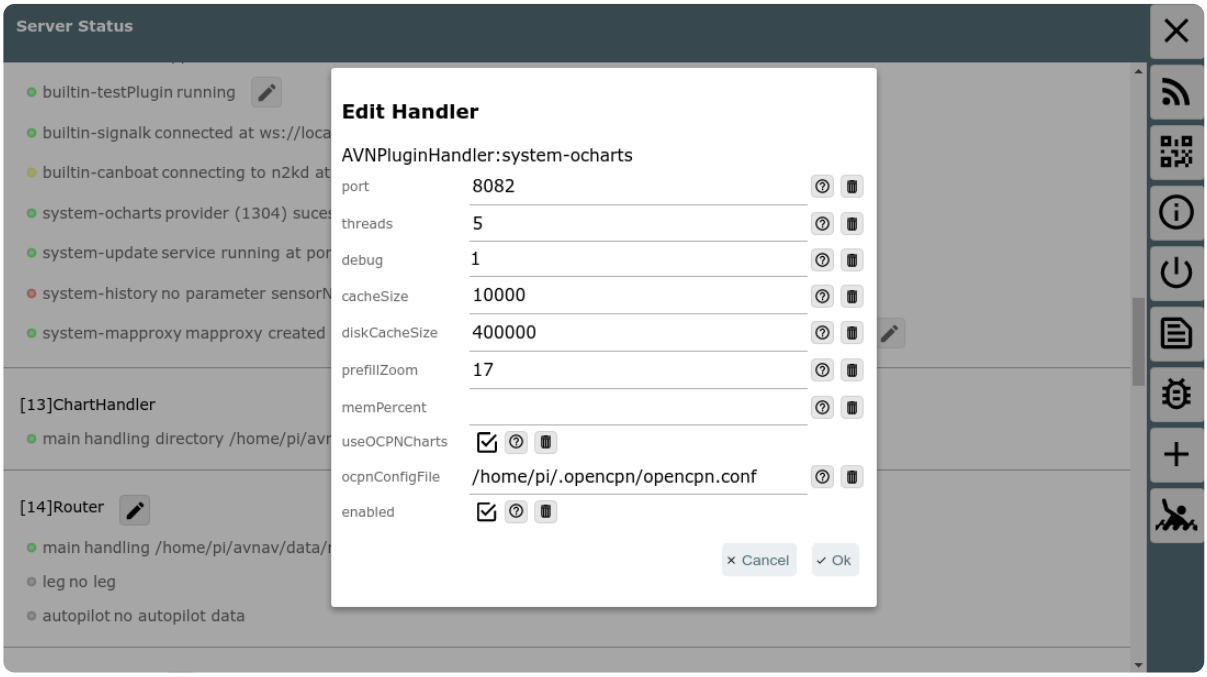
Die Lizenzbedingungen von o-charts sind dabei unbedingt zu beachten. Es ist insbesondere nicht gestattet, die Karten zu kopieren oder auf anderen als den lizensierten Systemen einzusetzen.

Der Zugriff auf die Karten innerhalb von AvNav ist nur aus dem lokalen Netz möglich, maximal können 5 Geräte (Clients) gleichzeitig die o-charts von einem AvNav Server nutzen.

Für die Software-Lizenzen siehe die Readme.

Konfiguration des Plugins

Einige Konfigurationen des plugins können über die Server/Status Seite unter "plugins/system-ocharts" vorgenommen werden (AvNav >= 20210322).



Das sind die folgenden Parameter:

Name	Bedeutung	Default
port	Http port für das plugin	8082
threads	Zahl der zu nutzenden Threads	5
debug	Level für das logging (<datadir>/ocharts/provider.log). <datadir> ist auf einem RaspBerry Pi /home/pi/avnav/data	1

cacheSize	Maximale Zahl von Karten-Kacheln, die im Speicher gehalten werden sollen. Der Provider berücksichtigt aber auch noch den erlaubten Speicher und verringert diese Zahl potentiell	10000
diskCacheSize	Maximale Zahl von Kartenkacheln für einen Kartensatz, die im Cache-File auf der SD Karte gehalten werden sollen	400000
prefillZoom	Bis zu welchem Zoomlevel sollen beim Prefill schon Kacheln berechnet werden. Wenn man diesen Wert höher setzt, dauert der Prefill entsprechend länger.	17
memPercent	Der prozentuale Anteil des Systemspeichers, den der Provider maximal nutzen soll. Wenn man diesen nicht setzt (oder zu klein) berechnet der Provider intern einen Minimalwert und nutzt diesen. Der kann u.U. insbesondere bei der Nutzung von Rasterkarten sehr klein sein und ihn damit zwingen ständig Karten-Dateien zu öffnen und zu schliessen - was die Geschwindigkeit stark reduzieren kann. Wenn man ausreichend Speicher hat (z.B. > 2GB), wird das Arbeiten beschleunigt, wenn man den Speicher auf 1GB setzt.	---
useOCPNCharts (seit 20220225)	Wenn dieses Flag gesetzt ist, können Karten, die von OpenCPN auf dem gleichen System genutzt werden, auch in AvNav direkt verwendet werden. Das geht allerdings nur mit den Karten, die	aus

	mit dem neuen o-charts_pi Plugin installiert wurden (ab 1.3. 2022).	
ocpnConfigFile (seit 20220225)	Der Pfad zum OpenCPN config file (normalerweise $HOME/.opencpn/opencpn.conf). Diese Datei muss gelesen werden, um die installierten Karten zu finden.	$HOME/.opencpn/opencpn.conf

Technische Details

Die eigentliche Bereitstellung der Karten erfolgt durch ein executable auf dem Raspberry, das standardmäßig über den Port 8082 erreichbar ist. Dieses executable lädt das oesenc-pi OpenCPN plugin.
Die Kommunikation mit AvNav erfolgt über ein plugin in AvNav.

Die GUI ist eine reactjs Anwendung, die ebenfalls durch das executable bereitgestellt wird und in AvNav als User App integriert ist.

Der Code ist verfügbar auf GitHub.

Die Installation erfolgt in das Verzeichnis /usr/lib/avnav/plugins/ocharts. Die Daten liegen im Verzeichnis /home/pi/avnav/data/ocharts. Für das Plugin können in der avnav_server.xml noch einige weitere Konfigurationen vorgenommen werden (die meisten direkt in der UI - siehe weiter oben). Im Normalfall ist das aber nicht nötig. Falls solche Konfigurationen erfolgen sollen, müssen sie unterhalb des Plugin-Managers stattfinden.

Ab Version 20200709 kann auch das Verzeichnis für die Karten separat gesetzt werden:

```
<AVNPluginHandler>
...
<system-ocharts uploadDir="$DATADIR/charts/ocharts"/>
</AVNPluginHandler>
```

Hier wird das Verzeichnis auf /home/pi/avnav/data/charts/ocharts gesetzt (es liegt sonst auf /home/pi/avnav/data/ocharts/charts). Das kann z.B. auf dem touch image hilfreich sein, da dort nur im Verzeichnis /home/pi/avnav/data/charts ausreichend Platz verfügbar ist.

Avnav Overlays

Ab der Version 20201219 ist AvNav in der Lage, neben der eigentlichen Karte noch Zusatzinformationen anzuzeigen. Das können z.B. Wegpunkte aus einer GPX Datei aber auch vorhandene Tracks oder andere Daten (wie z.B. aktuelle Tonnen von nautin.nl) sein.

Daneben können mit dieser neuen Funktionalität auch mehrere Karten übereinander gelegt werden (z.B. eine OpenSeamap Karte, die einen weiteren Bereich abdeckt unter eine o-charts Karte - oder auch o-charts Karten für verschiedene Regionen).

Innerhalb von AvNav wird diese Funktion als Overlays bezeichnet.

Im Einzelnen können die folgenden Daten als Overlays genutzt werden:

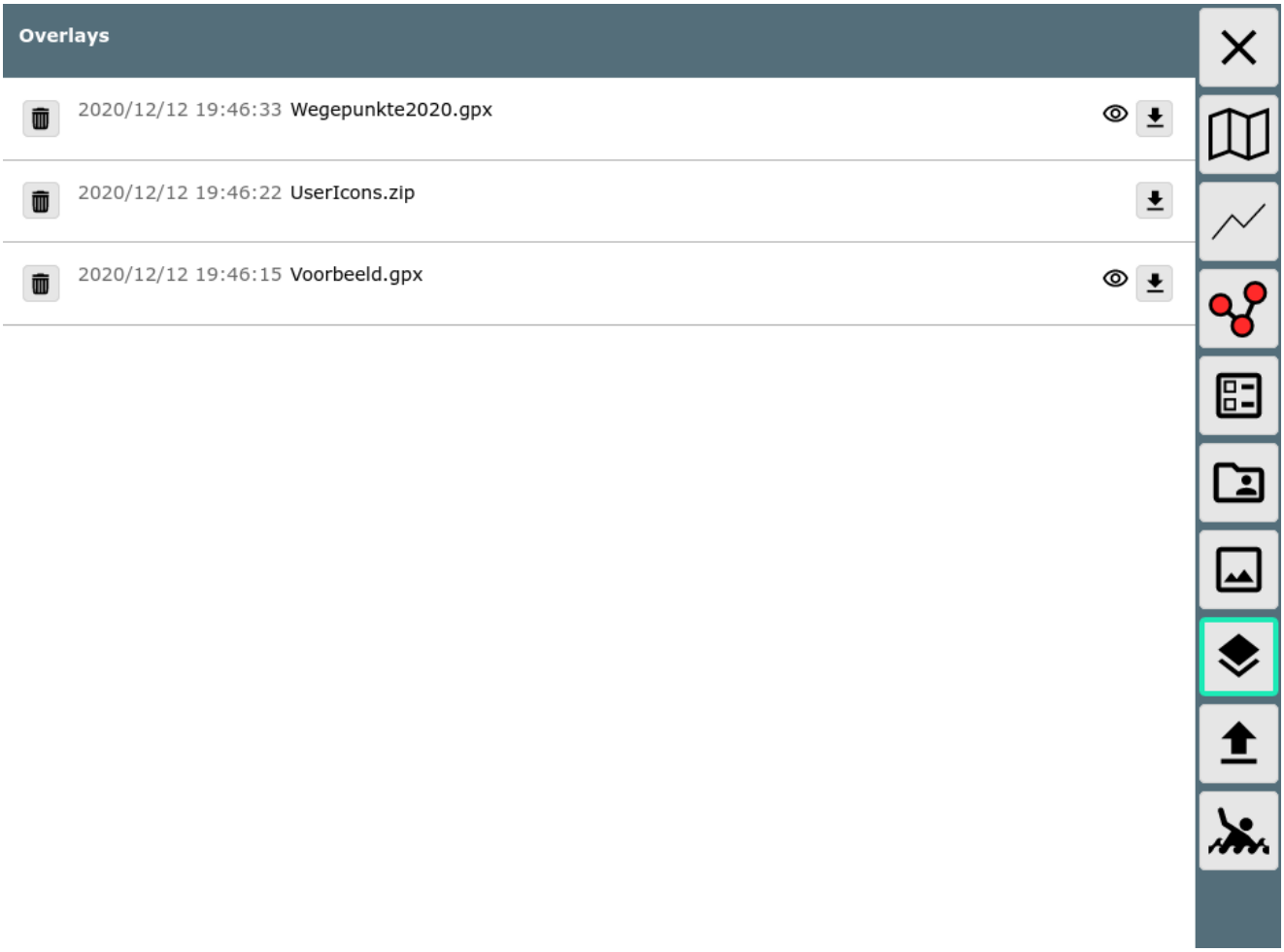
- GPX Dateien (Wegpunkte,Tracks,Routen)
- KML und KMZ Dateien (google) - ind der KMZ Datei wird eine doc.kml erwartet.
- geojson Dateien
- andere in AvNav vorhandene Karten
- AvNav Routen
- AvNav Tracks

Zu den GPX und KML Dateien können nutzerdefinierte Icons oder auch Daten, die über "link" Tags erreichbar sind, als zip Dateien gespeichert werden. KMZ Dateien enthalten diese Icons im Normalfall schon.

Wenn solche Overlays auf einer Karte vorhanden sind, können durch Klick auf entsprechende Punkte dazu Zusatzinformationen abgerufen werden.

Konfiguration

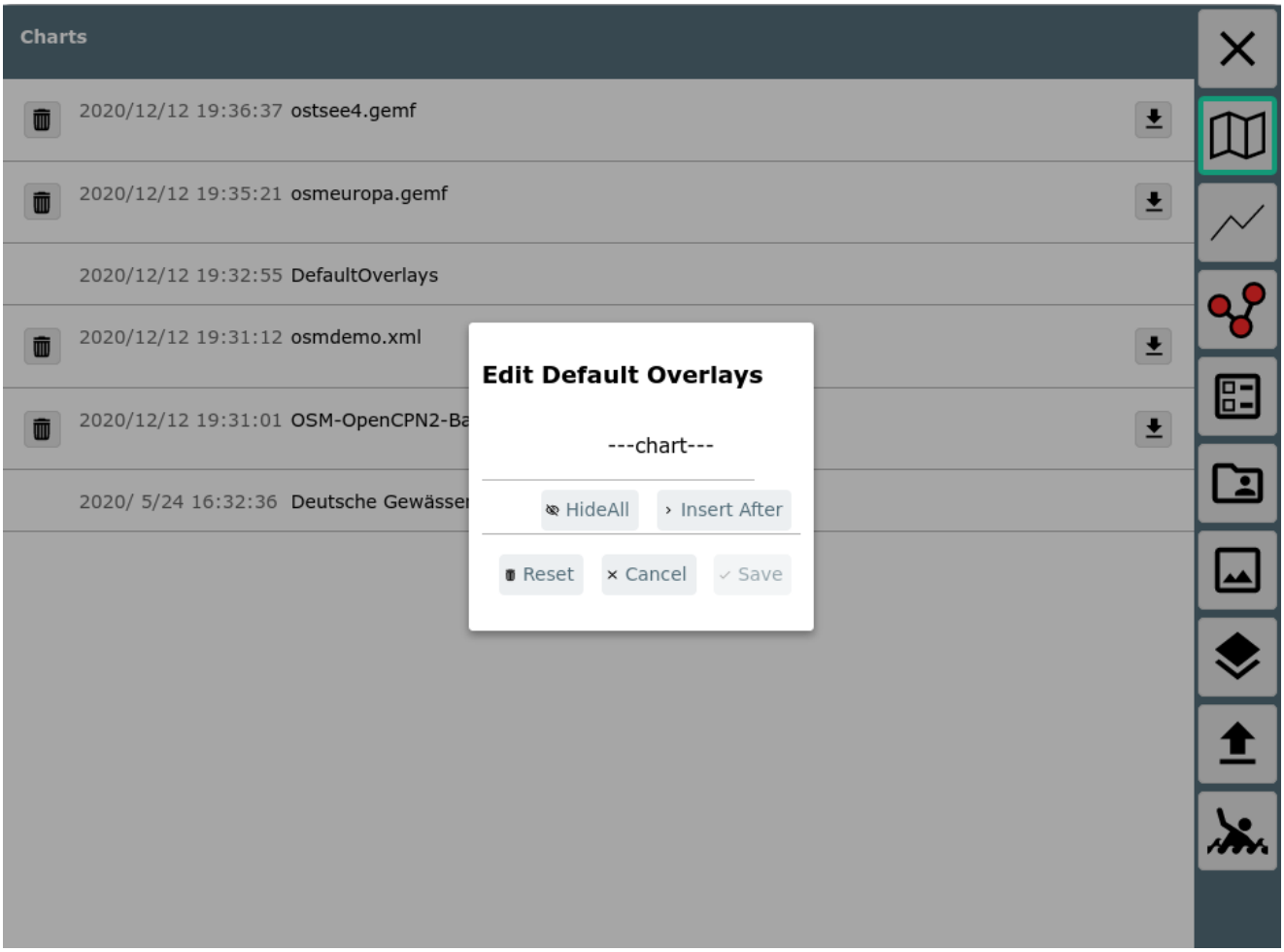
Bevor externe Overlays genutzt werden können, müssen die Dateien in AvNav hochgeladen werden.



Innerhalb von AvNav wird das Symbol für Overlays benutzt.

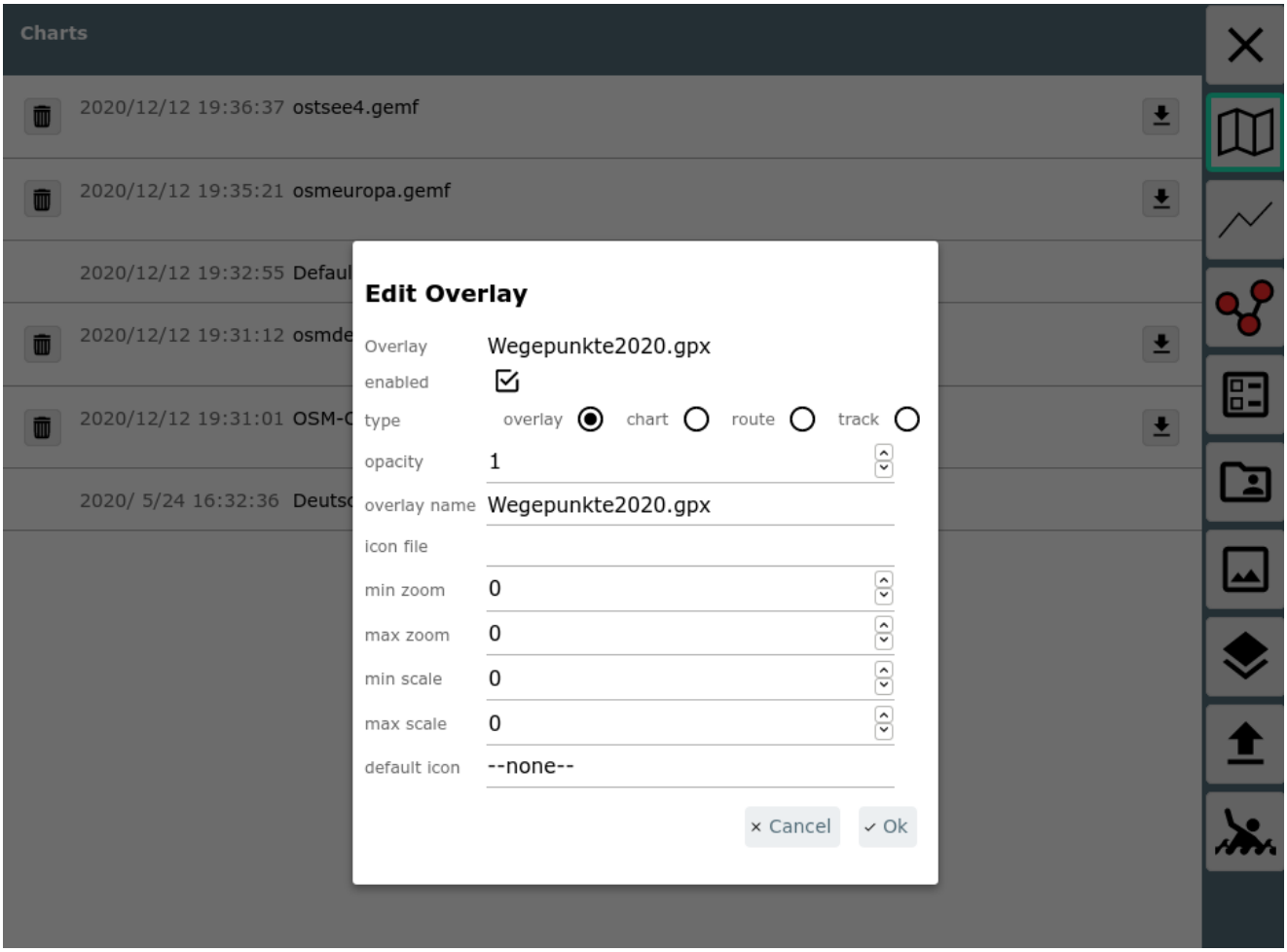
Nachdem die entsprechenden Dateien hochgeladen wurden, können diese Overlays den Karten zugeordnet werden. Man kann einen Satz von Overlays definieren, der für alle Karten genutzt werden soll (default overlays). Ausserdem kann man für jede einzelne Karte noch definieren, welche Overlays auf dieser angezeigt werden sollen.

Die Konfiguration der Overlays kann entweder auf der Files/Download-Seite im Reiter Karten erfolgen - oder auf der Hauptseite.



Im Bild ist der Dialog sichtbar, der nach Klick auf "DefaultOverlays" auf der Files/Download Seite erreichbar ist.

Hier können jetzt Overlays hinzugefügt, verändert oder gelöscht werden.

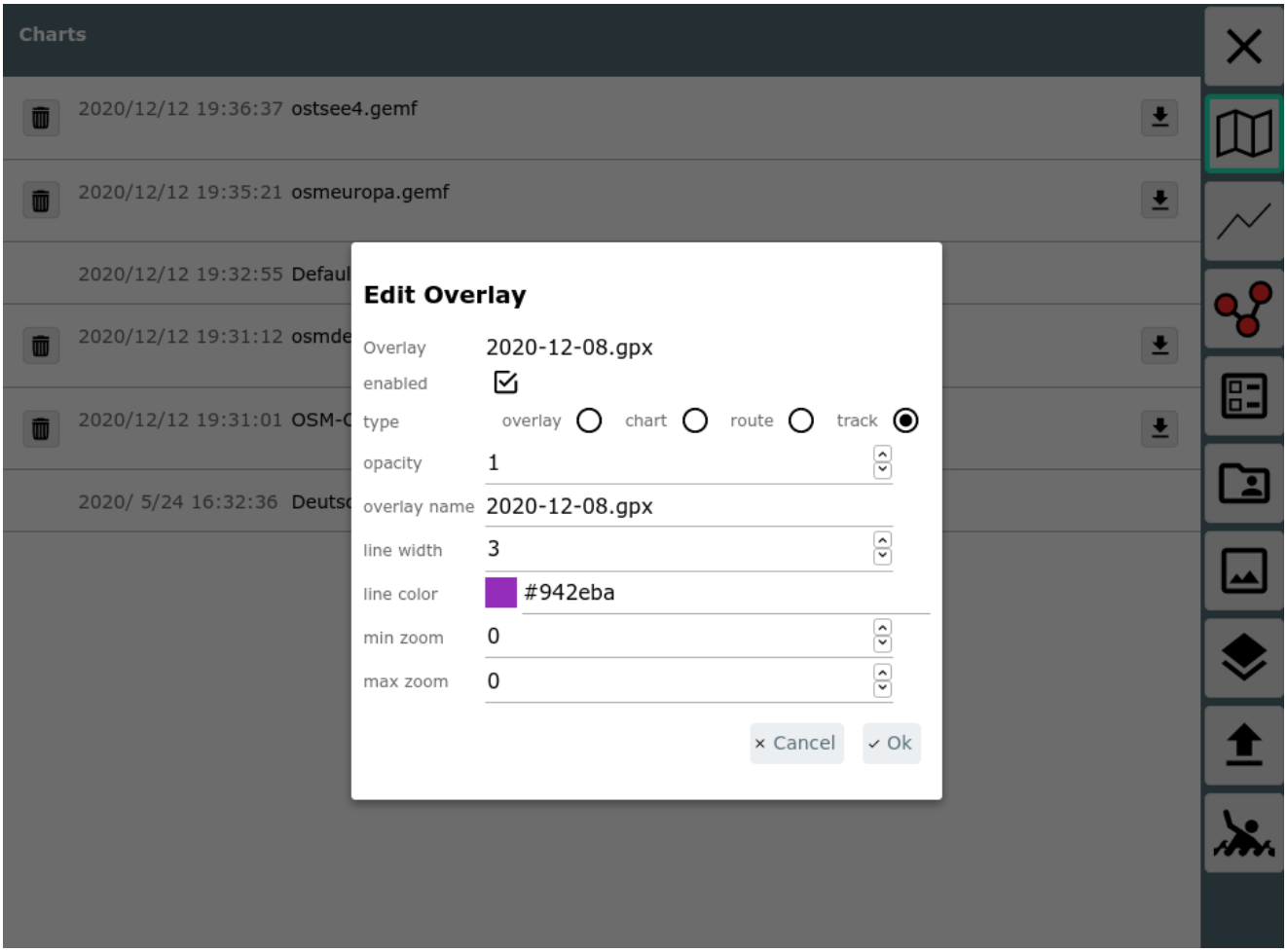


Im Bild wurde hier mit "InsertAfter" die Auswahl der vorhandenen Overlays aufgerufen.
Im Beispiel wurde die im overlays Verzeichnis vorhandene Datei Wegepunkte2020.gpx (NV Verlag) gewählt.
Es können jetzt noch Parameter für die Anzeige eingestellt werden.

Name	default	Beschreibung
enable	ja	Wenn das auf nein gesetzt wird, ist das Overlay zwar konfiguriert - wird aber aktuell nicht angezeigt. Man hat aber später die Möglichkeit, das mit einem Klick anzuzeigen.
type	overlay	Art des Overlays. overlay: eine Datei, die unter overlays hochgeladen wurde chart: eine in AvNav vorhandene Karte route: eine in AvNav vorhandene Route track: ein in AvNav vorhander Track
overlay name	---	hier wird je nach type die Liste der vorhandenen Elemente zur Auswahl angeboten

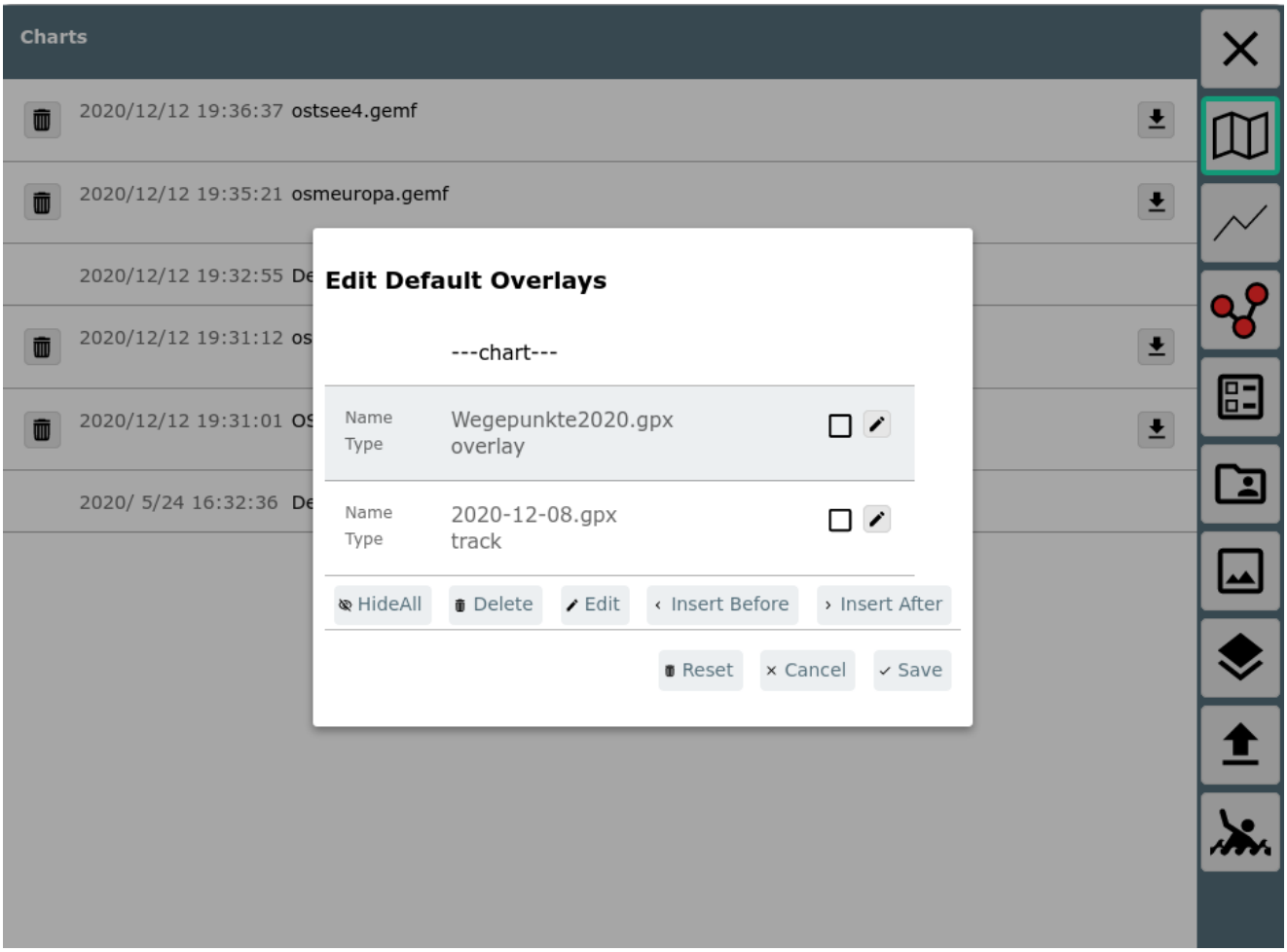
featureFormatter	---	Auswahl einer Java Script Funktion zum Aufbereiten von zusätzlichen Informationen bei Klick auf das Element
opacity	1	Ein Wert zwischen 0...1, der die Farbdeckung angibt (0: unsichtbar)
user icons	---	Hier kann eine zip-Datei angegeben werden, die die Bilder (oder sonstigen Inhalte) entält. Wenn in einer gpx-Datei eine "sym" Eigenschaft vorhanden ist (z.B. sym="b1"), wird in der Zip Datei zur Darstellung eine Datei b1.png gesucht. Wenn eine link Eigenschaft vorhanden ist (z.B. link="data/1.html") dann wird in der Zip Datei nach einem Eintrag data/1.html gesucht. Für KML Dateien gilt das sinngemäss. Allerdings empfiehlt sich hierbei eher die Nutzung des KMZ Formates.
min zoom	0	der minimale Karten-Zoom bei dem das Overlay noch angezeigt wird (0: immer)
max zoom	0	der maximale Karten-Zoom bei dem das Overlay noch angezeigt wird (0: immer)
min scale	0	wenn der Kartenzoom kleiner als min scale ist, werden die Icons verkleinert (0: Funktion deaktiviert)
max scale	0	wenn der Kartenzoom grösser als max scale ist, werden die Icons vergrößert (0: Funktion deaktiviert)
default icon	--none--	Hier kann eine Bild-Datein (png, svg) ausgewählt werden, die als Icon genutzt werden soll, wenn kein anderes Icon gefunden wurde. Die Bild-Datei muss vorher bei layouts (oder user Dateien oder Images) hochgeladen worden sein. Mit --DefaultGpxIcon-- kann ein Icon gewählt werden, das AvNav bereits mitbringt. 

Falls die Datei andere Inhalte hat als Wegpunkte, werden ggf. noch weitere Auswahlmöglichkeiten angeboten.



Name	default	Beschreibung
line width	track/route line width	Die Linienstärke für die Darstellung
line color	track/route color	Die Farbe für die Darstellung

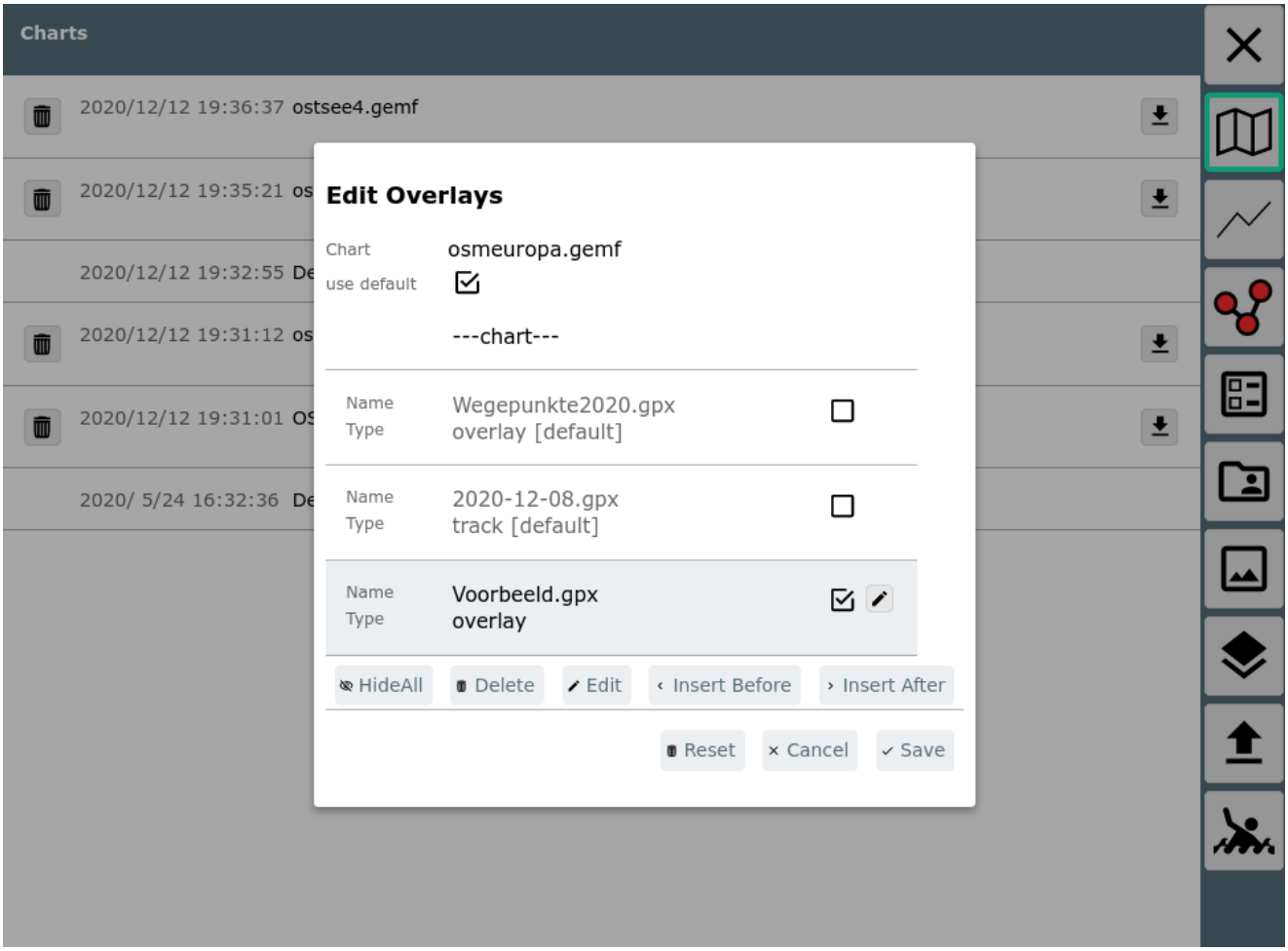
Nach Bestätigung mit Ok ist das Overlay jetzt hinzugefügt.



Die Reihenfolge der Overlays kann in der Liste per drag&drop geändert werden - sie können auch "unterhalb" der Karten angezeigt werden - dann vor die ---chart--- Reihe verschieben.

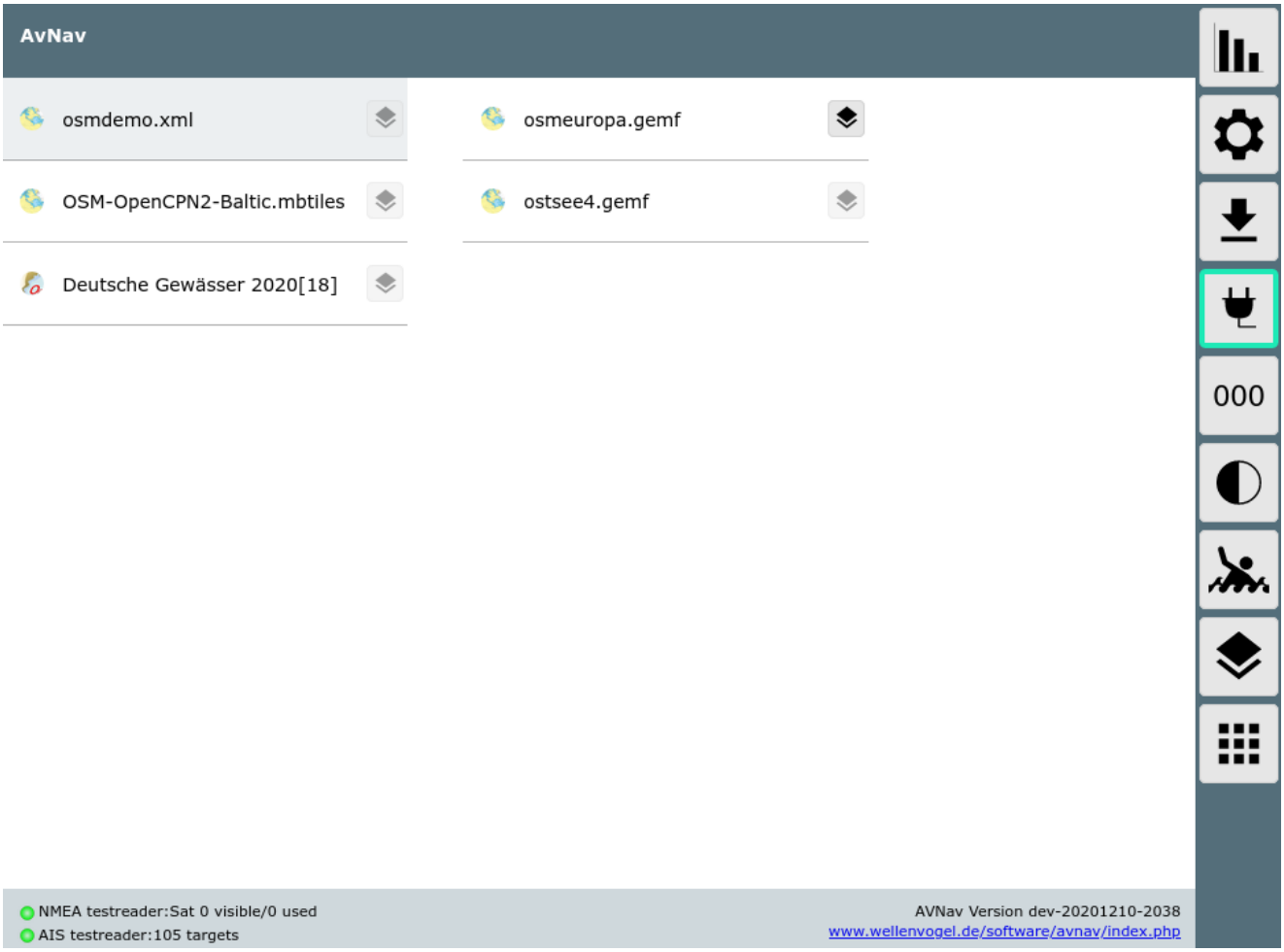
Für die default overlays macht es u.U. Sinn, diese zunächst disabled zu haben - und dann erst bei den einzelnen Karten (oder wenn sie benötigt werden) anzuschalten.

Für Karten sieht die Overlay Konfiguration dann so aus:



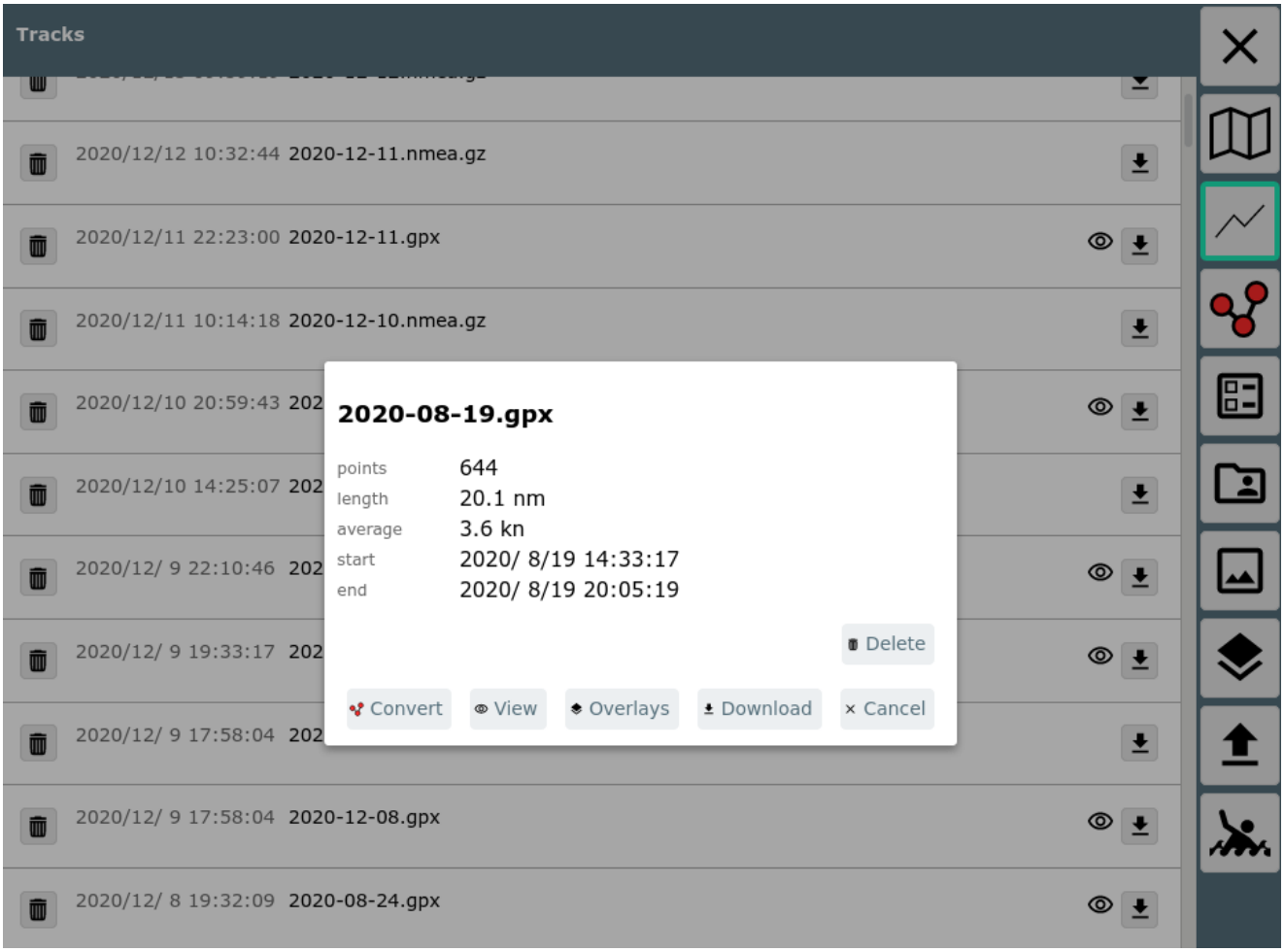
Hier können die unter default konfigurierten Overlays nur verschoben und an- bzw. abgeschaltet werden, die Overlays nur für diese Karte können bearbeitet werden.

Die Overlay-Konfiguration kann auch auf der Hauptseite durch Klick auf den Button neben der jeweiligen Karte erreicht werden.

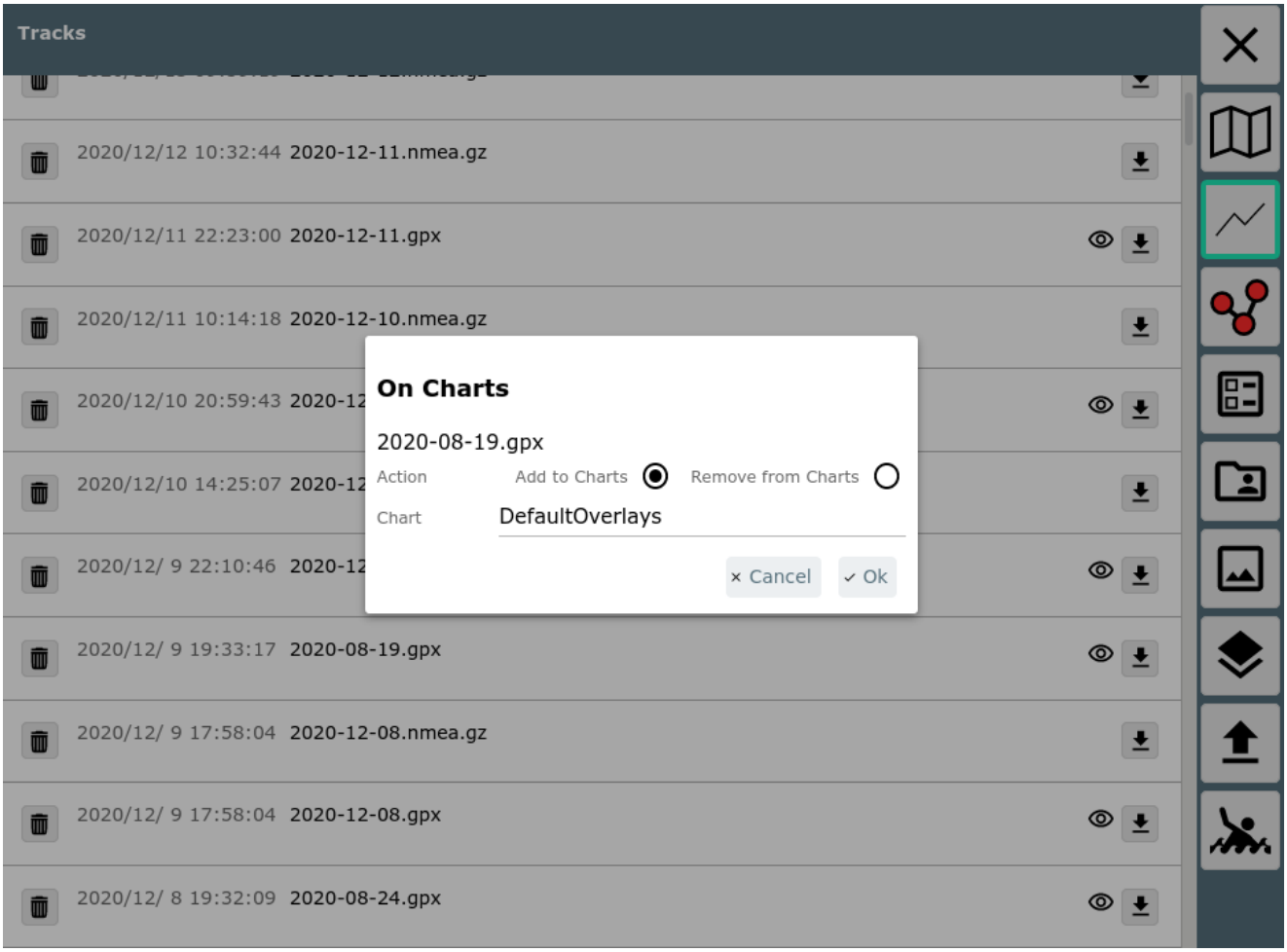


Der Button auf der rechten Seite führt zur Konfiguration der default overlays.

Für Tracks und Routen kann man die Overlay Konfiguration auch von der Files/Download Seite starten, indem man eine Route oder einen gpx Track auswählt.



Im Bild wurde auf einen gpx Track geklickt. Durch Auswahl von "Overlays" kann man diesen Track jetzt overlays zuordnen (oder ihn entfernen).

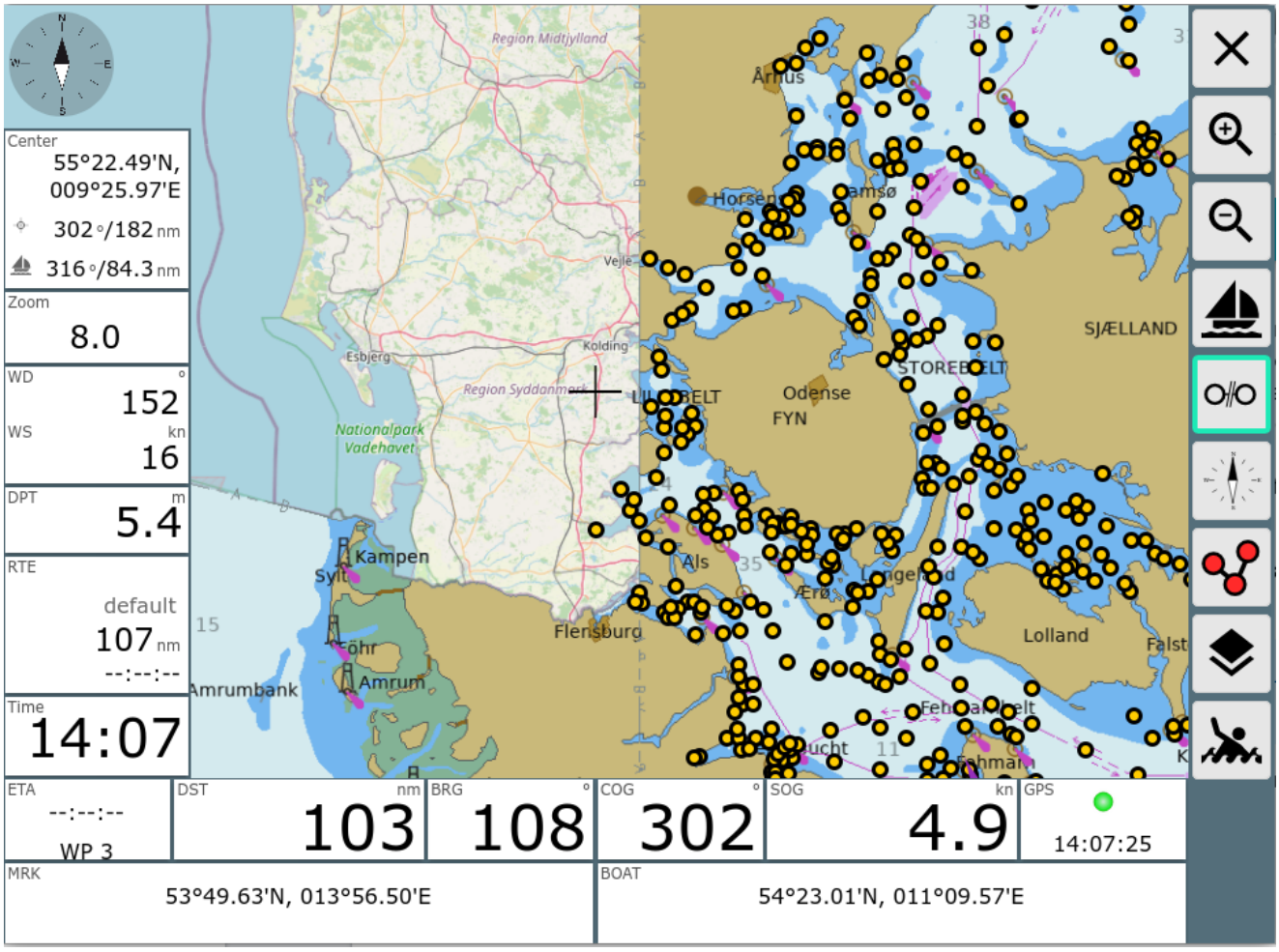


Nachdem man sich entschieden hat, ob man den Track den default overlays oder einer spezifischen Karte zuordnen möchte, kommt man zur normalen Overlay Konfiguration.

Wenn man "Remove from Charts" auswählt, wird der Track von allen overlays entfernt.

Nutzung

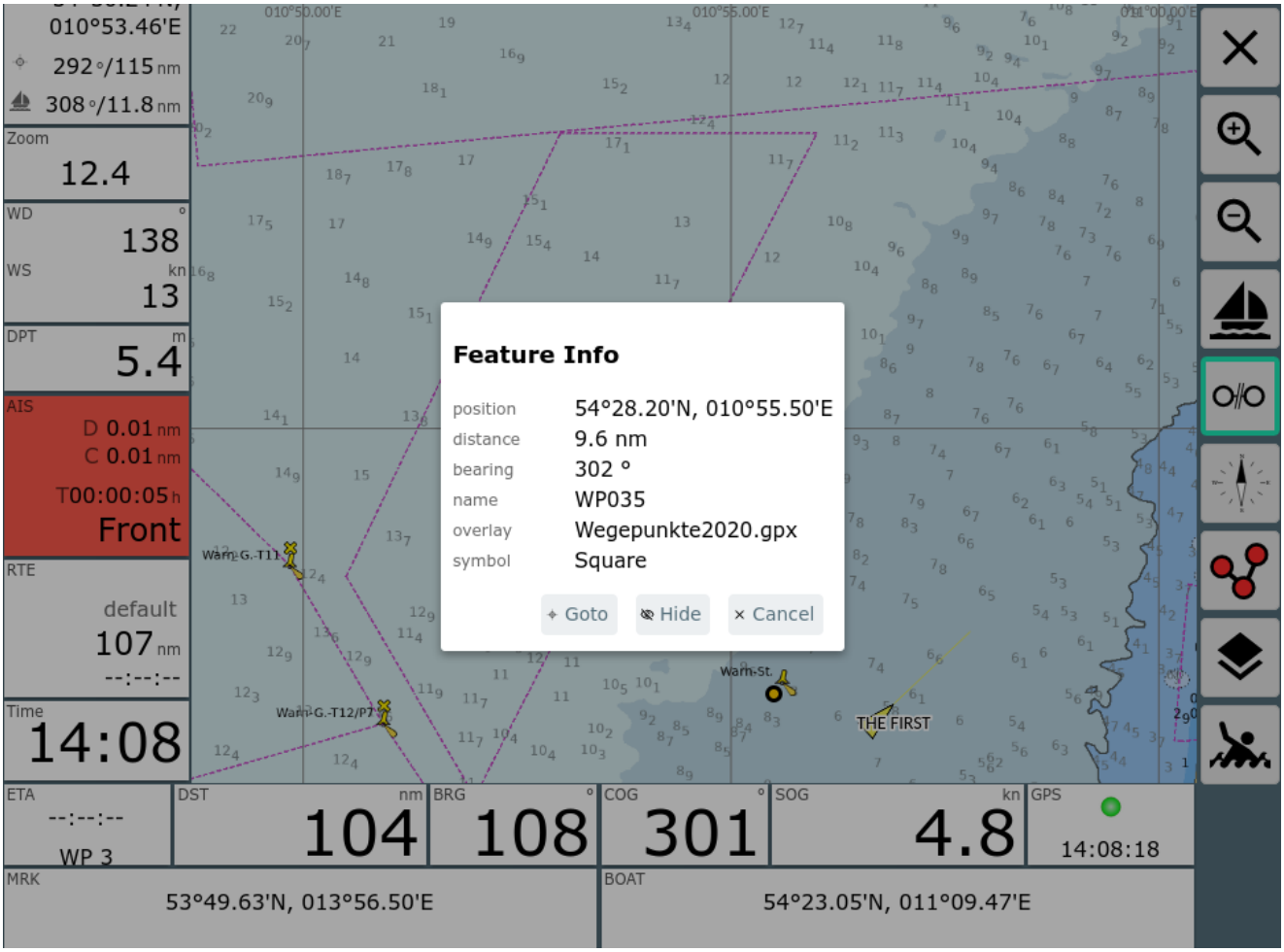
Wenn eine Karte ausgewählt wird, wird sie zusammen mit ihren Overlays dargestellt.



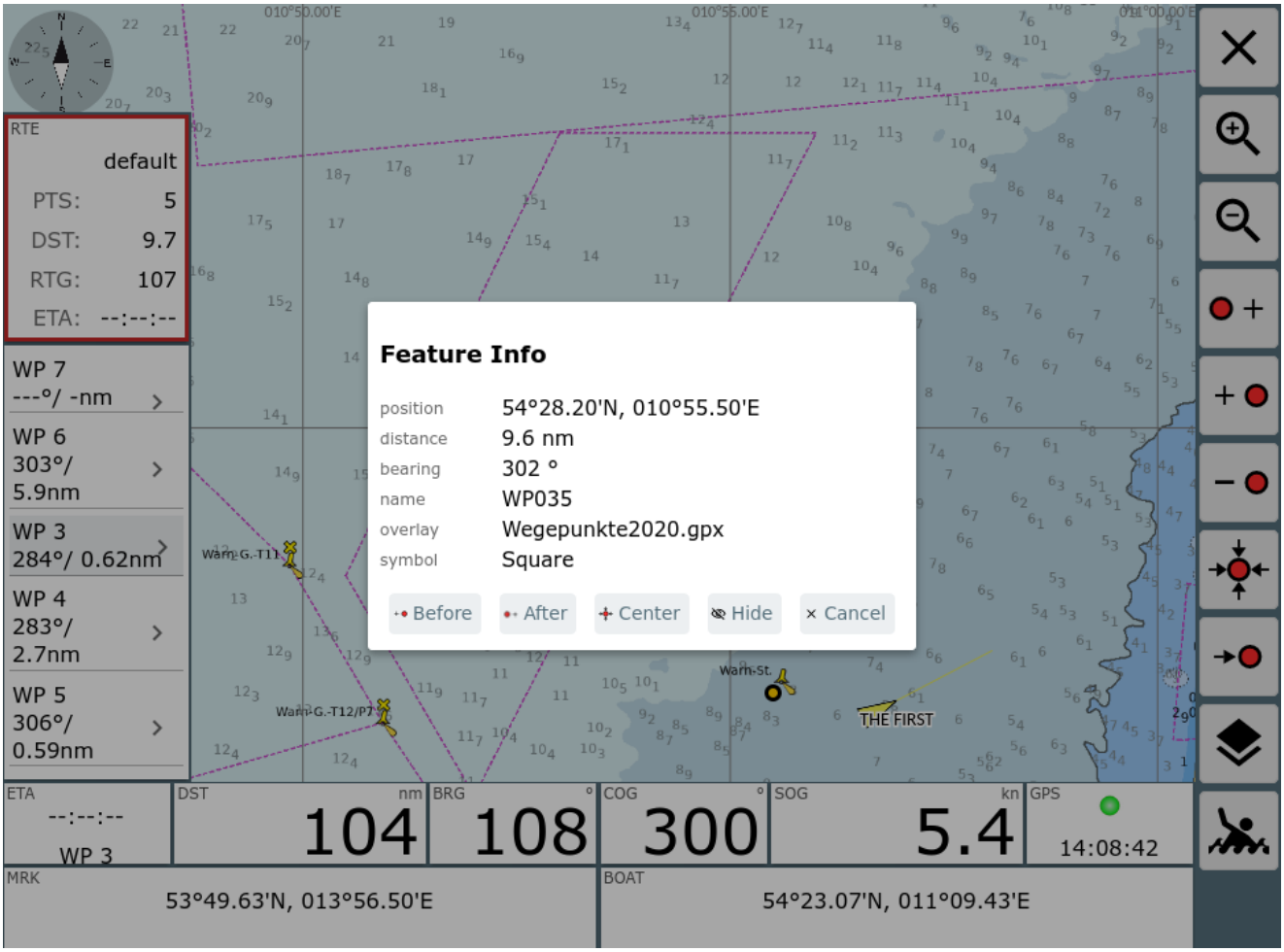
Im Beispiel eine o-charts Karte mit einer OpenStreetMap Karte darunter und einer Wegpunkt-Datei darüber.

Als Icon wurde das DefaultGpxIcon gewählt.

Wenn auf eines der Icons aus dem Overlay geklickt wird, erscheint ein Info Dialog.

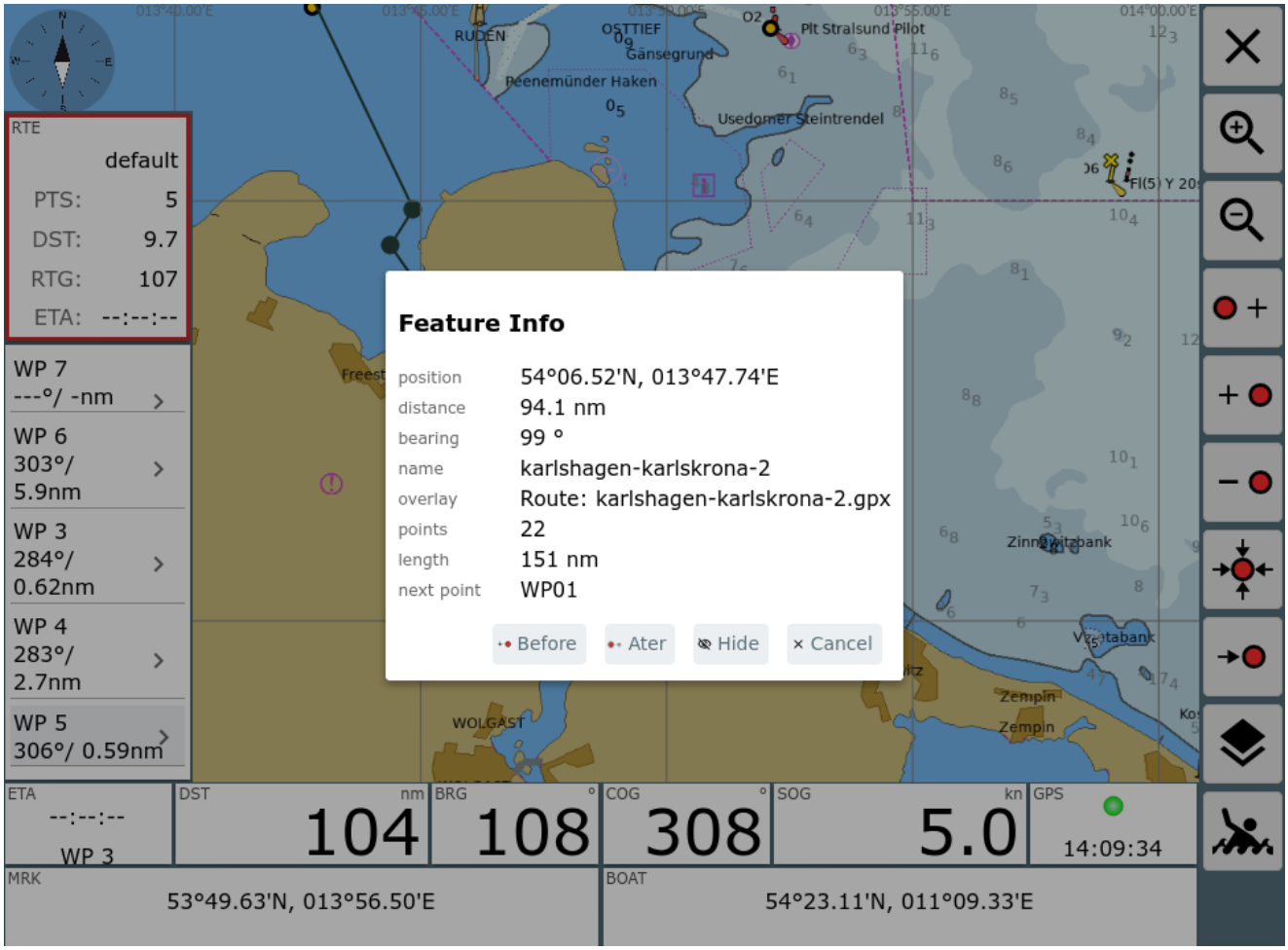


Man erhält die zum Punkt verfügbaren Informationen und die Möglichkeit z.B. diesen Punkt sofort als Zielpunkt zum Routing zu verwenden (Goto).
Über "Hide" kann das gesamte Overlay temporär ausgeblendet werden.
Wenn man sich momentan im Routen-Editor befindet, stehen noch weitere Funktionen zur Verfügung.



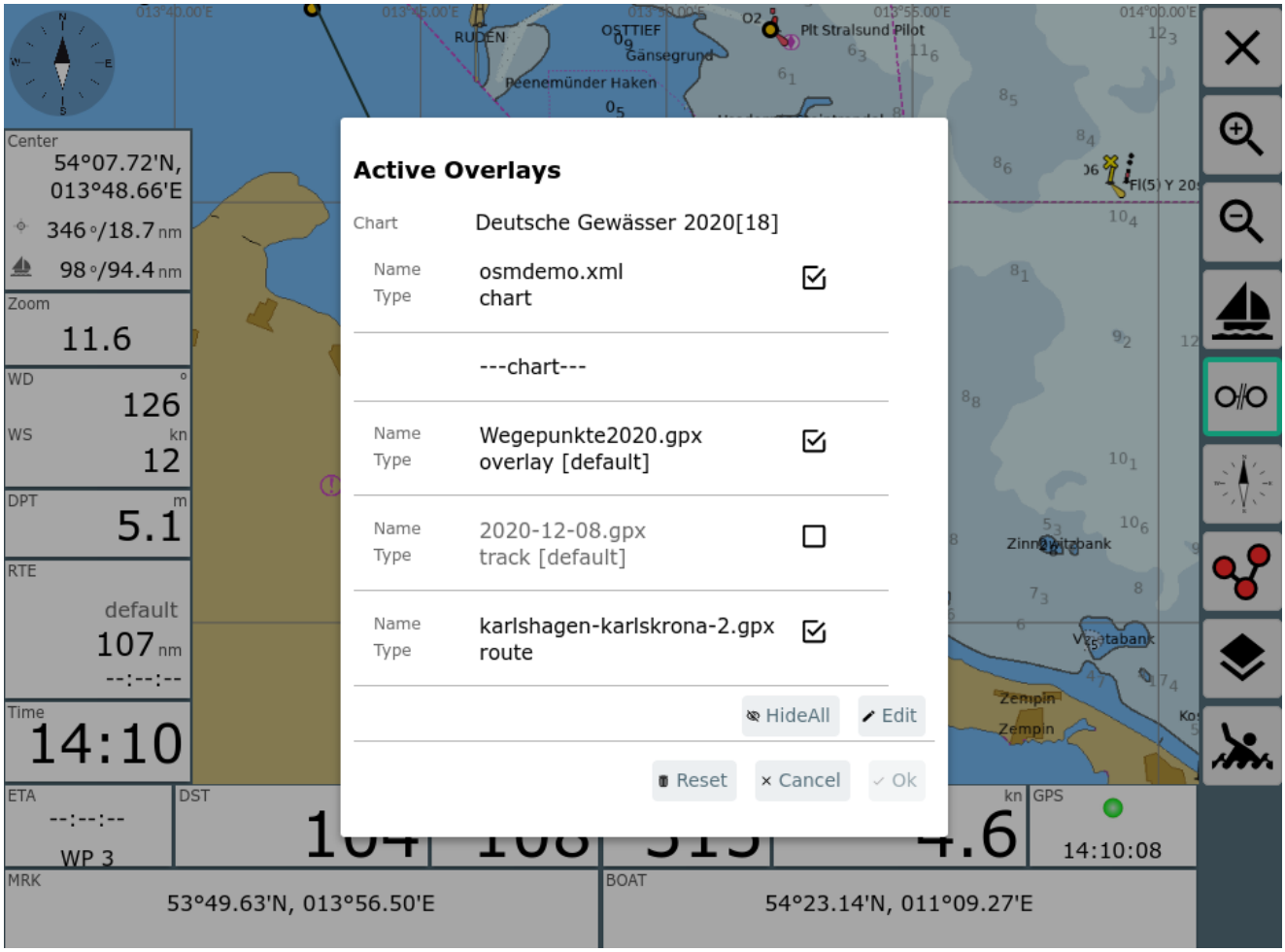
Der gewählte Punkt kann zur Route hinzugefügt werden (vor oder hinter dem aktuellen Punkt der Route) oder der aktuelle Punkt der Route kann auf diesen Wegpunkt verschoben werden.

Wenn es sich beim Overlay um eine Route handelt, kann diese Route (ab dem gewählten Punkt) zur aktuellen Route hinzugefügt werden.



Auf der Navigationsseite und im Route-Editor gibt es die Möglichkeit, Overlays aus- und einzublenden.

Dazu auf der rechten Seite den Button verwenden.



Hier kann man kurzfristig Overlays ein- und ausschalten. Diese Einstellungen bleiben wirksam, bis die gewählte Karte gewechselt wird. Sie betreffen auch nur das aktuelle Gerät, auf dem man arbeitet. Alle anderen Veränderungen an den Overlays finden immer auf dem Server statt und werden somit für alle Nutzer wirksam.

Über Edit kann man wieder zur normalen Bearbeitung der Overlays wechseln.

Hinweis:

Wenn man Overlay-Dateien mit sehr vielen Elementen nutzt, kann das die Arbeitsgeschwindigkeit stark reduzieren. Momentan ist dort kein festes Limit eingebaut, Dateien von einigen 100MB werden allerdings mit Sicherheit Probleme bereiten. 1000 Punkte in einem Overlay sind dagegen kein Problem.

Änderungen von Overlays

Ab Version 20220225 überwacht AvNav die Overlay Dateien und sorgt dafür, das die Kartenansicht neu aufgebaut wird, wenn sich ein Overlay ändert.

Anpassungen

Ab Version 20210114 kann man die Informationen anpassen, die aus der Overlay-Datei gewonnen werden und die dann bei "Feature Info" angezeigt werden. Das kann sinnvoll sein, da manchmal verfügbare Overlay Dateien Informationen enthalten, die AvNav von sich aus noch nicht kennt.

Zu diesem Zweck kann man beim Konfigurieren des Overlays einen "featureFormatter" angeben. Dabei handelt es sich um eine Java Script Funktion, die entweder in AvNav schon vorhanden ist, die in der user.js eingebaut wird, oder die von einem plugin kommt.

Diese Java Script Funktion bekommt als Parameter die in der Overlay Datei vorhandenen Eigenschaften des angeklickten Punktes und kann veränderte/neue Eigenschaften zurückgeben, die dann vom FeatureInfo Dialog angezeigt werden.

Die folgenden Eigenschaften können zurückgegeben werden:

Name	Bedeutung
sym	die URL für ein anzuzeigendes Icon. Das kann eine relative URL sein, diese ist dann eine Icon Datei innerhalb der konfigurierten userIcons Datei, ein absoluter Pfad wie z.B. /user/images/myImage.png oder eine mit http: beginnende externe URL (natürlich dann nur mit Internet Verbindung nutzbar).
name	der anzuzeigende Name
desc	der unter "description" anzuzeigende Text
htmlInfo	ein html String, der dann bei Klick auf den ⓘ Info Button angezeigt wird.
time	eine Zeitangabe (String oder java script Date)
link	eine URL, die bei Klick auf den ⓘ Info Button angezeigt werden soll (alternativ zu htmlInfo). Es gelten die gleichen Regeln wie für "sym".

Die übergebenen Parameter hängen von der Overlay Datei ab. Zusätzlich sind in jedem Falle die Werte "lat" und "lon" vorhanden.

Ein Beispiel für die eingebaute genericHtmlInfo Funktion, die alle vorhandenen Werte als HTML in den Wert htmlInfo schreibt.

```
let genericHtmlInfo=function(properties,extended){
    if (! extended) return {};
    let htmlInfo='<div class="featureInfoHtml">';
```

```
        for (let k in properties){
            if (!properties[k]) continue;
            if (htmlInfo !== "") htmlInfo+="<br/>";

    htmlInfo+=avnav.api.escapeHtml(k)+"="+avnav.api.escapeHtml(properties[k
        }
        htmlInfo+='</div>';
        return {htmlInfo:htmlInfo};
    }
```

Der zweite Parameter, der an die Funktion übergeben wird, gibt einen Hinweis, ob alle Parameter erzeugt werden sollen oder nur der "sym" Parameter. Falls extended auf "false" gesetzt ist, sollte die Funktion keine zeitraubenden Operationen ausführen, da sie potentiell für jedes Element aus dem Overlay aufgerufen wird.

Um selbst eine solche Funktion bei AvNav bekannt zu machen, muss sie wie folgt registriert werden (siehe user.js).

```
avnav.api.registerFeatureFormatter('myHtmlInfo',myHtmlInfoFunction);
```

Nachdem die Funktion registriert wurde, kann sie in der Overlay Konfiguration ausgewählt werden.

AvNav WebApp Beschreibung

Die WebApp kann man im Normalfall über den Link http://avnav.avnav.de oder http://avnav.local erreichen, wenn man sich im WLAN des Raspberry befindet. Für einige detaillierte Hinweise zur Erreichbarkeit siehe die Image Beschreibung.

Unter Android ist der Aufruf direkt in der App integriert.
Alternativ kann man auch auf seinem Mobilgerät einen Bonjour-Browser installieren - dann findet man den avnav Server ganz ohne URL-Eingaben.
Die passenden Apps dazu:

- IOS: 
- Android: 

Die WebApp gliedert sich in eine Reihe von Seiten die man entweder direkt von der Hauptseite - oder teilweise auch über bestimmte Zwischenseiten erreicht.
Über die Links in der 2. Spalte der Tabelle sind die Beschreibungen der einzelnen Seiten erreichbar.

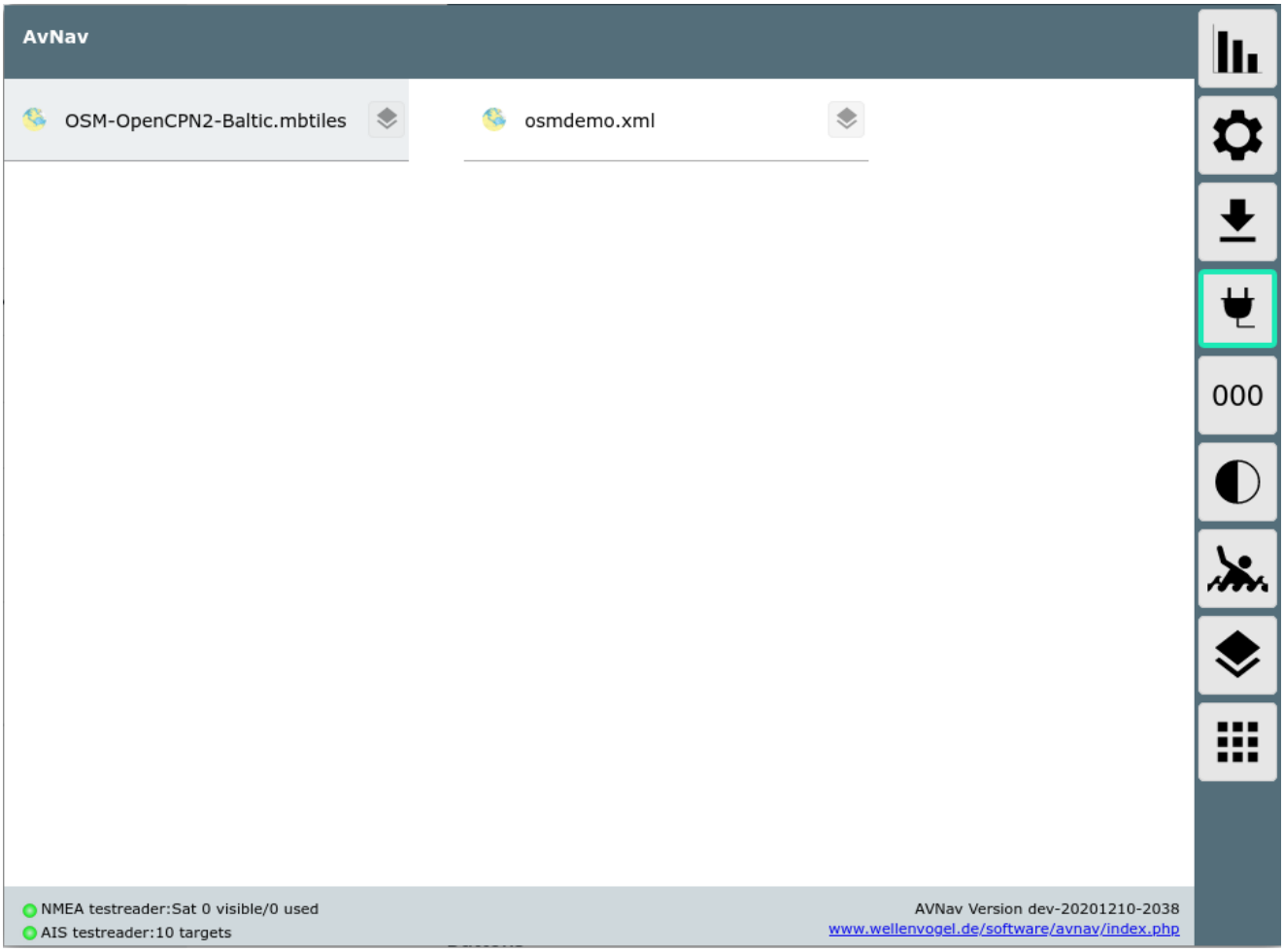
Icon	Seite	Erreichbar	Funktion
	Hauptseite	Direkt nach dem Start	Anzeige der Liste der Karten, NMEA Status, Verzweigung zu weiteren Seiten
	Navigationsseite	Klick auf eine Karte auf der Hauptseite	Basis-Navigationsfunktion, Karten und Instrumenten-Anzeige, Wegepunkte, Routen...
	Server/Status Seite	Button auf der Hauptseite	Status-Anzeige für den Server, Bearbeiten der Server Konfiguration, Weiterverzweigung zur Wifi Konfiguration, zur Anzeige der Server-Adressen, zum Herunterfahren und zur Lizenz-Info

	Einstellungen	Button auf der Hauptseite	Einstellungen für die Anzeige im Browser, von dort weiter zum Layout-Editor, zur User Apps Konfiguration und zu Android Settings (nur Android)
	Files/Download	Button auf der Hauptseite	Anzeige, Download, Upload, Bearbeiten von Karten, Tracks, Routen, Nutzerdateien, Bildern, Layouts
000	Dashboard	Button auf der Hauptseite, Klick auf bestimmte Anzeigen auf der Navigationsseite	Anzeige von Instrumentendaten (bis zu 5 Unterseiten)
	Route Editor	Button auf der Navigationsseite	Erstellen und Bearbeiten von Routen
	User Apps	Button auf der Hauptseite (nur sichtbar wenn user apps konfiguriert)	Anzeige von internen oder externen HTML-Seiten (z.B. signalK Web Interface wenn konfiguriert)
	Route Liste	Klick auf die Anzeige der aktuellen Route im Routen-Editor	Anzeige aller Wegepunkte einer Route, Kopierfunktion, Bearbeitung, Auswahl gespeicherter Routen
	Wifi Konfiguration	Button auf der Statusseite ()	Verbinden zu einem externen WLAN (nur wenn konfiguriert und WLAN Stick gesteckt ist, nicht Android)
	Anzeige der Serveradressen	Button auf der Statusseite ()	Anzeige der aktuellen Serveradressen mit QR-Code zum einfachen Scannen mit einem anderen Gerät
	Layout Editor	Button auf der Einstellungsseite ()	Bearbeiten der Anzeigen auf der Navigationsseite und auf den Dashboard Seiten

	User App Konfiguration	Button auf der Einstellungsseite ()	Definieren von externen oder internen HTML Seiten, die als User Apps angezeigt werden sollen
	Ais Info	Klick auf die AIS Anzeige auf der Navigationsseite oder ein AIS Ziel auf der Karte	Anzeige der Informationen zu einem AIS Ziel. Von dort weiter zur AIS Liste.
	Ais Liste	Über die Ais Info Seite, Button	Anzeige der Liste der AIS Ziele in der Umgebung, sortierbar

AvNav Hauptseite

Überblick



Buttons

Icon	Name	Beschreibung
	ShowStatus	Status-Seite für den Server

	ShowSettings	Einstellungen
	ShowDownload	Files/Downloads zum Herunterladen und Hochladen von Tracks, Routen, Karten, Layouts, User Files und Images
	Connected	Wenn aktiv wird die Navigation (Wegepunkte, Routen) auch auf dem Server aktiviert, sonst nur lokal
000	ShowGps	Anzeige des Dashboards
	Night	Nachtmodus aktivieren
	MOB	Mann über Bord (nur sichtbar wenn connected)
	NavOverlays	Bearbeitung der Default Overlays
	FullScreen	Fullscreen ein/aus (nur auf unterstützten Browsern)
	MainAddOns	Anzeige von konfigurierten User Apps (z.B. signalK)
	RemoteChannel	Erlaubt den Fernsteuerungskanal und -Modus zu wechseln
	Split	Schaltet den Split Mode ein oder aus

Im Hauptbereich der Seite befindet sich die Liste der auf dem Server gefundenen Kartensätze (beim Raspi-Server files unter /home/pi/avnav/data/charts, in der Android App unter charts im gewählten Verzeichnis).

Nach der ersten Installation sind hier einige Online Demo Karten sichtbar. Diese können nur mit Internet-Verbindung genutzt werden.

Weitere Karten kann man über die Files/Download Seite hochladen bzw. direkt in das entsprechende Verzeichnis kopieren (raspberry) bzw. aus einem externen Verzeichnis lesen (Android).

AvNav kann Karten im gemf Format lesen (bevorzugt), ab Version 202003xx auch im mbtiles Format. Ausserdem kann auch eine Online-Karten-Quelle über ein XML File eingebunden werden. Details dazu unter Kartenformate.

O-Charts Karten müssen über das o-charts Plugin (zu erreichen über den Button) hochgeladen werden.

Wenn in SignalK Karten installiert sind, werden diese hier ebenfalls angezeigt.

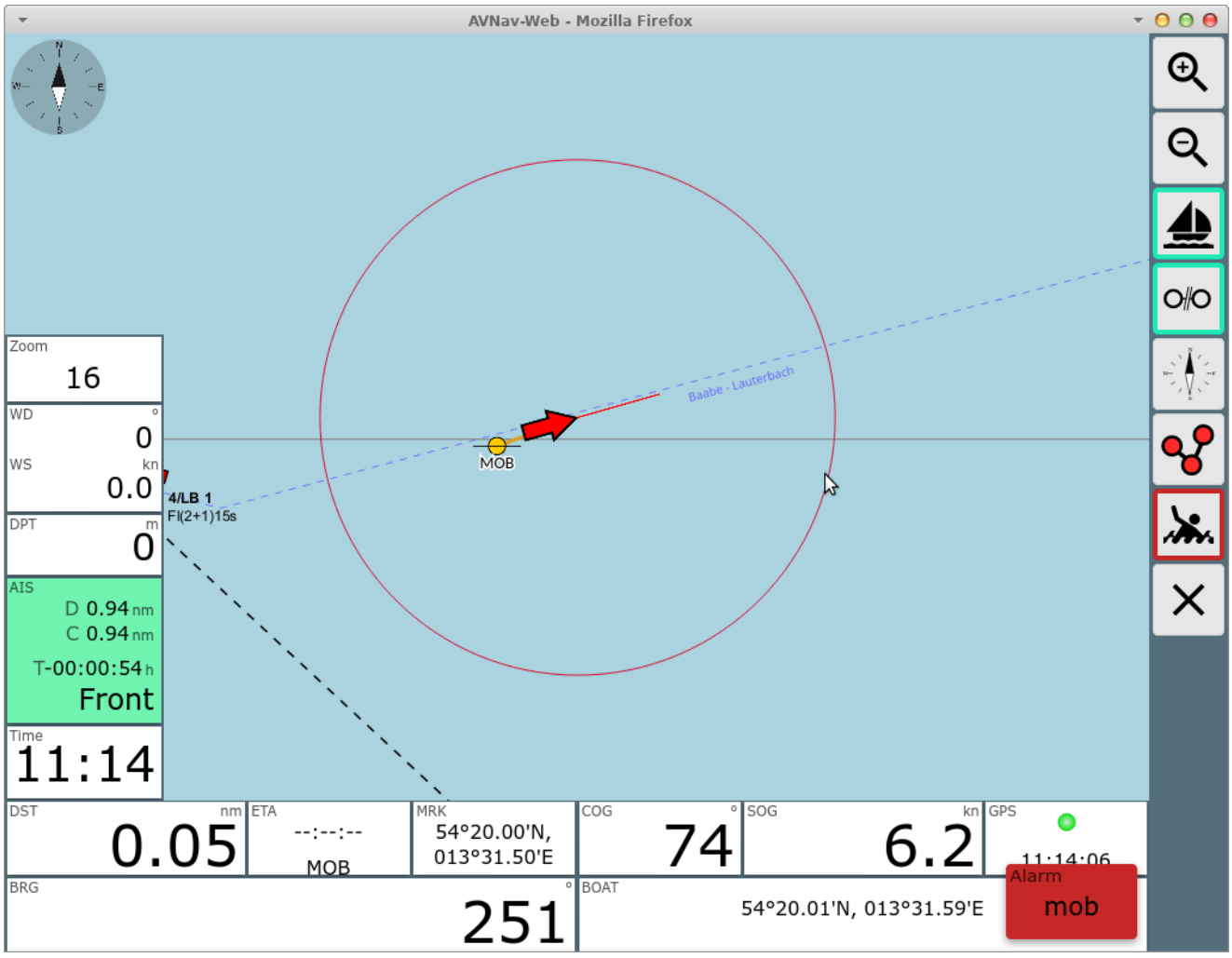
Über den Button neben jeder Karte kann man die Overlays für diese Karte bearbeiten.

Durch Anklicken eines Eintrages in der Kartenliste gelang man zur Navigationsseite mit dem enstprechenden Kartensatz.

Spezielle Funktionen

Mann über Bord

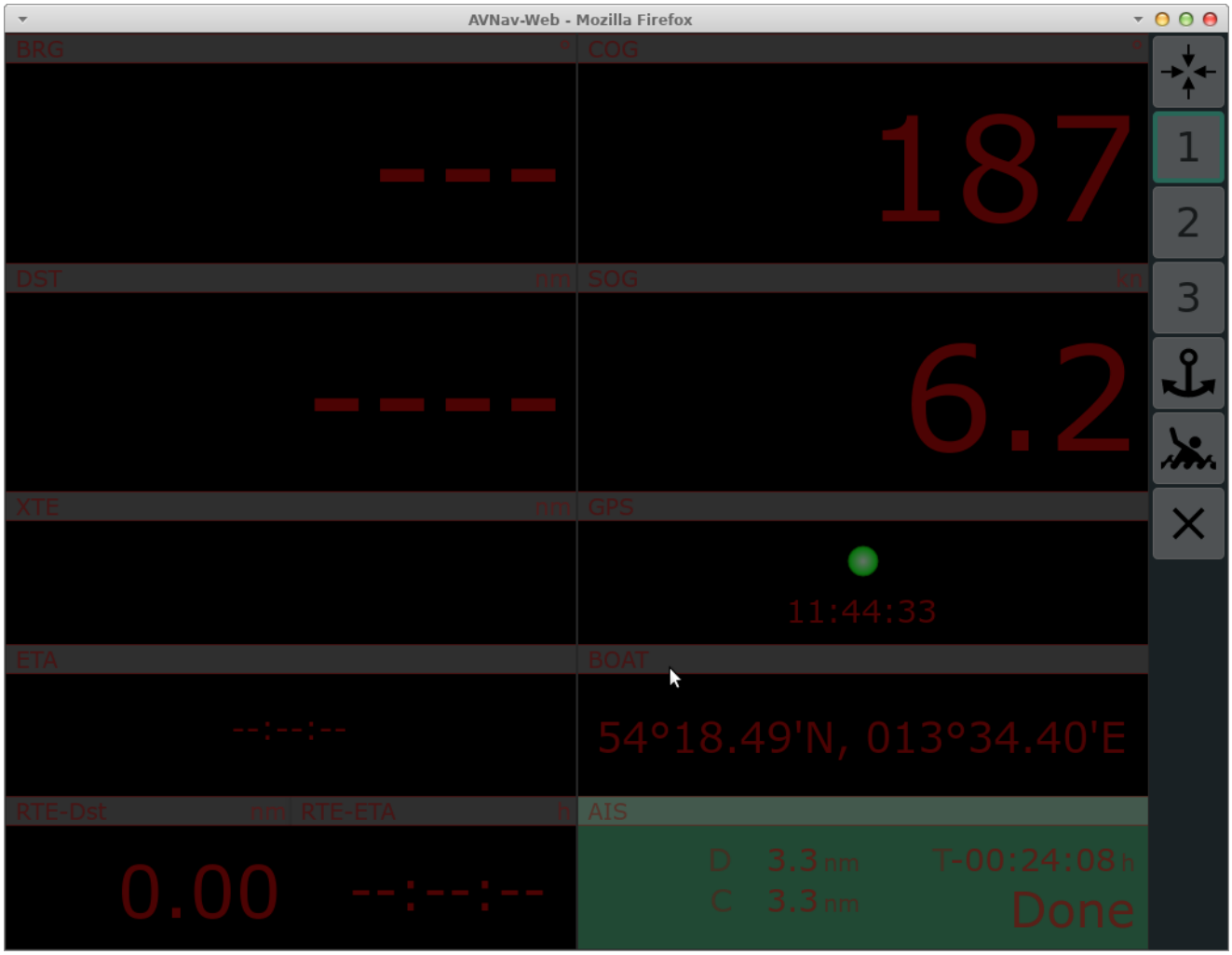
Sichtbar auf allen Seiten, aber nur wenn der "connected" Mode aktiv ist - Button ist grün.
Durch Klick wird die aktuelle Position zu einem Wegepunkt mit dem Namen "MOB", ein aktuelles Routing wird abgebrochen, ein Routing zum Wegepunkt wird aktiviert, und es wird auf die Navigationsseite mit der zuletzt gewählten Karte gewechselt. Die Karte wird auf das Boot zentriert, und es wird eine feste Vergrößerung eingestellt (in den Settings anpassbar). Ausserdem wird ein "MOB" Alarm ausgelöst. Dieser Alarm kann quittiert werden. Das Routing bleibt aktiv bis der MOB Button erneut gedrückt wird.



Falls noch nie eine Karte gewählt wurde, wird das Dashboard angezeigt.

Nachtmodus

Durch Klick auf den ◐ Button wird der Nachtmodus aktiviert. In den Einstellungen können die Dimm-Faktoren noch angepasst werden.



Verbunden/Nicht Verbunden (Connected)

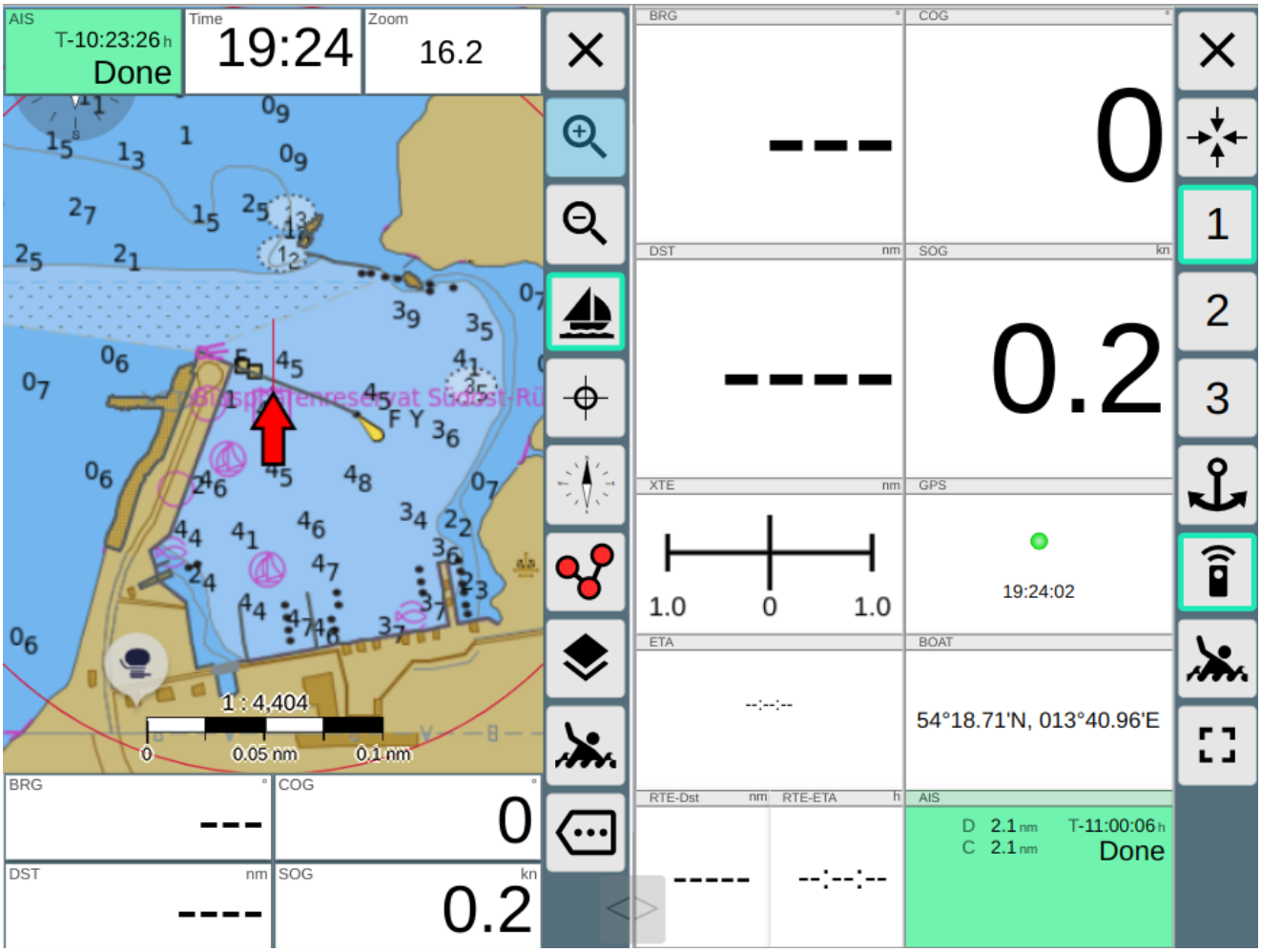
Über den Button wird gesteuert, wie die Routing Daten mit dem Server ausgetauscht werden. Wenn er aktiv ist (grüner Rand), werden alle Änderungen an den Routing-Daten (Wegepunkt, Starten, Stoppen, Routen-Änderungen) direkt mit dem Server synchronisiert und sind damit auf allen angeschlossenen Display sichtbar.

Wenn er ausgeschaltet wird, werden alle Routen-Änderungen nur lokal ausgeführt. Damit kann man z.B. eine alternative Route testen, ohne den Rudergänger oder den Autopiloten zu stören. Wenn man sich danach wieder verbindet, gewinnen im Zweifel die Daten vom Server, man muss also seine lokalen Änderungen z.B. in einer neuen Route speichern. Wenn eine solche Route dann akitiviert wird, wird sie auch zum Server hochgeladen. Auf der Files/Download Seite wird bei den Routen durch das Symbol angezeigt, ob sie mit dem Server synchronisiert sind.

Der Verbunden/Nicht Verbunden Zustand bezieht sich nur auf Routen und Wegepunkte! Alle anderen Funktionen sind immer mit dem Server verbunden (ausser MOB - das ist eine Routing Funktion - und daher nicht aktivierbar wenn man nicht verbunden ist).

Split Mode

Ab Version 20220819 kann AvNav sein Hauptfenster teilen. Dabei laufen praktisch 2 Instanzen der AvNav App nebeneinander.



Die beiden Instanzen sind weitgehend unabhängig voneinander und können jeweils getrennt bedient werden.
Sie teilen sich zu einem großen Teil die Einstellungen von AvNav.
Nur einige wenige Einstellungen sind jeweils spezifisch für die Seite (rechts/links). Diese werden beim Aufruf der Einstellungsseite mit einem * gekennzeichnet.
Wenn man Einstellungen ändert (ausser die mit dem *) werden die Änderungen auch sofort auf der anderen Seite wirksam.

Die folgenden Einstellungen sind jeweils spezifisch pro Seite:

Name	Bedeutung

connectedMode	Der "Stecker Button" - wenn der ausgeschaltet ist, kann die Instanz keine Änderungen auf dem Server vornehmen. Man kann hier z.B. eine eigene Route testen.
layoutName	Das zu verwendende Layout
remoteChannelName	Der Name für den Fernsteuerungskanal
remoteChannelRead	Fenrsteuerung lesen
remoteChannelWrite	Fernsteuerung schreiben

Falls man andere/weitere Einstellungen spezifisch je Seite vornehmen möchte, kann man eine Datei splitkeys.json im User-Verzeichnis anlegen. Die default Datei findet man auf GitHub. Die Namen für die keys muss man dem Source Code im Abschnitt properties entnehmen. Alle in dieser Datei aufgelisteten Eigenschaften sind dann spezifisch pro Seite.
Die pro Seite spezifisch eingestellten Eigenschaften werden beim Beenden des Split Modes nicht für die dann laufende Instanz übernommen (sind aber für den nächsten Split Mode Aufruf erhalten).

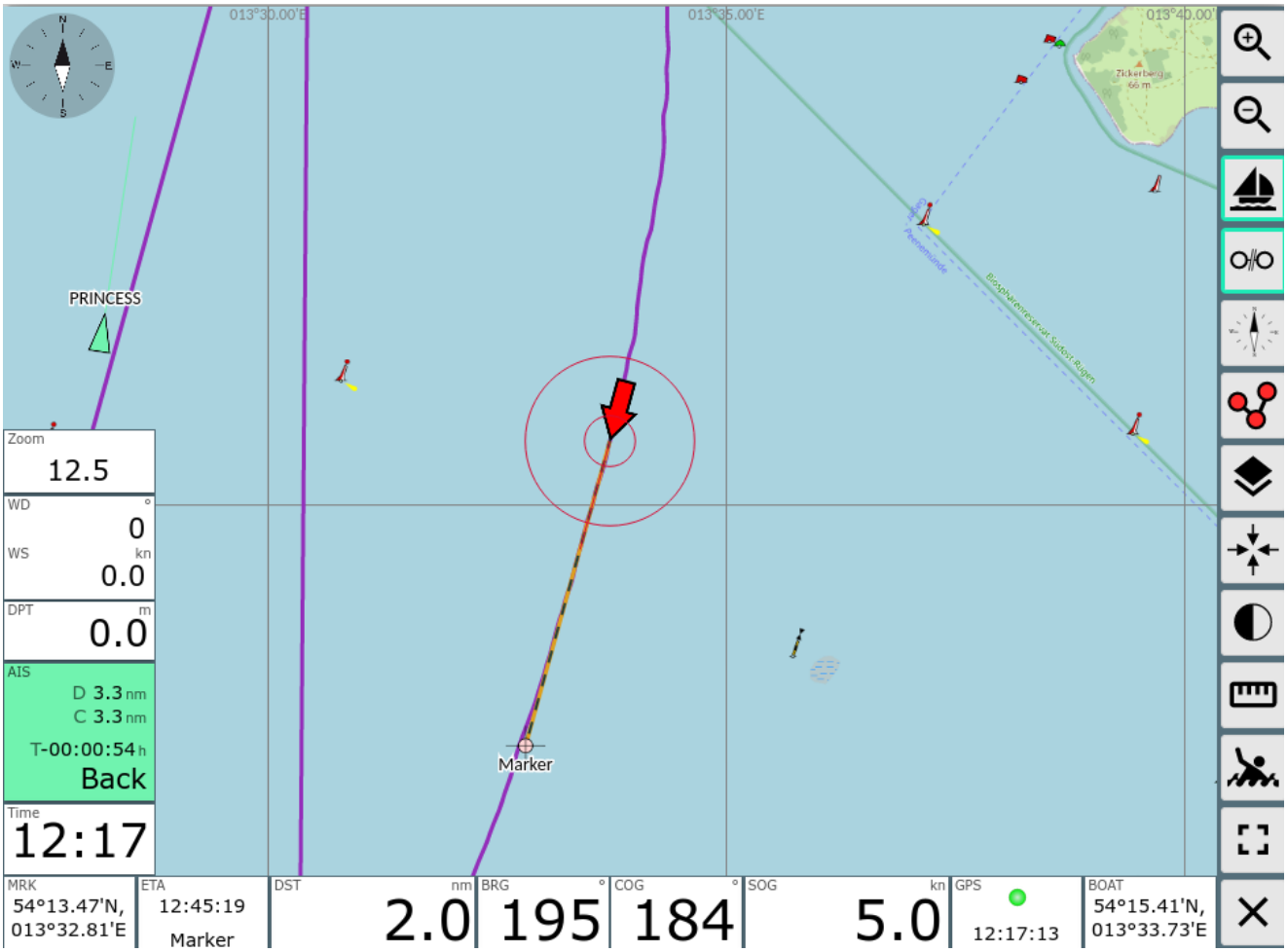
Alarme werden im Split Mode nur in der rechten Instanz angezeigt.

Über den Split Button in jeder Instanz kann der Split mode wieder verlassen werden.

Mit dem Slider kann die Aufteilung zwischen beiden Instanzen verändert werden.

Die Navigationsseite

Diese Seite ist die im Normalfall für die Navigation genutzte Darstellung.
Dieses Bild zeigt einen gesetzten Wegpunkt (Marker locked) – das Boot ist auf Kurs und der Kartenmittelpunkt wird permanent auf das Boot zentriert.



Buttons

Icon	Name	Funktion
	ZoomIn	Hereinzoomen
	ZoomOut	Herauszoomen
	LockPos	Kartenmittelpunkt auf Bootsposition setzen und dort halten (Karte bewegt sich mit dem Boot). Nur aktivierbar bei gültiger Position.

	StopNav	Navigation beenden. Nur sichtbar, wenn momentan ein Wegpunkt oder eine Route aktiv ist.
	LockMarker	Navigation starten. Der aktuelle Kartenmittelpunkt (Kreuz) wird zum Ziel-Wegpunkt. Nur sichtbar, wenn momentan keine Navigation aktiv ist.
	CourseUp	Karte drehen, so das die Kurs-Voraus Richtung oben ist.
	ShowRoutePanel	Wechsel zum Routen-Editor (auch durch Klick auf eine angezeigte Route)
	FullScreen	Fullscreen ein/aus (nur auf unterstützten Browsern)
	Measure	Setze eine Markierung auf die Kartenmitte und zeige fortlaufend Kurz und Entfernung zur MItte an, wenn die Karten bewegt wird (Mess-Tool)
	MOB	Mann über Bord (siehe Hauptseite)
	Overflow	Zeige eine zweite Liste von Buttons falls der Bildschirm zu klein für alle Buttons ist. Nur sichtbar, wenn unter Settings/Layout nicht "2 Button columns" ausgewählt ist
	NavOverlays	Aktivieren und Deaktivieren von Overlays
	Night	Aktivieren/Deaktivieren des Nachtmodus (ab 20210619, wenn Settings/Buttons/night mode on navpage gesetzt ist)
	GpsCenter	Zentriere die Karte auf die Bootsposition (ab 20210619)
	Dim	Dim Mode. Der Bildschirm wird abgedunkelt und alle Buttons werden inaktiv. Aufheben des Zustandes über Click auf eine beliebige Stelle auf dem Bildschirm. Der Button ist nur unter Android oder bei Nutzung des BonjourBrowsers (Version 1.5) sichtbar. Es wird dann auch der Bildschirm insgesamt gedimmt. Darüber kann der Stromverbrauch bei Nicht-Nutzung verringert werden

		- oder eine Überlast bei großer Helligkeit und hohen Temperaturen vermieden werden.
	RemoteChannel	Auswahl des Fernsteuerungskanals sowie des Fernsteuerungsmodus (senden/empfangen).
	Cancel	Zurück zur Hauptseite

Dies ist die Navigationsansicht. In der Mitte befindet sich die Kartenansicht mit der Schiffsposition (roter Pfeil). Die gelben und grünen Dreiecke mit Pfeilspitzen sind empfangene AIS Ziele in der Nähe (10nm, einstellbar) mit ihrem aktuellen Kurs sowie Name oder MMSI. Die organgefarbene Linie zeigt zum aktiven Wegpunkt. Die gepunktete Linie zeigt den Kurs vom Start der Navigation zum Wegpunkt (Sollkurs). Die Karte kann mit den normalen Gesten verschoben oder gezoomt werden, zum Zoomen können auch die Buttons +/- auf der rechten Leiste benutzt werden.

Um das Schiff kann man bis zu 3 Kreise in entsprechenden Entfernungen einstellen um Abstände zu schätzen (Über Einstellungen->Navigation Display, Standard 300m und 1000m).

(ab 202011xx) In Vorausrichting ("Course Vector") kann man die Länge der Linie in den Settings (Navigation/Boat Course Vector Length) einstellen. Man stellt die Sekunden ein, die Länge berechnet sich dann aus der aktuellen Geschwindigkeit (default: 10 Minuten). Die Dicke der Kursvektoren wird über die Einstellung für die Navigationskreise bestimmt.

Die gleiche Einstellung gilt auch für die Kursvektoren der AIS Ziele.

Für die Ausrichtung des Schiffes und die Kursvektoren gibt es verschiedene Modi (ab 20220421) - setzbar unter Einstellungen->Navigation->boat direction (siehe auch die Diskussion dazu auf GitHub):

Einstellung	Bedeutung	Symbol-Name
cog	Ausrichtung und Course Vector basieren auf COG	boatImage (Pfeil)
hdt	Ausrichtung basiert auf Heading True (fallback auf COG wenn nicht vorhanden), Course Vector auf COG. Optional HDT als gepunktete Linie (Einstellungen->navigation->add dashed vector for hdt/hdm)	boatImageHdg (Boot)
hdm	Ausrichtung basiert auf Heading Magnetic (fallback auf COG wenn nicht vorhanden), Course Vector auf COG.	boatImageHdg (Boot)

Optional HDM als gepunktete Linie (Einstellungen->navigation->add dashed vector for hdt/hdm)

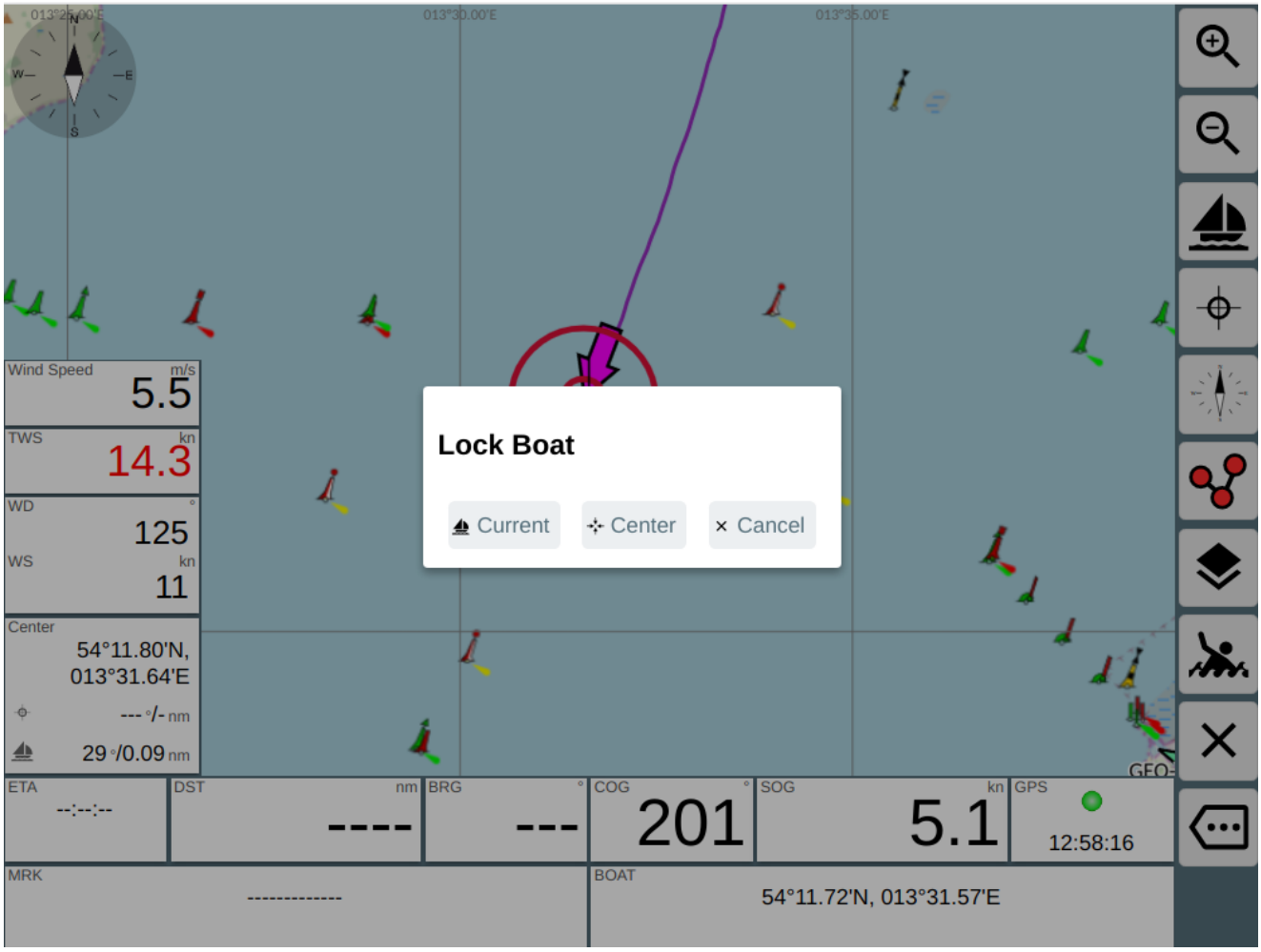
In den Einstellungen kann ausserdem noch eine Erkennung für "Boot nicht in Bewegung" aktiviert werden (Einstellungen->Navigation->zero SOG detect). Wenn das aktiviert ist, wird bei einer Geschwindigkeit < 0.2 kn (Einstellungen->Navigation->zero SOG detect below (kn)) das Boot Symbol geändert (boatImageSteady, roter Kreis).

Folge Modus

Wenn (wie im Bild) der Schiff-Button einen grünen Rand hat ("Schiff lock"), ist die Karten-Mitte auf die Schiffsposition fixiert und springt immer wieder dahin zurück.

Ab Version 20230614 kann unter Einstellungen/Map "allow move when locked" ein Verschieben der Karte im "Lock" Modus zugelassen werden.
Hierbei kann man die Karte verschieben und nach dem Loslassen wird die Schiff an der Stelle des Bildschirmes gehalten, wo es sich beim Loslassen befindet.
Allerdings wird die Position ggf. so verschoben, das das Schiff wieder auf der Karte sichtbar ist.

Ab Version 20210619 kann die Boot-Position auch von Anfang an auf eine beliebige andere Stelle des Bildschirmes fixiert werden. Dazu wählt man unter Einstellungen/Map/boat lock mode entweder "current" oder "ask" aus.
Bei "current" wird das Boot auf der Stelle am Bildschirm fixiert, an der es sich befindet, wenn "Lock" aktiviert wird. Die Karte wird dann entsprechend verschoben.
Bei "ask" wird bei jedem "Lock" Vorgang ein Dialog gezeigt:



Anzeigen

Links befinden sich (von oben nach unten):

- Die aktuelle Zoom-Stufe der Karte (wenn Auto Zoom aktiv, ist in Klammern die gewünschte)
- Das nächste AIS Ziel (ggf. rot bei Warnung) oder das ausgewählte AIS Ziel (gelb)
- Die aktuelle Uhrzeit

Die Darstellung des nächsten AIS Zieles (geringste momentane Entfernung) färbt sich rot, wenn eine CPA von 500m (einstellbar) unterschritten wird. Gelb bedeuted, dass nicht das nächste Ziel, sondern ein separat ausgewähltes Ziel (siehe unten AIS) angezeigt wird. Ein Klick auf diese Fläche oder ein AIS Ziel auf der Karte führt auf die AIS Info Seite .

(ab 202011xx) Die verwendeten Symbole für das Boot und die AIS Ziele können bei Bedarf über eine Datei images.json angepasst werden. Ausserdem können die AIS Symbole in den Einstellungen (AIS/Icon Scale) in ihrer Größe verändert werden, ein Rand kann ebenfalls hinzugefügt werden (AIS/Border Width). Falls die Berechnungen der AIS Kursvektoren zu aufwändig ist (Browser wird zu langsam), kann sie in den Einstellungen abgeschaltet werden (AIS/AIS Use Course Vector).

Im unteren Bereich der Navigationsseite befindet sich die Anzeige der wichtigsten Navigationsdaten ("Widgets"). Links die Daten des aktuellen Wegpunktes (Marker):

- Position
- ETA
- Kurs
- Distanz (nm)

Danach folgen die Schiffsdaten:

- Kurs
- Geschwindigkeit (kn)
- Position
- aktuelle (lokale) Zeit vom GPS
- GPS Indikator: grün – GPS Daten vorhanden, rot: keine Daten

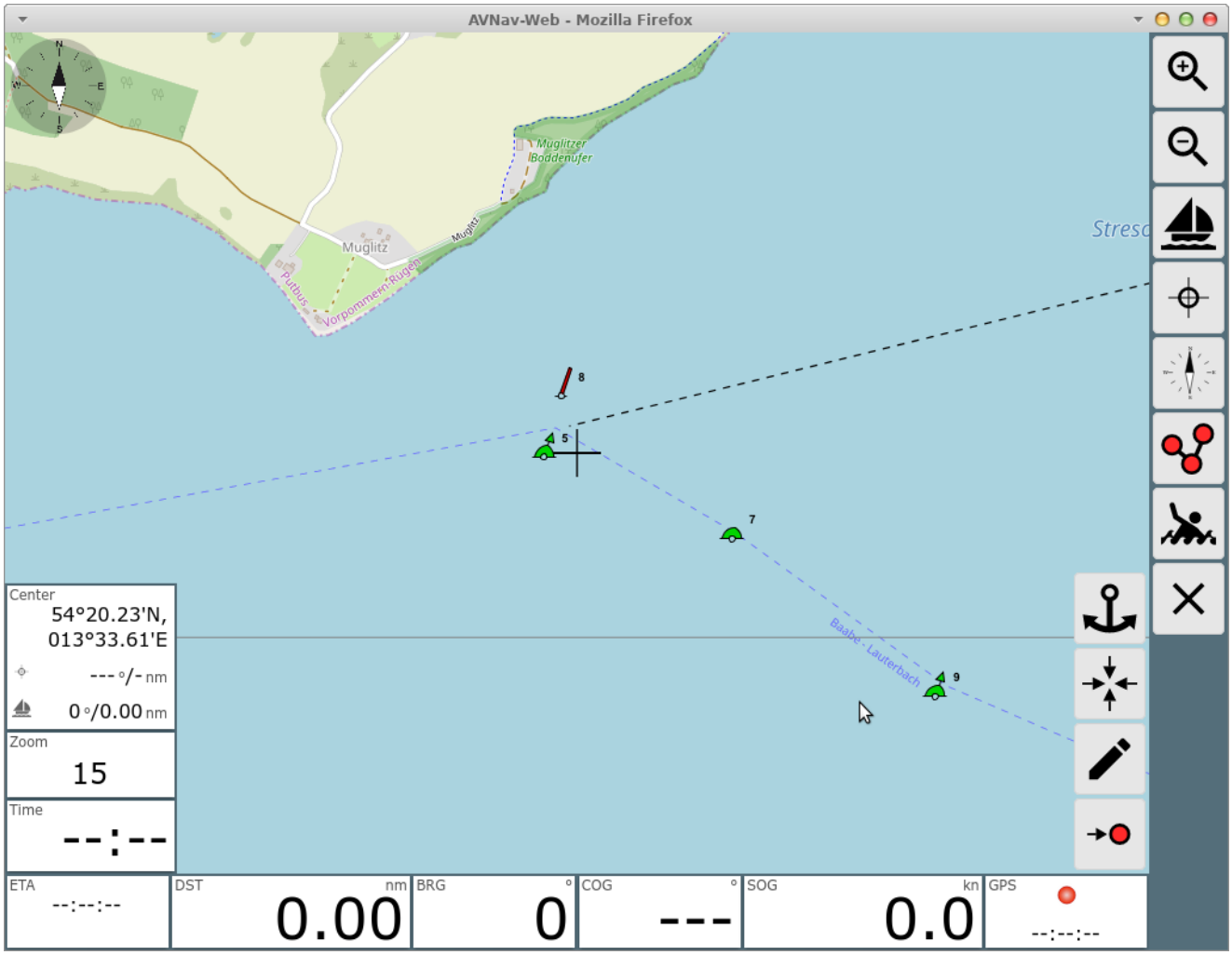
Je nach Breite des Bildschirms und den Einstellungen für die Schriftgröße der "Widgets" werden bis zu 2 Reihen an Daten angezeigt (unter Settings einstellbar). Daten, die nicht mehr auf den Bildschirm passen, werden unterdrückt.

Für die Schiffsposition, Kurs und Geschwindigkeit kann eine Mittelwertbildung eingestellt werden (Settings->Navigation). Wenn diese eingestellt ist, sind die Bezeichner rot.

Die Anzeigen auf der Seite können über den Layout Editor angepasst werden.

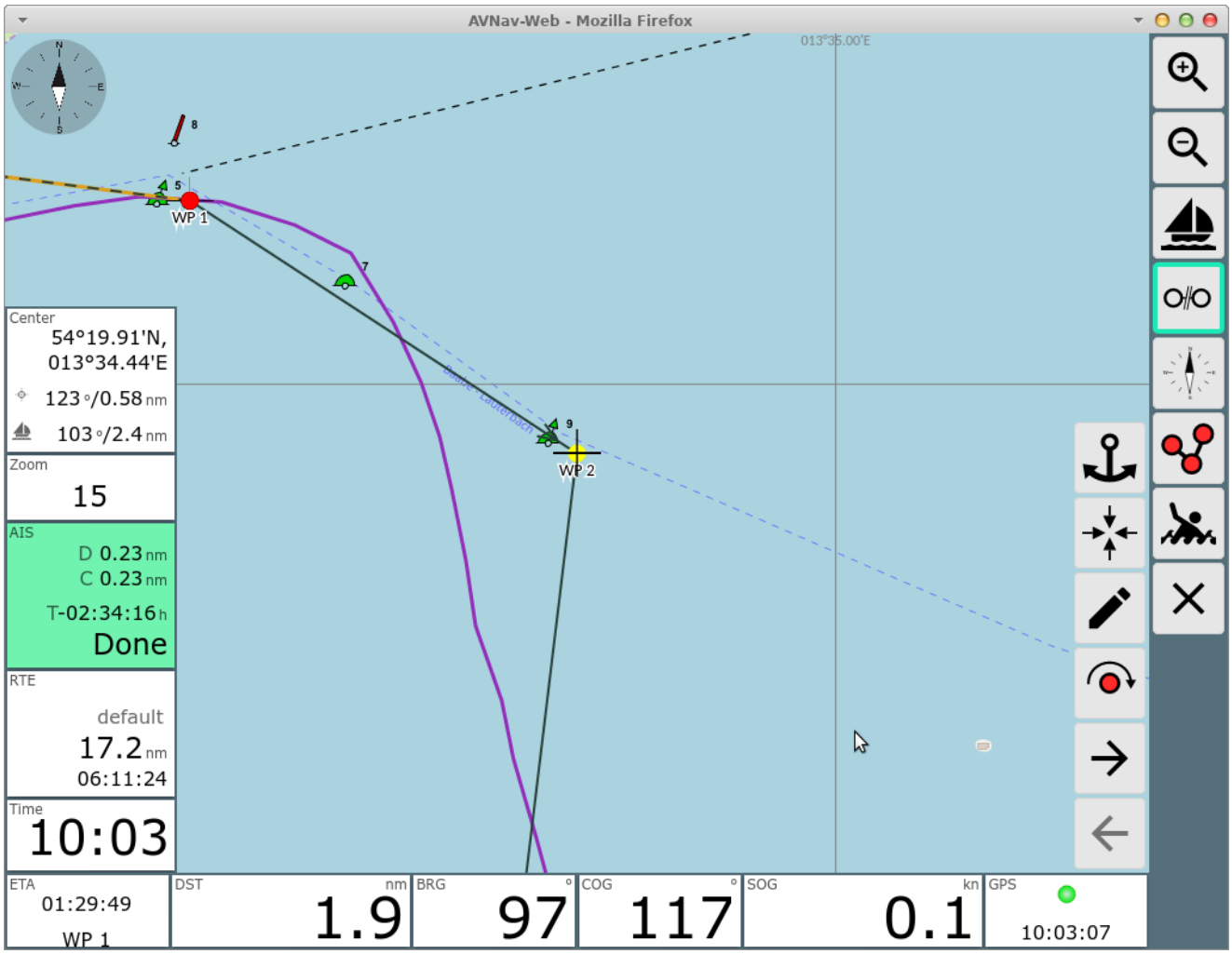
Ein Klick auf die rechten unteren Anzeigen führt zum Dashboard.

Ein Klick auf die linke Seite (Wegpunkt) zeigt einige zusätzliche Wegpunkt-Buttons.



Icon	Name	Funktion
	AnchorWatch	Einschalten der Ankerwache (siehe Dashboard)
	WpLocate	Zentrieren des Mittelpunktes auf den Wegpunkt
	WpEdit	Bearbeiten des Wegpunktes. Im angezeigten Dialog kann der Name und die Position des Wegpunktes editiert werden
	WpGoto	Start der Navigation zu diesem Wegpunkt. Wir nur angezeigt, wenn noch keine Navigation aktiv ist.
	NavRestart	Restarte die Navigation zum aktuellen Wegepunkt. Damit wird insbesondere der Soll-Kurs und damit die XTE Berechnung neu gestartet. Wird nur angezeigt, wenn eine Navigation aktiv ist.

Falls momentan eine Route aktiv ist, werden etwas andere Buttons beim Klick auf die Wegpunkt-Daten angezeigt.



Die zusätzlichen Buttons haben folgende Funktionen

Icon	Name	Funktion
	NavNext	Navigiere zum nächsten Wegpunkt der Route
	WpNext	Zentriere den Kartenmittelpunkt auf den nächsten Wegpunkt
	WpPrevious	Zentriere den Kartenmittelpunkt auf den vorigen Wegpunkt
	NavRestart	Restarte die Navigation zum aktuellen Wegepunkt. Damit wird insbesondere der Soll-Kurs und damit die XTE Berechnung neu gestartet. Wird nur angezeigt, wenn eine Navigation aktiv ist.

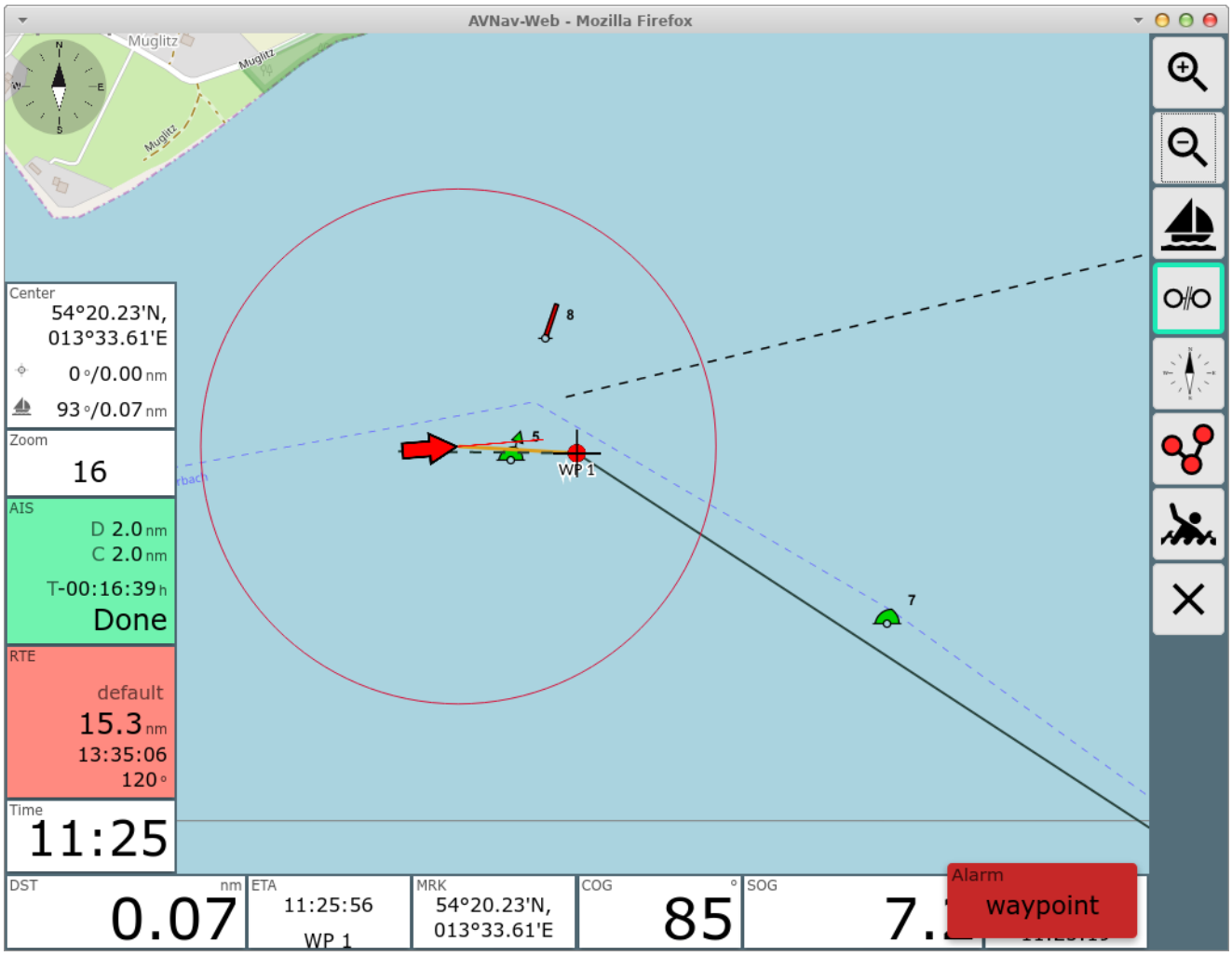
Wenn eine Route aktiv ist, hat man wie im Bild links eine Anzeige der Daten für die aktuelle Route (Name,verbleibende Distanz, Ankunftszeit).

In diesem Modus erfolgt eine automatische Weiterschaltung zum nächsten Wegpunkt, wenn die folgenden Bedingungen erfüllt sind:

- Das Boot befindet sich im "Fangbereich" (default 200m - settings->route->approach)
- Die weitere Bedingung entsprechend des eingestellten modes (early/90/late) ist erfüllt - siehe Wegepunkt Weiterschaltung.

Die Anzeige der Routen-Parameter wird rot und zeigt den Kurs zum nächsten WP an, wenn das Boot im "Fangbereich" ist. Ausserdem erfolgt eine Alarmierung mit einem akustischen Signal und einer Anzeige.

Falls keine automatische Weiterschaltung erfolgt (z.B. weil man nicht dicht genug am WP ist), kann man auf den Wegpunkt klicken und mit dem dann sichtbaren Button weiterschalten. Danach die Karte ggf. wieder auf die Bootposition fixieren (LockPos).



Spezielle Funktionen

Simple Wegpunkt Navigation

Schritte:

1. Unlock Boot (falls an)
2. Stop Nav
3. Karte verschieben bis Mittelpunkt (Kreuz) auf gewünschter Position (Zoom nutzen)
4. LockMarker
5. Lock Boot

Peilungen

Wenn man einen Wegpunkt aktiv hat (Marker Button grün) und die Karte nicht auf das Schiff "gelockt" ist (Schiffsbutton nicht grün), wird beim Bewegen der Karte ein Kreuz im Zentrum gezeigt, und links erscheint eine Anzeige des aktuellen Abstandes vom Marker bzw. Schiff zum Kartenmittelpunkt. Damit kann man einfache Peilungen machen - Mittelpunkt auf Peilziel verschieben und Peilung ablesen.

Feature Info

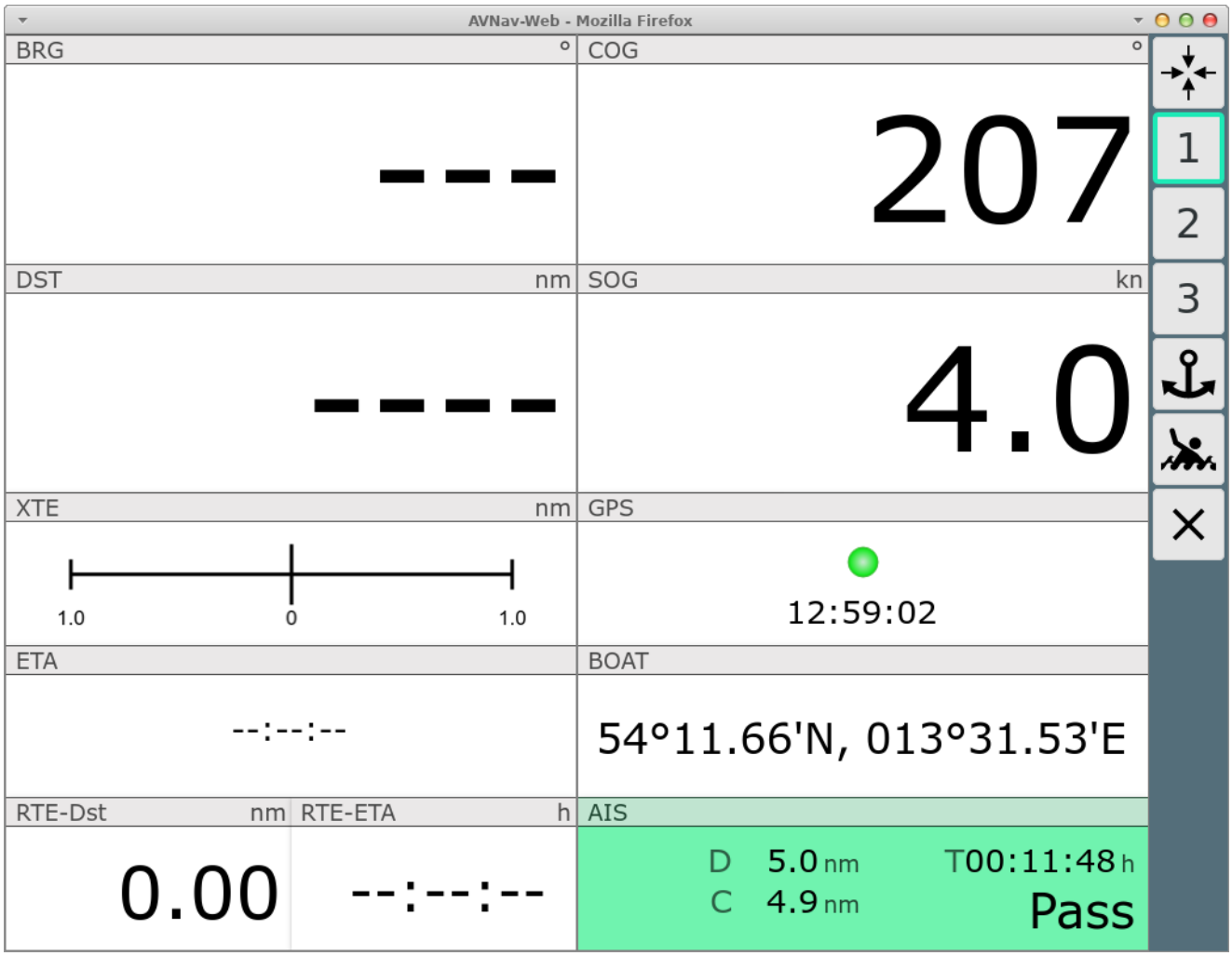
Wenn man auf die Karte klickt erscheint ein Dialog mit Informationen zum angeklickten Punkt. In diesem Dialog findet man einen Butoon um sofort eine Navigation zu diesem Ziel zu starten.

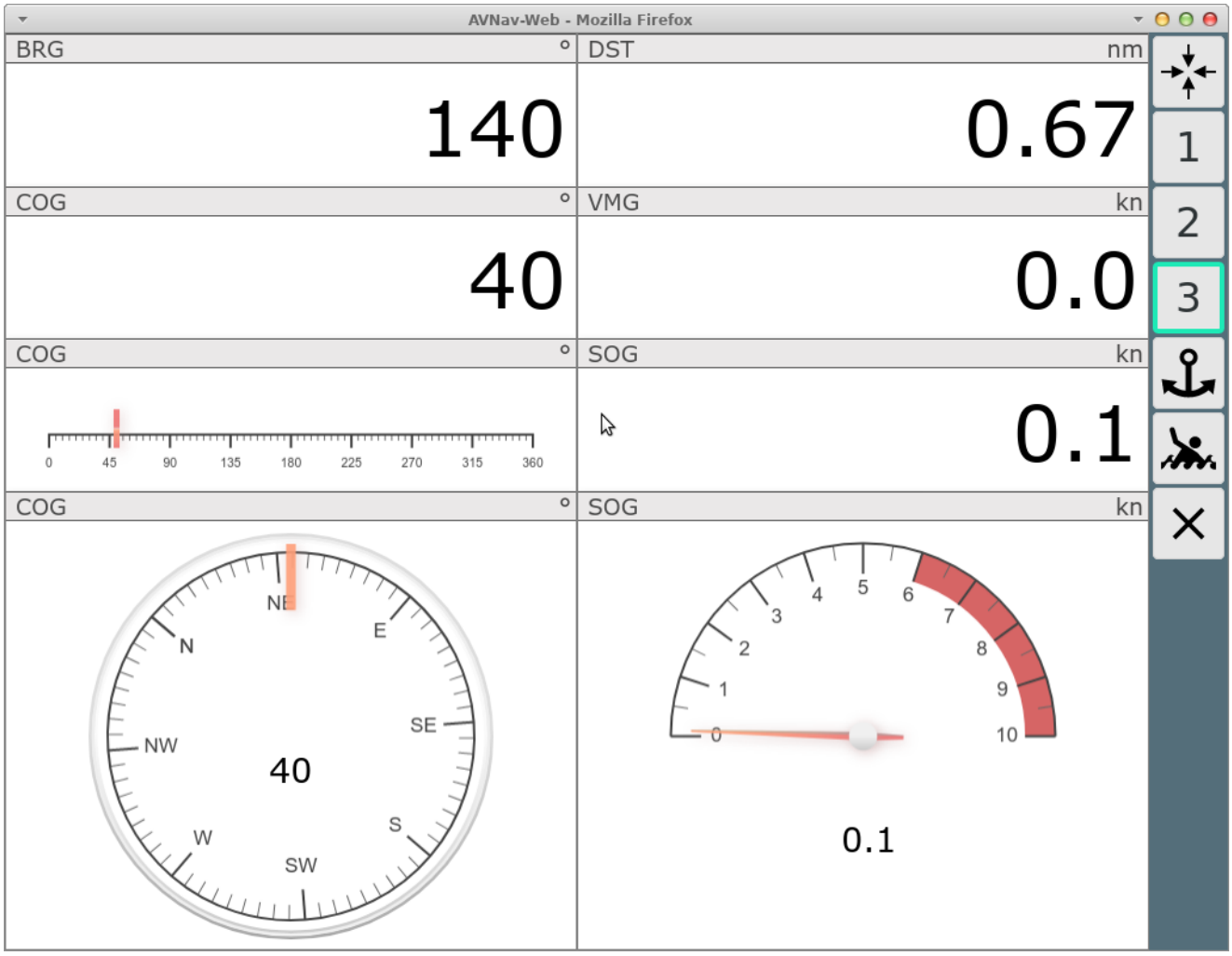
Für das Starten einer Route wechselt man zunächst mittels  zum Routen Editor.

Die Dashboard Seite

Über den Button (000) auf der Start-Seite oder einen Klick auf die Position (rechts unten) auf der Navi-Seite gelangt man zu einer Anzeige der GPS-Daten ohne Karte.

Im Layout können bis zu 5 solcher Seiten mit verschiedenen Anzeigen konfiguriert werden.





Buttons

Icon	Name	Funktion
	GpsCenter	Zentriere Karte auf den Wegpunkt und zurück zur vorigen Seite
1,2,...	Gps1, Gps2, ...	Auswahl des anzuzeigenden Dashboards. Die Konfiguration kann durch Auswahl eines Layouts oder Anpassung mit dem Layout Editor geschehen.
	AnchorWatch	Aktivieren der Ankerwache, siehe unten
	FullScreen	Fullscreen ein/aus (nur auf unterstützten Browsern)
	MOB	Mann über Bord (siehe Hauptseite)
	Overflow	Zeige eine zweite Liste von Buttons falls der Bildschirm zu klein für alle Buttons ist. Nur sichtbar, wenn unter Settings/Layout

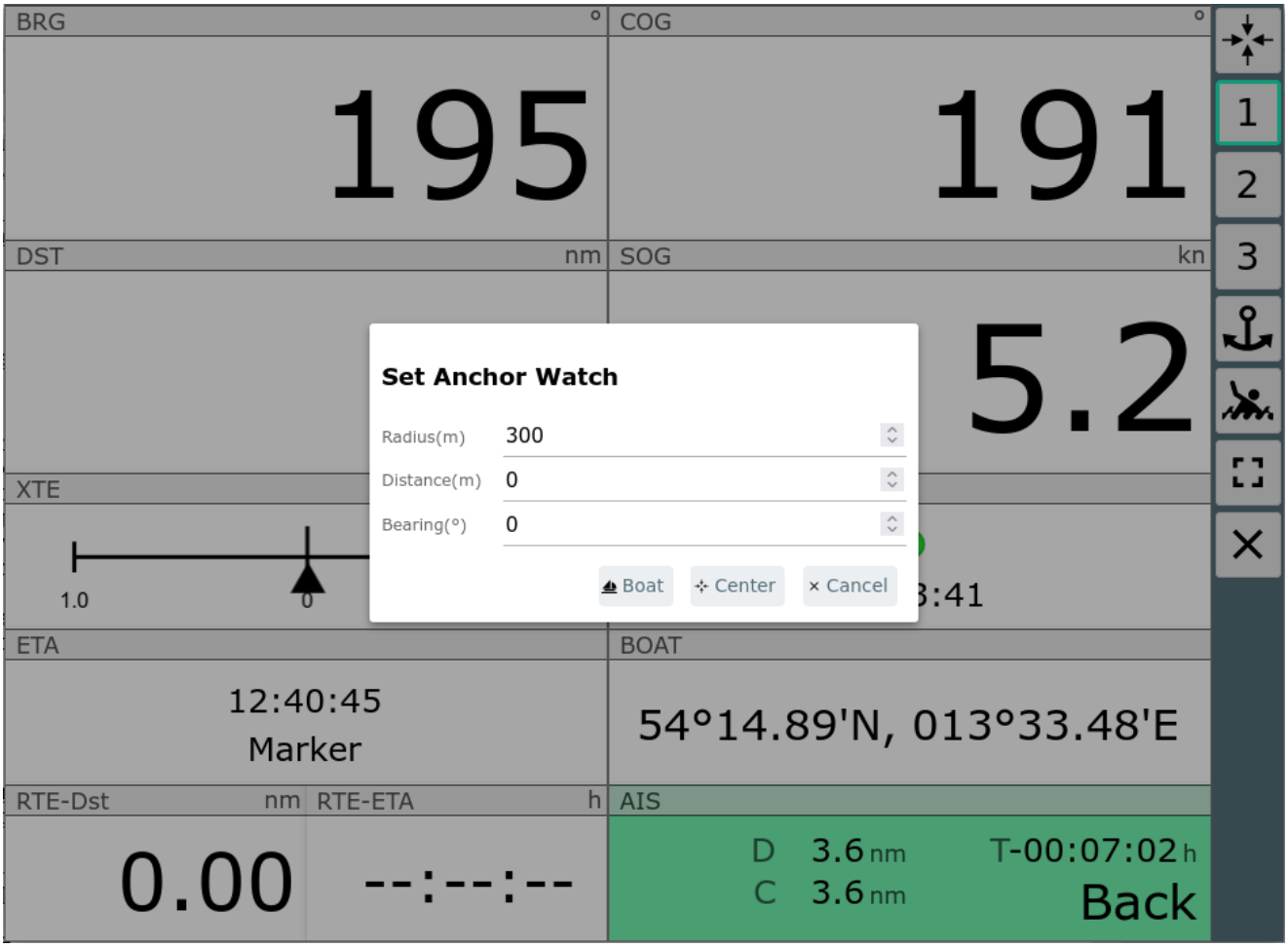
		nicht "2 Button columns" ausgewählt ist
	Dim	Dim Mode. Der Bildschirm wird abgedunkelt und alle Buttons werden inaktiv. Aufheben des Zustandes über Click auf eine beliebige Stelle auf dem Bildschirm. Der Button ist nur unter Android oder bei Nutzung des BonjourBrowsers (Version 1.5) sichtbar. Es wird dann auch der Bildschirm insgesamt gedimmt. Darüber kann der Stromverbrauch bei Nicht-Nutzung verringert werden - oder eine Überlast bei großer Helligkeit und hohen Temperaturen vermieden werden.
	Cancel	Zurück zur vorigen Seite

Ein Klick auf die Anzeige des nächsten AIS-Zieles (rechts unten) führt zur AIS-Seite, ein Klick auf jede andere Stelle oder den Cancel-Button zurück zur Seite von der man kam.

Spezielle Funktionen

Ankerwache

Durch Klick auf den  Button wird die Ankerwache aktiviert.

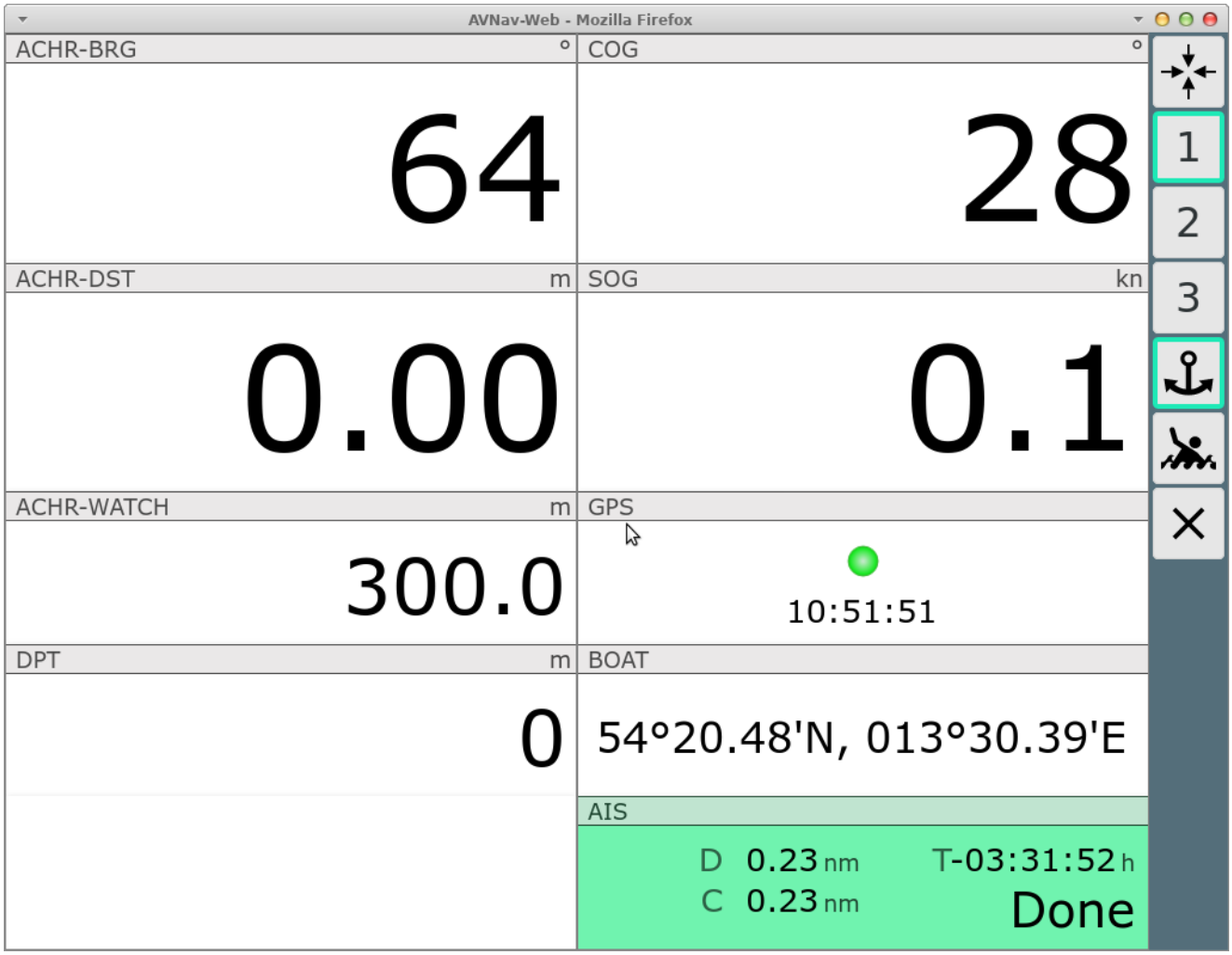


Im Dialog kann ausgewählt werden, ob als Anker-Position die Bootsposition oder alternativ der Kartenmittelpunkt gewählt werden soll. Ausserdem kann die zulässige Entfernung zur Ankerposition (Radius) und optional noch ein Offset zur aktuellen Position festgelegt werden (so kann man z.B. die Länge der Ankerkette und die Richtung in der sie liegt, mit berücksichtigen).

Nach Aktivierung bekommt der ⚓ Button einen grünen Rand und die Überwachung startet. Bei Überschreiten der Entfernung wird ein Alarm ausgelöst. Ausserdem erfolgt auch eine Alarmierung, wenn während der Ankerwache das GPS Signal ausfällt. Die Überwachung erfolgt dabei auf dem Server, d.h. es muss kein aktives Display verbunden sein. Voraussetzung ist natürlich, das auf dem Server eine Sound-Ausgabe möglich (und konfiguriert) ist.

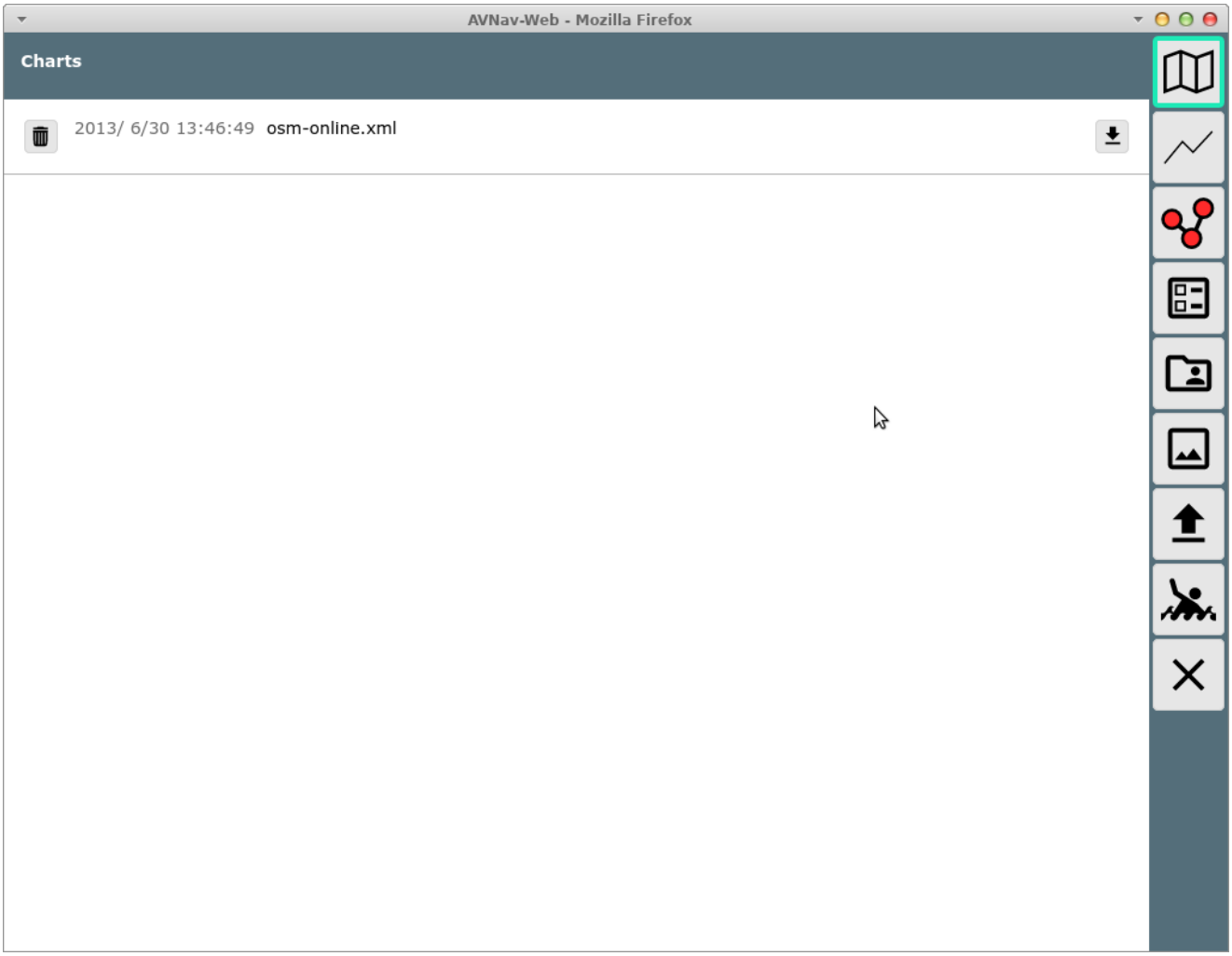
Die Ankerwache kann auch nach einem Klick auf die Anzeigen links unten auf der Navigationsseite aktiviert werden.

Mit aktiver Ankerwache ändern einige Dashboard Seiten ihren Inhalt.


AVNav-Web - Mozilla Firefox
ACHR-BRG °
64
COG °
28
ACHR-DST m
0.00
SOG kn
0.1
ACHR-WATCH m
300.0
GPS
10:51:51
DPT m
0
BOAT
54°20.48'N, 013°30.39'E
AIS
D 0.23 nm
C 0.23 nm
T-03:31:52 h
Done
1
2
3

Die Files/Download Seite

Von der Startseite kommt man mit dem Button zur Files/Download Seite. Auf dieser Seite kann man Karten, Tracks, Routen, Layouts, Nutzer-Dateien, Bilder herunterladen oder auf den Raspberry (Server) hochladen und sie dort auch löschen. Für einige Dateien ist auch eine direkte Bearbeitung möglich.

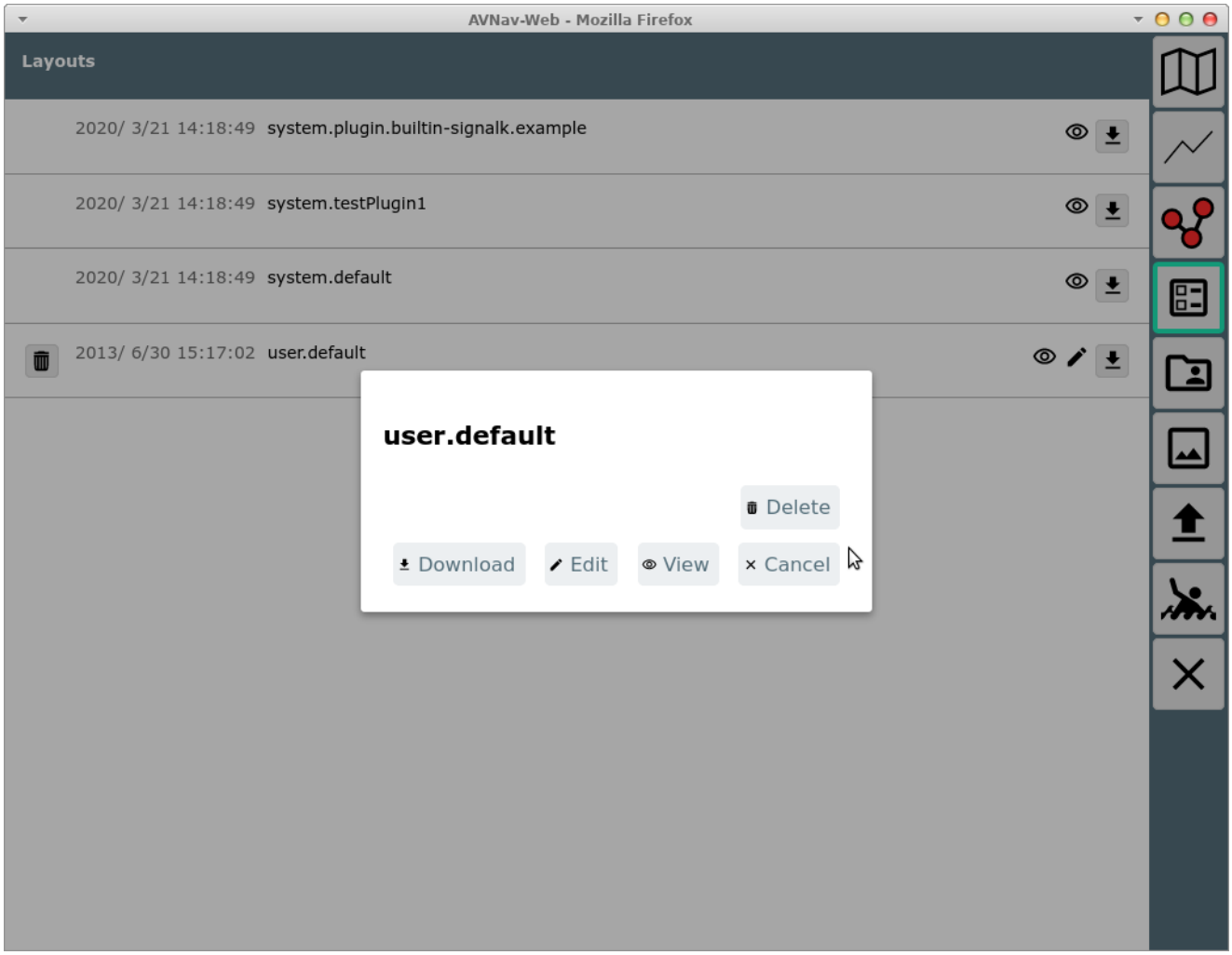


Buttons

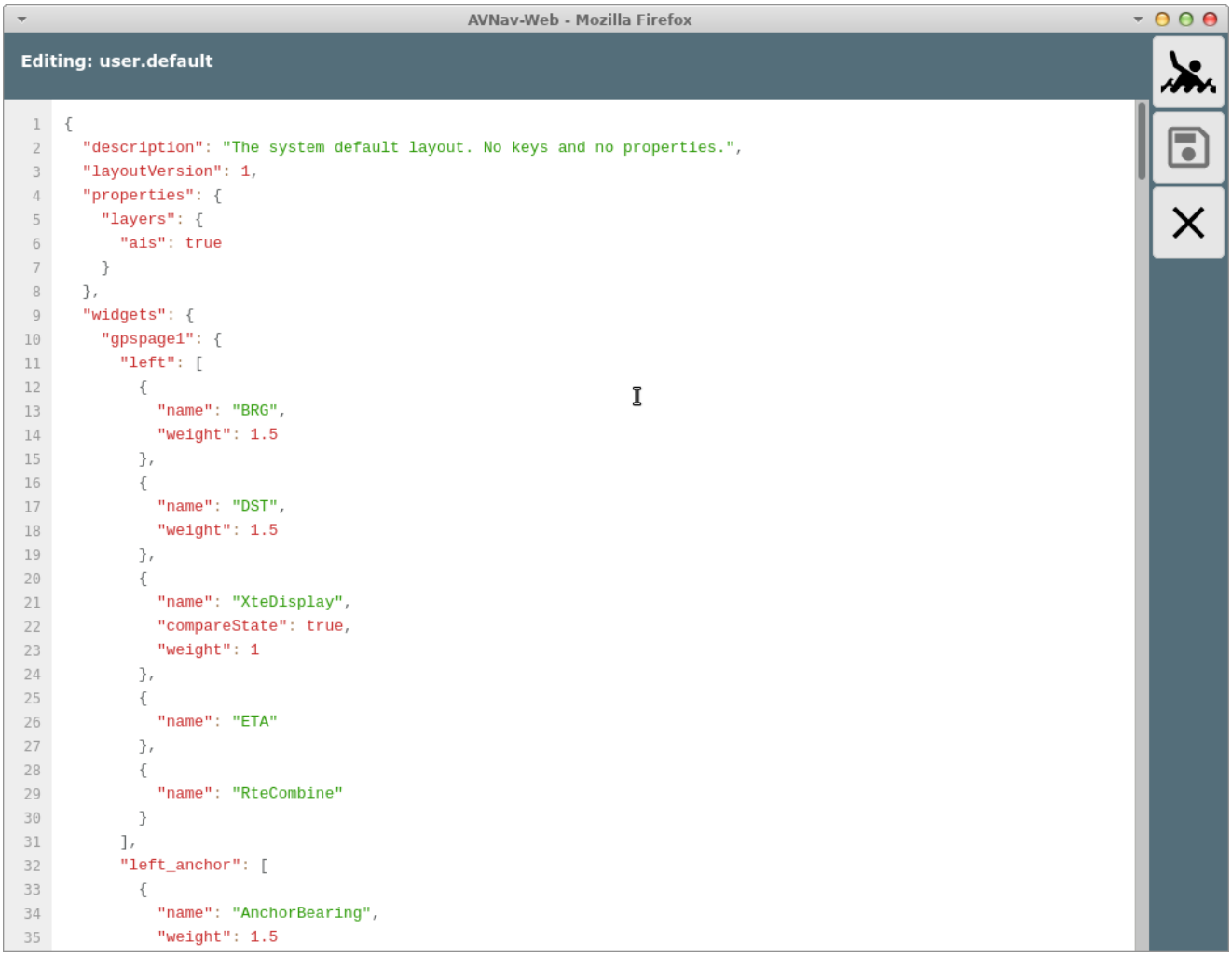
Icon	Name	Funktion
	DownloadPageCharts	Anzeige der Liste der Karten
	DownloadPageTracks	Anzeige der Liste der Tracks
	DownloadPageRoutes	Anzeige der Liste der Routen
	DownloadPageLayouts	Anzeige der Liste der Layouts
	DownloadPageUser	Anzeige der Liste der Nutzerdateien
	DownloadPageImages	Anzeige der Liste der Nutzer-Bilder
	DownloadPageOverlays	Zeige die Liste der Overlay Dateien
	DownloadPageUpload	Hochladen einer Datei für die gerade angezeigte Kategorie
	MOB	Mann über Bord (siehe Hauptseite)
	Cancel	Zurück zur vorigen Seite

In der angezeigten Liste stehen einige Informationen zum jeweiligen Eintrag und es gibt einen  Löschen und einen  Download Button. Diese aber nur, wenn es das jeweilige Element erlaubt (z.B. kein Löschen von demo charts oder system-Layouts).

Potentiell werden noch weitere Icons angezeigt, die die Bearbeitungsoptionen darstellen. Beim Klick auf ein Element in der Liste öffnet sich ein Dialog mit den verfügbaren Optionen.



Falls angeboten ("Edit" / "View") kann hier zu einer Anzeige- / Bearbeitungsansicht gewechselt werden. Für Routen öffnet man damit den Routen-Editor.



Auf dieser Seite können Änderungen vorgenommen werden (Überschrift "Editing") oder der Inhalt wird angezeigt. Durch Klick auf werden Änderungen gespeichert.

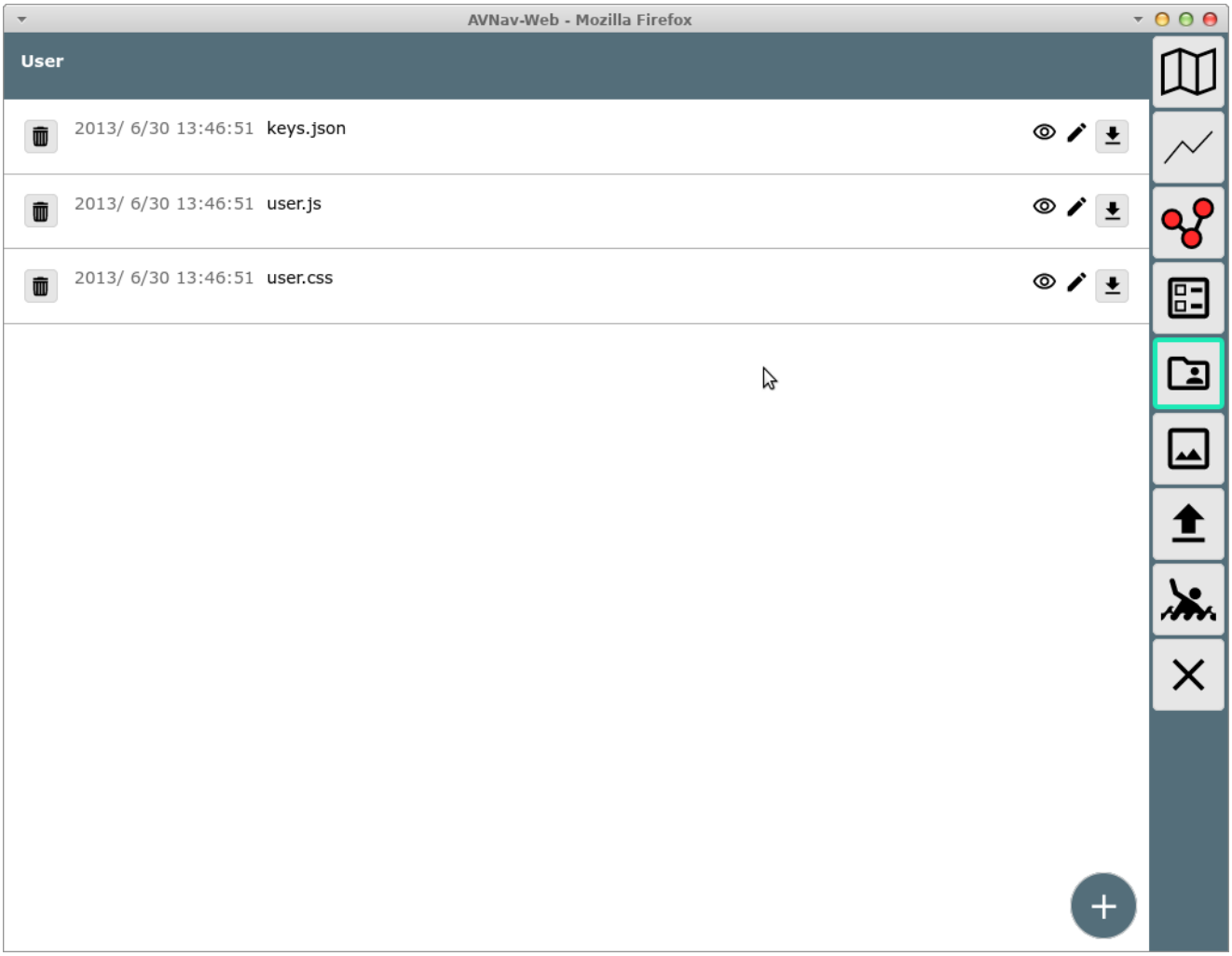
Besondere Funktionen

Nutzer Dateien

Wenn zur Ansicht Nutzer Dateien gewechselt wurde, können hier unter anderem die Datei user.js und user.css angesehen und bearbeitet werden.

Die Datei keys.json ermöglicht nutzerspezifische Tastaturkürzel.

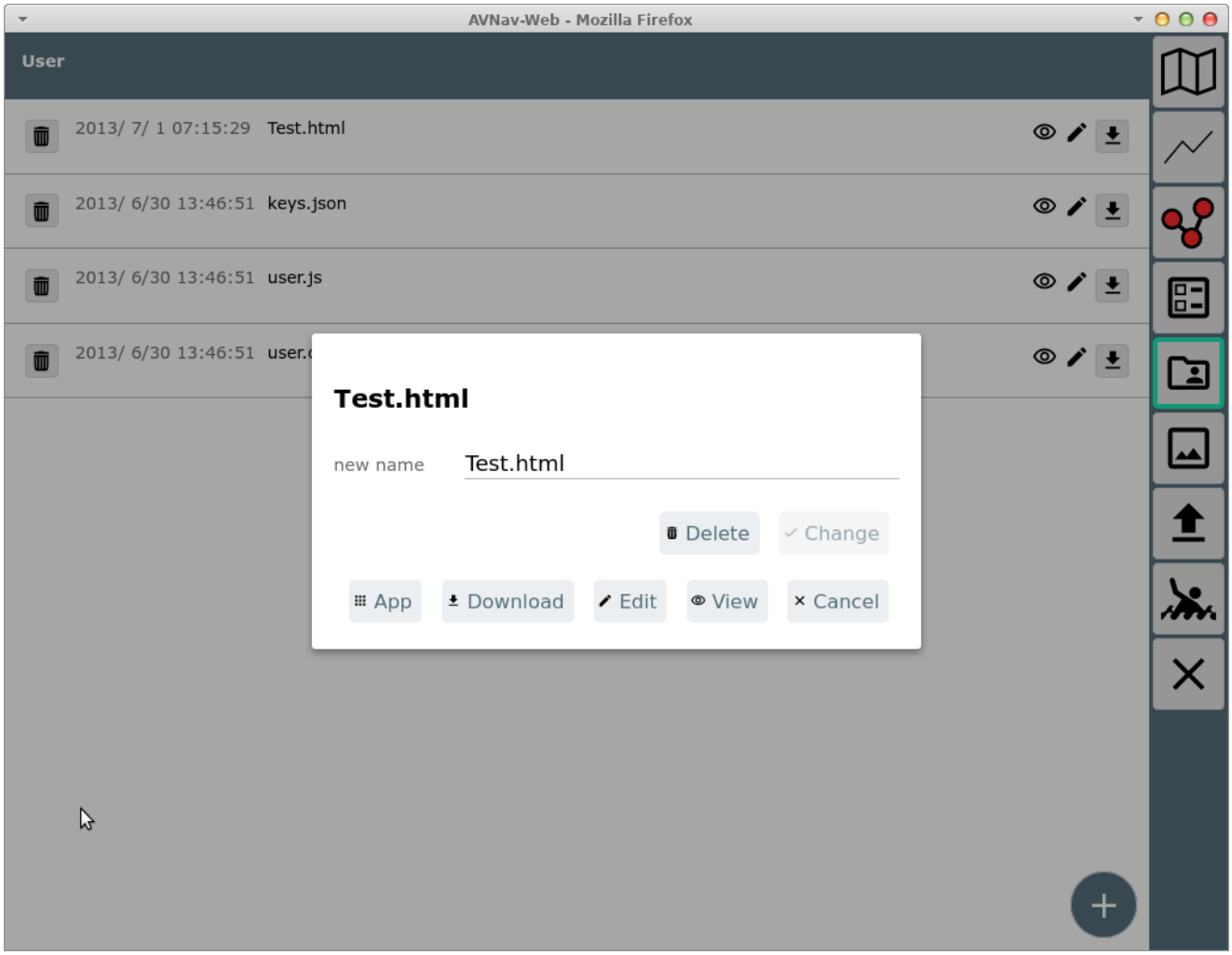
Mit der Datei images.json kann man Symbole anpassen.



Auf dieser Seite können auch weitere Dateien hochgeladen werden, die für die Anpassung von avnav benötigt werden (z.B. css Dateien, java script files oder auch HMTL Dateien). Diese Dateien sind jeweils über die URL /user/viewer/<name> erreichbar.
Bilder sollten vorzugsweise in der Ansicht hochgeladen werden und sind dann über /user/images/<name> erreichbar.

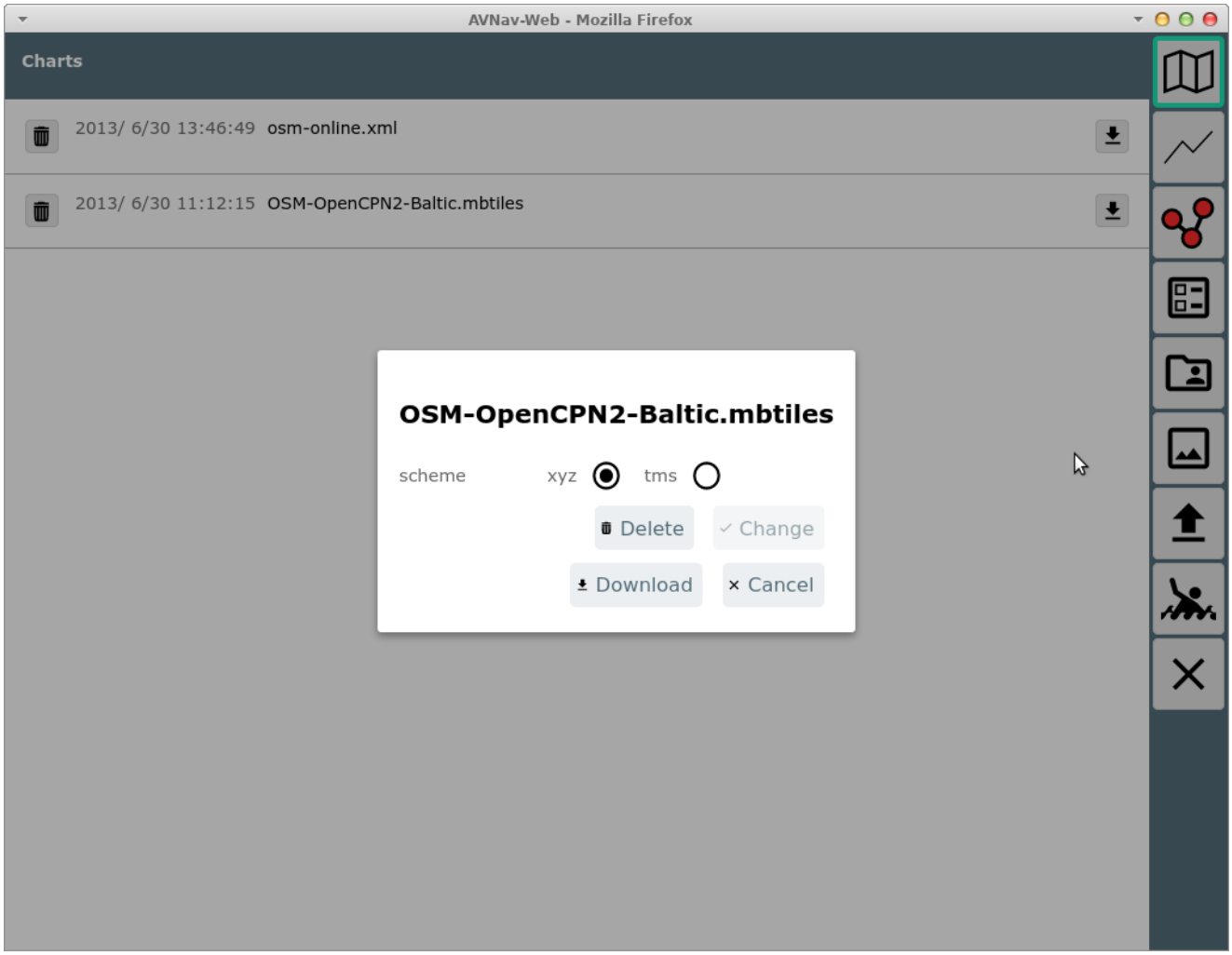
Über den "+" Button unten rechts kann eine neue Datei angelegt werden. Das könnte z.B. eine HTML Datei sein, die dann zu einer User App gemacht werden soll.

Für eine HTML Datei enthält der Aktionen Dialog den Button " App", mit dem direkt eine User App erzeugt werden kann (vorher muss allerdings ein Icon hochgeladen worden sein).



Karten - mbtiles Format

Bei einer Karte im mbtiles Format kann es sein, dass die Kacheln intern in einer anderen als der default Ordnung gespeichert sind. Der Default ist "xyz", optional gibt es "tms". Falls die Karte keine gültige Information enthält, wird "xyz" angenommen. Wenn eine solche Karte nicht korrekt dargestellt wird, kann man versuchen auf "tms" umzuschalten.

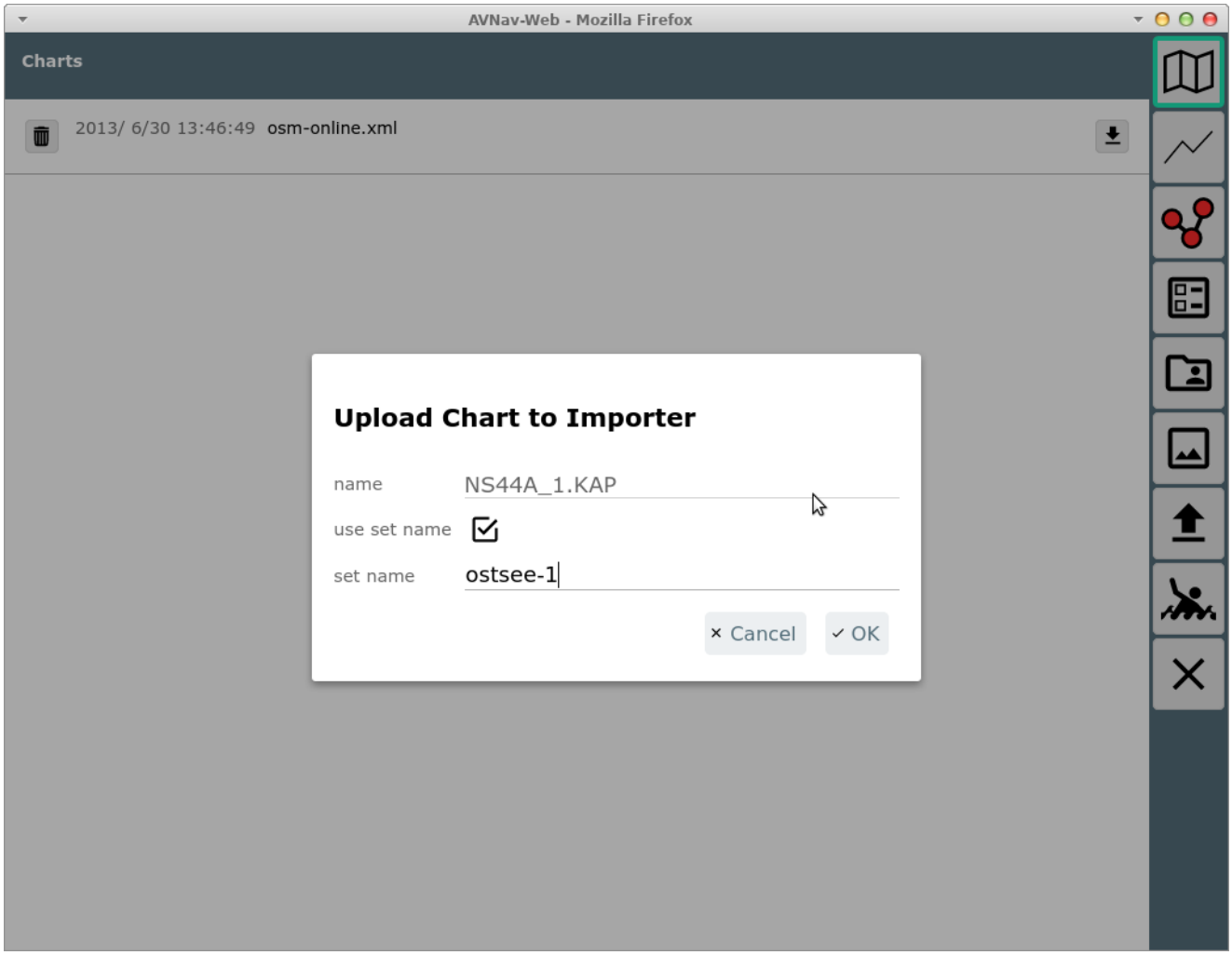


Diese Information wird dann dauerhaft in der Kartendatei gespeichert.

Hochladen von Karten

In der Kategorie Karten können auch Karten Dateien direkt hochgeladen werden. Dateien, die AvNav direkt verarbeiten kann (gemf, mbtiles, xml files), werden direkt hochgeladen und können sofort genutzt werden.

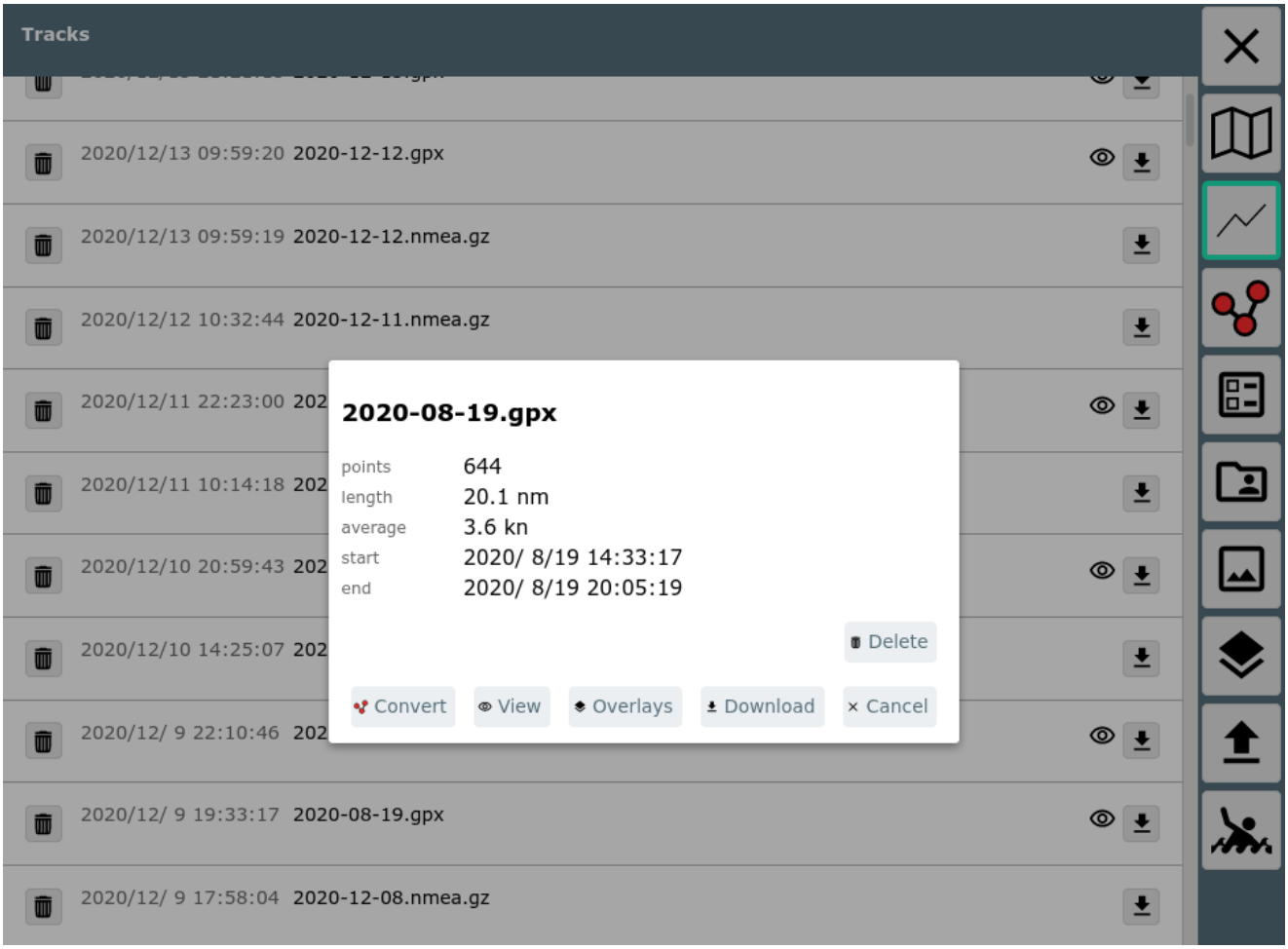
Für Karten, die erst nach Konvertierung nutzbar sind, wird angeboten, diese in das Eingangsverzeichnis des Importers zu laden (nicht unter Android).



Hierbei kann noch ein Name für ein "set" vergeben werden. Das wird dann der Name der erzeugten gemf Datei, es können so mehrere Karten in eine gemf Datei konvertiert werden. Der Zustand der Konvertierung kann anschliessend auf der Status Seite unter "importer" geprüft werden.
Nach einem Upload wartet der Importer noch eine gewisse Zeit, damit hat man die Möglichkeit nacheinander mehrere Dateien hochzuladen, die dann alle in einer Konvertierung bearbeitet werden.

Tracks

Der Info Dialog für Tracks enthält zusätzliche Informationen.

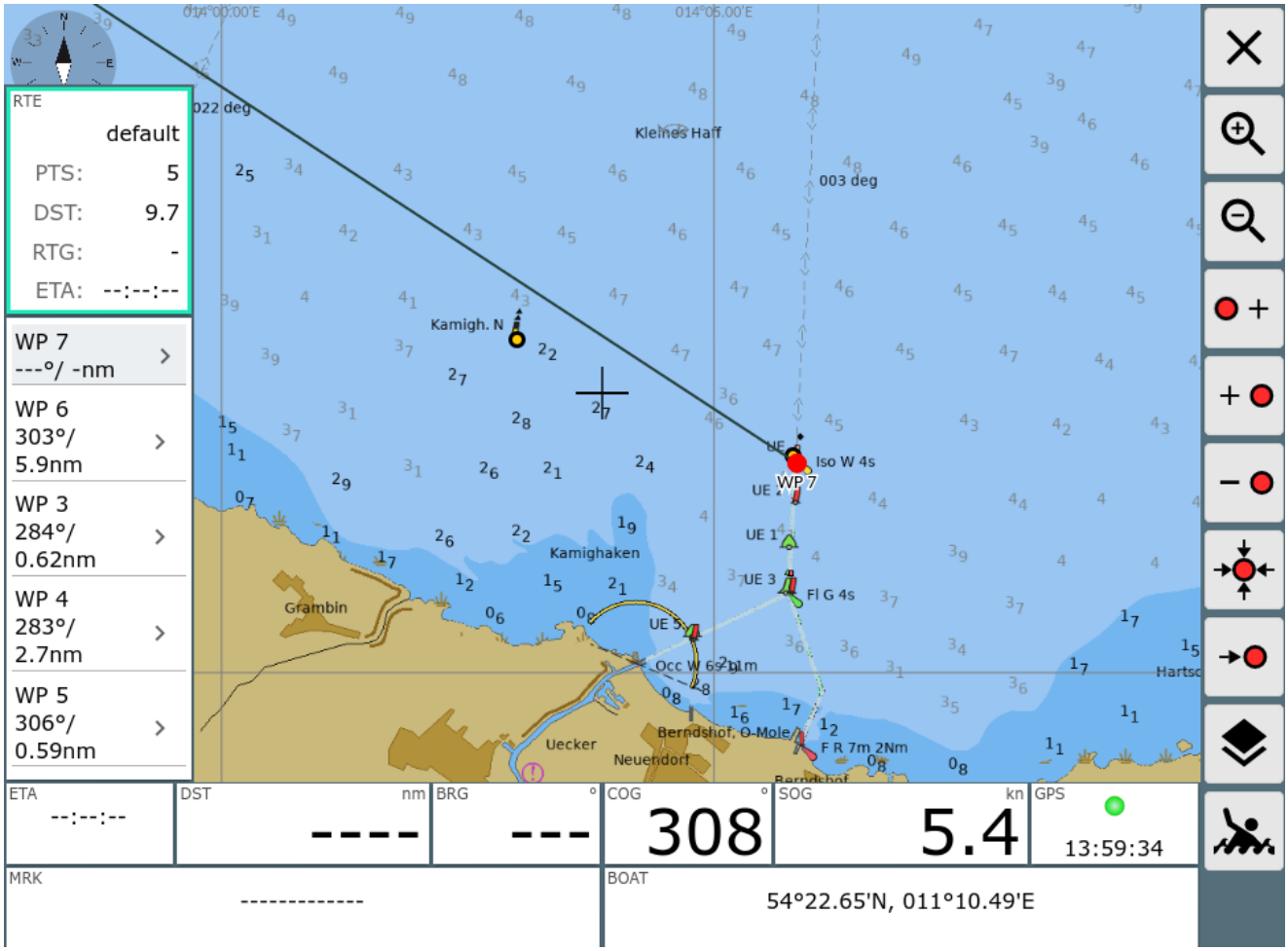


Ausserdem kann hier über den Button das Umwandeln des Tracks in eine Route gestartet werden. Über den overlay button kann der Track zu Overlays hinzugefügt oder gelöscht werden.

Der Routeneditor

Übersicht

Der Routen-Editor ist über den Button von der Navigationsseite erreichbar. Ausserdem kann er über den Button Edit im Info Dialog zu einer Route auf der Files/Download Seite erreicht werden.



Buttons

Icon	Name	Funktion

	ZoomIn	Hereinzoomen
	ZoomOut	Hinauszoomen
	NavAddAfter	Einen neuen Punkt zur Route hinzufügen (insert after) Der Kartenmittelpunkt (Kreuz) wird als neuer Punkt hinter dem momentan rot markierten Punkt der Route hinzugefügt.
	NavAdd	Einen neuen Punkt zur Route hinzufügen (insert before) Der Kartenmittelpunkt (Kreuz) wird als neuer Punkt vor dem momentan rot markierten Punkt der Route hinzugefügt.
	NavDelete	Löschen des momentan rot markierten Punktes der Route
	NavToCenter	Verschieben des momentan rot markierten Punktes der Route auf den Kartenmittelpunkt
	NavGoto	Starte die Navigation zum momentan rot markierten Punkt der Route
	NavOverlays	Ein- und Ausblenden von Overlays
	MOB	Mann über Bord(siehe Hauptseite)
	Cancel	Zurück zur Navigationsseite

Auf dieser Seite kann man eine Route bearbeiten bzw. erzeugen.

Zu jeder Zeit gibt es einen aktiven Wegpunkt (rot dargestellt, links in der Anzeige grau unterlegt. Die meisten Aktionen beziehen sich auf diesen Wegpunkt. Er kann durch Anklicken links in der Liste oder durch Anklicken auf der Karte geändert werden.

Wenn man die momentan aktive Route bearbeitet, wird das durch einen roten Rahmen links um die Routen-Info angezeigt. Falls eine andere Route bearbeitet wird, ist dieser grün.

Wenn man den Editor verlässt, geht die Anzeige zur aktiven Route zurück.

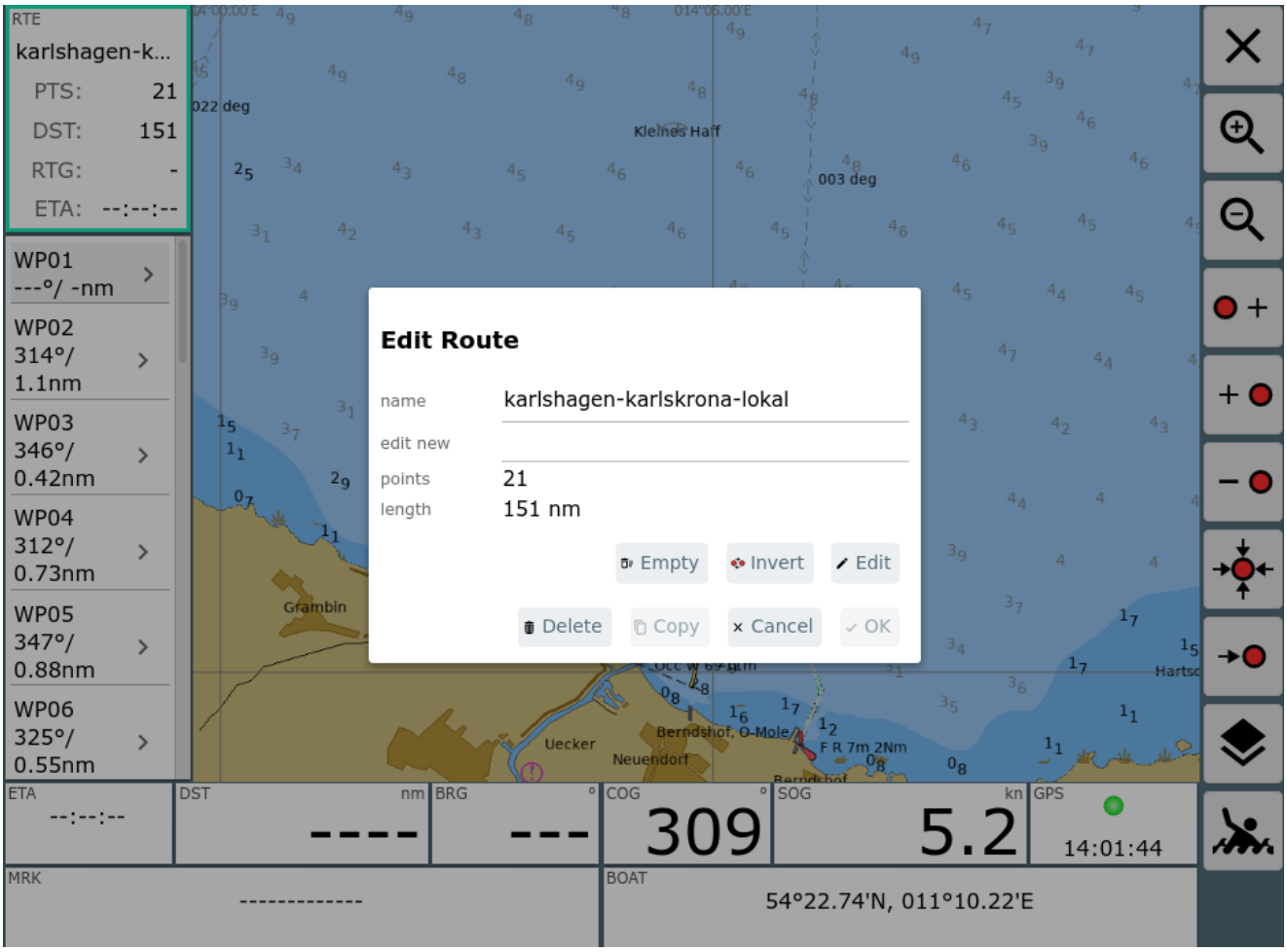
Alle Änderungen werden sofort wirksam, es gibt kein undo.

Auf der linken Seite werden die Wegpunkte der aktuellen Route angezeigt. Dazu jeweils die Kurse und Distanzen für die Abschnitte. Im oberen Teil die Gesamtlänge der Route sowie die Gesamtstrecke sowie die ETA. Durch Anklicken kann man einen Wegpunkt aktivieren.

Durch nochmaligen Klick auf einen links bereits markierten Punkt wird eine Eingabemaske aufgerufen, in der man den Wegpunkt bearbeiten kann.

Route Dialog

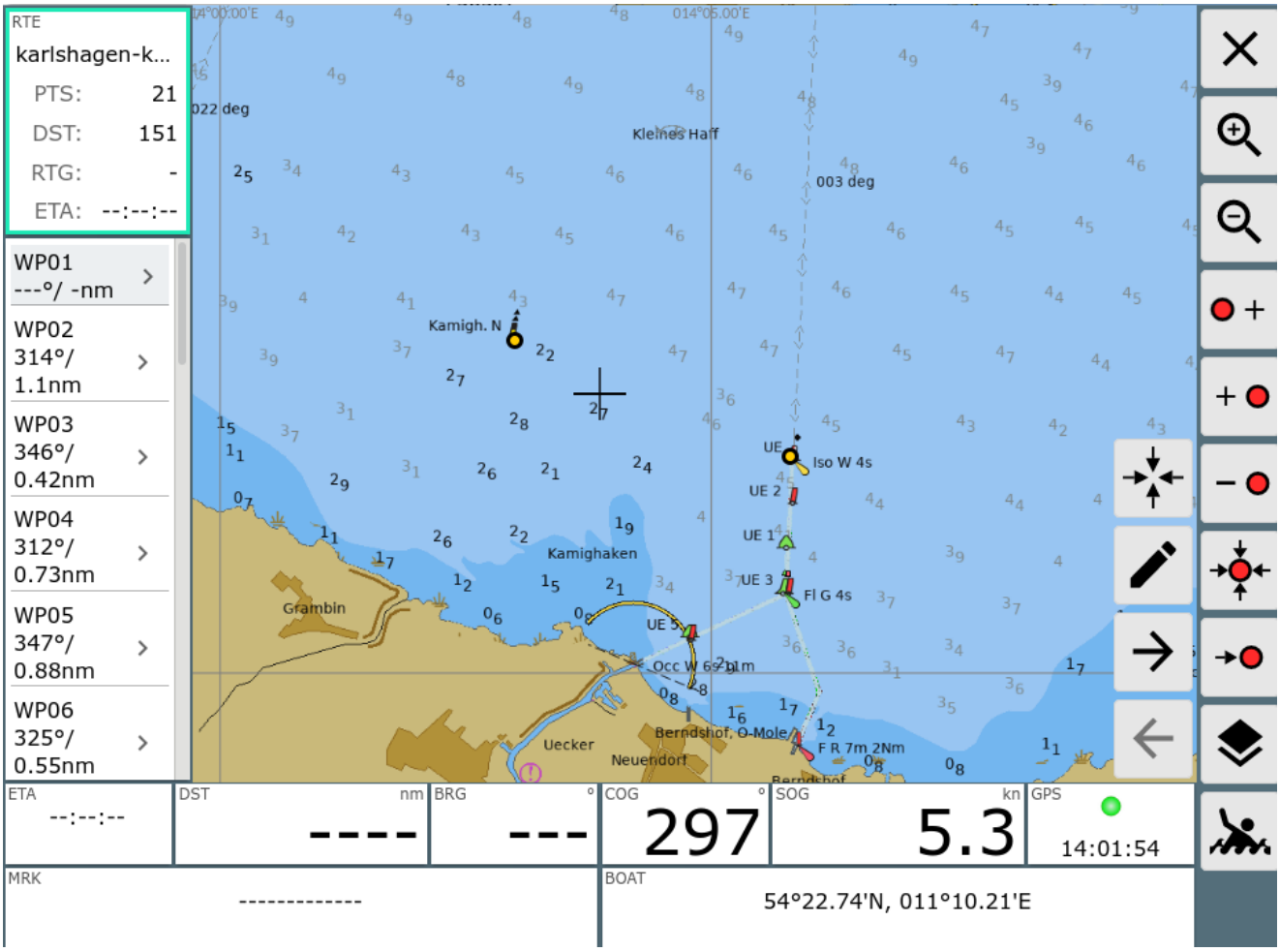
Ein Klick auf das Route Info Feld (links oben) führt zu einem Dialog.



Hier kann man den Namen der Route ändern, eine neue Route zum Editieren auswählen, die Route kopieren, sie löschen, invertieren oder alle Punkte löschen.

Ein Klick auf den Edit Button führt zur Route Liste. Hier kann man die Wegpunkte bearbeiten oder ebenfalls eine neue Route wählen.

Ein Klick auf die linken unteren Anzeigen zeigt einige zusätzliche Buttons.

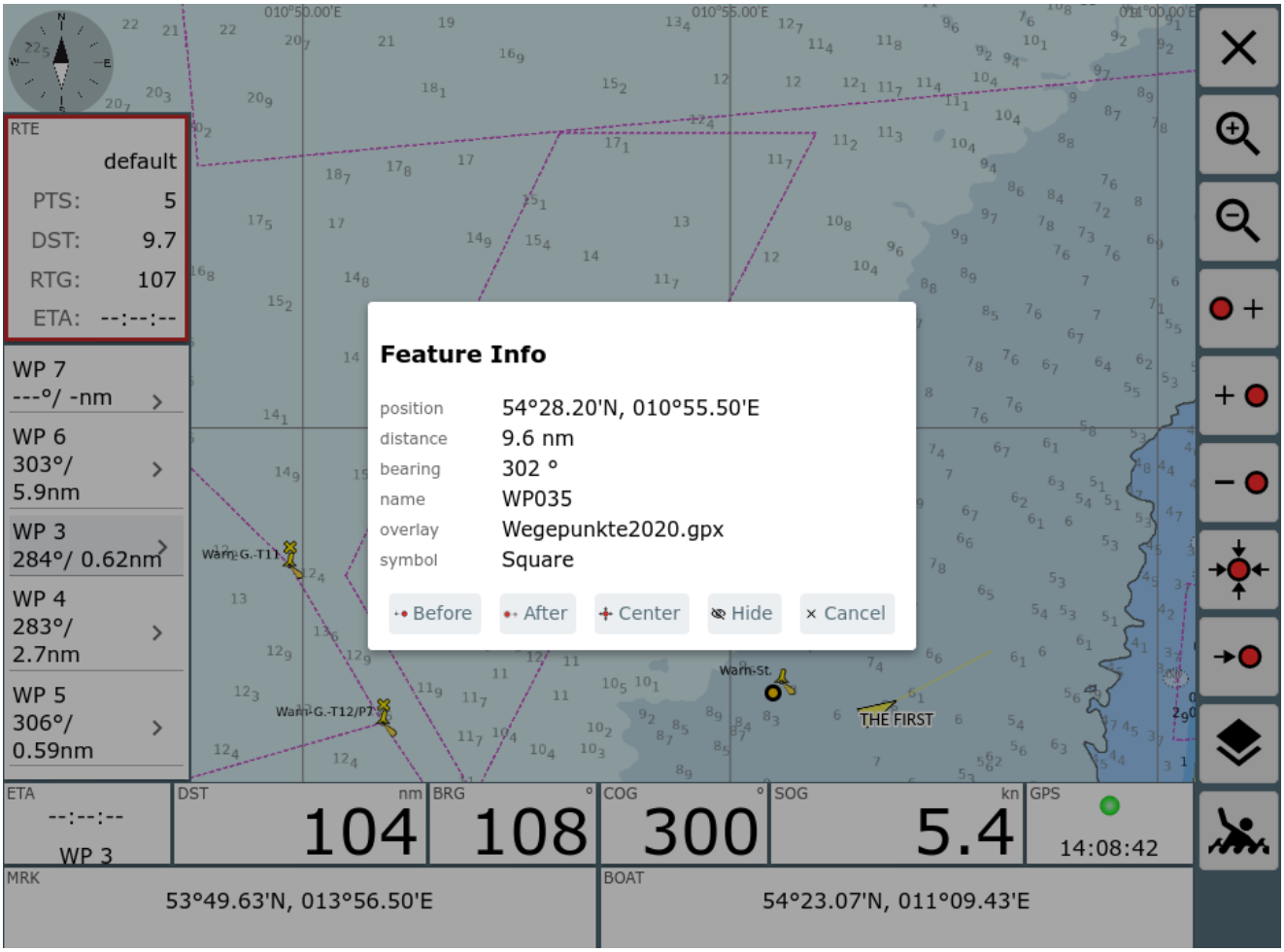


Mit diesen kann zwischen den Punkten der Route gewechselt werden bzw. der aktuelle Punkt kann bearbeitet werden.

Ein Klick auf die rechten Anzeigen zentriert die Karte auf die Bootsposition.

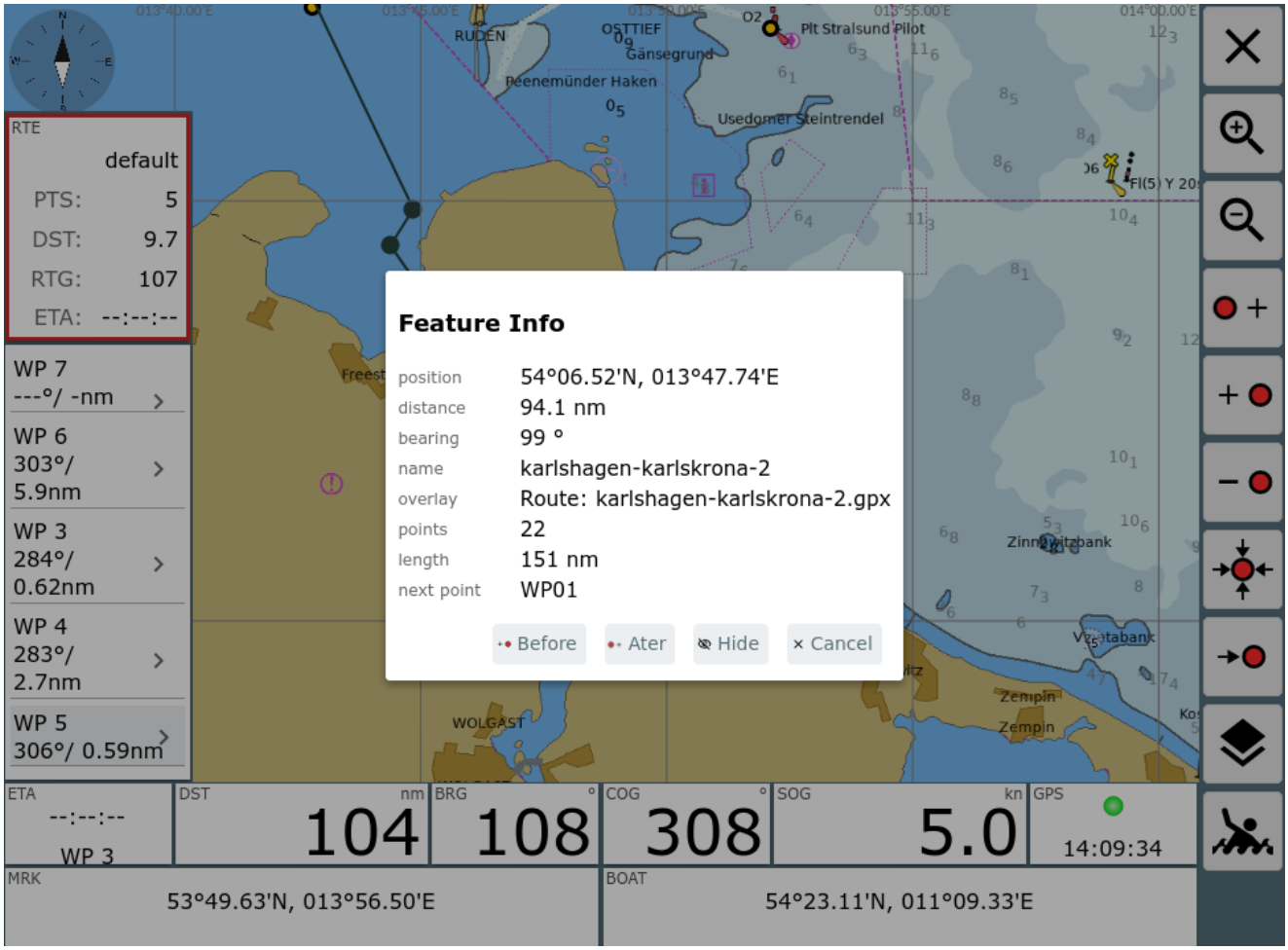
Nutzung von Overlays

Falls (wie im Bild) Overlays angezeigt werden (oder in ocharts-Karten) kann man durch Klick auf ein Objekt einen Dialog aufrufen, der weitere Routen-Funktionen bietet.



Der angeklickte Punkt kann jetzt in die Route eingefügt werden. Auf diese Weise können leicht Punkte z.B. aus einer Wegpunkt-Datei zum Erstellen von Routen genutzt werden.

Falls das angezeigte Overlay eine Route ist, kann der Teil nach dem angeklickten Punkt der aktuellen Route hinzugefügt werden.



Im Dialog wird der angeklickte Startpunkt der Route angezeigt und man kann auswählen, ob man die Route vor oder hinter dem aktuell markierten Punkt der in Bearbeitung befindlichen Route einfügen möchte.

Konvertierung von Tracks zu Routen

AvNav zeichnet permanent Tracks auf. Für jeden Tag wird eine neue gpx Datei mit dem aktuellen Datum geschrieben.

Diese Tracks (oder auch Tracks, die auf der Files/Download Seite hochgeladen wurden) können in Routen umgewandelt werden, um sie zur Navigation zu nutzen.

Verfahren

Für diese Umwandlung ist es typischerweise nötig, die Zahl der Punkte im Track zu verringern (Tracks haben oft weit über 1000 Punkte). Sonst ist eine sinnvolle Nutzung einer solchen Route kaum möglich.

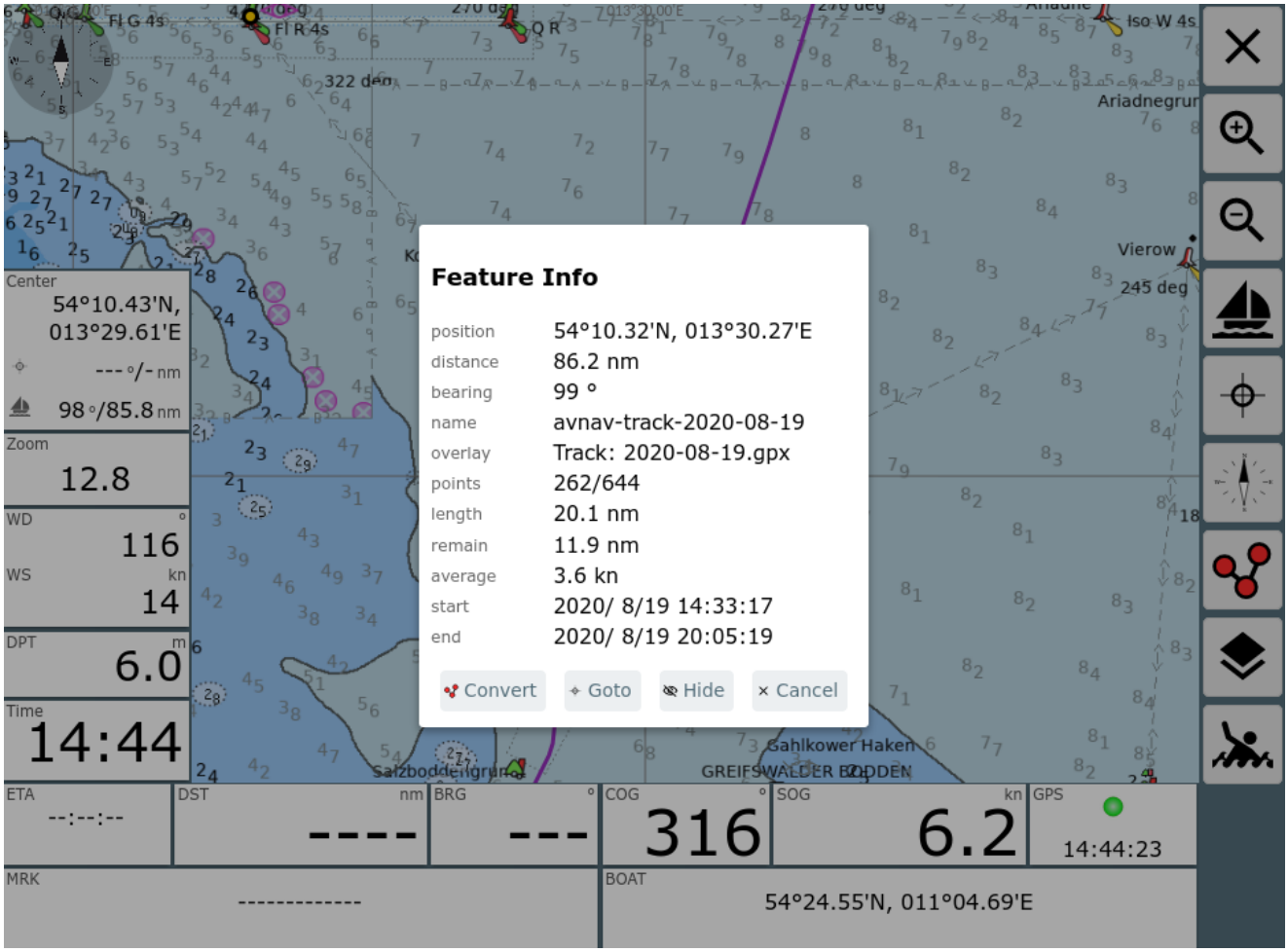
AvNav nutzt dazu ein Reduktionsverfahren, das an GPS Babel angelehnt ist. Es werden nacheinander die Punkte entfernt, bei denen die Abweichung zur neu entstehenden Strecke zwischen den Nachbarpunkten am kleinsten ist.

Man kann für dieses Verfahren eine maximale Abweichung (xte) angeben, die man tolerieren möchte. Je größer man diese zulässige Abweichung wählt, um so weniger Punkte bleiben am Ende übrig. Man sollte typischerweise versuchen, Routen auf ca. 50 Punkte zu begrenzen.

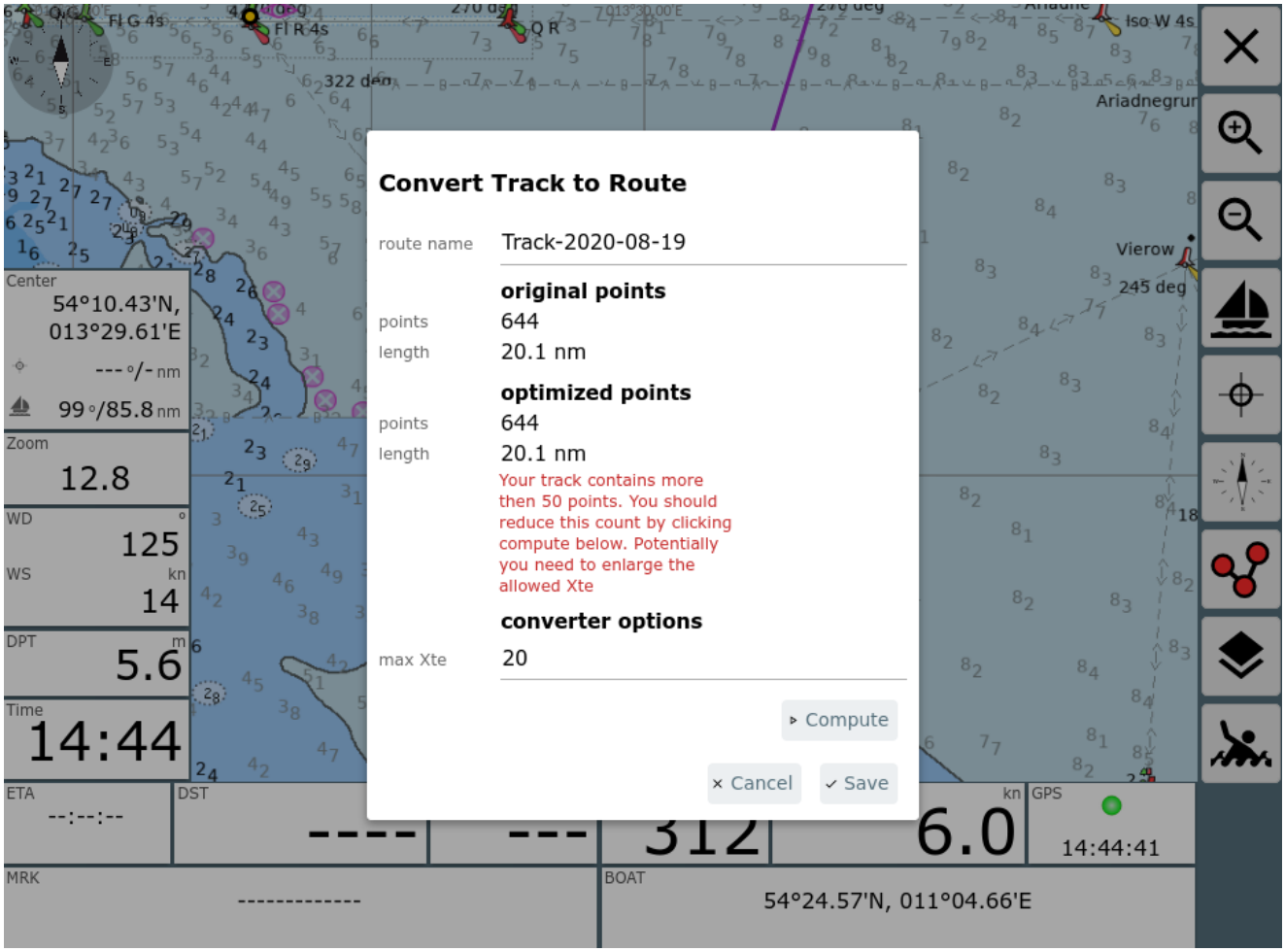
Nach der Umwandlung kann man die Route im Routen-Editor ggf. noch etwas nachbearbeiten.

Ablauf

Man kann den Dialog für die Konvertierung entweder von der Files/Download Seite aus dem Track Info Dialog erreichen ( Convert Button) oder aus dem Info Dialog, falls man einen Track als Overlay auf der Karte hat, und einen Punkt in diesem anklickt. Im Info Dialog wieder den  Convert Button nutzen - siehe Bild.

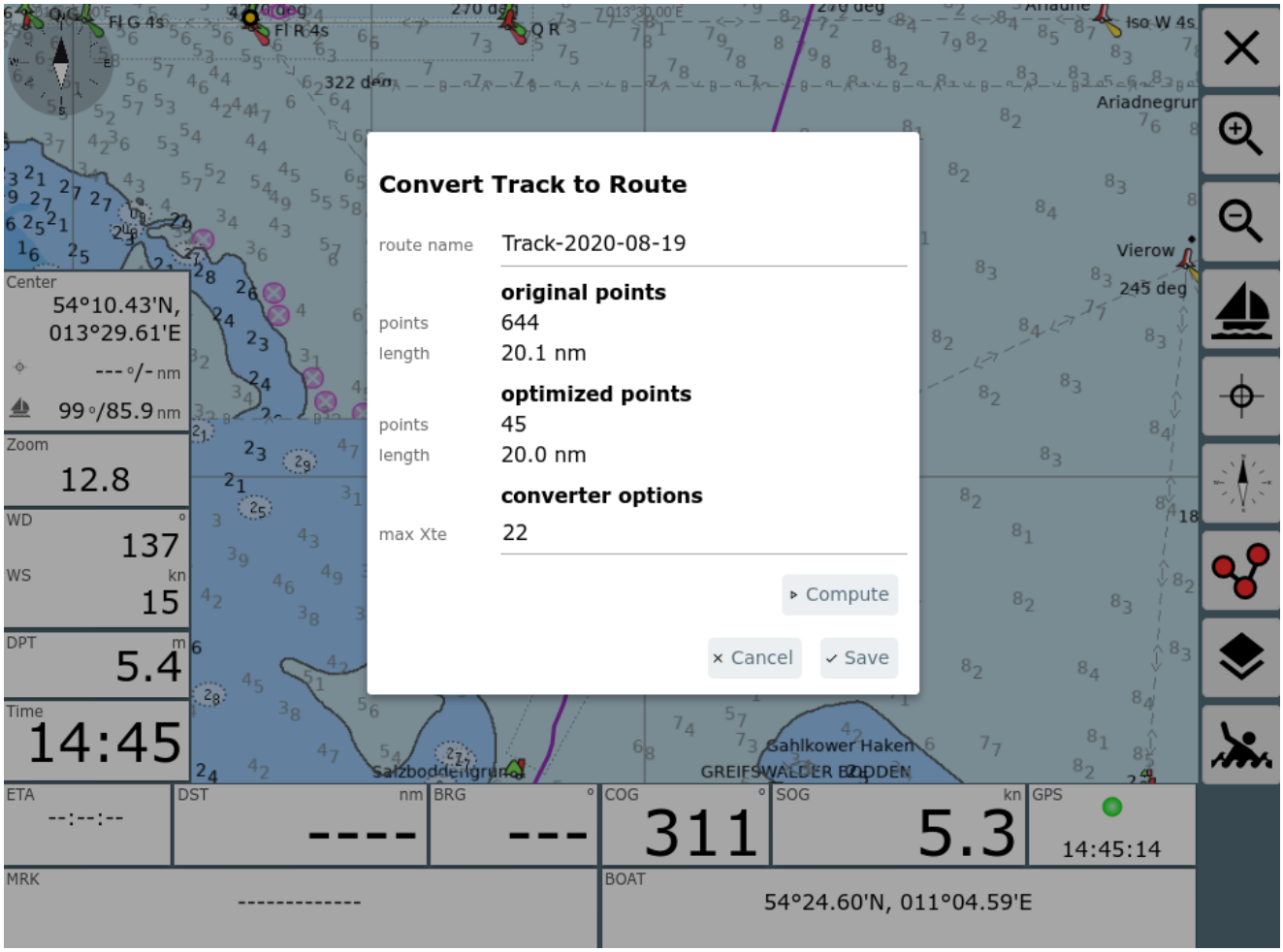


Nach Klick auf den Convert Button erhält man den Konverter-Dialog.



Wie im Bild zu sehen, wird man typischerweise eine Warnung bekommen, dass die Anzahl der Punkte über 50 liegt.

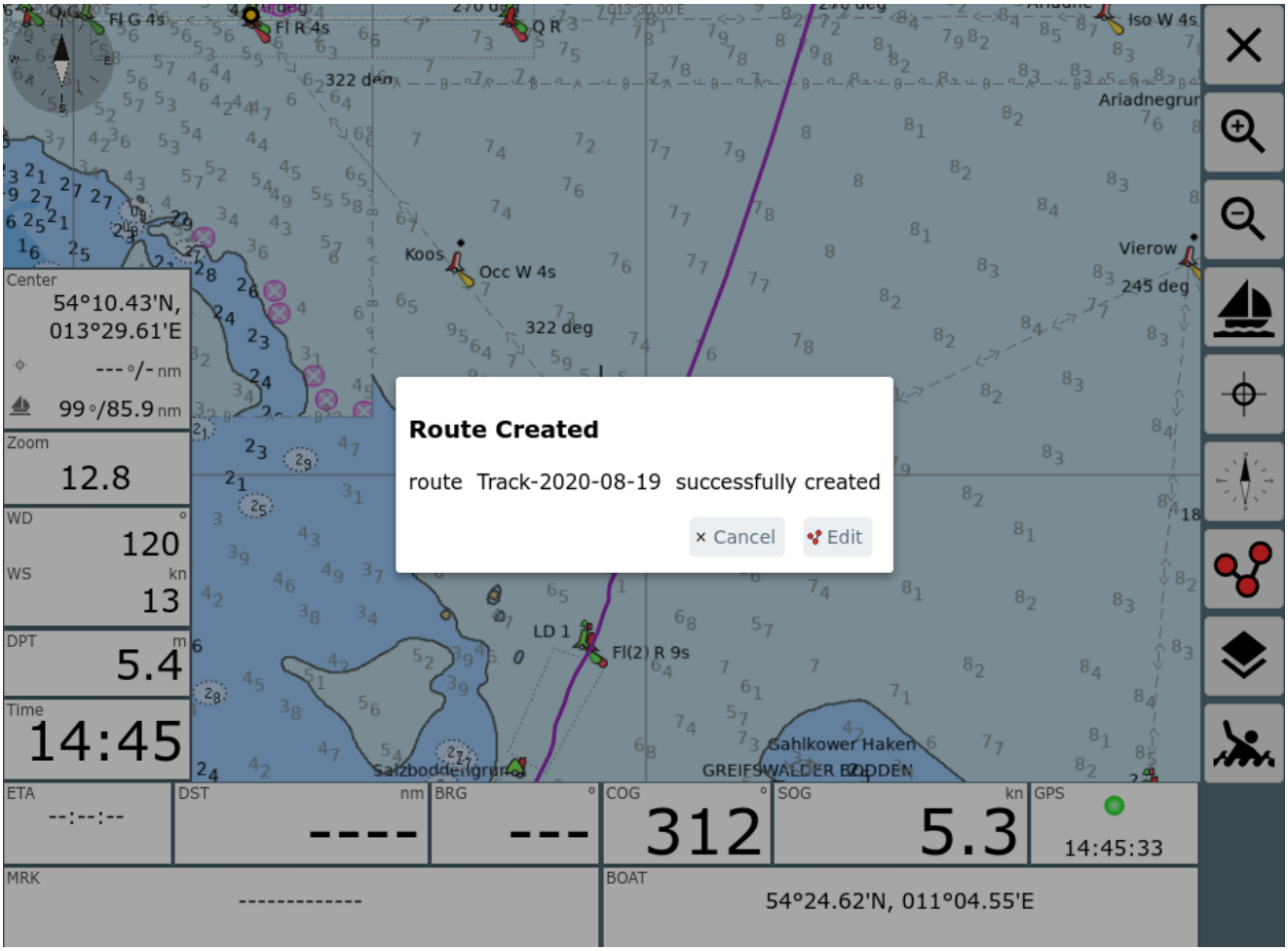
Über max Xte stellt man die maximal zulässige Abweichung ein und kann mit "Compute" eine Konvertierung starten. Falls das Ergebnis (Zahl der Punkte) noch nicht passend ist, kann man mehrere Durchläufe machen, mit veränderter Abweichung.



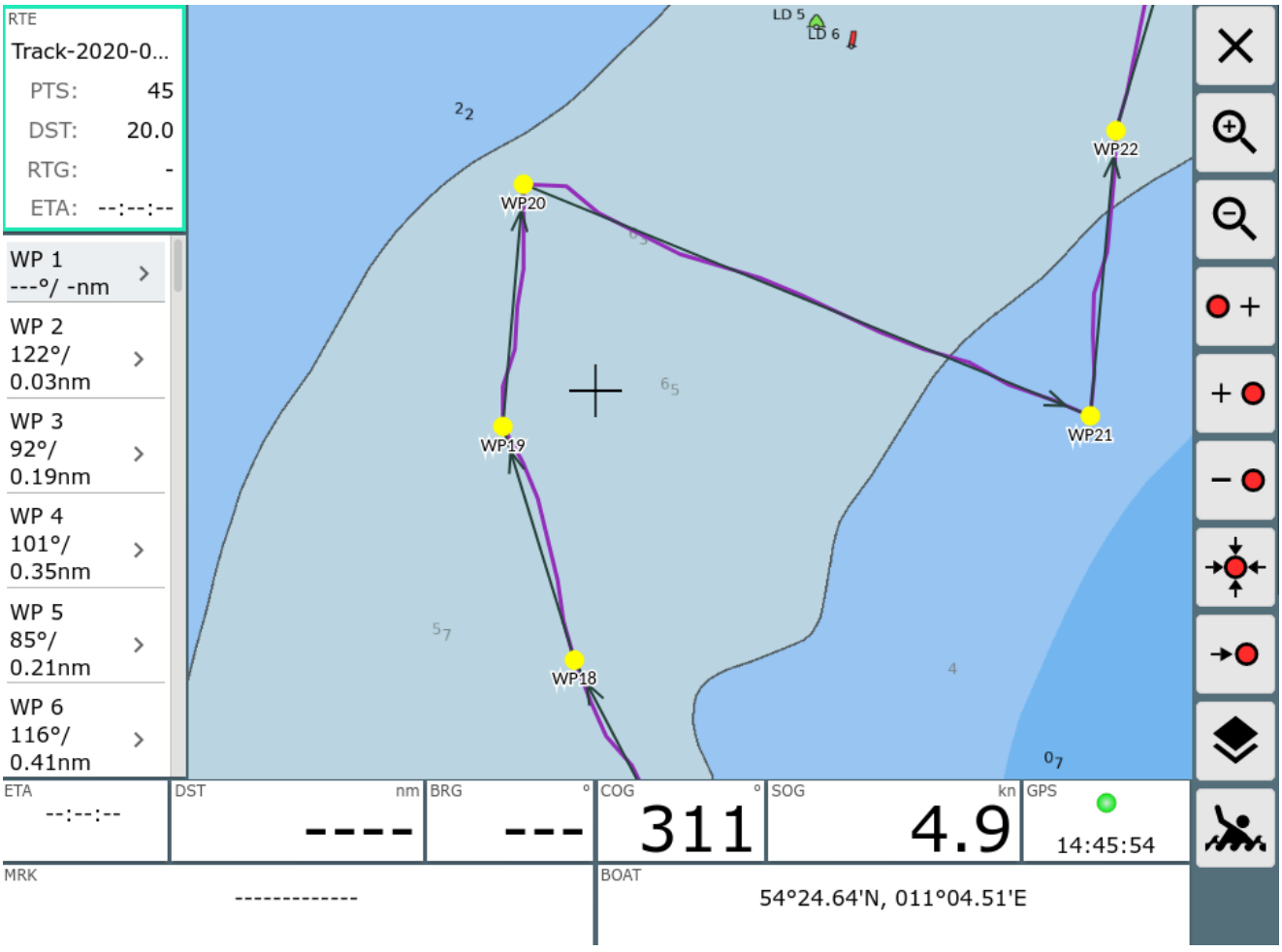
Im Beispiel wurde mit einer Abweichung von 22m eine ausreichend kleine Route erstellt.

Man kann im Dialog noch den Namen der enstehenden Route ändern (und ggf. entscheiden, ob eine bestehende Route gleichen Namens überschrieben werden soll).

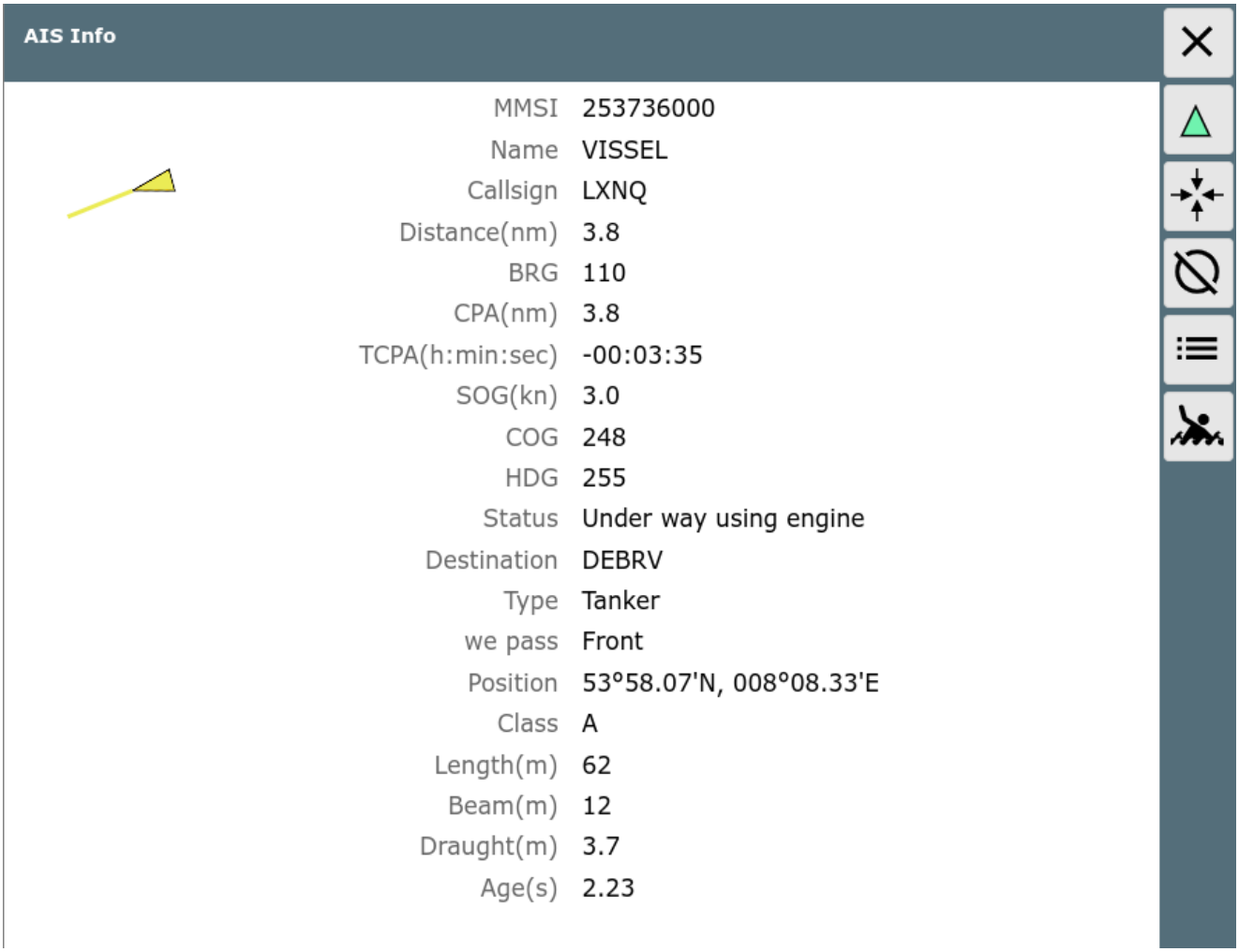
Nach dem Speichern kann man ggf. die enstandene Route direkt bearbeiten.



Hier ist es von Vorteil, wenn man den Track vorher als Overlay auf der Karte hatte, dann kann man jetzt leicht vergleichen, ob die Route alle wichtigen Punkte enthält.



Die AIS Info Seite



Von der Navigationsseite oder von einer Dashboard-Seite kommt man durch Klick auf die AIS-Anzeige oder auf ein AIS-Ziel in der Karte auf diese Info-Seite.

Die Details zum Ziel werden angezeigt.

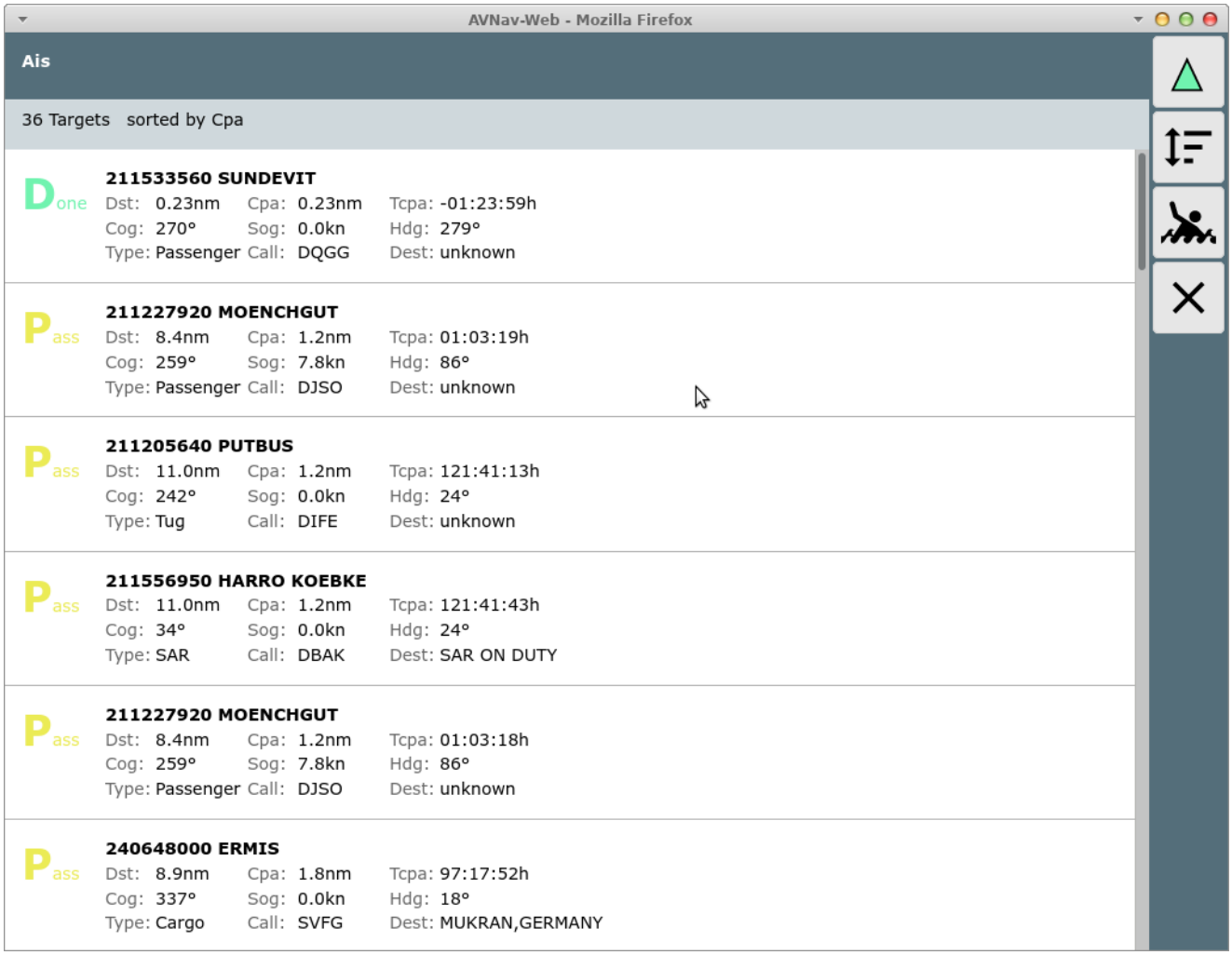
Buttons

Icon	Name	Funktion
	AisNearest	Zentriere auf das nächste Ziel und mache dieses wieder zum angezeigten Ziel auf der Navigationsseite
	AisInfoLocate	Zentriere die Karte auf dieses Ziel und zeige dessen Daten auf der Navigationsseite
	AisInfoHide	Verberge das AIS Ziel für einige Zeit (Einstellungen/AIS "hide time") - default 15s

	AisInfoList	Anzeige der Liste aller AIS Ziele
	MOB	Mann über Bord (siehe Hauptseite)
	Cancel	Zurück zur vorigen Seite

Die AIS Liste

Von der Ais Info gelangt man über den Button ☰ zu dieser Seite.



Buttons

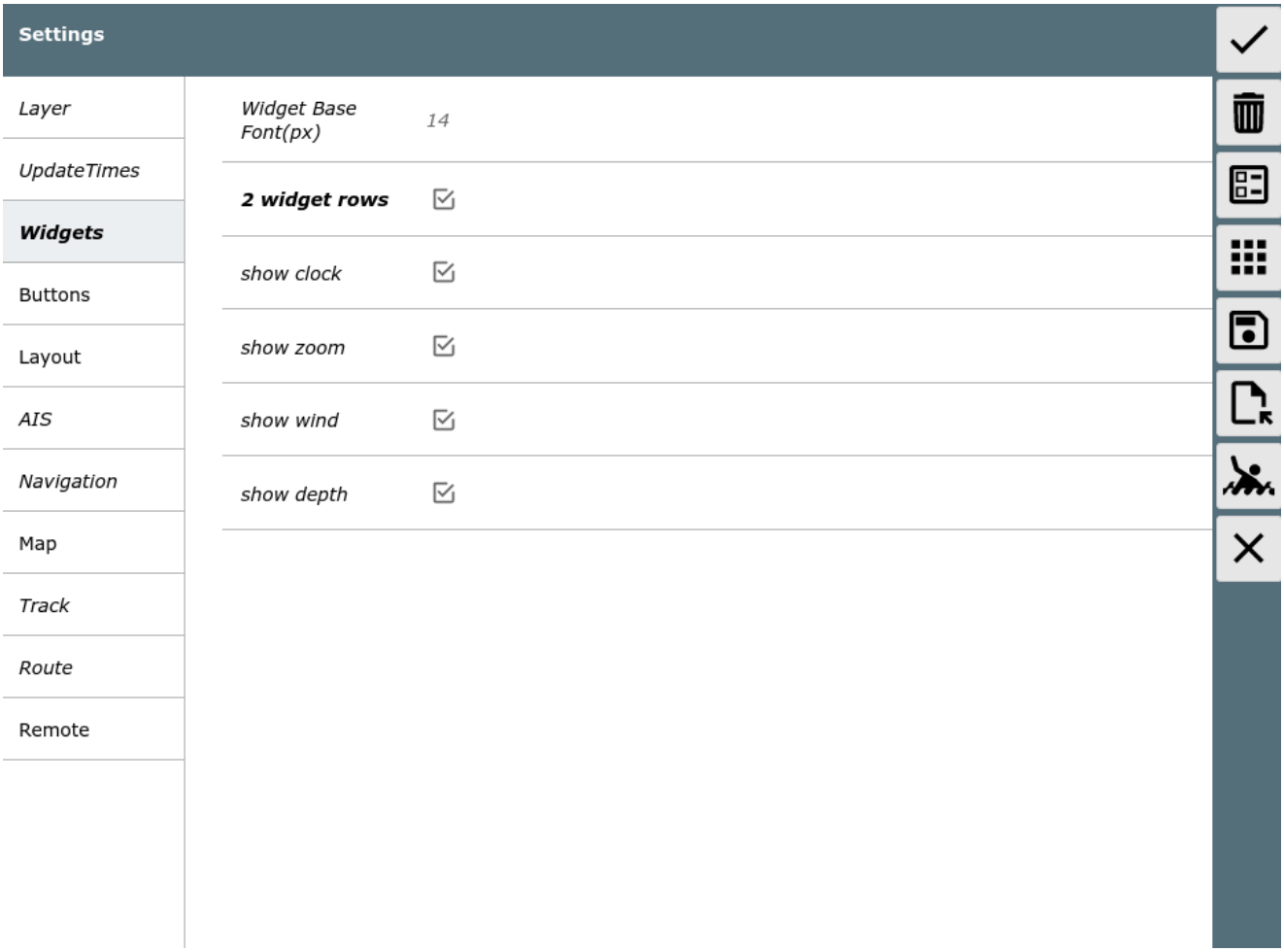
Icon	Name	Funktion
△	AisNearest	Zurück zum "Normalmodus", auf der Navigationsseite wird das nächste Ziel angezeigt.
☰	AisSort	Verändern der Sortierung der Liste
	MOB	Mann über Bord (siehe Hauptseite)
✕	Cancel	Zurück zur letzten Seite

Auf dieser Seite sieht man alle empfangenen AIS Ziele im Umkreis von ca. 10nm zur Bootsposition (sortiert nach CPA). Mit Klick auf eine Zeile springt man zurück zur Karten-Anzeige mit dem gewählten AIS Ziel im Zentrum. Das gewählte Ziel erscheint auch im AIS Info Fenster (dieses färbt sich gelb). Ein Klick auf den grünen Pfeil schaltet wieder in den „Normalmodus“ - d.h. Anzeige des nächsten AIS Zieles.

Durch Klick auf die 2. Überschriftszeile oder den Button  kann man die Sortierung verändern.

Die Einstellungsseite

Von der Hauptseite erreicht man über die Einstellungsseite.



Buttons

Icon	Name	Funktion
✓	SettingsOK	Akzeptieren der Änderungen und zurück zur vorigen Seite
🗑	SettingsDefaults	Rücksetzen aller Einstellungen auf Default-Werte
	SettingsLayout	Start des Layout Editors
	SettingsAddons	Wechsel zur Konfiguration von User Apps
	SettingsSave (ab 20220225)	Speichern der aktuellen Einstellungen auf dem Server

	SettingsLoad (ab 20220225)	Laden von Einstellungen vom Server (eine Auswahl wird angezeigt)
	MOB	Mann über Bord (siehe Hauptseite)
	Cancel	zurück zur vorigen Seite, Verwerfen der Änderungen

Auf dieser Seite können viele Parameter gesetzt werden, die die Darstellung beeinflussen. Alle diese Einstellungen werden für den aktuellen Browser (und die aktuelle URL von avnav) gespeichert. Sie sind also nicht auf dem Server gespeichert, sondern im Browser.

Die meisten Werte können separat auf ihre Defaults zurückgesetzt werden - das globale Rücksetzen kann über erfolgen. Wenn die Seite ohne Speichern verlassen werden soll, erfolgt eine Rückfrage.

Viele Einstellungen sollten selbst erklärend sein, einige etwas speziellere Einstellungen werden hier beschrieben.

Kategorie	Wert	Beschreibung	Default
Buttons	auto hide buttons on NavPage	Verberge die Button-Leiste auf der Navigationsseite nach einer einstellbaren Zeit Ein Klick auf den rechten Rand bringt dann die Button-Leiste zurück.	aus
Buttons	auto hide buttons on Dashboard Pages	Verberge die Button-Leiste auf den Dashboard-Seiten nach einer einstellbaren Zeit Ein Klick auf den rechten Rand bringt dann die Button-Leiste zurück.	aus
Buttons	time(s) to hide buttons on enabled pages	Zeit (in Sekunden) für das Verbergen der Button Leiste auf der Navigationsseite oder den Dashboard-Seiten	30
Buttons	show shade when buttons hidden	Zeige einen abgdunkelten Bereich auf der rechten Seite, wenn die Button-Leiste unsichtbar ist. Das markiert den Bereich auf	an

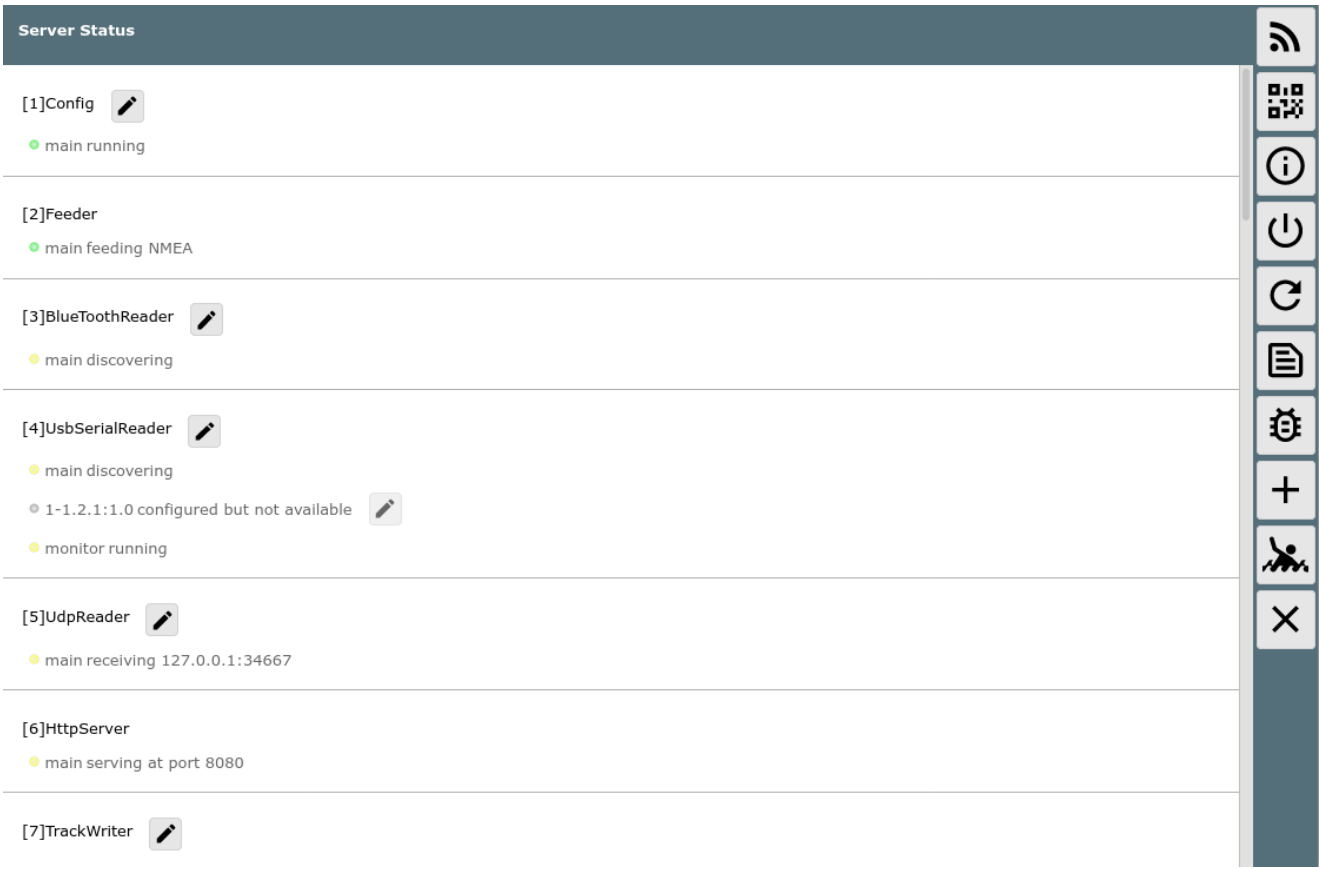
		der Karte, der geklickt werden muss, um die Button-Leiste zurück zu holen.	
Navigation	boat direction	Bestimmt, welcher NMEA Wert für die Anzeige der Richtung des Boot-Symbols genutzt wird. COG, HDT,HDM. Wenn HDT oder HDM gewählt wurde, aber der Wert nicht verfügbar ist, wird COG genutzt. Der Kursvektor wird in jedem Falle durch COG bestimmt! Ab 20220421: Wenn HDT oder HDM genutzt wird, wird das Boot als Boots-Symbol dargestellt, bei COG als Pfeil. Kann durch user icons angepasst werden.	COG
Navigation	add dashed vector for hdt/hdm	Zeige einen gestrichelten Kursvektor für HDT/HDM am Boot-Symbol an, wenn HDT/HDM für "boat direction" gewählt wurde	an
Navigation	Rotation Tolerance	Wenn Course Up gewählt wurde, wird die Kartenausrichtung nicht immer schnell an den aktuellen Kurs angepasst, solange die Kurs-Änderung unter diesem Wert bleibt. Das führt zu einer deutlich ruhigeren Anzeige.	15
Navigation	zero SOG detect (20220421)	Wenn das Boot nach COG ausgerichtet wird, kann entschieden werden, ab einer minimalen SOG keine Kurs-Vektoren mehr zu zeigen und das Boat als Kreis darzustellen	aus
Navigation	zero SOG detect below (20220421)	Switch off the course vectors, map rotation and show the boat as circle if SOG is below this value (only if zero SOG detect on)	
AIS	Class B rel size	Class B AIS Ziele können kleiner (oder grösser) als andere Ziele dargestellt werden.	0.6
AIS	reduce details in AIS	In der Listen-Anzeige kann die Menge der angezeigten Informationen pro Ziel begrenzt	aus

	list	werden. Besonders empfehlenswert, wenn auf langsamen Mobilgeräten die Anzeige der Liste träge wird.	
AIS	First AIS label Second AIS label Third AIS label	Auswahl der Informationen, die in der Nähe eines AIS Zieles auf der Karte gezeigt werden sollen.	first: name/mmsi
AIS	only show moving AIS targets	Wenn eingeschaltet werden nur bewegliche AIS Ziele angezeigt	aus
AIS	min speed (kn) for AIS target display	nur wenn "only show moving AIS targets" - Die minimale Geschwindigkeit, um ein AIS Ziel als bewegt zu sehen.	0.5kn
Map	automatic zoom	Wenn die Karte mit der Boot-Position verschoben wird, wird der Zoom angepasst, wenn sie in einen Bereich bewegt wird, auf dem z.B. auf demn aktuellen Zoom Level keine Kacheln vorhanden sind.	an
Map	float map behind buttons	Wenn dieser Schalter aktiv ist, "schweben" die Buttons und Anzeigen über der Karte und haben keinen dunklen Hintergrund.	aus
Map	Increase Fonts on High Res	Vergrößere die Symbole und Texte auf hochauflösenden Displays	an
Map	scale the map display	Manche Rasterkarten haben eine sehr feine Darstellung, so das u.U. Schrift schlecht lesbar ist. Mit dieser Einstellung kann die Anzeige der Karten-Kacheln vergrößert (oder auch verkleinert) werden, um sie an die eigenen Bedürfnisse anzupassen.	1
Map	zoom up lower layers	Wenn in einer Karte für den aktuell gewählten Zoom-Level keine Kacheln	4

		vorhanden sind, werden niedriger aufgelöste Kacheln geladen und enstprechend vergrößert. Da das jeweils neue Abfragen zum Server erfordert, kann das u.U. die Performance beeinflussen. Der Wert gibt an, wieviele kleinere Zoom-Level probiert werden, um eine Kachel zu finden.	
Map	zoom up lower layers for online sources	Das gleiche Verhalten wie "zoom up lower layers" für Karten, die direkt aus dem Netz geladen werden (wirkt auch für o-charts). In den meisten Fällen macht es für solche Quellen keinen Sinn - daher im default aus (0).	0
Map	Click Tolerance	Der "Einfangbereich" in Pixeln für den Click auf Kartenobjekte. Ein grosser Bereich führt u.U. dazu, das die falschen Objekte angeklickt werden. Ein zu kleiner Bereich macht den Klick sehr mühsam auf touch-Geräten.	60
Map	lock boat mode	Hier kann das Verhalten des Lock Buttons  auf der Navigationsseite beeinflusst werden, d.h. an welcher Stelle auf dem Bildschirm das Boot festgehalten werden soll. center: Bildschirm-Mitte current: aktuelle Position des Bootes ask: zeige einen Auswahldialog an	center
Route	Approach	Entfernung zum Wegpunkt (in Metern) ab der ein potentielles Weiterschalten zum nächsten Wegpunkt erfolgt wenn: - die Entfernung zum Wegepunkt zunimmt - die Entfernung zum nächsten Wegepunkt abnimmt	
Remote	...	Siehe die Beschreibung	

Die Server/Status Seite

Von der Startseite kann man über die Server/Status-Seite erreichen – hier werden interne Status-Informationen des Servers angezeigt.



Buttons

Icon	Name	Funktion
	StatusWpa	Zur Wifi Konfiguration für die Verbindung mit einem externen WLAN (nicht unter Android, nur sichtbar wenn konfiguriert)
	StatusAddresses	Zur Anzeige der Server Adressen. Es werden QR codes angezeigt, die sich einfach mit einem anderen Gerät scannen lassn, um dieses zu verbinden.
	StatusAndroid	Zu den Android Einstellungen wechseln (nur Android)
	StatusShutdown	Starte ein Herunterfahren des Servers (nicht Android). Der Server wird geordnet heruntergefahren.

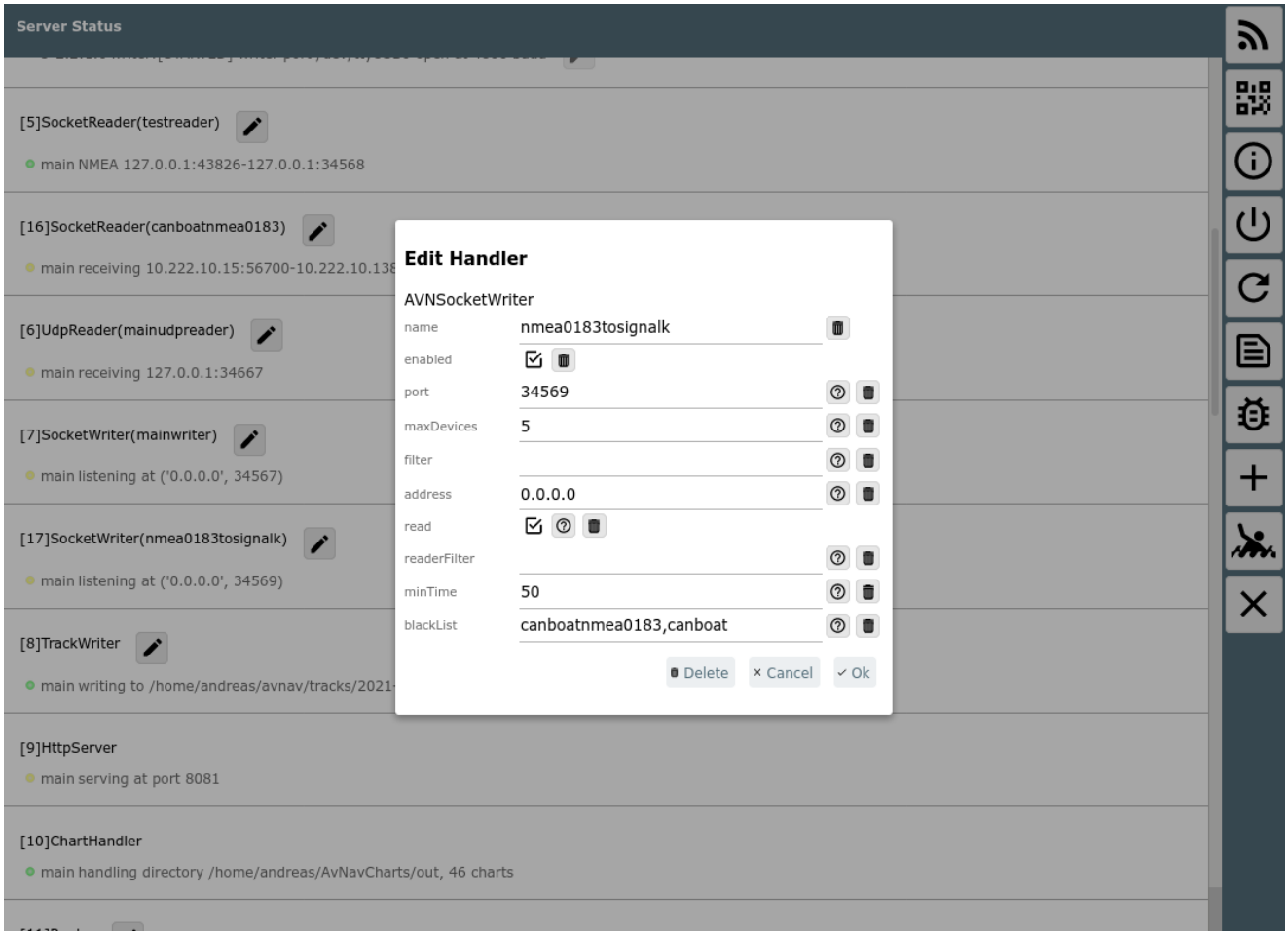
		Es erfolgt eine Sicherheitsabfrage. Danach sollte gewartet werden, bis die Titelzeile sich rot färbt.
	StatusRestart	Restart der AvNav Server Software (neu 20210322, nicht Android)
	StatusLog	Anzeige des Server Logs (ca. 300000kByte vom Ende), auch ein Download ist möglich (neu 20210322, nicht Android)
	StatusDebug	Anzeige eines Dialogs um für eine gewisse Zeit den debug Modus im Server anzuschalten. Ausserdem kann ein Filter für die Log-Ausgabe gesetzt werden. (nicht Android)
	MainInfo	Anzeige von Lizenz- und Datenschutz-Informationen
	StatusAdd	Hinzufügen eines neuen Handlers (z.B. serieller Input/Output, Socket Verbindung,...)
	MOB	Mann über Bord (siehe Hauptseite)
	Cancel	zurück zur vorigen Seite.

In der Liste werden die Status Informationen der verschiedenen Bestandteile des Servers angezeigt.
Man kann hier erkennen, ob z.B. Verbindungen vorhanden sind, ob NMEA Daten empfangen werden oder ob bestimmte Vorgänge ablaufen. Das ist insbesondere zur Fehlersuche wichtig. Die Seite aktualisiert sich ständig.

Server Konfiguration

Neu 20210322, Android ab 20210424 - siehe Android-Beschreibung.

Der Server hat eine Reihe von internen "Handlern". Die meisten können auf dieser Seite konfiguriert werden. Dazu nutzt man  neben der Status Information. Das erzeugt einen Dialog um die Einstellungen des Handlers anzupassen.



Für die meisten Einstellungen ist ein Hilfe Button verfügbar, der eine kurze Erklärung liefert. Mit dem Button kann man einen Wert auf seinen Default zurücksetzen. Wenn der Handler das unterstützt, kann er hier auch gelöscht werden. Die meisten Handler untertsützen auch ein enable/disable. So kann man einen Handler meist einfach auf disabled setzen, ohne ihn direkt zu löschen. Eine ausführlichere Beschreibung der Handler und ihrer Parameter findet sich in der Konfigurationsbeschreibung.

Alle Änderungen, die auf dieser Seite vorgenommen werden, werden in die Datei avnav_server.xml geschrieben, werden aber ohne Restart sofort wirksam. Die Anzeige auf der Seite benötigt typischerweise einige Sekunden bis die neuen Werte sichtbar werden.

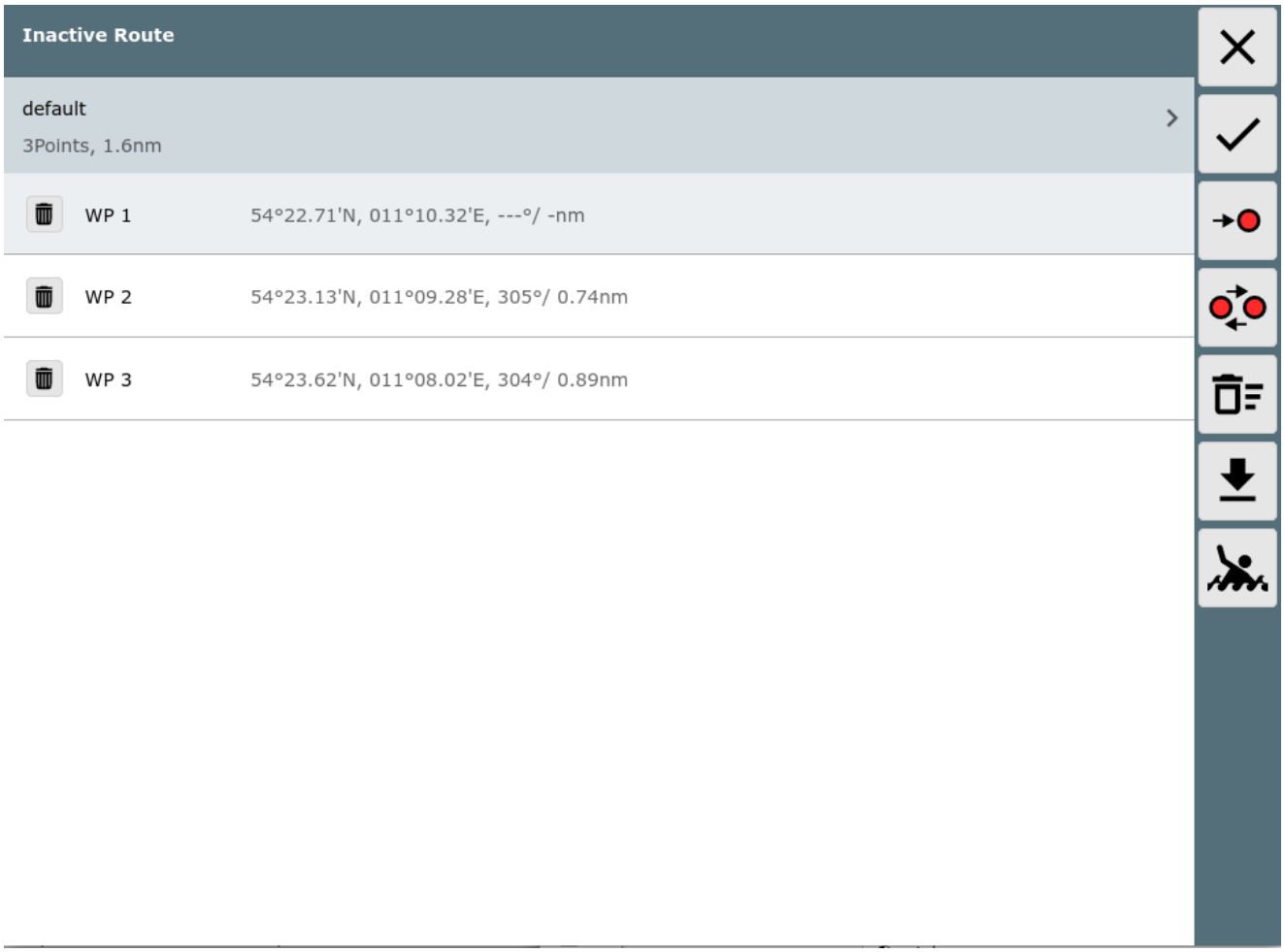
Ein neuer Handler kann mit dem + Button hinzugefügt werden.

Wenn man einen Port für eine IP Verbindung oder ein serielles Interface auswählt, prüft AvNav, das diese Resource nicht mehrfach genutzt wird.
Wenn die Speicherung unter Umständen zu einer korrupten avnav_server.xml führt, wird AvNav beim nächsten Start auf die letzte funktionierende Version zurückfallen. In der Sektion "Config" wird dann ein entsprechender Fehler angezeigt.

Route Liste Seite

Übersicht

Diese Seite wird erreicht durch Klick auf "Edit" im Route Dialog innerhalb des Route Editor.

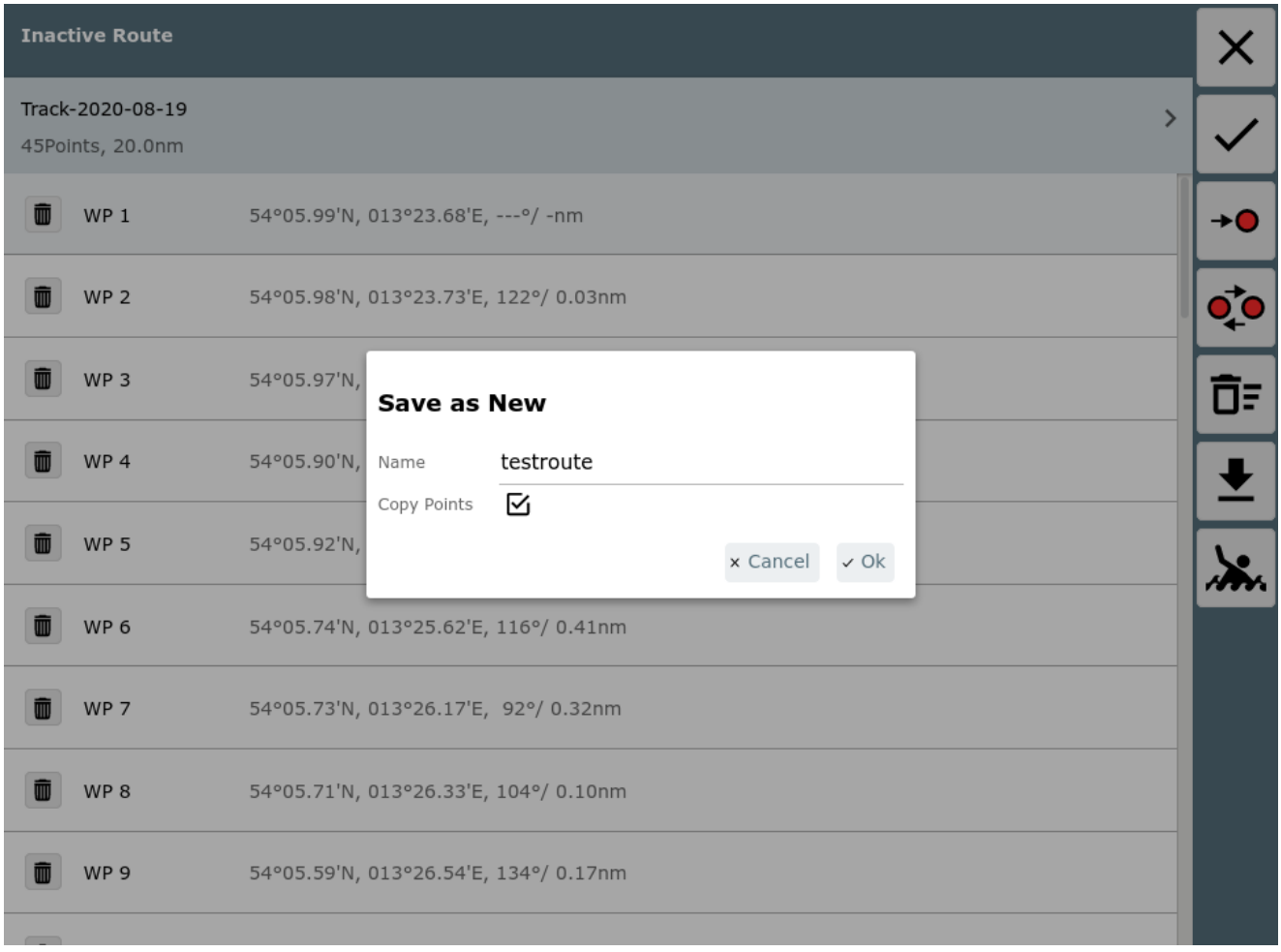


Buttons

Icon	Name	Funktion
✓	RoutePageOk	Änderungen übernehmen und zum Route Editor zurückkehren.
→●	NavGoto	Navigation zum ausgewählten Wegepunkt starten Der ausgewählte Punkt ist grau hinterlegt

	NavInvert	Reihenfolge der Punkte in der Route umkehren
	NavDeleteAll	Alle Punkte aus der Route löschen
	RoutePageDownload	Wechsel zur Files/Download Seite, Unterseite Routen. Es kann eine Route aus der Liste ausgewählt werden, Routen können heruntergeladen oder hochgeladen werden. Nach Auswahl einer Route Rückkehr zu dieser Seite mit der gewählten Route.
	MOB	Mann über Bord (siehe Hauptseite)
	Cancel	Bearbeitung abbrechen, zurück zum Route Editor

Auf dieser Seite kann man alle Punkte bearbeiten (Klick) oder die Route unter einem neuen Namen speichern (Kopie). Dazu klickt man auf den Namen der Route und erhält einen Dialog.

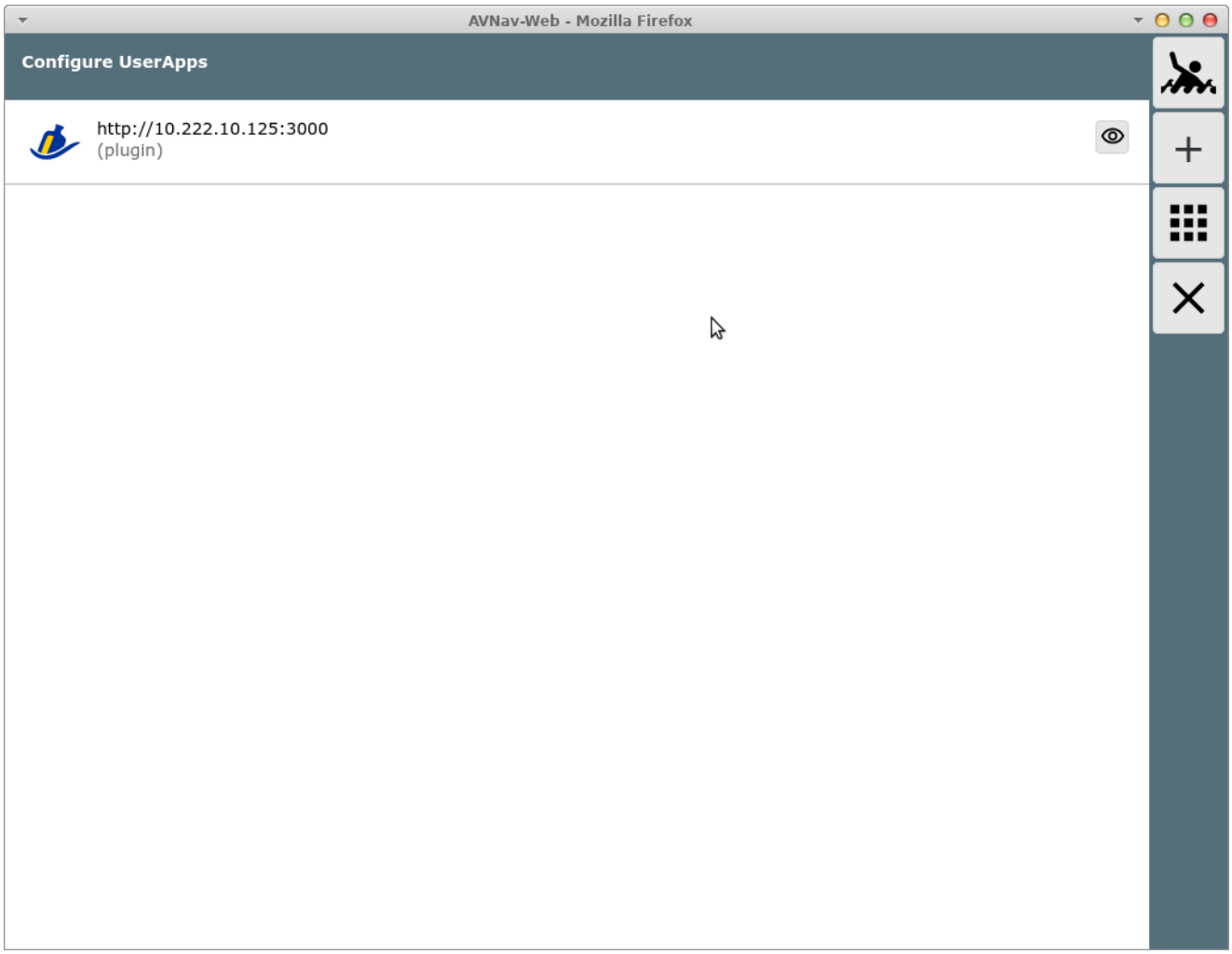


Danach über den OK Button zurück zum Route Editor um die Route weiter zu bearbeiten.

Konfiguration von User Apps

Von der Einstellungsseite erreicht man über den Button diese Konfigurationsseite.

Eine "User App" ist eine externe oder interne HTML Seite, die auf der "User App" Seite in einem iframe angezeigt wird.



Alle konfigurierten User Apps werden angezeigt.

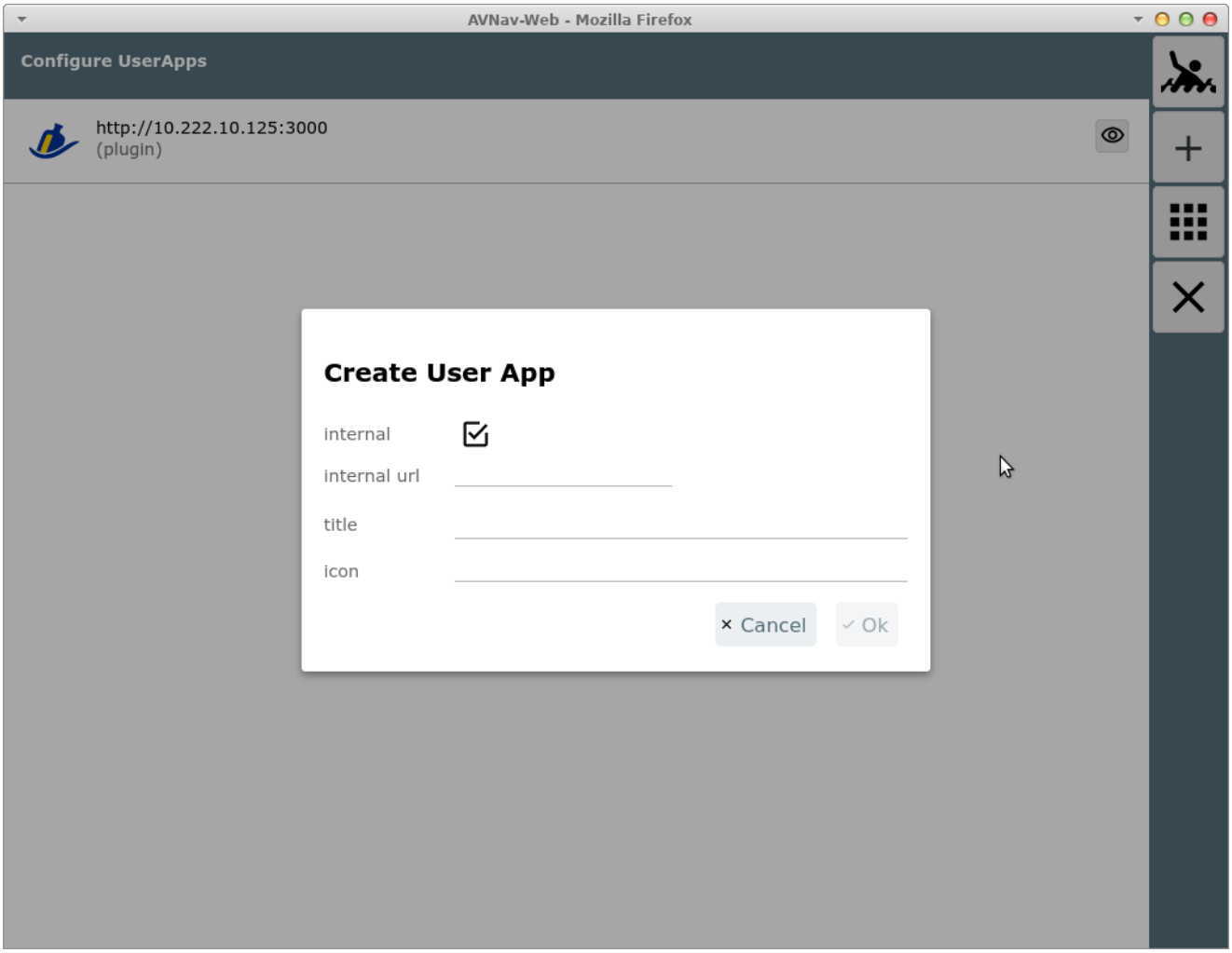
Buttons

Icon	Name	Funktion
	MOB	Mann über Bord (siehe Hauptseite)

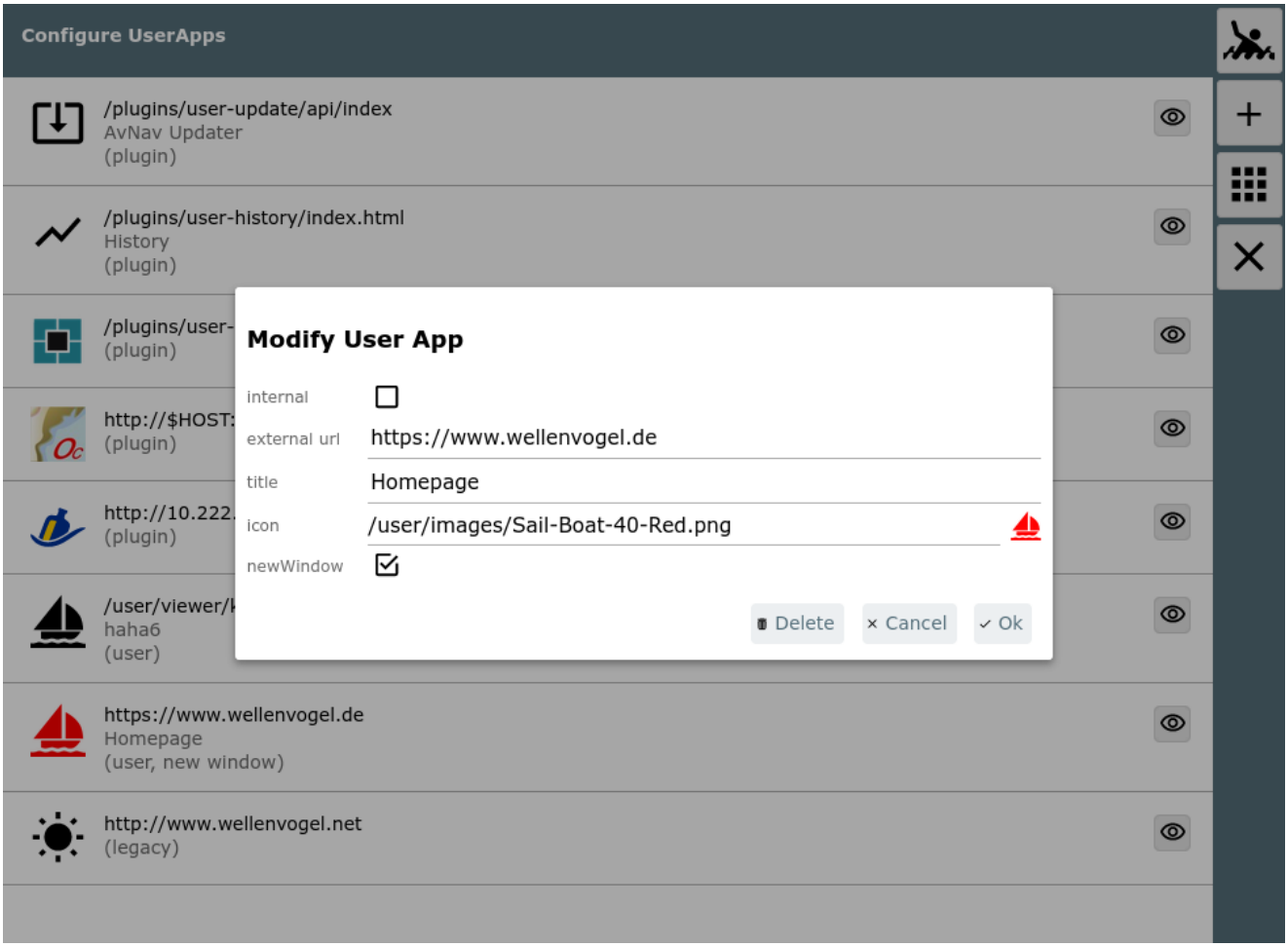
+	AddonConfigPlus	Neue Konfiguration hinzufügen
⠿	AddonConfigAddOns	User App Seite anzeigen
✕	Cancel	zurück zur letzten Seite

Für jede Konfiguration wird jeweils die URL und der optionale Titel angezeigt. Ausserdem wird angezeigt, woher die Konfiguration stammt (im Bild z.B. plugin). Falls die Konfiguration nicht gültig ist (z.B. Icon nicht gefunden) wird "invalid" dazu angezeigt. Durch Klick auf die Konfiguration kann sie bearbeitet werden (nur user Konfigurationen). Ein Klick auf 👁 geht zur User App Seite, und die entsprechende Konfiguration wird dort angezeigt.

Beim Hinzufügen einer Konfiguration oder beim Bearbeiten wird ein Dialog angezeigt.



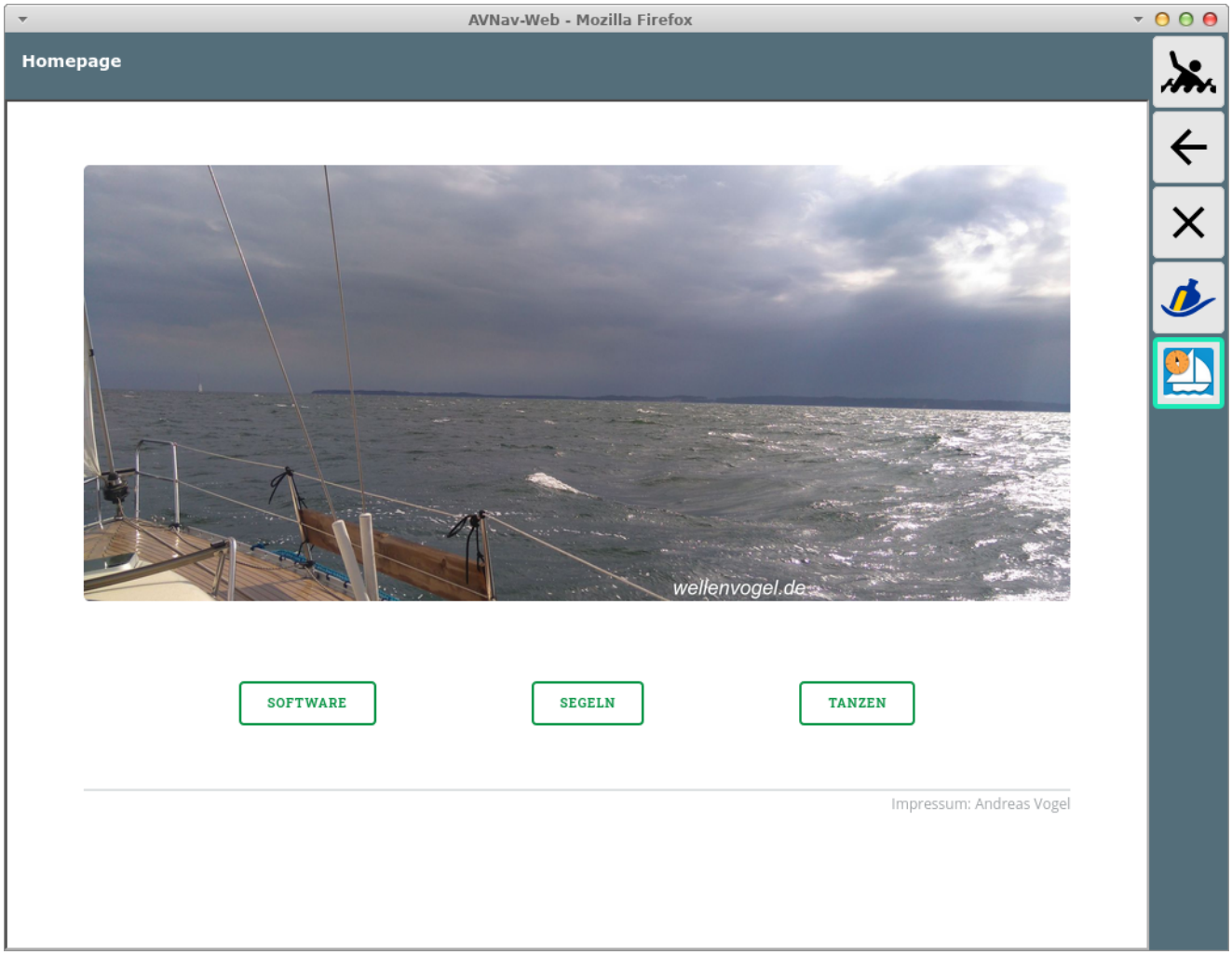
Hier kann zunächst gewählt werden, ob eine externe oder eine interne HTML Seite angezeigt werden soll (im Bild: interne aktiv). Externe Seiten müssen mit einer URL der Form http(s)://... angegeben werden. Falls sie auf dem gleichen Server liegen wie AvNav, sollte statt des Hostnamens der String $HOST angegeben werden. Das wird dann von AvNav automatisch in die richtige Adresse übersetzt.



Interne HTML Seiten müssen voher in das Nutzer Dateien Verzeichnis hochgeladen worden sein.

Ausserdem muss für jede User App eine Icon Datei (svg, png, jpeg) existieren. Diese muss ebenfalls über die Files/Download Seite entweder zu Nutzer-Dateien oder zu Bildern hochgeladen worden sein.
Nach Auswahl der URL und des Icons kann noch ein Titel angegeben werden. Wenn dieser leer bleibt, wird kein Titelbalken auf der Seite angezeigt.
Nach Speichern der Konfiguration kann sie über 👁 getestet werden.

Wenn "newWindow" (ab Version 20220225) ausgeschaltet ist, wird die Seite in einem iframe angezeigt, sonst in einem neuen Browserfenster oder Browsertab.

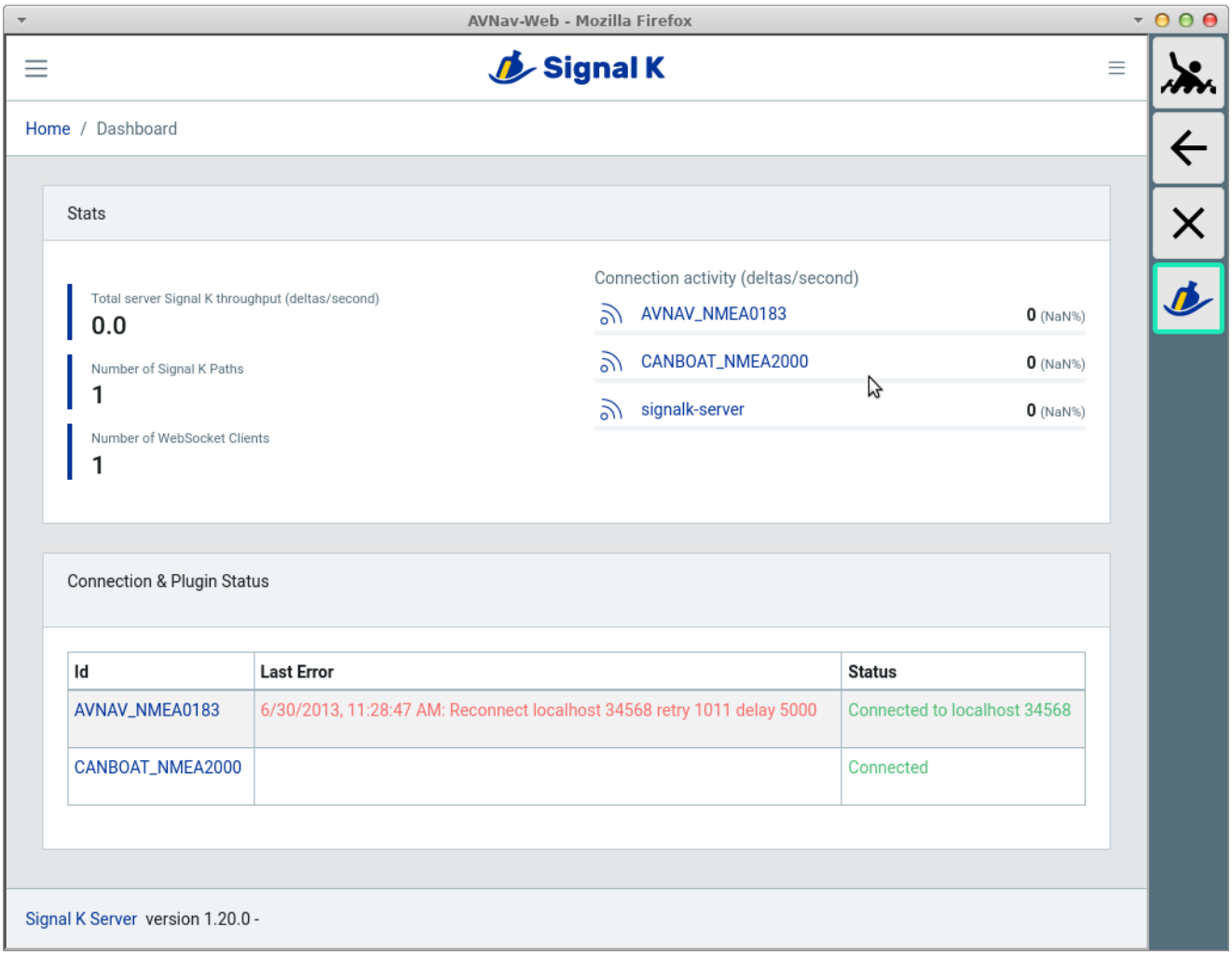


Wenn eine interne URL gewählt wurde und die Datei wird über die Files/Download Seite gelöscht, wird auch die User App Konfiguration entfernt.
Wenn eine Icon-Datei gelöscht wird, wird die User App Konfiguration nicht entfernt. Sie wird allerdings ungültig und wird auf der User App Seite nicht mehr angezeigt.
Man kann dann hier die Konfiguration wieder korrigieren.

Wenn eine User App Konfiguration gespeichert wird, schreibt der Server seine Konfigurationsdatei (avnav_server.xml) neu.
Damit er im Fehlerfall noch starten kann, wird beim jedem erfolgreichen Start eine Datei avnav_server.xml.ok erzeugt - diese wird beim nächsten Start genutzt, falls avnav_server.xml kaputt ist.

Die User App Seite

Von der Hauptseite erreicht man über den ⠿ Button die Seite mit den sogenannten User Apps. Der Button ist nur sichtbar, wenn solche User Apps konfiguriert wurden (oder z.B. durch Plugins eingerichtet wurden - signalk).



Buttons

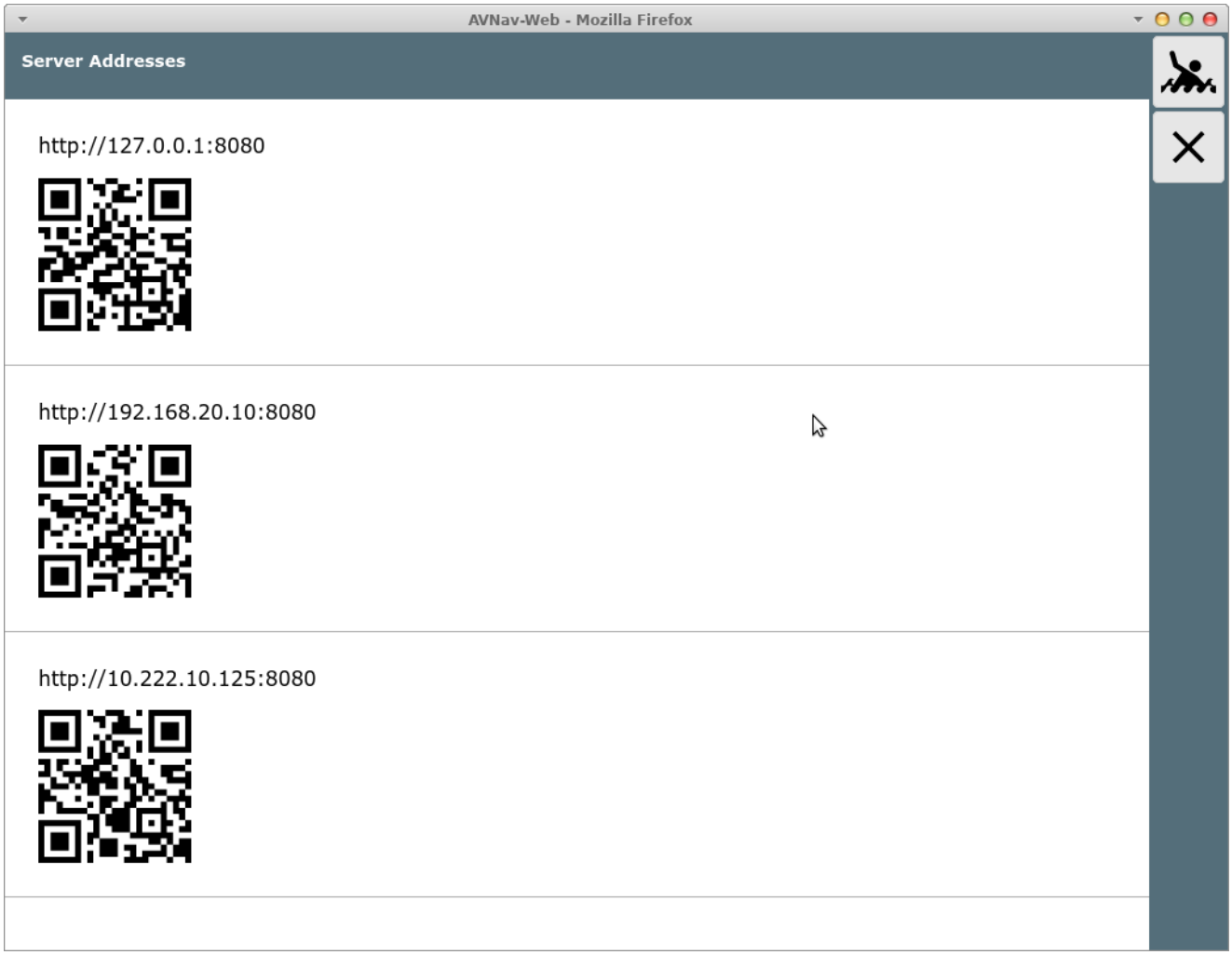
Icon	Name	Funktion
	MOB	Mann über Bord (siehe Hauptseite)
←	Back	Back Button des Browsers, kann für die Navigation in den angezeigten Seiten genutzt werden. Geht nicht zurück zur vorigen Seiten in AvNav!
✕	Cancel	zurück zur Hauptseite

andere Icons	---	Auswahl der konfigurierten User App

Auf dieser Seite werden externe oder interne HTML Seiten angezeigt, die als User Apps konfiguriert wurden. Die Anzeige erfolgt dabei in einem iframe. Manche externen Seiten lassen das nicht zu - das muss man ggf. ausprobieren.

Im Beispiel wird die signalk Web Oberfläche angezeigt, die durch das signalk Plugin konfiguriert wurde.

Die Adressen Seite



Es werden hier die Adressen angezeigt, über die AvNav externe Verbindungen annimmt. Im normalen Setup, wenn AvNav als Access Point arbeitet, und man sich mit dem avnav WLAN verbindet, sind es die mit 192.168.... beginnenden Adressen.
Wenn noch die Verbindung zu einem externen WLAN konfiguriert wurde, kann auch eine solche Adresse noch sichtbar sein (wie im Bild).

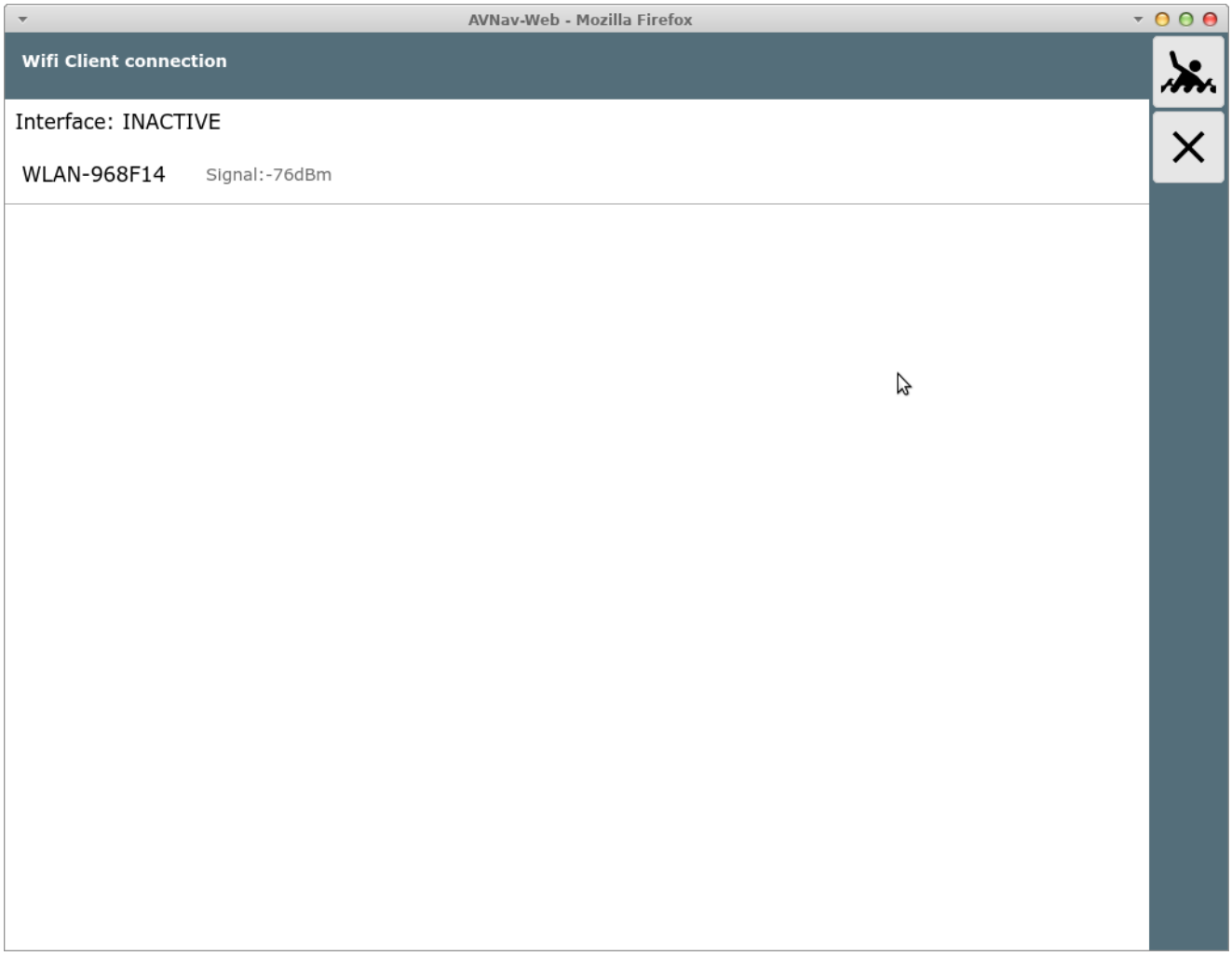
Durch Scannen dieser Codes mit einem anderen Tablet kann man leicht auf den gleichen Server (Raspberry) zugreifen. Man muss nur entscheiden, welche der Adressen zu benutzen ist - je nachdem, in welchem Netzwerk man mit dem Raspberry verbunden ist.

Diese Möglichkeit der Verbindung ist eine Alternative zu der in der Einführung genannten Variante der Adresseingabe bzw. der Bonjour Nutzung.

Die Wifi Konfigurationsseite

- nicht auf Android -

Von der Statusseite kommt man mit dem Button zu dieser Seite.



Diese Seite wird nur angezeigt, wenn das Wifi Client Handling konfiguriert ist (als default an) und ein WLAN-Adapter in der richtigen USB-Buchse eingesteckt wurde.

Im Bild gezeigt hier ein Raspberry Pi 3. Bei den neueren mit USB 3.0 (blaue Buchsen) muss er in der blauen Buchse auf der Leiterplatten-Seite stecken.

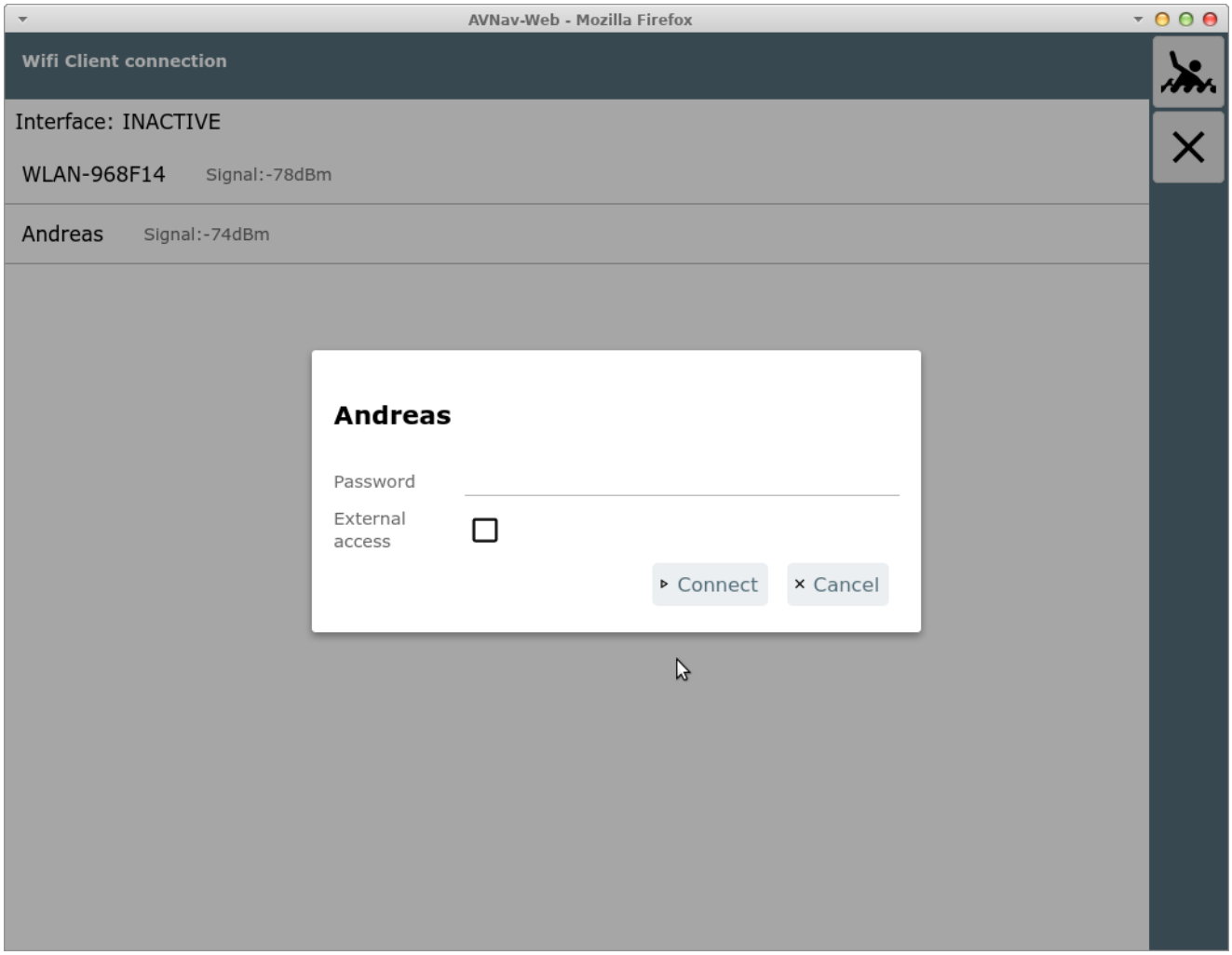
Auf Systemebene muss ein WLAN "wlan-av1" angezeigt werden. In der Konfiguration (avnav_server.xml) muss eine Eintrag der Form

```
<AVNWpaHandler wpaSocket="/var/run/wpa_supplicant/wlan-av1">
</AVNWpaHandler>
```

vorhanden sein.

Auf der Seite werden alle in Reichweite befindlichen oder konfigurierten WLANs angezeigt.

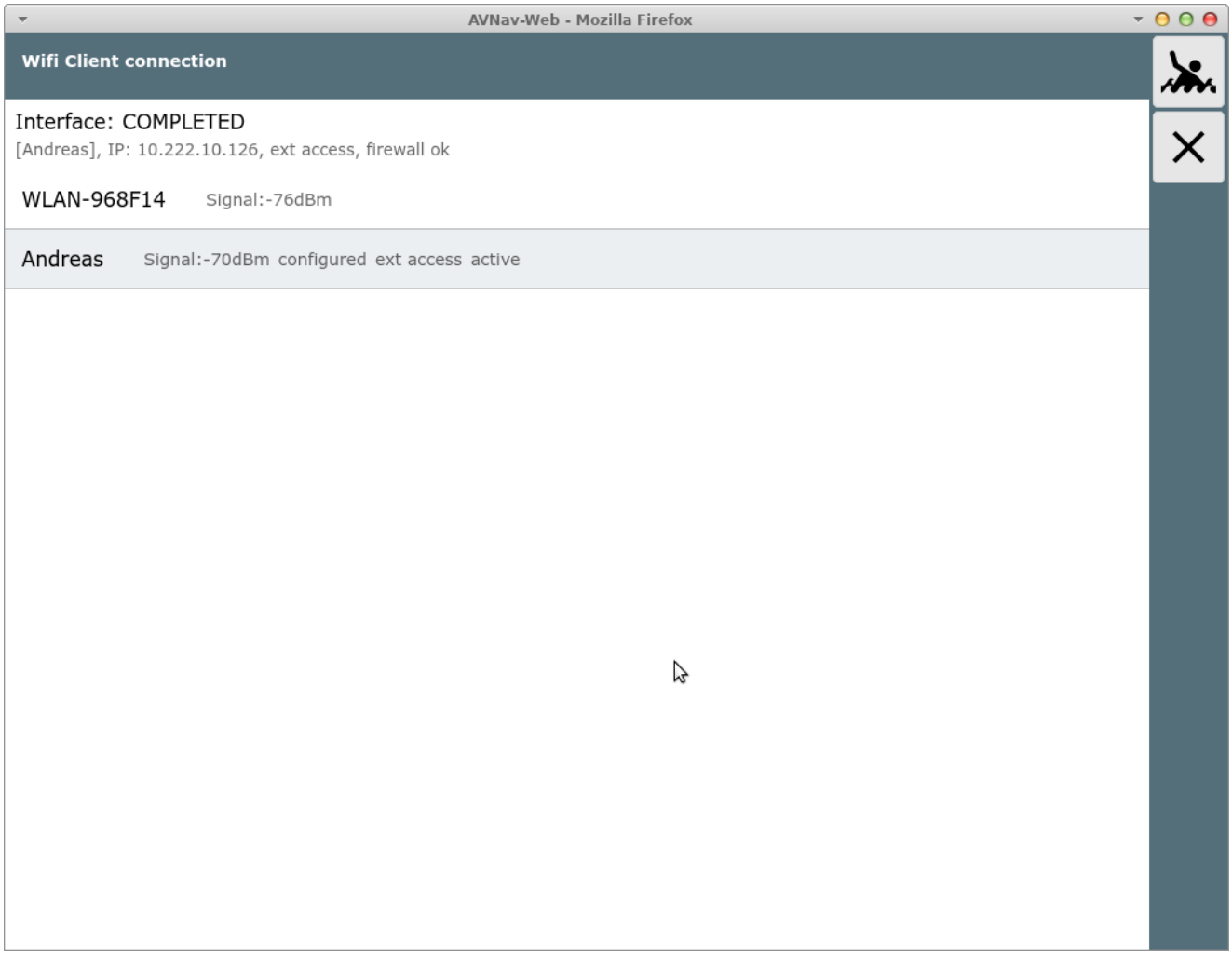
Durch Klick auf ein angezeigtes WLAN kann die Verbindung hergestellt werden.



Falls ein Zugriff von aussen auf den Raspberry erfolgen soll, kann hier "External Access" eingeschaltet werden.

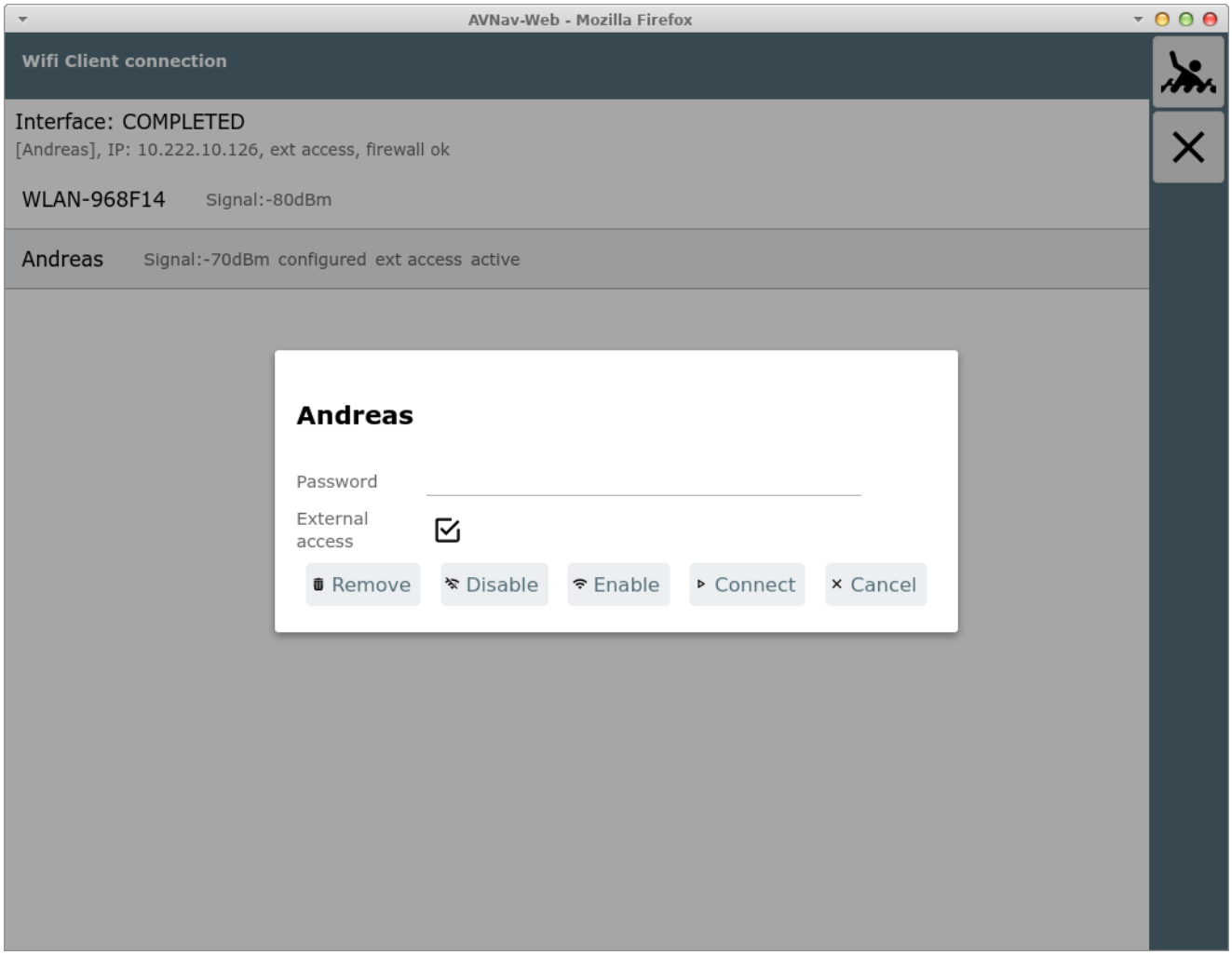
Achtung: Das sollte nicht in öffentlichen WLANs aktiviert werden, da AvNav keinen besonderen Schutz bietet und sonst jeder im gleichen WLAN ungehindert zugreifen kann. Es kann aber z.B. genutzt werden, wenn man einen LTE Router nutzt und seine Geräte direkt mit diesem verbunden sein sollen.

Wenn die Verbindung hergestellt wurde, wird das entsprechend angezeigt, das verbundene Netzwerk ist grau hinterlegt.



Während des Verbindungsaufbaus werden bei "Interface" Zustandsinformationen angezeigt.

Falls man das WLAN trennen möchte, kann man durch erneuten Klick auf das Netzwerk dieses trennen, deaktivieren oder auch komplett aus der Konfiguration entfernen.



Die WLAN Informationen werden in der Datei /etc/wpa_supplicant/wpa_supplicant.conf gespeichert.

Fernsteuerung

Ab Version 20210619 bietet AvNav die Möglichkeit, die Anzeige auf einem Display-Gerät durch ein anderes Gerät oder vom Server aus fernzusteuern.

Das kann man beispielsweise nutzen, wenn man ein Gerät zur Anzeige in der Plicht verwendet, dieses jedoch nicht in Griffweite ist. Dann kann man z.B. mit einem am Körper befindlichen anderen Gerät - oder mit einer Fernsteuerung - die Anzeige auf dem Display umschalten.

Funktionen

Per Fernsteuerung können Seiten umgeschaltet werden und auf den Seiten kann teilweise die Anzeige verändert werden (z.B. Auswahl der angezeigten Dashboard Seite oder Auswahl der Karte).

In der Navigationsansicht wird das Verschieben der Karte, das Zentrieren und das Zoomen übertragen.

Konfiguration

Insgesamt bietet AvNav 5 Fernsteuerungskanäle an. Für jedes Gerät kann ausgewählt werden, welchen Kanal es nutzen soll und ob es auf diesem Kanal Kommandos senden oder empfangen soll (Einstellungen/Remote). Für eine funktionierende Fernsteuerung müssen also mindestens 2 Display Geräte auf den gleichen Kanal eingestellt werden und eines davon muss senden und das andere empfangen.

Man kann auch die Geräte so einstellen, das sie sowohl senden als auch empfangen. Hier wird nach einer Wartezeit nach einer lokalen Bedieung jeweils automatisch in den Empfangsmodus geschaltet.

Im Server muss der Handler für die Fernsteuerung aktiv sein (standardmässig an).

In der Server Variante (nicht Android) können Fernsteuerungskommandos auch per UDP empfangen werden (nur Kanal 0) - oder über ein Plugin.

Es existiert ein Plugin für die Open Boat Projects IR Fernbedienung von chrhartz.

Per UDP oder über ein Plugin können verschiedene Fernsteuerungs-Kommandos gesendet werden (siehe unter Technik).

Technik

Intern verbindet sich jeder Browser mit dem konfigurierten Fernsteuerungskanal auf dem Server (unter Nutzung von WebSockets). Auf diesem Kanal sendet oder/und empfängt er dann Kommandos (je nach Konfiguration).

Diese Kommandos sind entweder Tastendrücke oder etwas komplexere Funktionen z.B. zum Umschalten einer Seite.

Die Kommandos werden jeweils als String erwartet, über UDP mit einem abschliessenden NewLine.
Sie bestehen jeweils aus einem Typ und Parametern.

Tasten-Kommandos

Ein Tasten-Kommando besteht aus einem "K", einem Leerzeichen und dem Tastencode.

```
K Ctrl-
K a
```

Die Tasten lösen die Aktion entsprechend ihrer Konfiguration aus (siehe Tastatur-Steuerung).

Komplexe Kommandos

Diese Kommandos lösen direkt bestimmte Aktionen in AvNav aus. Die Parameter sind meist in JSON kodiert.
Sie sind primär zur Steuerung von einem Display zu einem anderen gedacht und ihre Form kann sich durchaus ändern.
Prinzipiell sehen sie wie folgt aus:

```
CP navpage
```

In diesem Falle: setze die Seite "navpage". Eine Liste der Kommandos findet sich im source code unter remotechannel.js.

AvNav Android

Nachdem die Navigations-App für die Nutzung mit dem Raspberry Pi bereits einige Jahre erfolgreich im Einsatz ist, gibt es jetzt auch eine Version, die direkt (ohne einen Raspberry) Geräten mit Android läuft, z.B Tablets oder Handys. Sie erfordert Android ab 4.4 (KitKat).

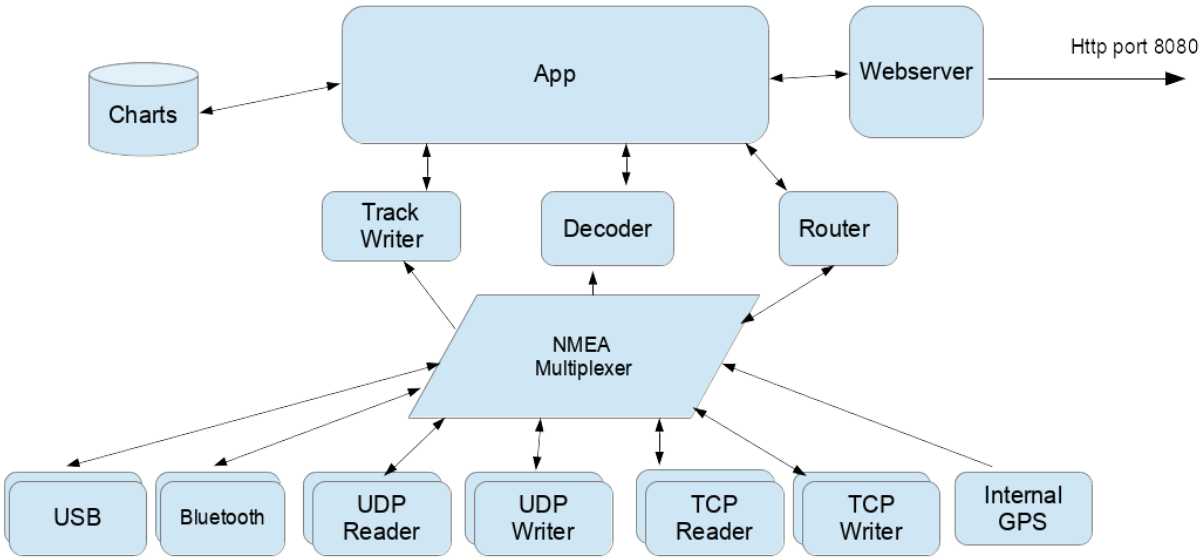
Die aktuellste Version kann man unter Downloads herunterladen oder im AppStore.
Ältere Versionen oder "daily builds" findet man über die Installationsbeschreibung.

Sie verpackt im Wesentlichen die Web-App (siehe Beschreibung) in eine Android Applikation - siehe dort auch die Verfahren zum Erzeugen/Konvertieren der Karten.

Funktion

(neu ab 20210424)

Die App besteht intern aus mehreren Funktionsblöcken .



Anav Android

Der NMEA Multiplexer verarbeitet NMEA0183 Daten von den verschiedenen Quellen. Neben dem internen Geräte-GPS können die Daten von Quellen wie TCP Verbindungen, UDP Ports, USB Geräten oder Bluetooth Verbindungen kommen. Die meisten Quellen unterstützen das gleichzeitige Senden und Empfangen von Daten. Es können jeweils mehrere Quellen des gleichen Typs konfiguriert werden (in der App-Konfiguration existiert für jede Quelle ein sogenannter "Handler").

Name in der App	Beschreibung
InternalGPS	Die Daten des internen GPS werden als NMEA-Daten eingespeist.
TcpReader	Eine TCP-Verbindung zu einem externen System. AvNav ist dabei ein TCP-Client und öffnet die Verbindung. Als Adresse kann sowohl eine IP-Adresse, als auch ein Hostname angegeben werden. Auch ein mDNS-Name wie avnav.local kann benutzt werden.
TcpWriter	AvNav ermöglicht es einer anderen App oder einem anderen System, sich zu verbinden und Daten zu empfangen. AvNav ist hier der TCP-Server.

UdpReader	AvNav empfängt UDP-Daten auf dem konfigurierten Port.
UdpWriter	AvNav sendet UDP-Daten zur konfigurierten Adresse und zum konfigurierten Port.
UsbConnection	Daten können über einen angeschlossenen USB-Seriell-Wandler gesendet umd empfangen werden. Das erfordert allerdings USB-OTG-Funktionalität auf dem Gerät.
Bluetooth	Eine Verbindung zu einem Bluetooth-Gerät. Das Pairing des Gerätes muss vorher außerhalb von AvNav erfolgen.
NMEA0183 service	Eine Verbindung zu einem System, das seine NMEA0183 Daten als TCP-Service über mDNS (Bonjour/Avahi) bereit stellt. AvNav öffnet eine TCP-Verbindung zu einem solchen Gerät.

Der NMEA-Multiplexer kann sehr flexibel konfiguriert werden. Zu jeder Verbindung lassen sich Eingangs- und Ausgangsfilter festlegen.

Der Multiplexer reicht intern die Daten auch an die anderen Bereiche der App weiter. Der Dekoder bereitet die Daten dann für die Nutzung innerhalb der App auf.

Die eigentliche App mit der Kartendarstellung und den Anzeigen kann einerseits ganz normal als Android-App genutzt werden. Parallel dazu kann der integrierte Webserver aktiviert werden. Das ermöglicht den Zugriff mit einem Browser von weiteren Geräten aus, so wie auch in der Server-Variante.

Der Anzeige-Teil der App kann beendet werden, sodass der Multiplexer allein im Hintergrund weiter läuft. Damit kann AvNav auch genutzt werden, um NMEA-Daten für andere Android-Apps bereitzustellen. In AvNav konfiguriert man dazu einen TcpWriter, in den zugreifenden Apps verbindet man sich über die Adresse 127.0.0.1 und den vorher konfigurierten Port).

Karten und gespeicherte Daten

Die Karten werden im gemf- Format erwartet.Ab Version 20200325 auch als "mbtiles" oder "*.xml files". Sie können in 2 Verzeichnissen liegen:

- im "Charts"- Verzeichnis unterhalb des Arbeitsverzeichnisses. Das Verzeichnis kann beim Start der App ausgewählt werden. Das Verzeichnis muß beschreibbar sein, es wird daher ab Android 4.4 und höher auf dem internen Speicher liegen müssen

- in einem beliebigen anderen Verzeichnis, vorzugsweise auf der SD Karte, als - "additional charts dir" in den Einstellungen
  Hier können allerdings keine "mbtiles"-Karten genutzt werden, diese müssen zwingend in das interne Verzeichnis hochgeladen werden.

Die Karten müssen auf das Gerät in eines der Verzeichnisse kopiert oder in der App hochgeladen werden . Zusätzlich sind einige Demo-Karten vorinstalliert, die allerdings eine Online-Verbindung erfordern.

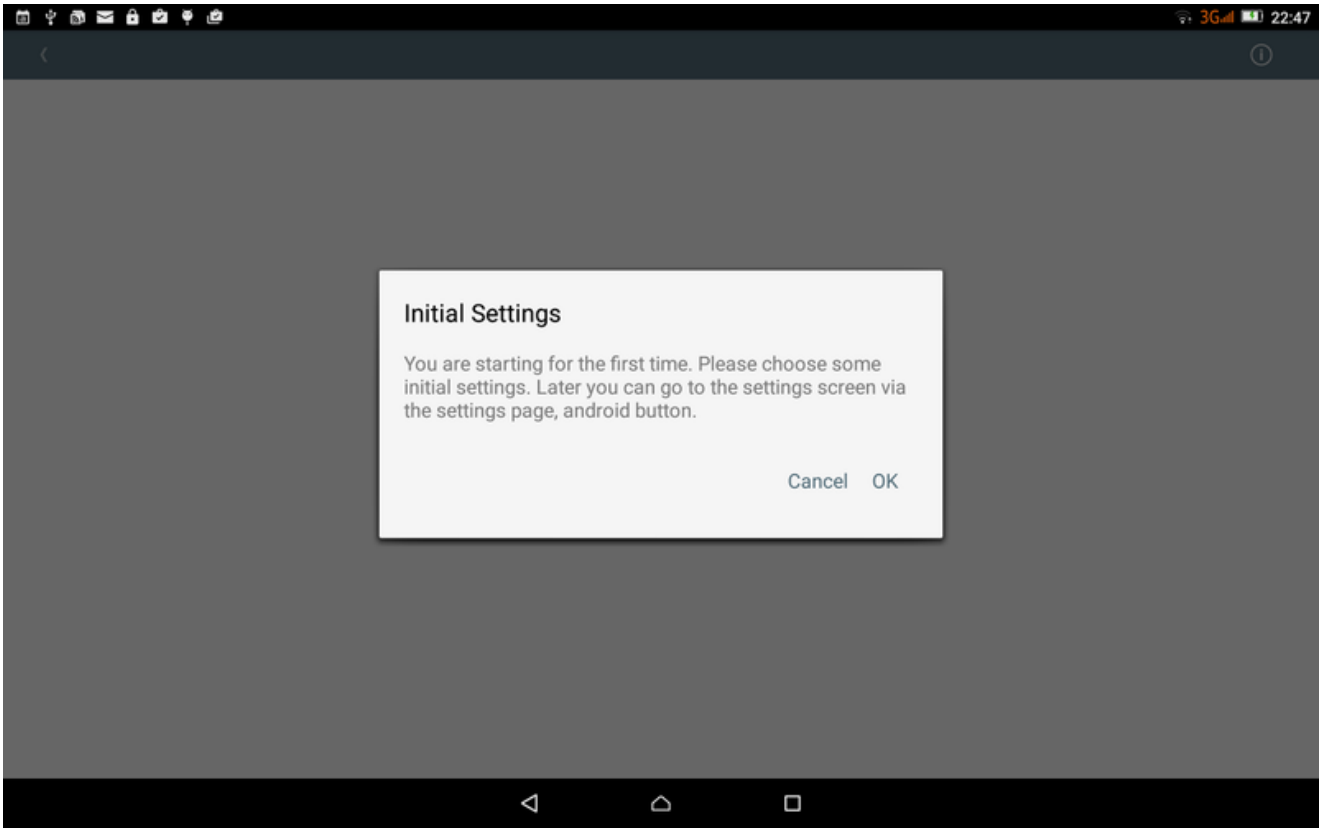
Wenn das gewählte Arbeitsverzeichnis (Standard: interne-sd-Karte/avnav) noch nicht existiert, wird es beim ersten Start angelegt.

Die Track-Daten und Log-Dateien werden in das Verzeichnis "tracks" unterhalb des Arbeitsverzeichnisses geschrieben.

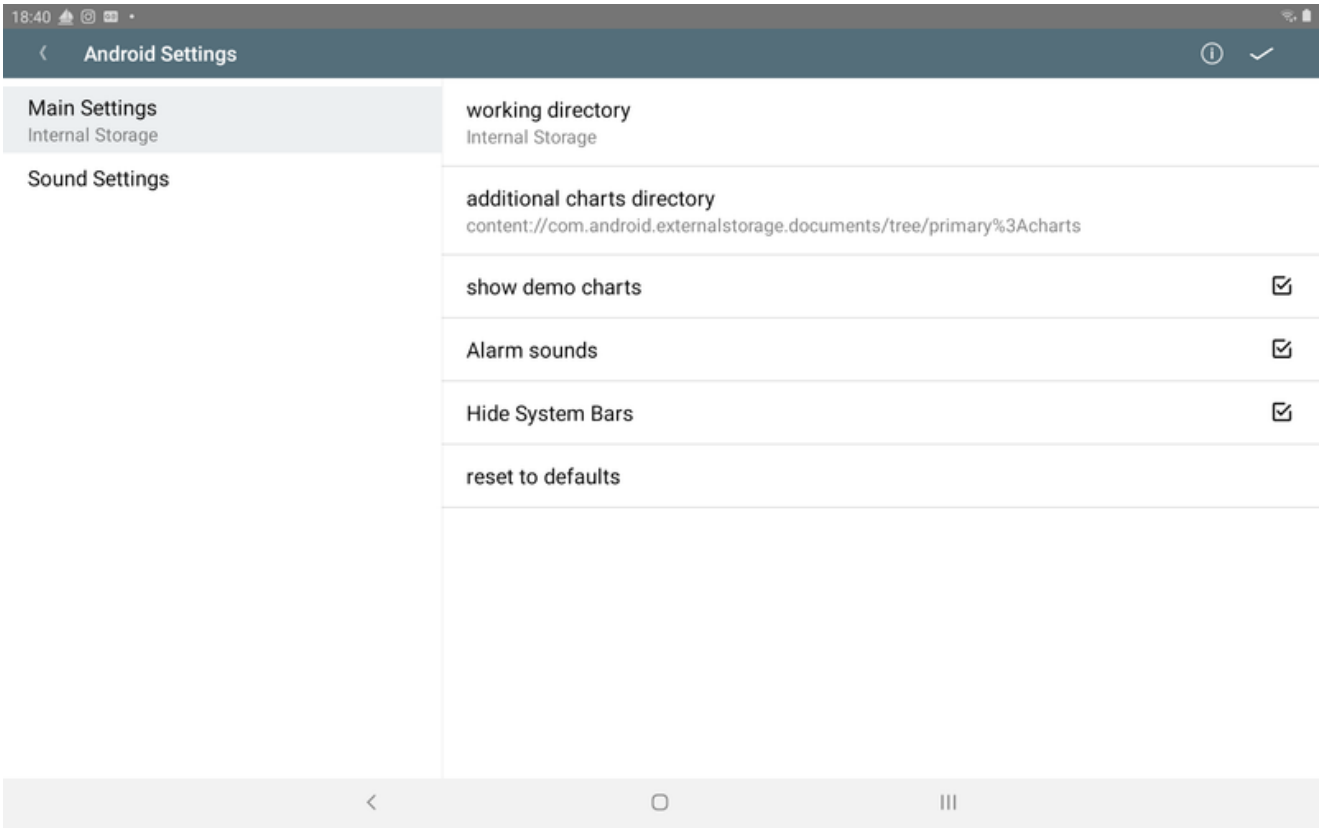
Im Unterverzeichnis "routes" werden die Routen im "gpx"-Format gespeichert. Routen und Tracks können aus der App über die Download-Funktion an andere Apps weitergegeben werden (siehe auch die Beschreibung der App). Um sie in andere Verzeichnisse zu speichern oder Routen aus anderen Verzeichnissen zu laden (Upload-Funktion), sollte sinnvollerweise ein Dateimanager auf dem Android-Gerät installiert sein.

Benutzung

Nach dem erstmaligen Start der App befindet man sich auf einer Einführungsseite:

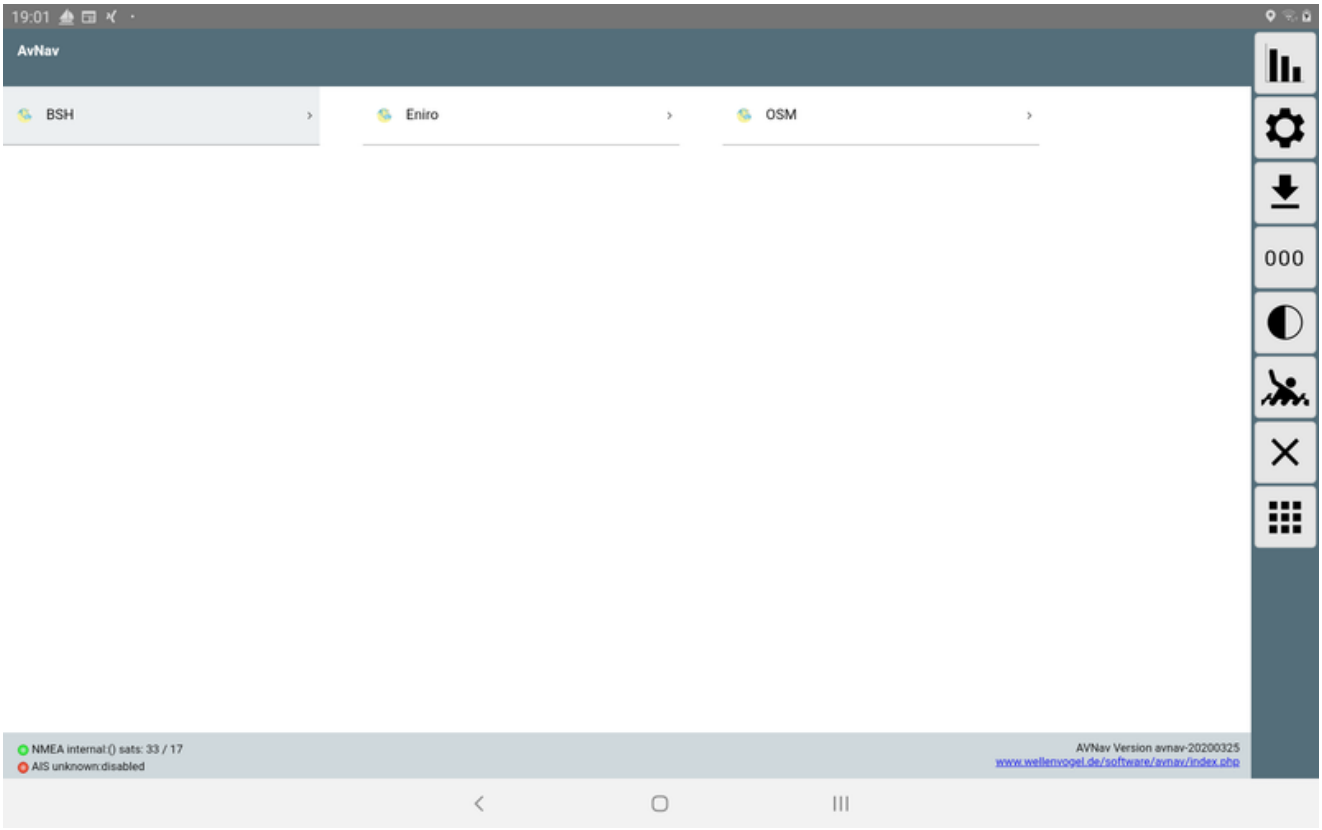


Nach dem Klick kommt man auf die Einstellungsseite:



Hier können neben anderen Einstellungen z. B. die Verzeichnisse für Karten und Tracks gesetzt werden.
Eine Liste aller Einstellungen findet sich hier.
Die Einstellungen für die NMEA-Datenquellen und einige Weitere erreicht man in der App über die "Server/Status"-Seite .

Die "Settings"-Seite kann über den "OK"-Button (oben rechts) oder über den "Zurück"-Button verlassen werden. Man erreicht dadurch die Hauptseite der App.



Im Bild ist wird eine gültige Position (oberer grüner Punkt) und bisher noch keine gültigen AIS Daten (unterer roter Punkt) empfangen.
Um zu den Android Einstellungen zu gelangen, muss zunächst die Einstellungsseite in der App aufgerufen werden, dort klickt man dann auf den "Android"-Button.
Bei weiteren Starts erreicht man sofort die Hauptseite der App.

Externer Zugriff

(neu ab 20210424).

Die App ermöglicht es, das man sich mit einem Browser von anderen Geräten verbinden kann. Dazu muss in der App der Web-Server aktiviert werden (Status/Server Seite).
Bei der Aktivierung des Web Servers muss zunächst "externalAccess" aktiviert werden. Mit "mdnsEnabled wird dafür gesorgt, dass sich eine Bonjour-fähige App (z.B. BonjourBrowser)sich mit dem Server der App verbinden kann.

In älteren Versionen hat die App dafür 2 Modi genutzt.

- Normal
- External Browser

Im Normal-Modus ist der Browser in die App integriert, sie wird wie eine normale Android-App genutzt.
Im "External-Browser"-Modus startet die App einen kleinen Webserverauf über einen einstellbaren Port (voreingestellt: 34567).

Hintergrund

Der NMEA-Multiplexer und auch der Web Server von AvNav können ohne Anzeige nur im Hintergrund laufen. Das kann genuzt werden, wenn die Anzeige für Benutzer auf einem anderen Gerät erfolgen soll - oder wenn eine andere App für die Navigation genutzt wird und nur der Multiplexer von AvNav benötigt wird.

Dazu wird nach dem Start auf der Hauptseite über ✕ der Beenden-Dialog aufgerufen und dort "BACKGROUND" ausgewählt.

Über die Benachrichtigung (in der Android-Nachrichtenzeile) kann die App wieder in den Vordergrund geholt - oder direkt beendet werden.

Einstellungen

Die Einstellungen gliedern sich in zwei Teile:

- Spezielle Android-Einstellungen
- Einstellungen für den Multiplexer und andere Hauptbestandteile

Die Android-Einstellungen erreicht man über den  Button auf der Einstellungsseite bzw. der Server/Status Seite.

Android Main Einstellungen

Name	Bedeutung	Default
working directory	das Arbeitsverzeichnis (mit den Unterverzeichnissen charts,tracks, routes)	/storage/sdcard/avnav
additional charts directory	ein zusätzliches Kartenverzeichnis, das sinnvollerweise auf einer externen SD-Karte angelegt werden sollte	/storage/sdcard/avnav/charts
show demo charts	Anzeige der Demo-Karten. Das erfordert eine aktive Internetverbindung)	ein
Alarm-Sounds	Hier können die durch den Server erzeugten Alarm-Sounds abgeschaltet	

	werden. Im Browser müssen diese ggf. zusätzlich abgeschaltet werden.
Hide System Bars	Verbergen der Android Kopf- und Fusszeile
reset to defaults	Rücksetzen der Multiplexer-Einstellungen auf Default-Werte

Android Sound-Einstellungen

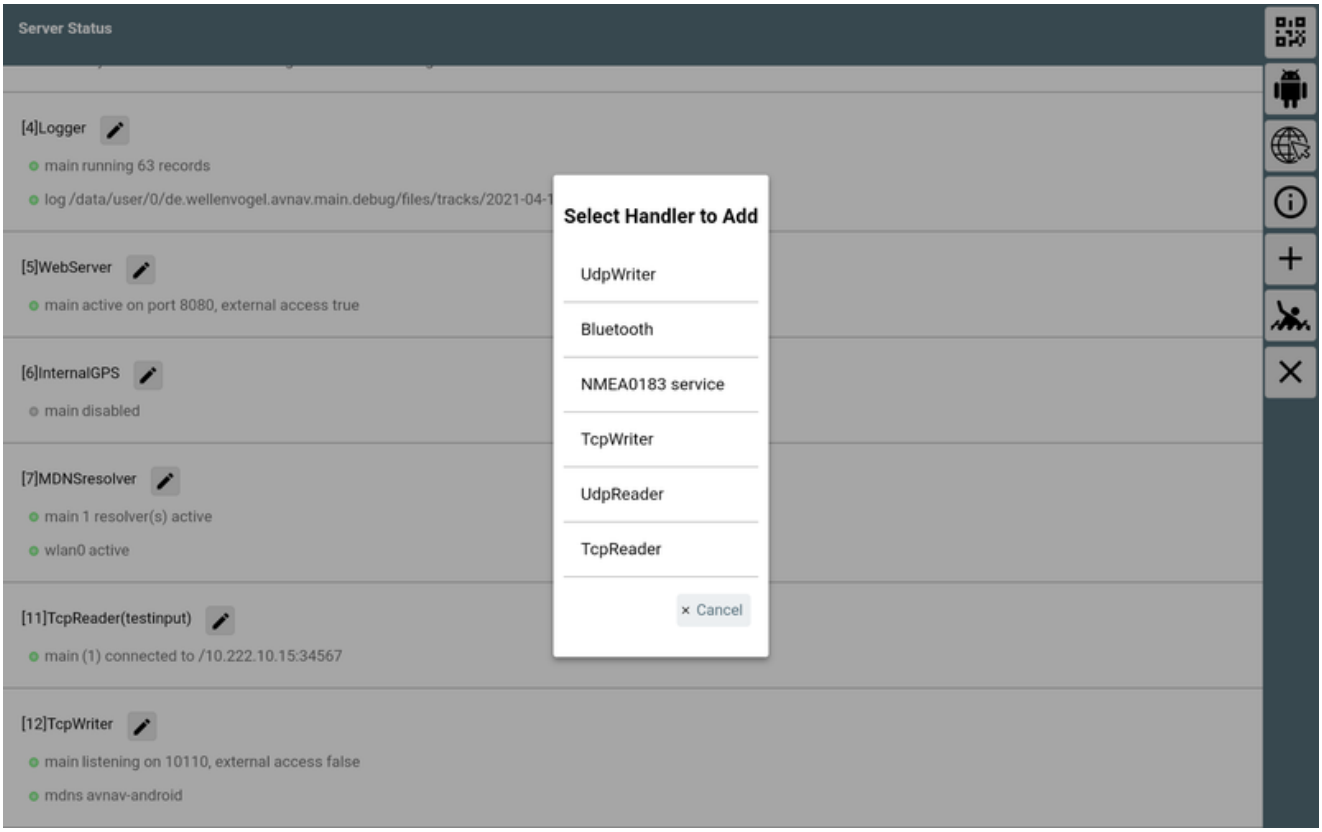
Name	Bedeutung	Default
Sound for XXX alarm	Hier kann der Ton für die verschiedenen Alarme gewählt werden	
reset to defaults	Rücksetzen der Sound-Einstellungen auf Defaults	

Multiplexer Einstellungen

(neu ab 20210424)

Die Einstellungen für den Multiplexer sind auf der Status/Server Seite verfügbar. Für jede Funktion, z.B. für jede Quelle im Multiplexer, ist ein sogenannter "Handler" vorhanden, der hier konfiguriert werden kann.

Weitere Quellen können über den + Button hinzugefügt werden. Es werden dann die Handler angeboten, die möglich sind, z.B. wird "UsbConnection" nur angezeigt, wenn auch ein USB-Gerät angeschlossen ist.



Neben den angezeigten Handlern in der Statusübersicht existiert im Normalfall ein Button, mit dem man den Handler bearbeiten kann.

Die im Bearbeitungs-Dialog angezeigten Parameter haben meist einen Button, über den eine Hilfe angezeigt wird. Mit dem Button kann der jeweilige Parameter auf seinen Default-Wert zurückgesetzt werden.

Einige Parameter tauchen bei mehreren Handlern auf:

Name	Beschreibung	default
enabled	aktviert/deaktiviert diesen Handler	je nach Handler
name	Name für den Handler. Dieser kann in Blacklists verwendet werden.	leer
port	TCP- oder UDP-Port	
filter/readerFilter/ sendFilter	Das ist ein NMEA-Filter. Hier kann definiert werden, welche NMEA-Daten durchgelassen werden sollen. Mehrere Filter werden durch ein Komma (,) getrennt. Für Datensätze, die mit dem Dollarzeichen ($) beginnen, werden die nächsten beiden Zeichen ignoriert (die Talker-Id). Ein Filter für alle RMC-Sätze sieht z.B. so aus: $RMC	leer

	Ausschließlich für alle AIS-Daten: ! Alle RMC- und RMB-Sätze: $RMC,$RMB Falls der Filter negiert werden soll, muss ein ^ dem Ausdruck vorangestellt werden. ^$RMB,^$APB
blacklist	eine durch Kommas (,) getrennte Liste von Namen. NMEA Daten von diesen Quellen werden nicht ausgesendet.

Die vorhandenen "Handler" und ihre Parameter. Es werden nur die spezifischen Parameter beschrieben.

Decoder

Parameter	Beschreibung	default
ownMMSI	eigene MMSi, diese wird in der AIS-Anzeige ausgeblendet	leer
posAge	erlaubtes Alter für die GPS-Position (in sec), nach dieser Zeit wird der Eintrag gelöscht, wenn keine neuen GPS-Daten empfangen wurden	10
nmeaAge	erlaubtes Alter für NMEA-Daten (in sec), die keine Positionsdaten enthalten	600
aisAge	erlaubtes Alter für AIS-Daten	1200
readTimeout	Timeout in s für die Anzeige ob valide NMEA Daten empfangen werden	10

Route

Parameter	Beschreibung	default
computeRMB	erzeuge NMEA-RMB-Datensätze, wenn ein Routing aktiv ist.	an

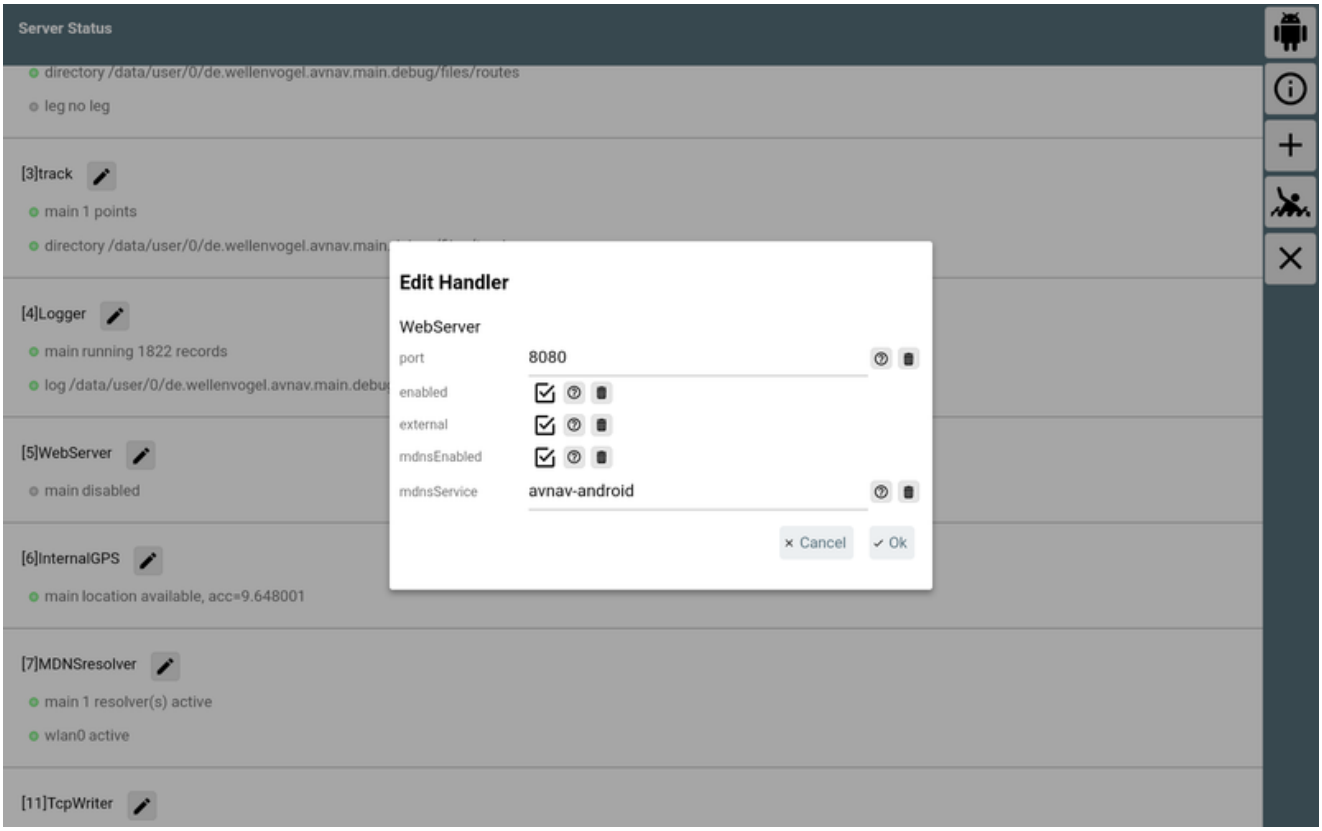
Track

Parameter	Beschreibung	default
interval	Intervall (in sec) für das Schreiben des Tracks als "gpx"-Datei.	300
distance	Minimaler Abstand (in m) bevor ein neuer Trackpunkt geschrieben wird	25
minTime	Minimale Zeit (in sec) bevor ein neuer Trackpunkt geschrieben wird	10
length	Länge des angezeigten Tracks in Stunden (h).	25

Logger

NMEA-Logger

WebServer



Parameter	Beschreibung	default
port	der TCP-Port, auf dem der Server Verbindungen annimmt.	8080
external	Wenn aktiv, dann können sich auch andere Geräte verbinden (sonst nur auf dem eigenen Gerät). Hinweis: Dieser Parameter sollte mit Vorsicht und nur in	aus

	vetrauenswürdigen Netzen aktiviert werden. Innerhalb der App gibt es keinen weiteren Schutz!	
mdnsEnabled	macht den Service per "mDNS" bekannt.	an
mdnsService	Der Name unter dem die App per "mDNS" erreichbar ist.	avnav-android

InternalGPS

Das GPS des Gerätes.

MDNSResolver

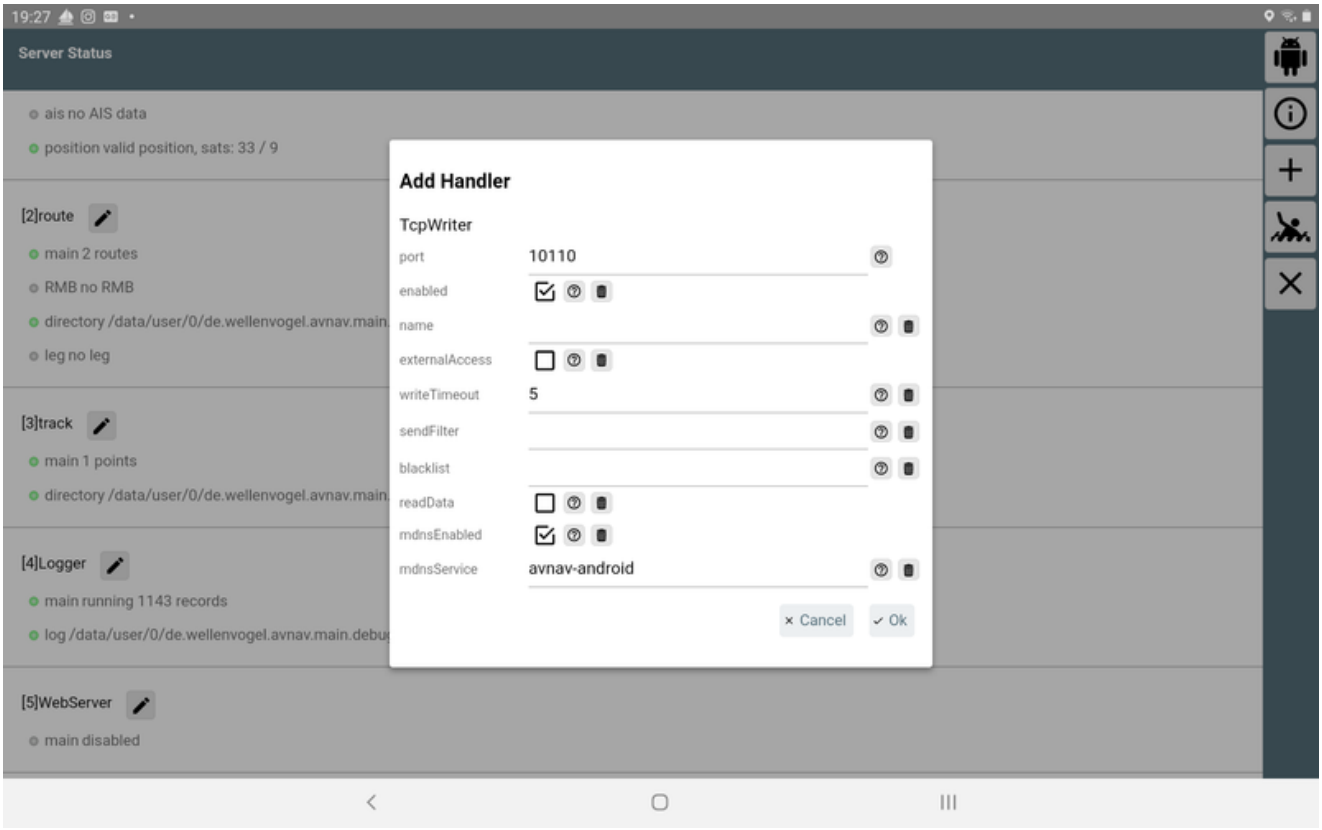
Der Handler für die Auflösung und das Bekanntmachen von "mDNS" (Bonjour/Avahi) Services.

TcpReader

Parameter	Beschreibung	default
ipAddress	Die IP-Adresse oder der Hostname für den Server, der kontaktiert werden soll. Das kann auch ein mDNS Name wie z.B. avnav.local sein.	---
port	Der Ip-Port, zu dem eine Verbindung aufgebaut werden soll.	---
sendOut	Wenn aktiviert, werden auch NMEA-Daten gesendet. Ansonsten wird nur emfangen.	aus
readTimeout	Markiere die Verbindung als inaktiv, wenn keine Daten nach der eingestellten Zeit (sec) eingegangen sind.	10
writeTimeout	Schliesse die Verbindung, wenn ein NMEA-Datensatz nicht innerhalb der eingestellten Zeit (sec) geschrieben werden konnte.	5
connectTimeout	Timeout für den Verbindungsaufbau (in sec, 0 = System-default)	0
closeOnTimeout	Schliesse die Verbindung und öffne sie erneut, wenn das readTimeout erreicht wird.	an

TcpWriter

Ein TCPWriter stellt die NMEA-Daten für andere Anwendungen bereit.



Parameter	Beschreibung	default
port	Der Port, auf dem der Server Verbindungen annimmt	---
externalAccess	Wenn aktiv, können auch andere Geräte sich verbinden. Andernfalls können sich nur Apps auf dem selben Gerät verbinden.	aus
writeTimeout	Schliesse die Verbindung, wenn ein NMEA-Datensatz nicht innerhalb der eingestellten Zeit (sec) geschrieben werden konnte.	5
readData	Wenn aktiv, empfängt AvNav auch NMEA-Daten über eine aufgebaute Verbindung	aus
mdnsEnabled	mache den Service per mDNS im Netz bekannt (type: _nmea-0183._tcp)	aus
mdnsService	Der Name, unter dem dieser Server per mDNS erreichbar ist	

UdpReader

Ein UDP-Reader empfängt Daten von anderen Apps/Systemen.

Parameter	Beschreibung	default
port	der UDP-Port, auf dem Daten empfangen werden	---
externalAccess	wenn aktiv, können Daten von anderen Geräten empfangen werden, sonst nur innerhalb des selben Gerätes.	aus
readTimeout	zeige die Verbindung als inaktiv, wenn für die eingestellte Zeit (sec) keine Daten empfangen wurden.	10

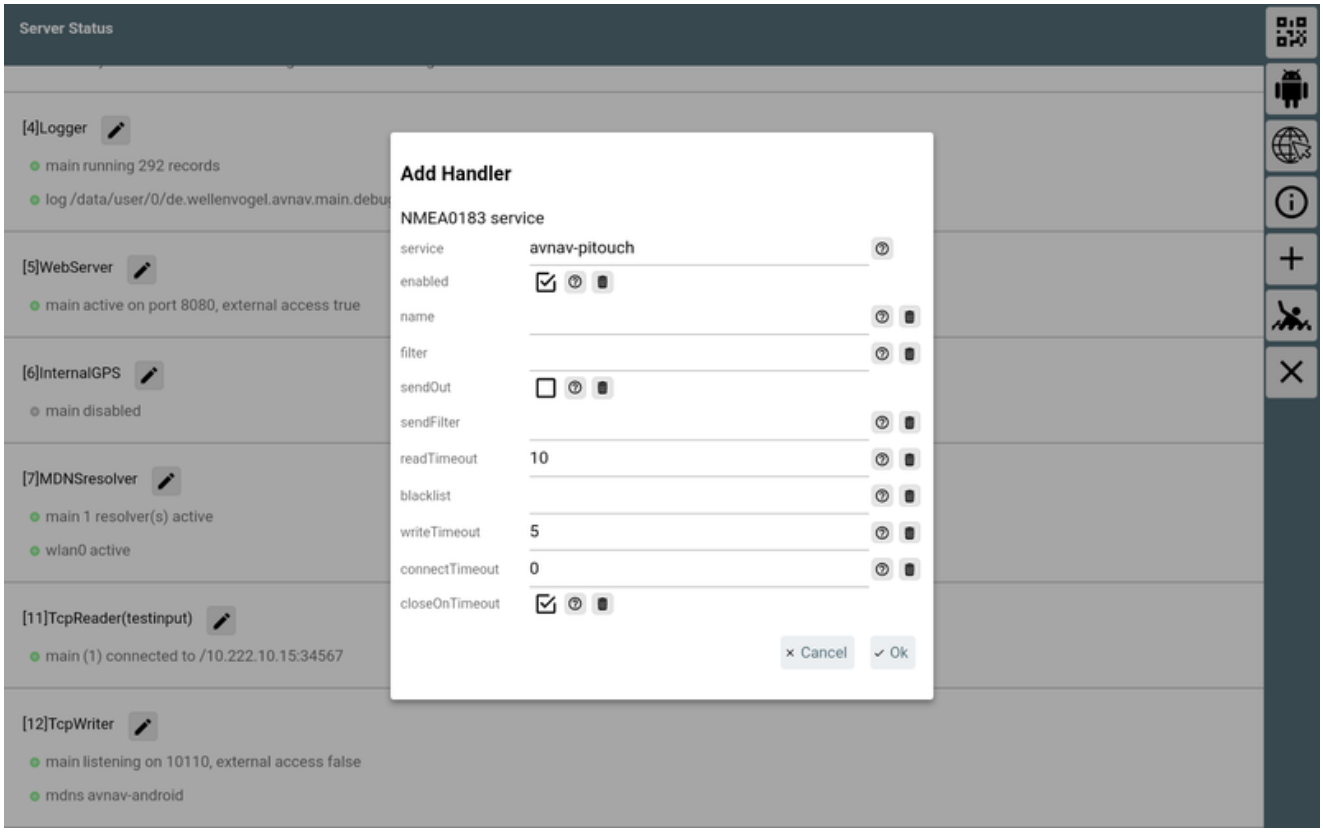
UdpWriter

Ein UDP-Writer sendet NMEA-Daten per UDP an eine andere App.

Parameter	Beschreibung	default
ipaddress	die IP-Adresse oder der Hostname des Ziel-Computers. Es kann auch ein mDNS-Name wie "avnav.local" genutzt werden.	---
port	Der Ziel-Port , an den Daten gesendet werden sollen	---
broadcast	Sende die Daten als Broadcast, die IP-Adresse muss dann eine gültige Broadcast-Adresse sein.	aus

NMEA0183 Service

Ein NMEA0183-Service arbeitet im Wesentlichen wie ein TcpReader. Allerdings werden hier nicht Zieladresse und Port vorgegeben. Stattdessen wird der Name eines (mDNS) Services (type: _nmea-0183._tcp) aus einer Liste der verfügbaren Services gewählt. Falls z.B. im Netz ein SignalK-Server oder ein AvNav-Server (> 20210415) vorhanden sind, geben diese ihre NMEA-Ausgänge auf diese Weise im Netzwerk bekannt.
Der Vorteil ist, dass auch bei Wechsel des Netzwerkes der Zugriff im Normalfall wieder funktionieren wird und Verbindungen so automatisch wieder aufgebaut werden können.

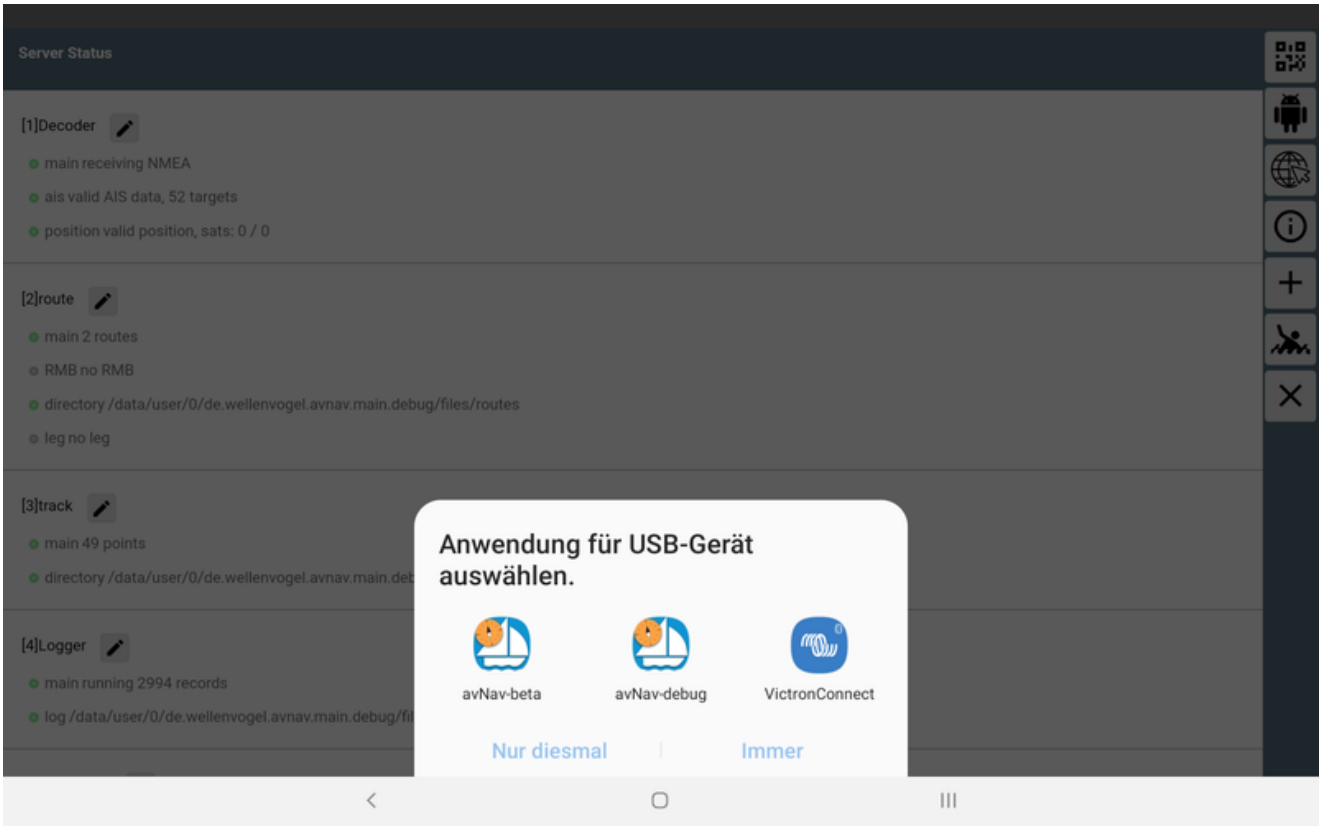


Parameter	Beschreibung	default
service	der Name des Services (Auswahl aus der Liste der gefundenen)	---
sendOut	sende NMEA-Daten auf dieser Verbindung	
readTimeout	Zeige die Verbindung inaktiv, wenn keine Daten nach der eingestellten Zeit (sec) aufgelaufen sind.	10
writeTimeout	Schliesse die Verbidung, wenn ein Satz nicht innerhalb der eingestellten Zeit (sec) geschrieben werden konnte	5
connectTimeout	Timeout für den Verbindungsaufbau (in sec, 0 = System-default)	0
closeOnTimeout	Schliesse die Verbindung und öffne sie erneut, wenn das readTimeout erreicht wird.	an

UsbConnection

AvNav wird aktiv, wenn ein USB-Gerät angeschlossen wird. Es ist daher sinnvoll, zunächst AvNav zu starten und danach das USB-Gerät anzuschließen. Das Gerät kann dann direkt so

konfiguriert werden, dass AvNav immer auf dieses Gerät zugreifen darf. AvNav startet dann sofort den Dialog zur Konfiguration des Gerätes.



Parameter	Beschreibung	default
device	Das angeschlossene USB-Gerät, genauer: der Anschluss des Gerätes, Auswahl aus einer Liste	---
baud rate	Die Baudrate	9600
flowControl	none|xon/xoff|rts/cts - Auswahl der Flusssteuerung (muss vom Adapter unterstützt werden)	none
sendOut	sende NMEA-Daten auf dieser Verbindung	aus
readTimeout	Markiere die Verbindung als inaktiv, wenn keine Daten nach der eingestellten Zeit (sec) aufgelaufen sind.	10

Bluetooth

Zum Verbinden z.B. mit einem Bluetooth-GPS. Das Gerät muss zunächst über die Bluetooth-Einstellungen des Android-Geräts verbunden werden ("pairen").

Parameter	Beschreibung	default

device	Das Bluetooth-Gerät. Bei der Einrichtung werden alle Geräte angeboten, die bereits bekannt sind, erst danach wird versucht, das Gerät zu erreichen.	---
sendOut	sende NMEA-Daten auf dieser Verbindung	aus
readTimeout	Markiere die Verbindung als inaktiv, wenn keine Daten nach der eingestellten Zeit (sec) aufgelaufen sind.	10

Konfiguration und Anpassung

AvNav kann auf verschiedene Weise an die eigenen Bedürfnisse angepasst werden

- Anpassung der Anzeigen in der WebApp - Einstellungsseite
- Anpassung der Anzeigen auf der Navigationsseite und den Dashboard-Seiten (Widgets): Layout-Anpassung.
- Anpassung des Aussehens per CSS. Da die App selbst in einem Browser läuft, kann man das Aussehen weitgehend über die Anpassung von Stylesheets ändern. Das ist unter Anpassung per CSS beschrieben
- Anpassung verschiedener Icons (Boot, AIS,...)
- Anpassung der Tastaturbelegung
- Anpassung der Serverkonfiguration (avnav_server.xml) - nicht unter Android
- Verwaltung der Verbindungen des Servers zu WLAN-Netzwerken Wlan Seite - nicht unter Android und OpenPlotter

Layout-Anpassung

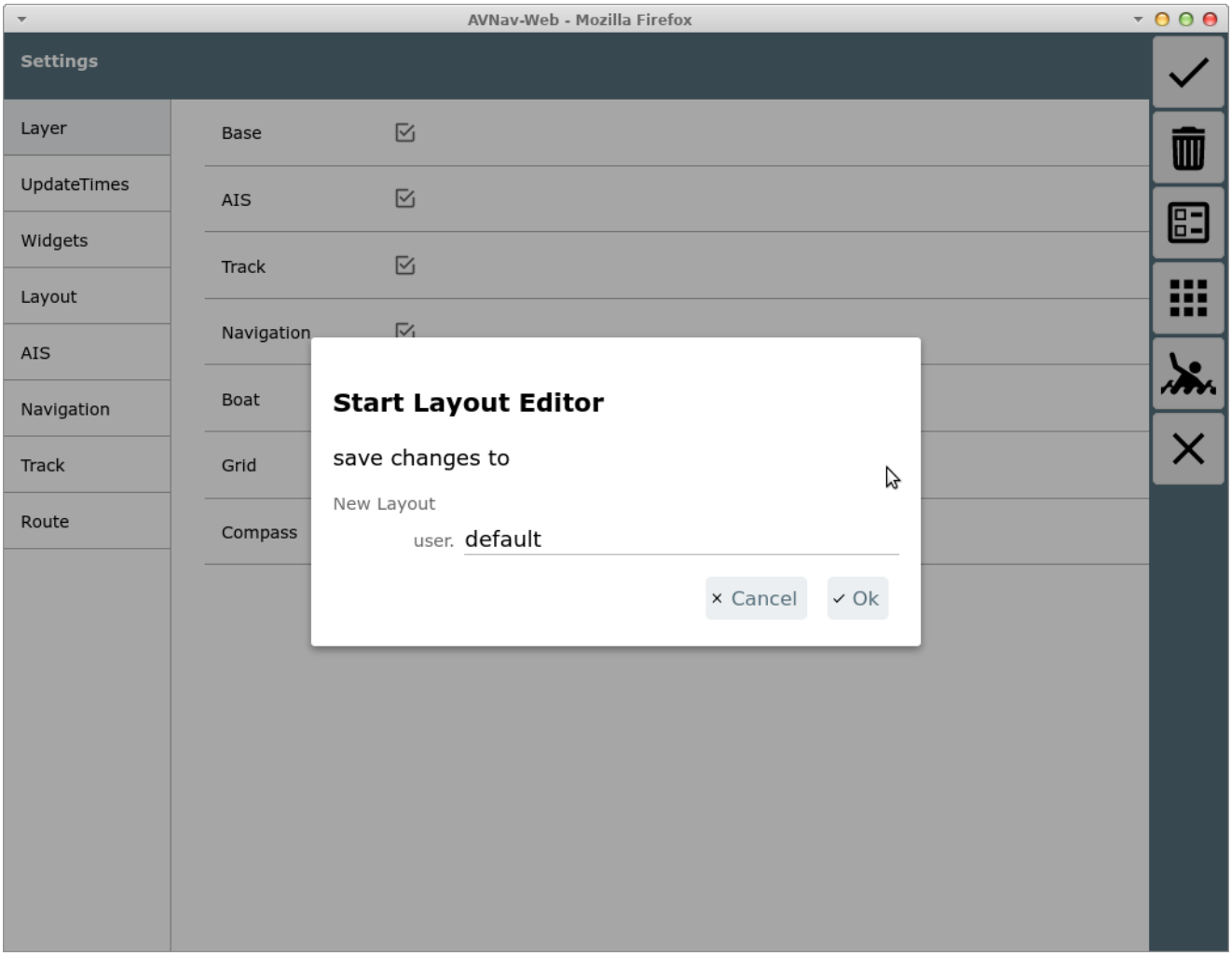
Die Anzeigen auf der Navigationsseite und im Dashboard werden über eine Konfigurationsdatei ("layout") gesteuert. Das ist eine Json-Datei. AvNav bringt selbst einige solcher Dateien mit (zu erkennen am Namens-Prefix system.). Der Benutzer kann eigene Layouts speichern (Namensprefix: user.). Diese Dateien werden unterhalb des Datenverzeichnisses (/home/pi/avnav/data auf dem raspberry) im Ordner "layout" abgelegt. Man kann diese Dateien direkt bearbeiten (innerhalb von avnav), sie herunterladen (und wieder hochladen) oder man kann das Layout innerhalb von AvNav editieren ("layout editor").

Der bevorzugte Weg sollte hierbei der Layout-Editor sein, da dieser die wenigsten Fehlerquellen bietet.

Die Auswahl des aktuell genutzten Layouts erfolgt über Einstellungen /Layout

Layout Editor

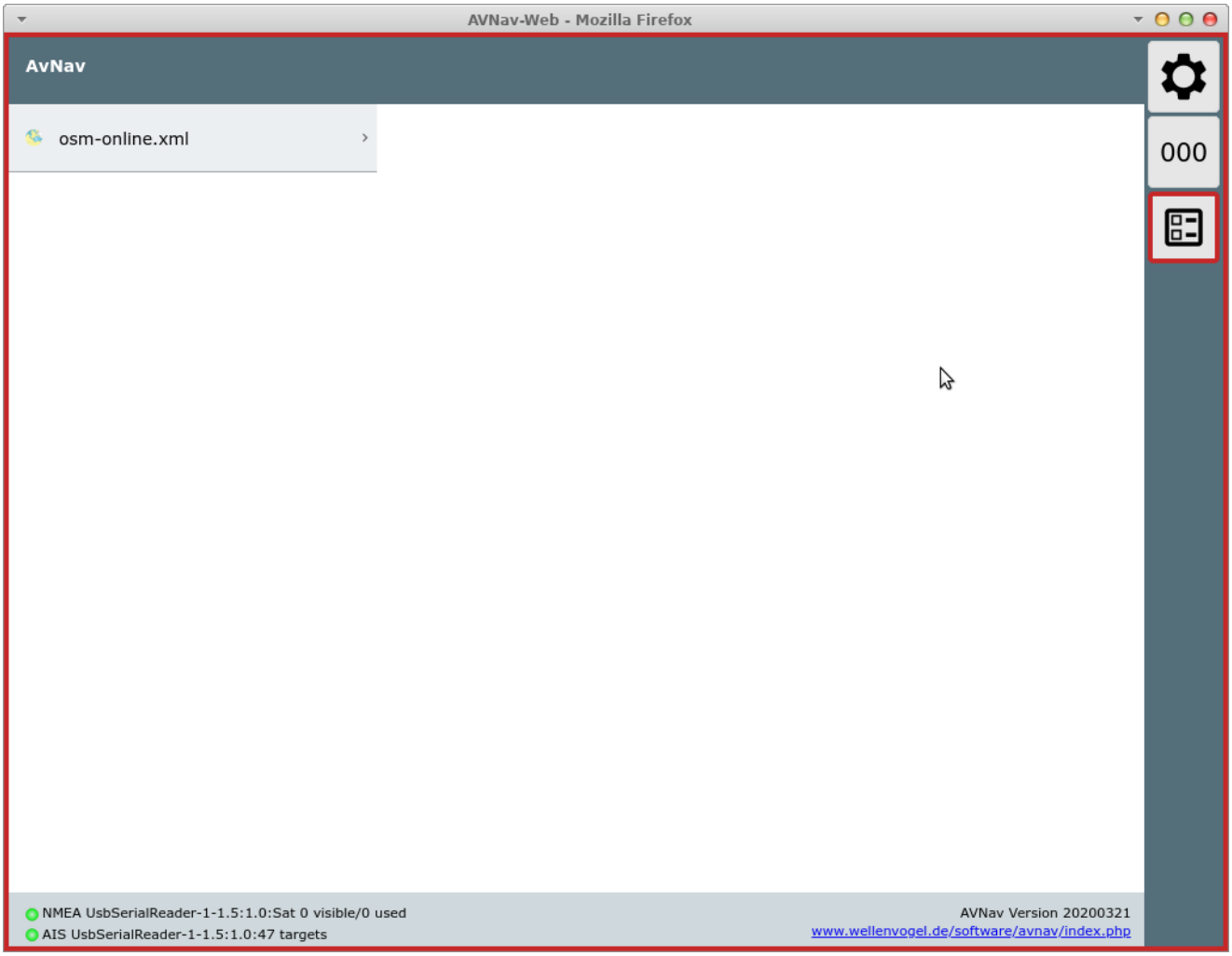
Der Start des Layout-Editors erfolgt über die Einstellungsseite , Layouts .



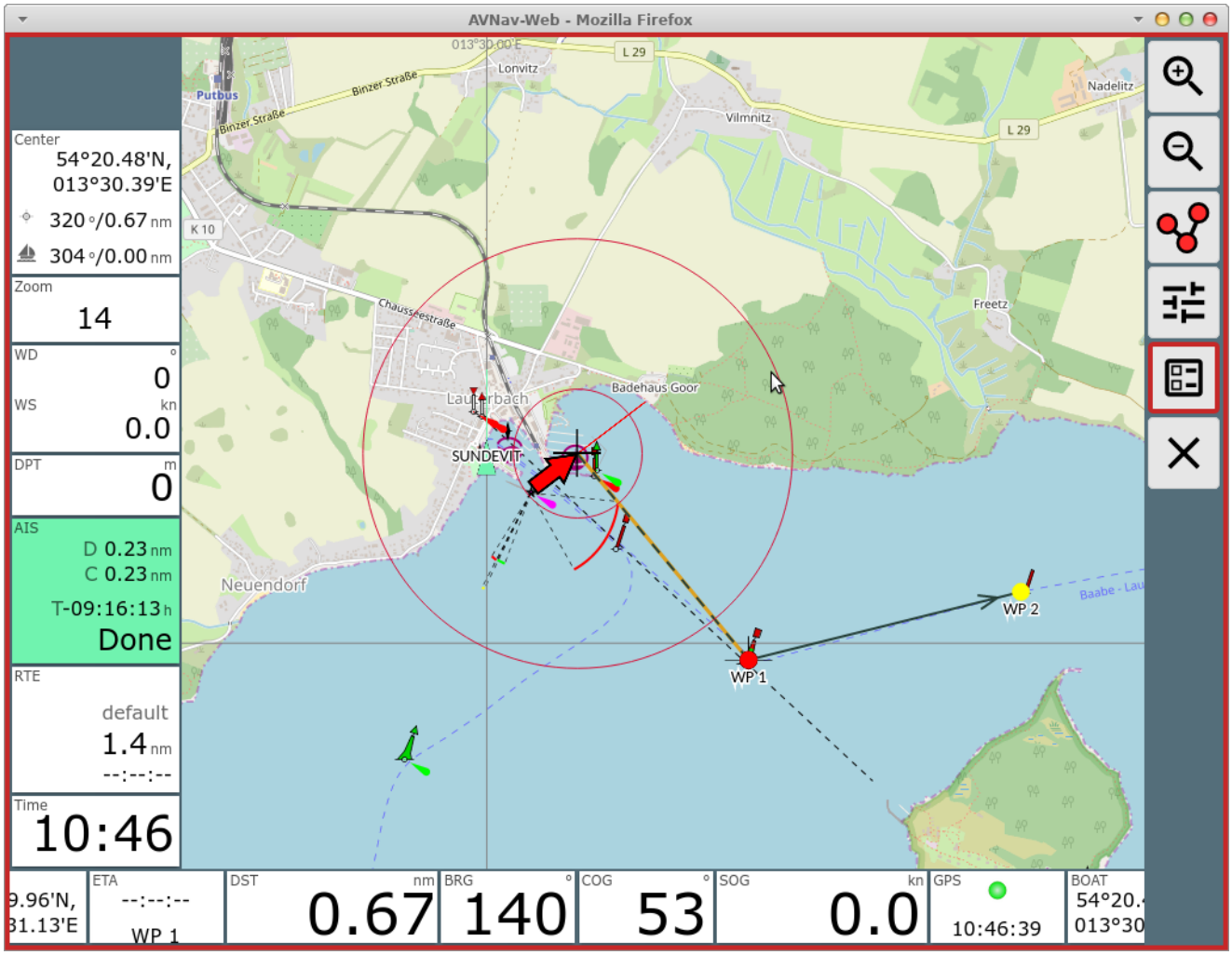
Anpassung von AvNav

Hier wird der Name für das layout-File vergeben. Wenn bisher ein System-Layout aktiv war, wird ein user-Layout mit dem gleichen Namen erzeugt und aktiviert. Falls das Layout schon existiert, wird gefragt, ob es überschrieben werden soll (das kann am Ende noch verhindert werden, in dem alle Änderungen verworfen werden).

Nach dem Start bekommt die App einen roten Rand und es sind nur noch die Seiten sichtbar, auf denen das Layout angepasst werden kann.

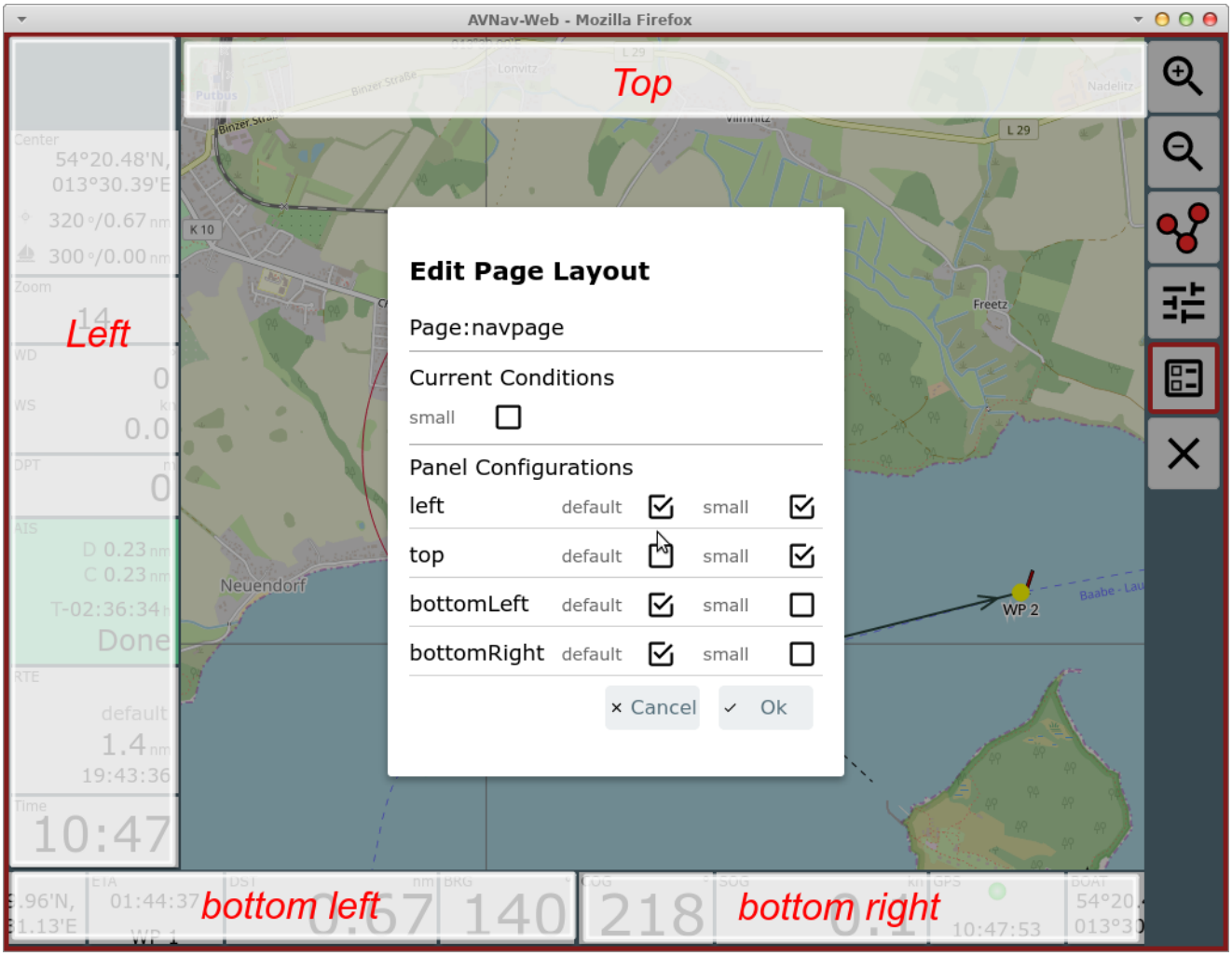


Durch Auswahl einer Karte gelangt man wie immer zur Navigationsseite.



Die Anzeigen sind auf allen relevanten Seiten in sogenannten "Panels" angeordnet.
Innerhalb des jeweiligen Panels können die Anzeigen im Editor-Modus einfach verschoben werden.
Auf der Navigationsseite können die Panels unterschiedlich belegt werden, je nachdem ob ein eher schmales Display vorhanden ist (small) oder ein breiteres (die Grenze dafür kann in den Settings eingestellt werden).

Über kann die Konfiguration für die Panels auf der jeweiligen Seite aufgerufen werden.



Im Bild sind die vorhandenen Panels markiert. Im Dialog kann ausgewählt werden, welche der Panels jeweils sichtbar sein sollen. Wenn in der Spalte "small" ein Häkchen gesetzt ist, bedeutet das, dass bei schmalem Bildschirm das Panel anders belegt werden kann. Im Beispiel ist das Top-Panel im normalen Modus nicht sichtbar, sondern nur im schmalen, das Left-Panel ist jeweils unterschiedlich belegt. Die beiden unteren Panels sind in beiden Modi gleich belegt. Unter "Current Conditions" kann ausgewählt werden, ob die Panels für die Normal-Ansicht oder für die schmale Ansicht bearbeitet werden sollen.

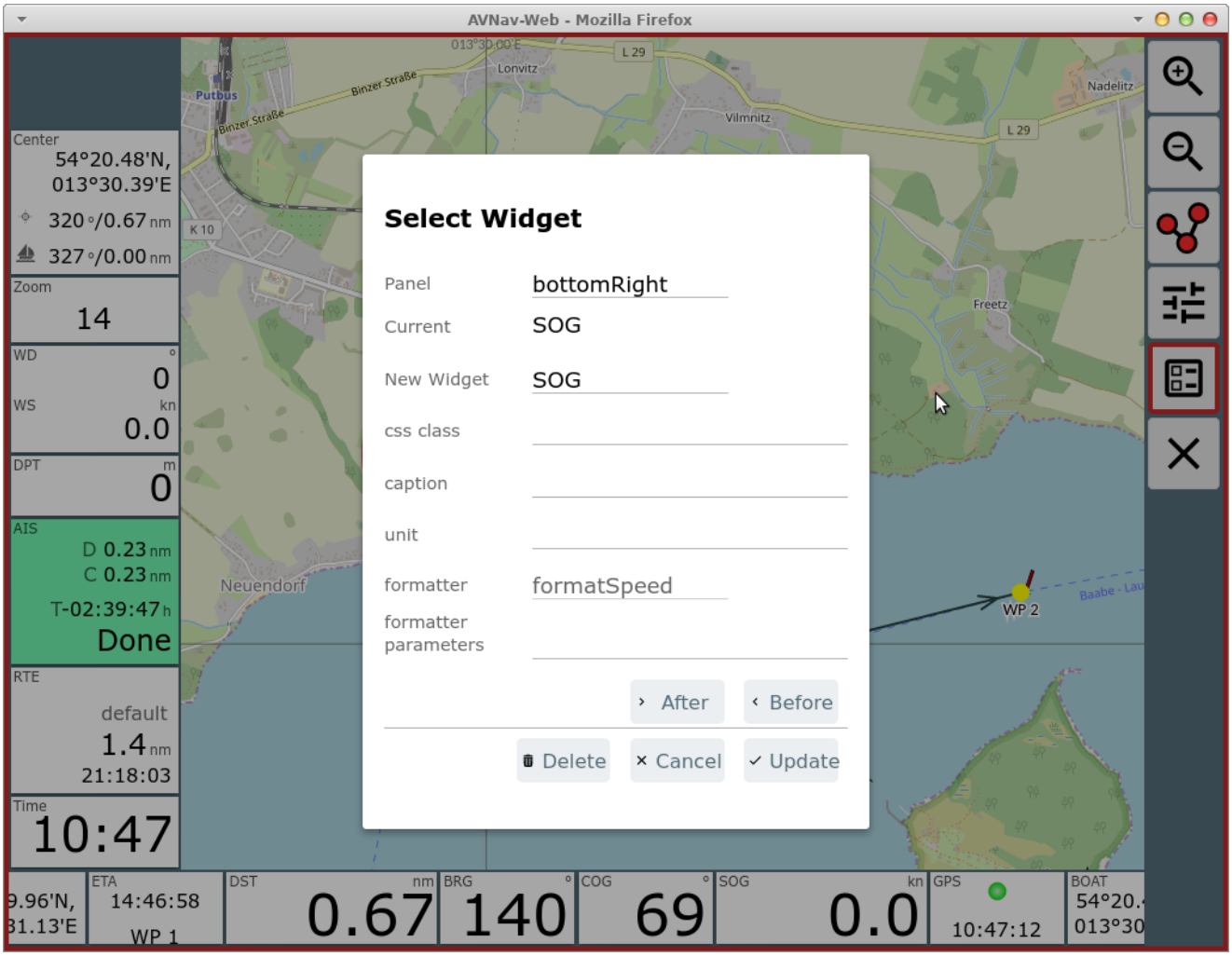
Die beiden unteren Panels können potentiell 2 Reihen von Anzeigen aufnehmen (Einstellung: "2 widget rows"), nicht mehr passende Anzeigen werden ausgeblendet.

Nach Beenden des Panel-Dialogs können jetzt Anzeigen (Widgets) innerhalb der Panels verschoben werden (jedoch nicht von einem Panel zum anderen).

Widget Dialog

Zum Ändern oder Einfügen eines Widgets klickt man auf ein vorhandenes oder auf ein freies Feld in einem Panel.

Man erhält den folgenden Dialog.



Je nach Art des Widgets sind hier unterschiedliche Werte sichtbar bzw. änderbar. Nicht ausgefüllte Werte haben meist defaults.

Falls das Widget in ein anderes Panel verschoben werden soll, kann man oben das neue Panel auswählen.

Änderungen schliesst man mit "Update" ab.

Unter "New Widget" kann man ein anderes Widget aus der Liste der vorhandenen Widgets auswählen.

Jedes Widget benötigt eine Information, welche Daten es anzeigen soll. Die verfügbaren Daten sind intern in einem Speicher verfügbar und werden über einen Key ausgewählt.

Je nach Art des Widgets ist dieser Key (oder auch mehrere) fest im Widget eingebaut, bei anderen kann man das frei wählen. Fast immer wählbar sind der Titel (caption), die Einheit (unit) und die css Klasse (falls man das Aussehen per css anpassen möchte).

Formatierer (formatter)

Die meisten Widgets benötigen für die Darstellung einen Formatierer, der den internen Wert in die gewünschte Darstellung wandelt. Meist ist der beim Widget fest vorgegeben. Einige

Formatierer akzeptieren eine Komma-getrennte Liste von Parametern um ihr Verhalten anzupassen (z.B. m/s statt kn).

Die folgenden Formatierer sind vorhanden:

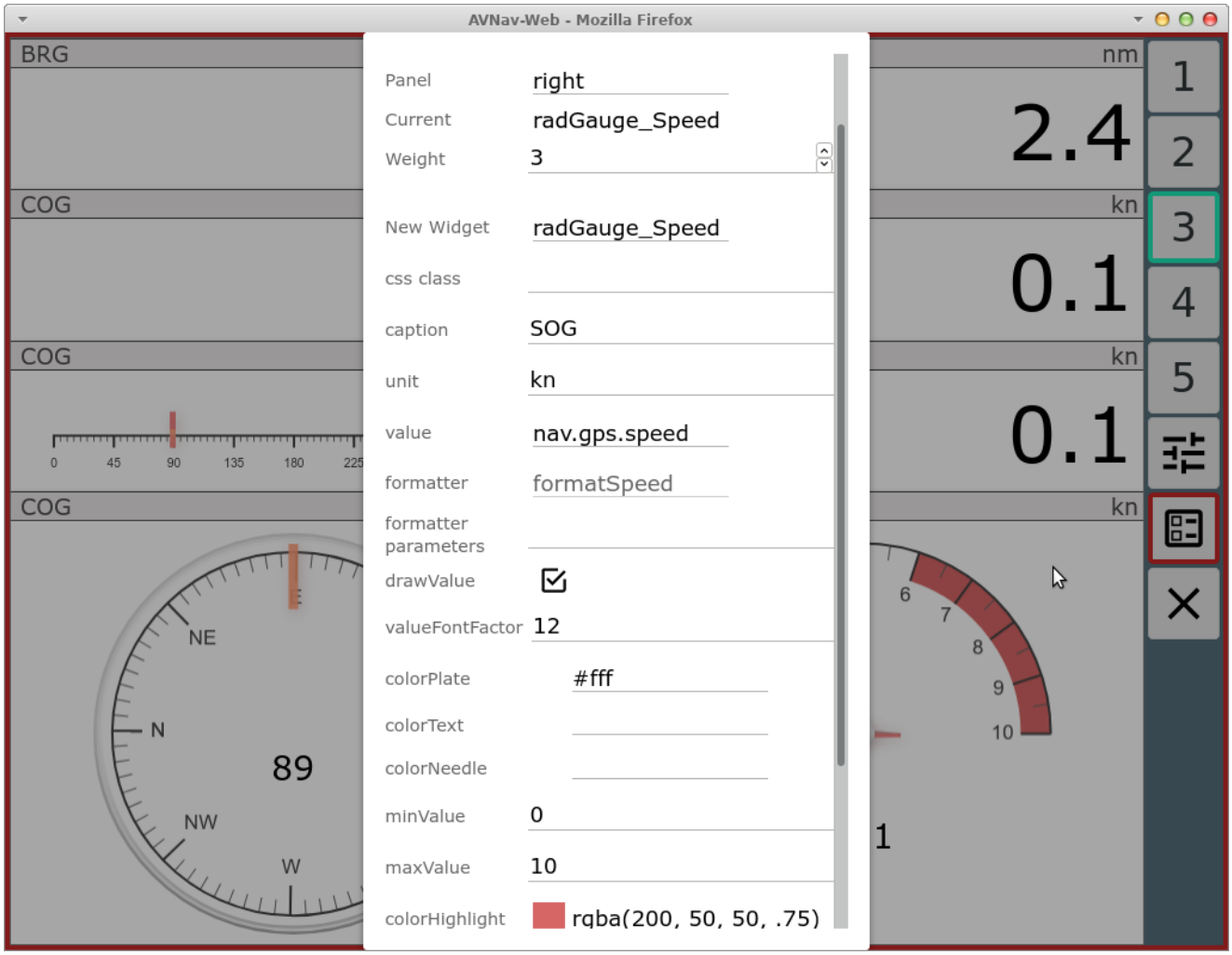
Name	Beschreibung	Parameter
formatDecimal	einfache Formatierung als Dezimalzahl	fix, fract, addSpace z.B.: 3,1
formatDistance	Entfernung in nm	ein Parameter (unit): nm - Enterfnung in nm m - Entfernung in m statt nm km - Entfernung in km statt nm
formatSpeed	Geschwindigkeit in kn	ein Parameter (unit): kn - knoten ms - m/s statt kn kmh - km/h statt kn
formatDirection	Formatieren einen Gradwert	ein Parameter: inputRadian - Input in rad statt °
formatTime	Formatiere einen Zeitwert (Wert muss intern ein Date Wert sein) (hh:mm:ss)	
formatClock	Formatiere einen Zeitwert (Wert muss intern ein Date Wert sein) (hh:mm)	
formatDateTime	Formatiere Datum und Uhrzeit (Wert muss intern ein Date Wert sein)	
formatDate	Formatiere Datum (Wert muss intern ein Date Wert sein)	
formatString	gibt den Input unverändert weiter	
formatTemperature	Formatiere eine Temperatur (seit	ein Parameter (unit)

	20210106), Input in Kelvin	celsius, kelvin
formatPressure	Formatiere einen Druck (seit 20210106), input in Pa	ein Parameter (unit) pa, hpa, bar

Plugins oder eigene Erweiterungen können ggf. weitere Formatierer hinzufügen.

Falls man das vorhandene Widget nicht ersetzen möchte, kann man das geänderte Widget davor oder danach einfügen (Before/After).

Falls das Widget weitere Parameter unterstützt, dann werden diese im Dialog angezeigt.



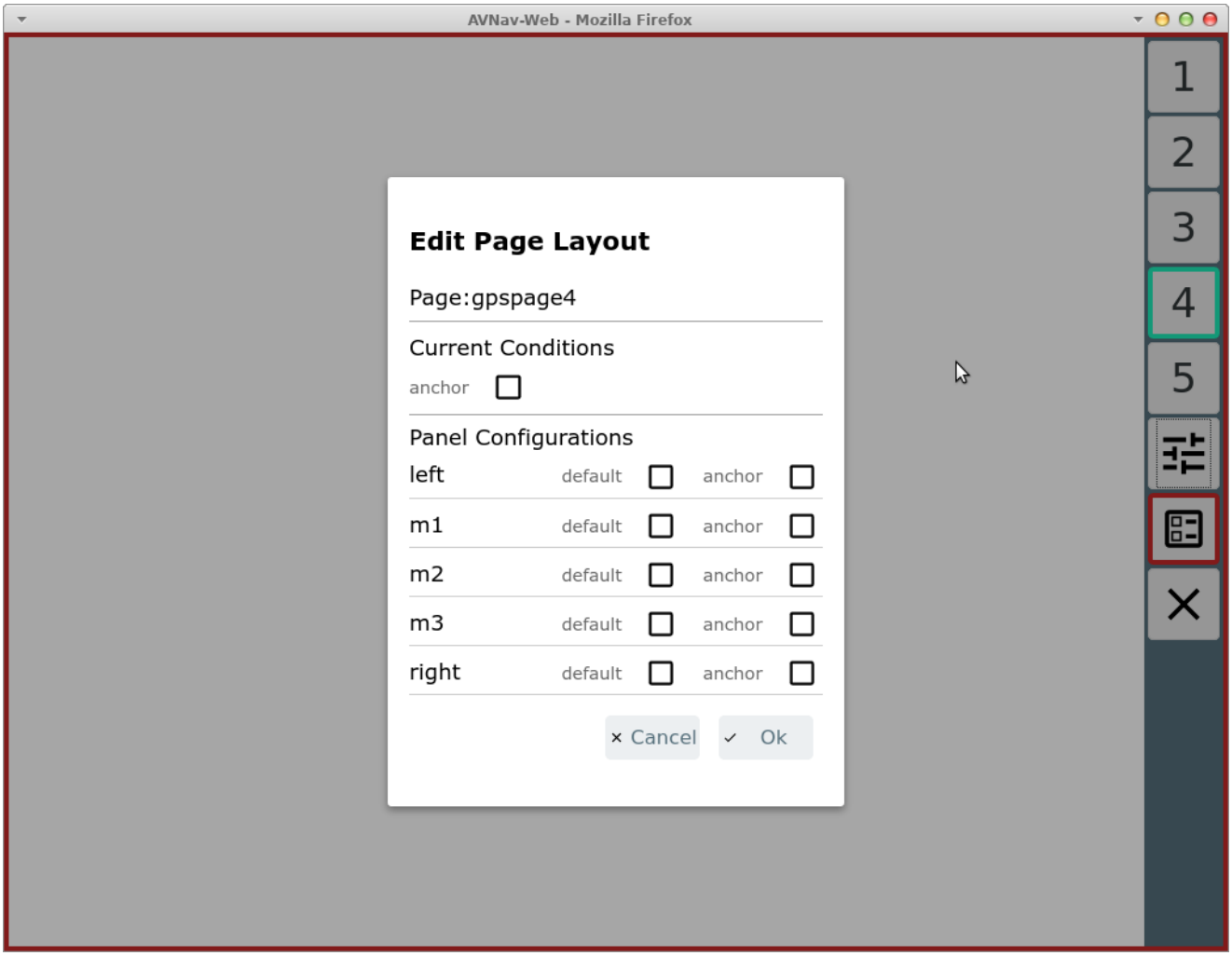
Im Beispiel die Konfiguration für eine grafische Anzeige (avnav nutzt hierzu canvas-gauges).

Von der Navigationsseite kann über die Hauptseite zu den Dashboard Seiten gewechselt werden.

Dashboard Seiten

Es können bis zu 5 Dashboard Seiten konfiguriert werden. Auf jeder Seite sind bis zu 5 Panels möglich.

Auf einer leeren Dashboard Seite muss zunächst die Panel-Konfiguration erfolgen - .

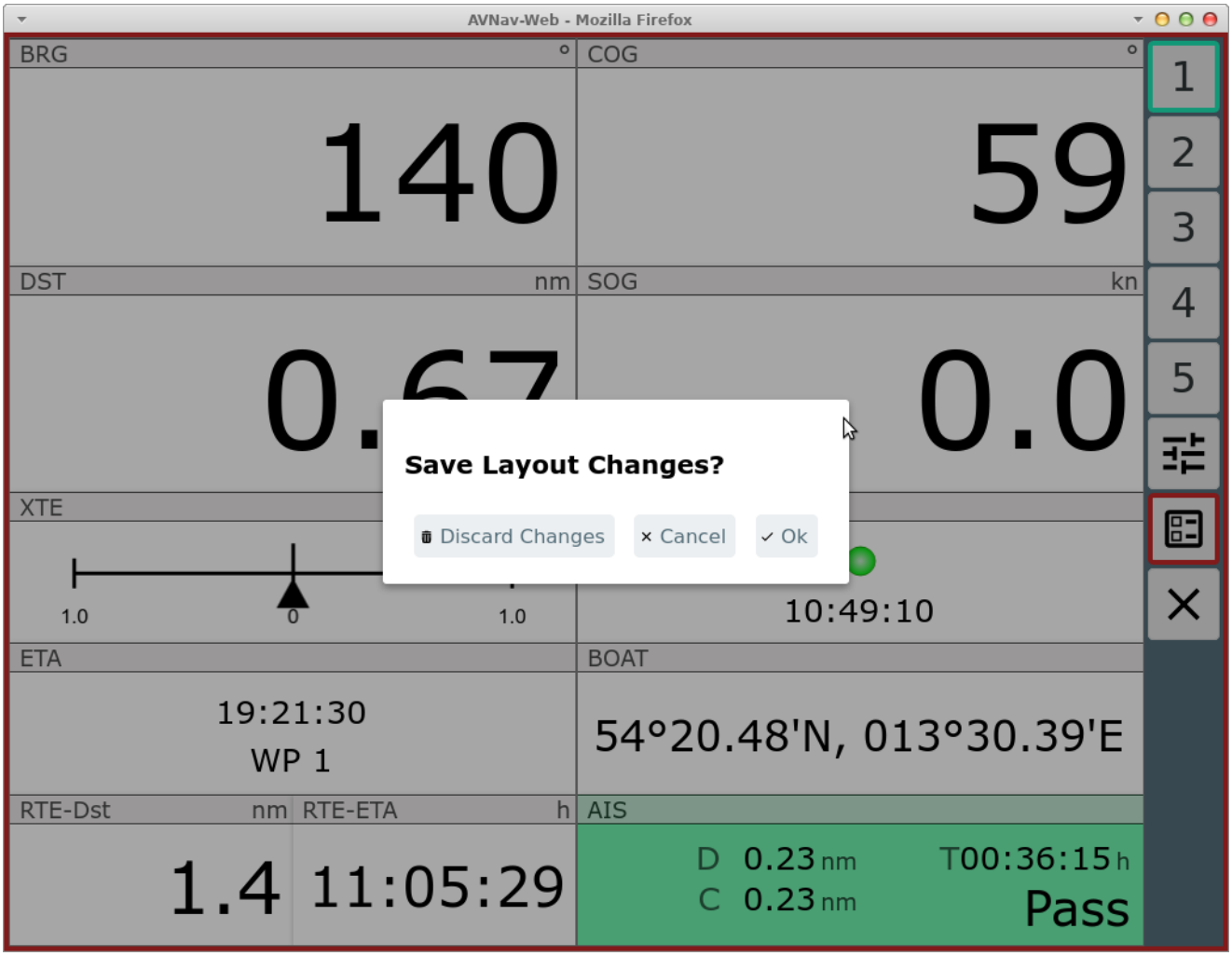


Auf den Dashboard Seiten kann eine abweichende Konfiguration eingestellt werden, wenn die Ankerwache aktiv ist (im default layout z.B. auf Dashboard Seite 1 genutzt).

Leere Dashboard Seiten erscheinen später nicht in der Anzeige.

Es ist auch eine Konfiguration auf der Seite des Routen-Editors (von der Navigationsseite aus) möglich, hier muss allerdings die Liste der Wegepunkte und die Anzeige der editierten Route immer sichtbar bleiben.

Nach Abschluss der Layoutbearbeitung muss das Layout noch gespeichert werden - rot umrandeter Button auf jeder Seite.



Hier kann ggf. noch einmal entschieden werden, die Änderungen zu verwerfen.

Das Layout wird im aktuellen Browser sofort aktiv (und auf dem Server gespeichert). Andere Browser erhalten es erst, wenn dort die App neu geladen wird oder das Layout gewechselt wird.

Layout Download/Upload

Über die Files/Download Seite können unter die Layout-Files heruntergeladen/hochgeladen/gelöscht bzw. als Datei bearbeitet werden.

Das momentan aktive Layout kann hier allerdings nur heruntergeladen werden.

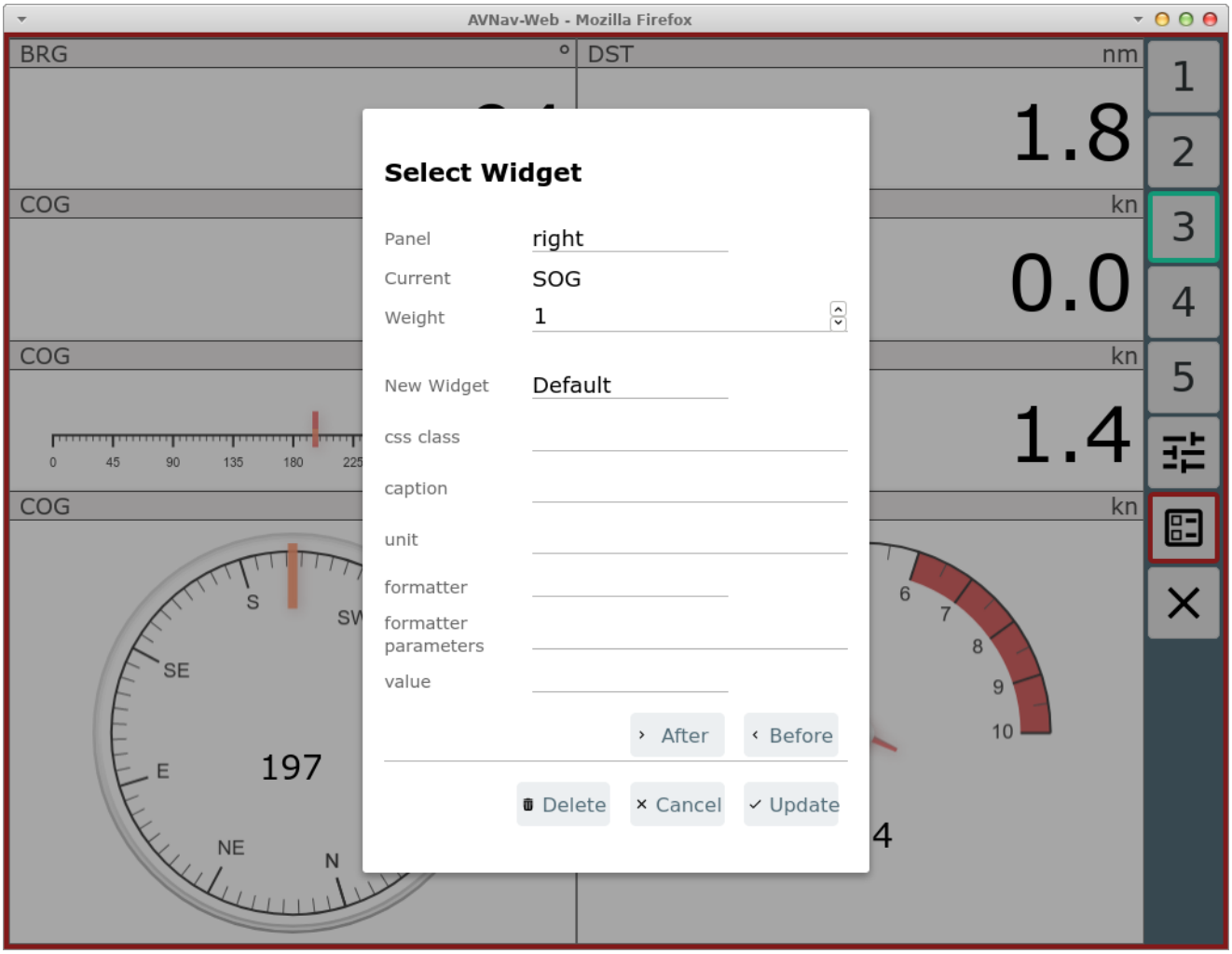
Wenn ein Layout gelöscht wird, das momentan in einem anderen Browser noch aktiv ist, wird dieser es wieder zum Server hochladen, wenn dort die App neu geladen wird. Das muss man ggf. beachten.

Spezielle Widgets

Neben einer ganzen Liste von vorkonfigurierten Widgets (wie z.B. SOG, COG, BRG, AisTargetWidget,...) bei denen ggf. nur die Beschriftung und die Parameterisierung des

Formatters geändert werden kann, gibt es einige spezielle Widgets

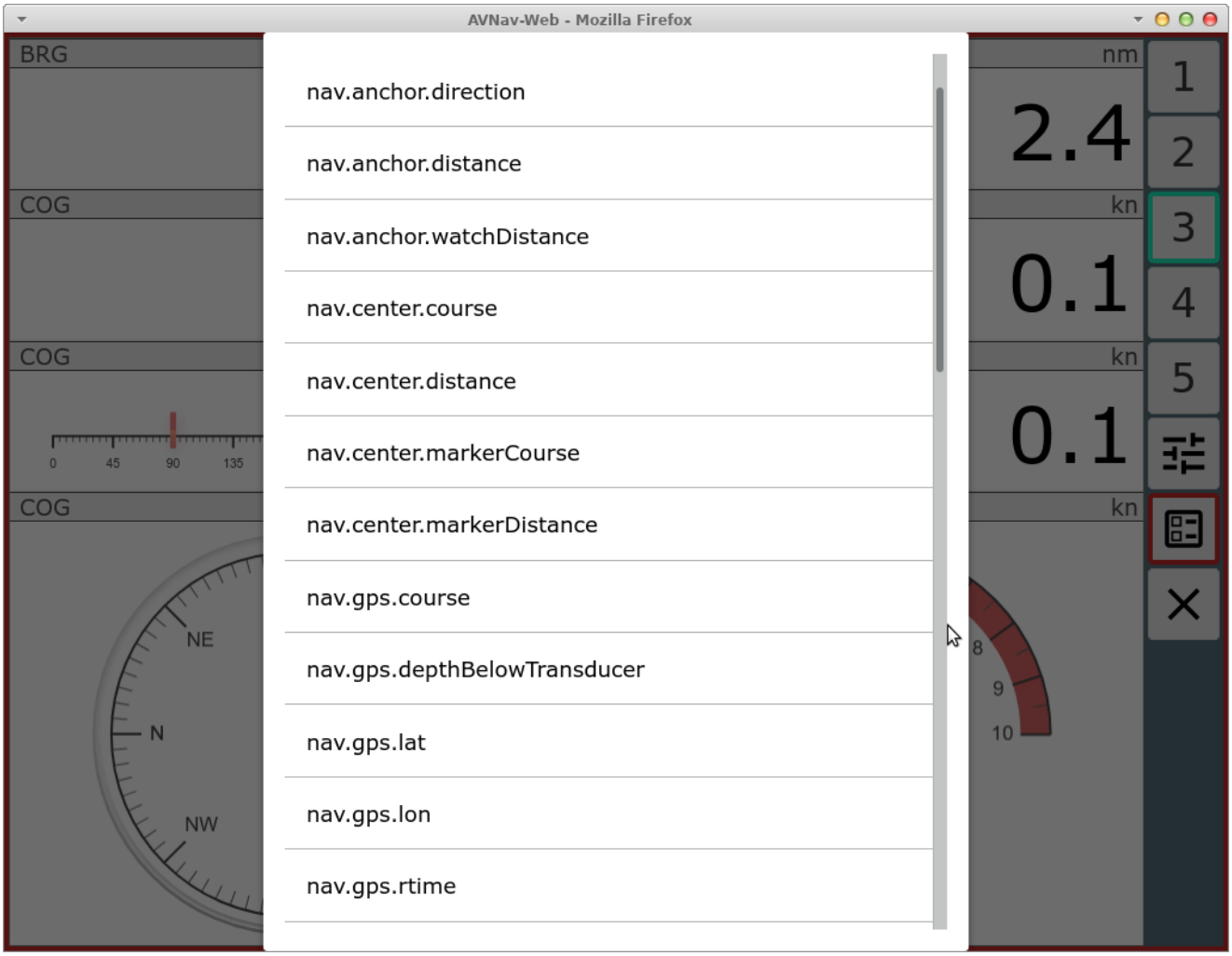
Default



Dieses Widget ist intern die Basis für die meisten anderen Widgets. Man kann damit recht einfach Anzeigen für bestimmte Werte realisieren.

Wichtig ist hier, dass in jedem Falle ein Formatter und ein "value" gewählt werden müssen.

Unter Value bekommt man eine Liste der zur Verfügung stehenden Anzeige-Daten.



Das sind einmal die in AvNav intern vorhandenen Daten. Falls die signalk Integration aktiv ist (nicht unter Android), erscheinen hier auch vorhandene Werte unter vessels/self mit dem Prefix nav.gps.signalk.

Die Auswahl des Formatters muss passend zum Wert erfolgen.

Gauges

Wie oben bereits erwähnt, stellt AvNav eine Integration von canvas-gauges bereit. Basierend auf dieser Bibliothek bringt AvNav einige vorbereitete Widgets mit, deren Parameter jeweils konfigurierbar sind.

Es gibt lineare Anzeigen (linGauge...) oder radiale Anzeigen (radGauge...).
Über den Layout-Editor sind nur bestimmte Parameter direkt anpassbar.

Falls die Anzeigen weitergehend verändert werden sollen, sollte dazu eine Anpassung über nutzerspezifischen JavaScript code erfolgen.

SignalK

Das SignalK plugin bringt auch einige Widgets mit, die insbesondere Formatter für Daten enthalten, die so in avnav sonst nicht verfügbar sind - z.B. signalKCelsius zur Anzeige von

Temperaturen. Diese machen natürlich nur Sinn, wenn entsprechende SignalK Daten vorhanden sind.

Anpassung des Aussehens

Falls man das Aussehen eines Widgets anpassen möchte, empfiehlt es sich, eine neue css class zu vergeben - auf diese kann man dann mit nutzerspezifischem css zugreifen.

Eigene Widgets

Mit nutzerspezifischem Java Script Code kann man relativ leicht auch eigene Anzeigen einbauen - sowohl mit simplem HTML als auch mit grafischen Elementen.

Nutzer Symbole

Seit Version 20201030.

Eine Reihe der in AvNav genutzten Symbole kann man an die eigenen Bedürfnisse anpassen. Man kann die vorhanden Symbole in ihrer Größe ändern, verschiedene Eigenschaften einstellen, oder sie durch eigene Symbole ersetzen.

Falls man eigene Symbole nutzen möchte, müssen diese als .png Dateien in das Images Verzeichnis hochgeladen werden - siehe bei der Beschreibung zu Nutzer-Dateien.

Welche Symbole verändert werden sollen (und wie), wird in einer JSON Datei im Nutzer Verzeichnis beschrieben - images.json.

Diese Datei hat den folgenden Aufbau (Beispiel):

```
{
  "boatImage": {
    "anchor": [20,0],
    "size":[40,71]
  },
  "boatImageHdg:{
    "src": "/user/images/SpecialBoat.png",
    "anchor": [15,15],
    "size":[30,70]
  }
  "markerImage":{
    "src": "/user/images/Marker.png",
    "anchor": [15,15],
    "size":[30,30]
  },
  "aisNormalImage-Sail":{
    "src": "/user/images/Sail-Boat-40.png",
    "anchor": [15,15],
    "size":[30,30],
    "courseVectorColor": "#ff00ff",
    "rotate":false
  },
```

```
    "aisNormalImage-Military":{
      "anchor": [32,0],
      "size":[64,120]
    }
  }
```

Ab 20230614 existiert eine Basis-Konfiguration die das System nutzt. In der Nutzer-Datei können Einträge überschrieben werden.

Für jedes zu ersetzende Symbol muss ein Eintrag mit dem entsprechenden Namen existieren.

Allgemeine Images

boatImage	Das Symbol für das Boot auf der Navigationsseite
boatImageHdg (20220421)	Das Symbol für das Boot wenn hdm oder hdt zur Anzeige genutzt werden
boatImageSteady (20220421)	Das Symbol für das Boot, wenn zero SOG detect aktiviert ist und das Boot sich nicht bewegt
markerImage	Das Symbol für den aktuellen Ziel-Wegepunkt
anchorImage	Das Symbol für den Anker bei aktiviertem Anker-Alarm
measureImage	Das Symbol für den Startpunkt der aktuellen Messung

AIS Images

Bei den AIS Images gibt es eine ganze Reihe von Optionen. Jedes AIS Ziel kann in einem bestimmten Zustand sein (gekennzeichnet durch eine entsprechende Farbe)

Zustand	Bedeutung
Normal	AIS Ziel
Warning	Das nächste AIS Ziel, das die minimal eingestellte CPA unterschreitet
Tracking	Das AIS Ziel, das über die AIS Info Seite ausgewählt wurde

Nearest	Das nächste AIS Ziel

Daneben können die AIS Images noch abhängig von der Art (Normal/Aton) und verschiedenen Parametern (ship type, navigation status, aid type) unterschieden werden.

Es ist prinzipiell möglich für jede der möglichen Kombinationen ein eigenes Icon zu definieren. Einfacher ist es jedoch (ab 20230614) , die Handling des Zustandes (Farbe) AvNav zu überlassen. Dazu müssen die Icons eine Farbe enthalten, die dann bei der Darstellung durch die entsprechende Farbe des Zustandes ersetzt wird. Diese Farbe muss dazu AvNav über den Parameter "replaceColor" mitgeteilt werden (siehe die defaults für Beispiele).

Es sind damit im Wesentliche folgende Icon-Angaben möglich (Beispiele):

Key	Bedeutung
aisImage	default Ais icon
aisImage-status1	Icon für AIS Ziele mit dem navigational Status 1 (At Anchor), für eine Liste der Werte siehe den source code.
aisImage-typeFishing	Icon für AIS Ziele mit dem ship type 30 (Fishing), siehe den code für die Werte
aisatonImage	default AIS Icon für atons
aisatonImage-type9	AIS Icon für atons mit type 9 (Beacon, Cardinal N), siehe den code für die Werte.

Falls man nicht mit "replaceColor" arbeiten möchte, kann man unterschiedliche Icons für die Zustände angeben:

aisWarningImage, aisNormalImage, aisWarningImage-status1, ...

Icon Parameter

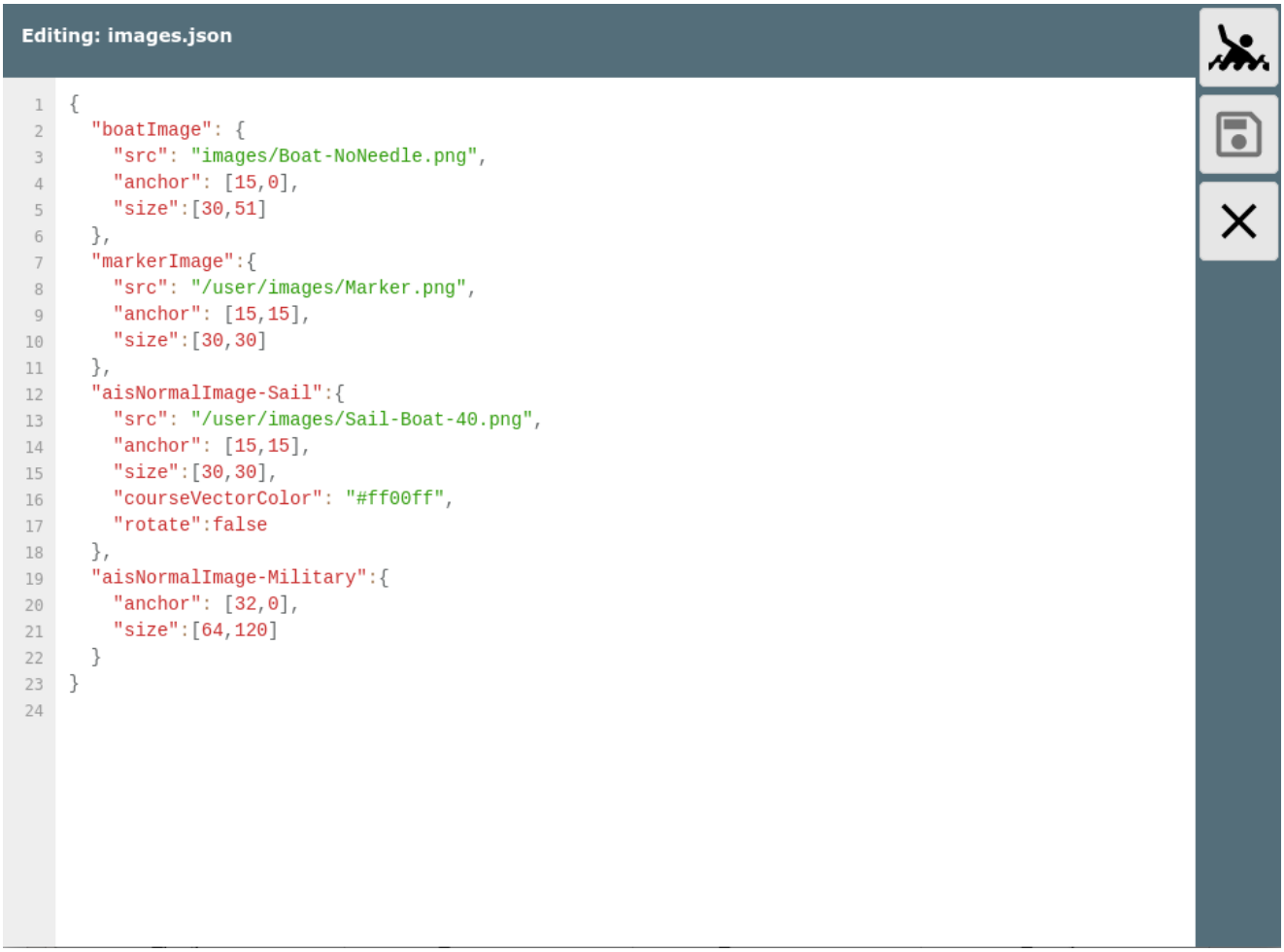
Die folgenden Parameter können für jedes Symbol definiert werden:

src	Die URL für die Image Datei. Typisch /user/images/XYZ.png für eine Datei, die über die Download Seite hochgeladen wurde. Falls dieser Parameter nicht angegeben wird, wird

	das in AvNav vorhandene Symbol genutzt - man kann aber z.B. mit den anderen Parametern die Größe ändern. Die Bilddateien sollten ein wenig grösser sein, als das was man bei size angibt - z.B. Faktor 2 (aber nicht zu gross, da sonst die Performance leidet). Wenn man Vektorgrafiken hat, kann man z.B. inkscape nutzen, um daraus png's zu erzeugen	
size	[breite,höhe] - muss als Array (siehe Beispiel) angegeben werden. Das beschreibt die Größe des Symbols (die Bilddatei wird auf diese Größe skaliert). Falls man keinen src Parameter angibt, kann man hiermit die Größe des internen Symbols verändern.	
anchor	[x,y] - der Punkt des Symbols (bezogen auf Breite und Höhe), der auf die aktuelle Position auf der Karte gesetzt werden soll.	
rotate	true oder false - wenn auf false gesetzt, wird das Symbol nicht entsprechend des aktuellen Kurses rotiert	nicht für markerImage
courseVector	true oder false - wenn auf false, wird für dieses Symbol kein Voraus-Vektor gezeichnet (auch wenn es über die Einstellungen aktiv ist)	nicht für markerImage
courseVectorColor	die Farbe für den course Vektor. Hier kann man eine Farbe wählen, die zu den genutzten Bildern passt.	nicht für markerImage
replaceColor (ab 20230614)	Die Farbe, die je nach Zustand zu ersetzen ist	nur ais...Image
textOffset (ab 20230614)	Ein Array [x,y] für den Basis-Text-Offset. Zusätzlich wird noch ein weiterer Offset je nach Kurs berechnet (primär y). Der X Wert muss sich an der Größe des Icons (size Parameter) orientieren	nur ais...lImage

Nicht angegebene Parameter werden jeweils durch default Werte ersetzt. Es ist auch möglich, für bestimmte Kombinationen nur Parameter anzugeben und kein eigenes Icon - damit kann man z.B. die AIS Symbole unterschiedlich groß gestalten. Typischerweise muss bei Änderung von size auch anchor geändert werden.

Bei der Bearbeitung der images.json muss darauf geachtet werden, valides json zu erzeugen. Wenn man es innerhalb von AvNav auf der Files/Download Seite (Unterseite) bearbeitet, wird beim Speichern automatisch eine Syntax-Prüfung vorgenommen.



Nach der Änderung von images.json muss AvNav neu geladen werden.

Tastatur Unterstützung

AvNav hat eine Unterstützung für die Bedienung wichtiger Funktionen über Tastenkürzel.
Die Zuordnung zwischen Tasten und Funktionen kann dabei relativ frei konfiguriert werden.

Prinzip

Die Zuordnung erfolgt dabei über 3 Stufen:

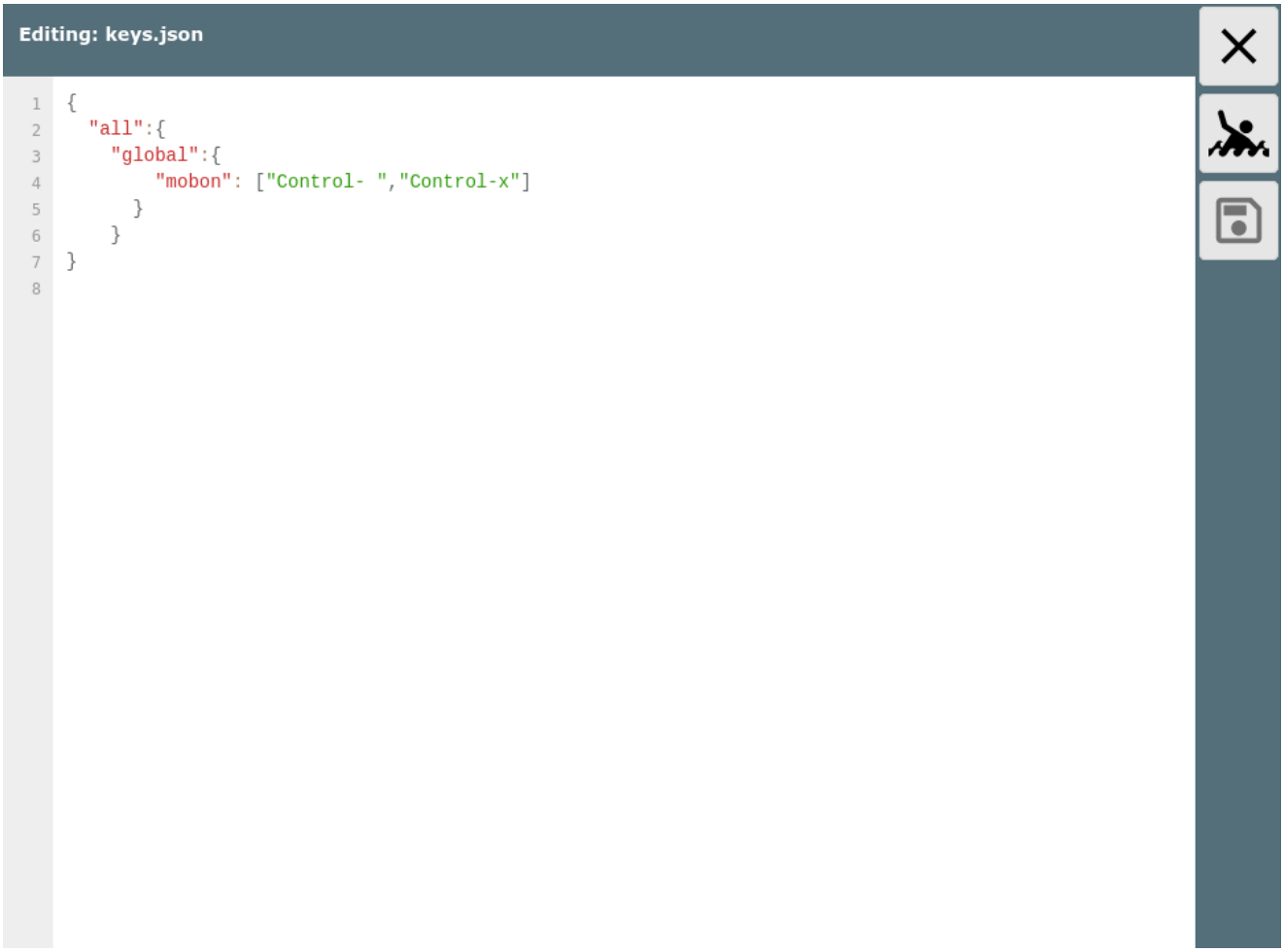
1. Seite
   Das ist die in AvNav momentan angezeigte Seite (siehe Nutzerbeschreibung). Über den speziellen Namen "all" kann die Funktion auf allen Seiten zugeordnet werden.
2. Gruppe
   Hier sind die Funktionen noch einmal gruppiert - z.B. "button"
3. Funktion
   Die eigentliche Funktion, die ausgelöst werden soll (z.B. der Klick auf einen Button)

Es kann dabei den jeweiligen Funktionen eine oder mehrere Tasten zugeordnet werden. Ein spezifischere Konfiguration gewinnt dabei (also wenn es z.B. eine Zuordnung für die Seite "all" gibt und eine andere für z.B. die Seite "navpage", dann gewinnt die letztere).

Konfiguration

Die Zuordnung der Tasten erfolgt über eine Datei keys.json im Nutzer-Verzeichnis. Diese Datei kann dort direkt bearbeitet werden. Es gibt dazu noch eine in AvNav eingebaute Datei mit den default-Zuordnungen.

In der Datei im user Verzeichnis können die Werte aus der default-Datei überschrieben werden.



Mit diesem Beispiel werden auf allen Seiten dem Button "Mann über Board" die Tasten Ctrl-Leer und Ctrl-x zugeordnet.
Wenn nur eine Taste zugeordnet werden soll, müssen keine eckigen Klammern angegeben werden. Nach dem Speichern der Änderungen muss die AvNav Seite neu geladen werden.

Seiten Gruppen und Funktionen

Die Liste der Seiten, Gruppen und Funktionen ist hier immer nur der aktuelle Stand beim Erstellen der Dokumentation. Es werden Stück für Stück weitere hinzu kommen.

Die Namen für die Keys entsprechen den Werten laut der Dokumentation. Wenn die Control (Strg) Taste dazu gedrückt ist, wird ein "Control-" vor den Namen gesetzt.
Die Namen der Funktion in der Gruppe "button" sind jeweils die Namen der Buttons, so wie sie in der Nutzerbeschreibung dokumentiert sind.
Ein Klick auf ein Widgets kann über die Gruppe "widgets" und den Namen des Widgets erreicht werden (die Namen sieht man im Layout editor). Ein SOG widget wäre z.B. mit

```
"all":{
   "widgets": {
      "SOG": "s"
```

```
    }
  }
```

mit der Taste s auf allen Seiten anklickbar.

Die Buttons in Dialogen sind über die Gruppe "dialogButton" und den Namen des buttons erreichbar. Diesen kann man leicht aus dem HTML code z.B. mit den Entwicklertools des Browsers ablesen. Man sollte allerdings nur spezielle Keys den dialogButtons zuordnen, da sonst potentiell keine normale Werte-Eingabe mehr möglich ist.

In der folgenden Tabelle sind die Gruppen und Funktionen aufgelistet, die entweder in den default Einstellungen bereits eine Taste zugewiesen haben - oder aber weder button, dialogButton noch widget sind. Texte in Klammern in der Tabelle sind Hinweise zur Funktion.

Zuweisungen

Seite	Gruppe	Funktion	Default Keys
all	button	Cancel	"Escape"
	map	zoomIn	["+","PageUp"]
		zoomOut	["-","PageDown"]
		up	"ArrowUp"
		down	"ArrowDown"
		left	"ArrowLeft"
		right	"ArrowRight"
		lockGps (Kartenmitte auf Position)	"l"
		unlockGps	"u"
		toggleGps	["t","Control-a"]
		toggleCourseUp	"b"
		centerToGps (einmalig Boot in	

		Kartenmitte)	
	alarm	stop	"a"
	global	mobon	["Control- "]
		moboff	
		mobtoggle	
		anchoron (Anker Alarm an an aktueller Position, seit 20220421)	"i"
		anchoroff(seit 20220421)	"Control-i"
	addon	0 (erstes addon)	"Control-0"
		1	"Control-1"
		2	"Control-2"
		3	"Control-3"
		4	"Control-4"
		5	"Control-5"
		6	"Control-6"
		7	"Control-7"
gpspage (Dashboard)	button	Cancel	["d","Escape"]
navpage (Navigationsseite)	widget	AisTarget	"a" (geht zur Ais Info Seite)
		COG	"d" (geht zum Dashboard, mit d kann man so zwischen Navigationsseite und

			Dashboard hin- und herschalten)
	button	LockMarker (starte Navigation zur Kartenmitte)	"g"
		StopNav	"s"
		ShowRoutePanel (gehe zum Routen-Editor)	["Control-r","r"]
	map	centerToGps (einmalig Boot in Kartenmitte)	"c"
	page	centerToTarget (aktuellen Wegpunkt in Kartenmitte)	"w"
		navNext (Navigation zum nächsten Punkt in der Route)	["n","Control-n"]
		toggleNav (Navigation ein/aus)	["Control-g"]
mainpage (Hauptseite)	page	selectChart (wähle die selektierte Karte und gehe zur Navigationsseite)	"Enter"
		nextChart	["Tab","ArrowDown"]
		previousChart	"ArrowUp"
	button	ShowSettings	"Control-+"
		ShowStatus	"Control-s"
		ShowGps	"d"

	Night	["c","Control-c","Control-g"]
infopage (Lizenz)		
addonpage (andere webseiten)		
addresspage (Anzeige der QR codes)		
statuspage (Server Status)		
wpapage (Wifi Steuerung)		
routepage (Routen Liste)		
downloadpage (Files/Download)		
settingspage (Einstellungen)		
editroutepage (Routen Editor)		
addonconfigpage (Konfiguration von User Apps)		
viewpage (Anzeige/Editieren)		

AvNav Server Konfiguration

== nicht für Android ==

Einführung

Der AvNav server liest beim Start seine Konfiguration aus einer xml Datei - avnav_server.xml. Diese Datei befindet sich normalerweise unter /home/pi/avnav/data auf dem Raspberry, sonst unter $HOME/avnav - siehe Installation.

Wenn diese Datei beim ersten Start noch nicht existiert, wird sie aus einem Template erzeugt - passend für den Raspberry oder andere Systeme.
Dieses Template ist auf dem Raspberry Pi (mit dem Paket avnav-raspi) die Datei /etc/avnav_server.xml (ab 20230426). Wenn diese nicht existiert, wird eine Datei aus dem Paket als Template genutzt.
Falls AvNav von der Kommandozeile über das Kommando "avnav" gestartet wird, kann mit der Option -t ein Template angegeben werden.

Bei Updates der AvNav Software wird diese Datei im Allgemeinen nicht geändert. Es kann aber sein, dass für neue Funktionen neue Einträge nötig werden. Dann wird in den Release Notes darauf hingewiesen.

Mit jedem erfolgreichen Start (ab Version 20200325) schreibt AvNav eine Kopie diese Datei mit der Endung .ok. Falls beim nächsten Start das Parsen der xml Datei fehlschlägt, liest er stattdessen die .ok Datei. Diese Funktion soll verhindern, dass nach einer Änderung, die AvNav in manchen Situation selbst vornimmt, der nächste Start ggf. scheitert.

Wenn AvNav nicht mehr starten kann wegen Fehler in der Konfiguration, kann man die avnav_server.xml komplett entfernen und danach noch einmal starten. AvNav startet dann wieder von einem "sauberen" Template.

Beginnend ab Version 20210322 **es ist nicht mehr nötig, die Datei "per Hand" zu bearbeiten**.
Stattdessen sollte AvNav selbst (Server/Status Seite) für die Bearbeitung der meisten Parameter genutzt werden. Das vermeidet auch die Notwendigkeit eines Restarts nach Änderungen.
In den folgenden Beschreibungen wird in der Spalte "online" angezeigt, ob die Parameter direkt auf der Server/Status Seite geändert werden können.

Wenn Parameter geändert werden müssen, die nicht direkt bearbeitbar sind, sollte das avnav-update-plugin genutzt werden, um die Datei direkt im Browser zu bearbeiten.
Die Hinweise hier darunter sind also nur noch eine Zusatzinformation.

Wenn man Änderungen an der Konfiguration vornimmt, muss AvNav danach neu gestartet werden (das gilt aber nicht für Änderungen, die direkt auf der Server/Status Seite vorgenommen wurden). Wenn AvNav als Systemdienst läuft, macht man das mit dem Kommando

```
sudo systemctl restart avnav
```

Es empfiehlt sich jedoch, nach einer Änderung AvNav zunächst einmal nur von der Kommandozeile zu starten, um zu sehen, ob es schwerwiegende Fehler gibt. Die Kommandofolge ist dann

```
sudo systemctl stop avnav
avnav -e
^C
sudo systemctl start avnav
```

Die option -e gilt erst ab Version 20200325. Sie verhindert, dass im Fehlerfall die avnav_server.xml.ok geladen wird. ^C bricht das laufende AvNav wieder ab.

Inhalt

Innerhalb der avnav_server.xml sind Einträge für die einzelnen Bestandteile von AvNav enthalten. In den Templates sind bereits viele kommentierte Beispiele für entsprechende Einstellungen.

Grundsätzlich gibt es 3 Kategorien von solchen Bestandteilen:

1. Anteile, die nur genau einmal auftreten dürfen, die aber unbedingt in der avnav_server.xml stehen müssen
   Beispiele: AVNConfig, AVNHttpServer,...
2. Anteile, die im Normalfall nicht in der avnav_config.xml stehen müssen, nur wenn etwas Spezielles konfiguriert werden soll
   Beispiele: AVNAlarmHandler, AVNChartHandler,...
3. Anteile, die ein- oder mehrfach in der avnav_server.xml stehen können. Das sind insbesondere die Eingangs- und Ausgangskanäle. Wenn kein solcher Eintrag vorhanden ist, steht die Funktion nicht zur Verfügung.

Es gibt einige Eigenschaften, die an mehreren Bestandteilen auftauchen, für diese hier eine Erklärung.

Name	Beschreibung	Beispiel
enabled	Viele Handler können auf der Server/Status Seite mit diesem Parameter ein- bzw. ausgeschaltet werdem	ein
name	Name eines Input oder Output Kanals. Dieser wird auf der Status-Seite angezeigt und kann auch im Parameter blackList für Filterungen genutzt werden	nmea0183tosignalk
filter	Filterung von NMEA Daten. hier können durch Komma getrennte Filter angegeben werden, die bestimmen, welche NMEA Daten durchgelassen werden. Um sie unabhängig von Talker Ids zu machen, werden die 2 Zeichen nach einem $ nicht berücksichtigt. Ein Filter für $GPRMC sieht dann so aus: $RMC. Wenn dem Filter ein ^ vorangestellt wird, wird er negiert, also ^$RMC heisst: keine RMC records. AIS Daten kann man mit dem Filter "!" oder "!AIVDM" matchen.	$RMC,^$RMB,!AIVDM
readFilter	Für kombinierte Reader/Writer ein Filter für die Eingangsseite. Siehe filter	
blackList	Liste von Kanal-Namen, deren Daten nicht ausgesendet werden sollen. Schreibweise beachten (grosses L)	nmea0183tosignalk
priority (since 20220421)	Alle NMEA Input Kanäle haben ein priority Feld. Dieses beeinflusst, welcher Wert gewinnt, wenn die gleichen Werte von mehreren Kanälen dekodiert werden. Die default priority ist 50, sie kann nach oben und unten geändert werden. Die SignalK Integration hat die default Priority 40.	50

Im Folgenden sind die wichtigsten Bestandteile mit ihren Parametern aufgeführt. Falls Parameter hier nicht beschrieben sind, aber ggf. in einem Template auftauchen, sollte sie so belassen werden, wie sie dort sind.

AVNConfig

Basis Konfiguration und Systemzeit, Kategorie 1 (1x,nötig)

Name	Online	Beschreibung	default/template
settimecmd		Ein Kommando, das aufgerufen wird, um die Systemzeit zu setzen. Der Parameter ist ein Zeitstempel in UTC so wie er für date -u benötigt wird.	nur gesetzt mit dem avnav-raspi Paket
settime (ab 203304xx)	X	Wen ein, setze die Systemzeit (settimecmd muss ebenfalls gesetzt sein)	ein
maxtimeback	X	maximale Zeit, die die Systemzeit rückwärts gesetzt wird, bevor alle internen Daten gelöscht werden (s)	5
systimediff	X	maximale Zeitabweichung der Systemzeit von der gps-Zeit bevor die Systemzeit neu gesetzt wird (s)	5
settimeperiod	X	Zeit in s bevor die Systemzeit erneut gesetzt wird	3600
ntphost (ab 20220421)	X	Ein ntp server. Dieser wird befragt, wenn keine gültige GPS Zeit vorhanden ist und settime aktiv ist	pool.ntp.org
switchtime (ab 20220421)	X	Zeit(in s) die nach dem Setzen der Zeit mindestens gewartet wird, bevor von gps zu ntp oder zurück gewechselt wird. Diese Zeit wird auch nach dem Start auf eine gültige GPS Zeit gewartet wird	60

ownMMSI	X	MMSI des eigenen Bootes, diese wird aus den AIS Daten ausgefiltert	
expiryTime	X	Zeit (in s) die empfangene NMEA Daten gültig bleiben	30
aisExpiryTime	X	Zeit (in s) die empfangene AIS Daten gültig bleiben	1200

AVNFeeder

Die interne NMEA Liste und Dekodier-Einheit. Kategorie 2 (1x, nicht nötig ab Version 20200325).

Diese Einheit hat keine Parameter mehr. In früheren Versionen wurde hier der gpsd für die Dekodierung genutzt, daher sind u.U. noch Parameter zu finden, die sich darauf beziehen. In neueren Versionen sollte der Eintrag so geändert werden, dass useGpsd="false" enthalten ist.

AVNHttpServer

Der interne HTTP server. Kategorie 1 (einmal, erforderlich).

Neben den Parametern für AVNHttpServer gibt es einige Unter-Einträge, die sich mehrfach wiederholen können. Im Normalfall sollten hier aber keine Änderungen nötig sein (Directory,MimeType).

Ausser dem httpPort sollten normalerweise keine Änderungen erforderlich sein.

Parameter für AVNHttpServer

Name	Beschreibung	default/template
httpPort	der Port, auf dem der HTTP Server Anfragen annimmt	8080
navurl	REST interface, nicht änderbar	/viewer/avnav_navi.php
index	Startseite, nicht änderbar	/viewer/avnav_viewer.html
httpHost	Die Bind Adresse, man kann hier z.B. auf ein bestimmtes Netzwerk beschränken	0.0.0.0

numThreads	Die Zahl der vom Server genutzten Threads	5

Parameter für Directory

Diese Werte werden meist durch den Aufruf (Parameter -u bei avnav) überschrieben und sollten nicht geändert werden.

Name	Beschreibung	default/template
urlpath	die URL (ohne /)	
path	der reale Pfad auf dem System	

Parameter für MimeType

Hier werden mime types für Dateinamensendungen konfiguriert. Falls eine eigene Anwendung hier ggf. etwas spezielles benötigt, kann man das ergänzen.

Name	Beschreibung	default/template
extension	Namens-Endung (z.B. .avt)	
type	Mime type (z.B. text/plain)	

AVNBlueToothReader

Lesen von Bluetooth Geräten mit seriellem Profil. Kategorie 3 (einmal möglich, optional) Nur möglich, wenn das Gerät ein Bluetooth Device hat.

Name	online	Beschreibung	default/template
maxDevices	X	Anzahl der maximal gleichzeitig verbundenen Bluetooth Geräte	5
deviceList	X	Komma-separierte Liste von Bluetooth Geräte-Ids. Wenn gesetzt, werden nur diese Geräte verbunden.	

filter	X	filter für NMEA Daten
name	X	
enabled	X	
priority	X	

AVNSerialReader

Lesen von seriellen Geräten. Kategorie 3 (mehrfach, optional). Dieser Reader sollte nur für direkt per Hardware (UART) verbundene Geräte genutzt werden, für Geräte, die per USB angeschlossen sind ist der AVNUsbSerialReader zuständig.

Name	online	Beschreibung	default/template
name	X	Kanal Name für die Nutzung in blackList und für die Anzeige	intern gebildeter Name
port	X	Gerätename, z.B. /dev/ttyAMA0	
baud	X	Baudrate. Wenn minBaud auch angegeben ist, die maximale Baudrate, die für das automatische Feststellen der Baudrate genutzt wird	4800
minbaud	X	Minimale Baudrate, die für eine automatische Erkenung genutzt wird. Wenn nicht gesetzt oder 0 - automatische Erkennung aus	
timeout	X	Timeout in s, nach dem das Gerät ohne Daten geschlossen und wieder geöffnet wird	1
bytesize	X	serielle Byte Größe	8
parity	X	Parity	N
stopbits	X	Anzahl der Stopbits	1

xonxoff	X	Nutzung xon/xoff Protokoll (0: aus)	0
rtscts	X	RTS/CTS Nutzung (0: aus)	0
numerrors	X	Anzahl der Fehler, nach der das Gerät geschlossen und neu geöffnet wird.	20
autobaudtime	X	Zeit in s, die versucht wird, ein Newline in den Daten zu erkennen (während der automatischen Baudraten-Erkennung)	5
filter	X	NMEA Filter, siehe filter	
enabled	X		
priority	X		

AVNSerialWriter

Ausgang über ein serielles Gerät. Auch kombiniert Ein- und Ausgang. Kategorie 3 (optional)
Nur für direkte serielle Geräte, nicht für USB-Wandler (AVNUsbSerialReader für diese)

Name	online	Beschreibung	default/template
name	X	channel name	
combined	X	wenn "true", dann gleichzeitig Eingang und Ausgang	false
readFilter	X	filter für die Eingangsrichtung. Der Parameter "filter" bezieht sich auf die Ausgangsrichtung!	
blackList	X	blackList, Komma getrennte Liste von Kanalnamen, deren Daten nicht ausgegeben werden sollen.	
....		alle Parameter von AVNSerialReader	

AVNUsbSerialReader

Behandelt über USB angeschlossene serielle Geräte. Kategorie 3(einmal, optional).
Dieser Worker sucht alle über USB verbundenen Geräte. Solche mit einem seriellen Profil versucht er zu öffnen, automatische die Baudrate einzustellen und dann NMEA Daten zu lesen. Damit werden solche Geräte normalerweise komplett automatisch von AvNav erkannt. Man kann für einzelne Geräte Regeln definieren, um sie speziell zu behandeln. Als Identifikation für ein Gerät wird dabei eine ID genutzt, die die enstprechende USB Buchse identifiziert. Mann kann diese ID am einfachsten ermitteln, indem man bei Einstecken des Gerätes die Status Seite beobachtet.

Die Parameter gliedern sich in 2 Teile:

- Attribute für den Eintrag selbst
- Darunter liegende Einträge des Types UsbDevice

Beispiel

```
<AVNUsbSerialReader maxDevices="5" allowUnknown="true" baud="38400"
minbaud="4800">
        <UsbDevice usbid="1-1.2.1:1.0" baud="38400" minbaud="4800"
filter="$RMC"/>
</AVNUsbSerialReader>
```

Parameter für AVNUsbSerialReader

Name	online	Beschreibung	default/template
maxDevices	X	maximale Zahl von gleichzeitig verbundenen USB Geräten	5
allowUnknown	X	nur wenn dieser Eintrag auf "true" steht, werden Geräte eingebunden, die nicht explizit mit UsbDevice konfiguriert sind	true
...	X	alle Parameter von AVNSerialReader bis auf port. Diese werden für nicht explizit konfigurierte Geräte gesetzt.	

Parameter für UsbDevice

Name	online	Beschreibung	default/template

usbid	X	USB Port identifikation z.B. "1-1.2.1:1.0", erforderlich	
type	X	Type des Gerätes reader, writer, combined, ignore, setze ignore, wenn das Gerät nicht genutzt werden soll	reader
...		alle Parameter von AVNSerialReader wenn der type = "reader" ist (bis auf port, dieser wird intern gesetzt)	
...		alle Parameter von AVNSerialWriterwenn der type combined oder writer ist (bis auf port, dieser wird intern gesetzt)	

AVNUdpReader

Öffnet einen UPD port und verarbeitet dort hereinkommende Daten. Kategorie 3(optional, mehrfach).

Name	online	Beschreibung	default/template
name	X	Kanalname für die Nutzung in blackList und in der Anzeige	intern berechnet
port	X	UDP port	
host	X	Bind Adresse für den Port. Damit kann der Empfang z.B. auf localhost begrenzt werden.	0.0.0.0
minTime	X	wenn gesetzt: Wartezeit in s bevor ein weiterer Datensatz empfangen wird. Hiermit kann u.U. die Datenrate begrenzt werden.	0
filter	X	filter für NMEA Daten	
enabled	X		
priority	X		

stripLeading	X	Entferne alle Zeichen vor $ oder ! in einer Zeile.	aus

AVNUdpWriter

Sendet NMEA Daten per UDP. Kategorie 3 (optional, mehrfach)

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
port	X	UDP Ziel port	
host	X	UDP Zieladresse	
filter	X	filter NMEA Daten, die gesendet werden	
broadcast	X	muss auf true gesetzt werden, wenn die Daten als broadcast geschickt werden sollen	false
blackList	X	blackList für Kanalnamen, deren Daten nicht gesendet werden sollen	
enabled	X		

AVNSocketWriter

Ein Ausgang, der auf einem Port auf Verbindungen wartet und an diese die NMEA Daten ausgibt (TCP server). Kategorie 3 (mehrfach, optional).

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
port	X	der Listener Port	
address	X	wenn gesetzt, binde auf diese Adresse (sonst any: 0.0.0.0)	
filter	X	filter für NMEA Daten	

read	X	wenn true, werden auch Daten vom Socket gelesen	false
priority	X	nur wenn read aktiv ist	
readFilter	X	falls auch gelesen wird, NMEA filter für die Eingangsrichtung	
blackList	X	blackList durch Komma getrennte Liste von Kanalnamen, für die keine Daten ausgegeben werden	
minTime	X	minimale Zeit in s zwischen 2 gesendeten Nachrichten. Damit kann die Datenrate begrenzt werden.	0
avahiEnabled	X	wenn eingeschaltet, wird der Service über avahi als _nmea-0183._tcp bekannt gemacht	aus
avahiName	X	der Name für den avahi service	avnav-server
enabled	X		

AVNSocketReader

Ein Eingang, der sich mit einem TCP Server verbindet und von dort Daten liest (TCP client). Kategorie 3 (mehrfach, optional)

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
port	X	TCP Port zu dem eine Verbindung aufgebaut wird	
host	X	TCP Zieladresse zu der eine Verbindung aufgebaut wird	
timeout	X	Verbindungs-timeout in s	10

minTime	X	Minimale Zeit zwischen 2 empfangenen Nachrichten	0
filter	X	filter für NMEA Daten	leer
writeOut	X	sende NMEA Daten auf dieser Verbindung	aus
writeFilter	X	Filter für gesendete NMEA Daten	leer
blackList	X	, separierte Liste von Source-Namen, deren Daten nicht gesendet werden	leer
enabled	X		
priority	X		
stripLeading	X	Entferne alle Zeichen vor $ oder ! in einer Zeile.	aus

AVNNmea0183ServiceReader

Dieser handler ist dem AVNSocketReader sehr ähnlich. Aber anstelle der Konfiguration von host und port wird hier der Name des Services konfiguriert. AvNav sucht im Netz nach (MDNS/Bonjour/Avahi) Services vom Typ_nmea-0183._tcp . Wenn der Eintrag über die Web Oberfläche erfolgt, bietet AvNav die Liste der gefundenen Services zur Auswahl an. MIt diesem Handler kann eine Verbindung auch dann wieder aufgebaut werden, wenn sich z.B. die IP Adressen ändern.

Name	online	Description	default/template
serviceName	X	Der Name des Services (AvNav bietet eine Liste)	--
timeout	X	Verbindungstimeout in Sekunden	10
minTime	X	minimale Zeit zwischen 2 empfangenen Nachrichten.	0
filter	X	Filter für NMEA Daten	leer
writeOut	X	sende NMEA Daten auf dieser Verbindung	aus

writeFilter	X	Filter für gesendete NMEA Daten	leer
blackList	X	, separierte Liste von Source-Namen, deren Daten nicht gesendet werden	leer
name	X		
priority	X		
enabled	X		

AVNBME280Reader

Reader für BME280 per I2C. Kategorie 3 (optional)
Schreibt MDA und XDR Datensätze.
Nur sichtbar, wenn python3-smbus installiert ist.

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
addr	X	I2C Adresse des Sensors	0x77
interval	X	Zeit zwischen 2 NMEA Datensätzen in s	5
writeXdr	X	Schreibe XDR wenn true	true
writeMda	X	Schreibe MDA wenn true	true
namePress	X	XDR Transducer Name für Luftdruck	Barometer
nameHumid	X	XDR Transducer Name für Feuchtigkeit	Humidity
nameTemp	X	XDR transducer Name für Temperatur	TempAir
enabled	X		
priority	X		

AVNBMB180Reader

Reader für BMP180 per I2C. Kategorie 3 (optional)
Schreibt MDA und XDR Datensätze.

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
addr	X	I2C Adresse des Sensors	0x77
interval	X	Zeit zwischen 2 NMEA Datensätzen in s	5
writeXdr	X	Schreibe XDR wenn true	true
writeMda	X	Schreibe MDA wenn true	true
namePress	X	XDR Transducer Name für Luftdruck	Barometer
nameTemp	X	XDR transducer Name für Temperatur	TempAir
enabled	X		
priority	X		

AVNSenseHatReader

Reader für SenseHat I2C. Kategorie 3 (optional)
Schreibt MDA und XDR Datensätze.
Erfordert python3-sense-hat

Name	online	Beschreibung	default/template
name	X	Kanalname	intern berechnet
interval	X	Zeit zwischen 2 NMEA Datensätzen in s	5
writeXdr	X	Schreibe XDR wenn true	true
writeMda	X	Schreibe MDA wenn true	true
namePress	X	XDR Transducer Name für Luftdruck	Barometer

nameHumid	X	XDR Transducer Name für Feuchtigkeit	Humidity
nameTemp	X	XDR transducer Name für Temperatur	TempAir
nameRoll (ab 20220421)	X	XDR transducer Name für Roll	Roll
namePitch (ab 20220421)	X	XDR transducer Name für Pitch	Pitch
enabled	X		
priority	X		

AVNTrackWriter

Schreiben von Tracks im gpx Format und einem simplen ASCII Format. Kategorie 3 (einmal, optional)

Name	online	Beschreibung	default/template
interval	X	minimaler Abstand in s zwischen dem Schreiben von 2 Einträgen	10
mindistance	X	minimaler Abstand in m zwischen 2 Track Punkten	50
trackdir	X	Verzeichnis für tracks	<datadir>/tracks
cleanup	X	Maximale Länge des intern vorgehaltenen Tracks in Stunden. Trackdaten werden weiter in Dateien geschrieben, aber die App kann maximal diese Zeit (rückwärts) als Track bekommen.	25
writeFile	X	Schreibe eine Track Datei. Wenn ausgeschaltet Aufzeichnung nur im Speicher.	ein

AVNRouter

Verwalten von Routing Daten (Wegpunkte, Routen, Ankeralarm). Berechnung der AP Daten. Kategorie 1 (einmal, erforderlich).

Name	online	Beschreibung	default/template
name	X	Kanalname (genutzt für AP Daten)	intern berechnet
routesdir		Verzeichnis für routen	<datadir>/routes
interval	X	Intervall (in s) zwischen RMB Datensätzen	5
computeRMB	X	berechne einen RMB Datensatz wenn ein Wegpunkt aktiv ist	true
computeAPB	X	berechne einen APB Datensatz	false
useRhumbLine (ab 20220819)	X	benutze den rhumb line Modus für Routen	false
nextWpMode (ab 20220819)	X	Auswahl des Weiterschaltungs-Modus für den nächsten Wegepunkt in einer Route (late, 90, early)	late
nextWpTime (ab 20220819)	X	Die Wartezeit nach dem Wegepunktalarm (in Sekunden) bis zur Weiterschaltung zum nächsten Wegepunkt (nur nextWpMode = early)	10

AVNNmeaLogger

Schreibt NMEA logs in das track Verzeichnis. Kategorie 3 (einmal, optional).

Name	online	Beschreibung	default/template
maxfiles	X	Anzahl der Dateien (1 pro Tag), die aufgehoben werden	100
filter	X	filter für NMEA Daten	"$RMC,$DBT,$DBP"
interval	X	Minimale Zeit in s bevor ein Satz des gleichen Typs erneut geschrieben wird	5

enabled	X

AVNImporter

Importiert Karten, die noch konvertiert werden müssen. Kategorie 2 (einmal, nicht notwendig)

Name	online	Beschreibung	default/template
importDir		Verzeichnis aus dem die zu importierenden Karten gelesen werden	<datadir>/import
workDir		Arbeitsverzeichnis für den import	<datadir>/work
waittime	X	Zeit in s, die nach dem Finden einer Datei im Import-Verzeichnis gewartet wird, bevor der Konverter startet	30
knownExtensions	X	Liste der Dateinamensendungen, die in der App zum Hochladen in den Importer angeboten werden	kap,map,geo
keepInfoTime		Zeit in s, die eine Information über einen Import noch stehen bleibt	30
enabled	X		

AVNWpaHandler

Konfiguration von externen WLAN Verbindungen. Kategorie 3 (einmalig, optional)

Name	Beschreibung	default/template
wpaSocket	die Steuerverbindung zu wpa_supplicant	/var/run/wpa_supplicant/wlan-av1
ownSsid	eigene SSIDs, diese werden ausgeblendet	avnav,avnav1,avnav2
firewallCommand	wenn konfiguriert, kann damit der externe Zugriff über ein	sudo -n $BASEDIR/../raspberry/iptables-

WLAN freigeschaltet werden	ext.sh wlan-av1

AVNCommandHandler

Ausführen von Kommandos, u.a. für Alarme. Kategorie 2 (einmalig, nicht notwendig).

Der AVNCommandHandler selbst hat keine Parameter. Es können jedoch verschiedene Kommandos konfiguriert werden, die dann jeweils per Name angesprochen werden. Die default Konfiguration ist:

```
<AVNCommandHandler>
     <Command name="sound" command="mpg123 -q" repeat="1"/>
</AVNCommandHandler>
```

Parameter für Command

Name	Beschreibung	default/template
name	Name des Kommandos	
command	Auszuführender Befehl	
repeat	Zahl der Wiederholungen	1

AVNAlarmHandler

Management von Alarmen. Kategorie 2 (einmal, nicht notwendig).

Stark verändert ab 20220421.

Die default Konfiguration ist:

```
<AVNAlarmHandler>
        <Alarm name="waypoint" category="info" repeat="1"/>
        <Alarm name="connectionLost" category="info" repeat="1"/>
        <Alarm name="anchor" category="critical" repeat="20000"/>
        <Alarm name="gps"  category="critical" repeat="20000"/>
        <Alarm name="mob" category="critical" repeat="2"/>
```

```
</AVNAlarmHandler>
```

Ab der Version 20220421 sollten vorhandene Alarm-Einträge in avnav_server.xml gelöscht werden, falls nicht ein spezielles Kommando dort eingetragen werden soll.
Damit können die Sounds über die Sound Auswahl für die Kategorie definiert werden.

Falls mit Alarmen spezielle Kommandos ausgelöst werden sollen, können diese jedoch in der avnav_server.xml explizit gesetzt werden.

```
<AVNAlarmHandler>
        <!-- legacy way of configuring alarms - still supported but
not recommended, use category at least and optionally parameter -->
        <Alarm name="gps" category="critical" command="gpsAlarm"
parameter="$BASEDIR/../sounds/anchorAlarm.mp3" repeat="20000"/>
        <!-- with the next line we configuer a special command that
will be called when we receive a "sinking" notification from SignalK
             the sound is determined by the category - and this is
also the parameter that the command will receive -->
        <Alarm name="sk:sinking" command="sinkingAlarm"
category="critical" repeat="2"/>
</AVNAlarmHandler>
```

Parameter für Alarm

Name	Beschreibung	default
name	Name des Alarms	leer, erforderlich
category	Kategorie (info,critical)	leer
command	Kommando, das ausgeführt werden soll (muss bei AVNCommandHandler konfiguriert sein)	leer
autoclean	Schalte den Alarm ab, wenn das Kommando beendet wurde	aus
sound	Der Name einer Sound Datei - wenn angegeben wird diese genutzt (relative Namen beziehen sich auf das interne Sound Verzeichins oder auf das user-Verzeichnis). Wenn nicht gesetzt, wird der sound aus der Kategorie ermittelt oder wenn nicht gesetzt aus dem 'parameter'	leer

repeat	Anzahl der Kommando (und sound) Wiederholungen	1
parameter	Falls angegeben wird dieser Parameter dem Kommando übergeben. Falls weder sound noch category gesetzt sind, wird der Name als der Pfad zu einer Sound-Datei interpretiert.	

Parameter für AlarmHandler

Name	online	Beschreibung	default/template
infoSound	X	Name einer mp3 Datei für den Sound in der Kategorie info. Kann aus einer Liste von eingebauten sounds und Dateien im user-Verzeichnis gewählt werden (über Download Seite hochladbar)	waypointAlarm.mp3
criticalSound	X	Name einer mp3 Datei für den Sound in der Kategorie critical. Kann aus einer Liste von eingebauten sounds und Dateien im user-Verzeichnis gewählt werden (über Download Seite hochladbar)	anchorAlarm.mp3
defaultCommand	X	Kommando für Alarme die nicht explizit ein Komando definiert haben. Dieses Kommando muss beim AVNCommandHandler konfiguriert werden.	sound
stopAlarmPin	X	Nur auf Raspberry Pi. Wenn gesetzt (board Nummerierung), schaltet ein Low an diesem Pin Alarme aus	leer

AVNPluginHandler

Management von plugins. Kategorie 2 (einmalig, optional).
Der AVNPluginHandler verwaltet Plugins, die in verschiedenen Verzeichnissen installiert werden können. Es gibt 3 Verzeichnisse in denen Plugins gesucht werden:

- builtin: /usr/lib/avnav/server/plugins
- system: /user/lib/avnav/plugins
- user: /home/pi/avnav/data/plugins

Neben den Parametern für den Plugin Handler selbst können die jeweiligen Plugins Parameter erwarten. Der Name für das Plugin ergibt sich dabei aus der Kategorie und dem plugin Verzeichnis. Beispiel:

```
<AVNPluginHandler>
        <builtin-signalk enabled="true"/>
        <builtin-canboat enabled="true" allowKeyOverwrite="true"
autoSendRMC="30" sourceName="canboatgen"/>
</AVNPluginHandler>
```

Parameter für AVNPluginHandler

Name	Beschreibung	default/template
builtinDir	Verzeichnis für eingebaute Plugins, nicht änderbar	/usr/lib/avnav/server/plugins
systemDir	Verzeichnis für Plugins, die als separate Pakete installiert werden	/usr/lib/avnav/plugins
userDir	Verzeichnis für Nutzer Plugins	/home/pi/avnav/data/plugins

Parameter für builtin-canboat

Name	online	Beschreibung	default/template
enabled	X	Nur wenn auf true, ist das Plugin aktiv	false
allowKeyOverride	X	Muss gesetzt werden, wenn Datum und Zeit von canboat gelesen werden sollen	false
port	X	canboat json Port	2598

host	X	Host für den n2kd	localhost
autoSendRMC	X	falls für diese Zeit in Sekunden kein RMC im NMEA-Datenstrom gesehen wird, aber gültige Positionsdaten + Zeit von canboat vorhanden sind: sende RMC (ist wichtig für Datum/Zeit auf NMEA0183)	0 (aus)
sourceName	X	Kanalname, der für RMC genutzt wird	plugin-Name
timeInterval	X	minimale Zeit zwischen 2 NMEA2000 Zeit Werten, bevor diese gespeichert werden (Sekunden)	0.5
timePGNs	X	PGNs, die für das Setzen der Zeit genutzt werden	126992,129029

AVNChartHandler

Verwaltung der Karten. Kategorie 2 (einmal, muss nicht enthalten sein)

Name	Beschreibung	default/template
period	Zeitintervall zwischen 2 Lesevorgängen für das Kartenverzeichnis (Sekunden)	30
upzoom	Anzahl von zoom Stufen über der höchsten vorhandenen Stufe	2

AVNUserHandler

Verwaltung der Nutzer-Dateien. Kategorie 2 (einmal, muss nicht enthalten sein)

Name	Beschreibung	default/template
interval	Zeitintervall zwischen 2 Lesevorgängen für das Verzeichnis (Sekunden)	5

AVNImagesHandler

Verwaltung der Nutzer-Images. Kategorie 2 (einmal, muss nicht enthalten sein)

Name	Beschreibung	default/template
interval	Zeitintervall zwischen 2 Lesevorgängen für das Verzeichnis (Sekunden)	5

AVNUserAppHandler

Verwaltung der konfigurierten User Apps. Kategorie 2 (einmal, muss nicht enthalten sein)

Dieser Handler ist etwas speziell. Initial sind hier keine Konfigurationen zu finden, über die WebApp können aber Konfigurationen angelegt werden. Eine händische Änderung ist nicht empfohlen.

AVNAvahiHandler

Steuert die Registrierung von AvNav bei Avahi(MDNS/Bonjour).

Name	online	Beschreibung	default/template
serviceName	X	Der Name der in Tools sichtbar wird. Das ist nicht der Host Name, den man z.B. in avnav.local benutzt!	avnav
maxRetries	X	Wie viele Wiederholungen macht AvNav, wenn der gewählte Name bereits vergeben ist. Wiederholungen hängen ein "-nn" suffix an den Namen an.	20
timout	X	Timeout bei der Verbindung mit dem avahi daemon (s)	10
enabled	X		

AVNSignalKHandler

New with 20220421.
For a description refer to the SignalK Documentation.

Erweiterungen und Zusätze

Die Funktionalität von AvNav kann auf verschiedenen Wegen erweitert werden. Einige solcher Erweiterungen werden bereits mit AvNav installiert, andere als separate Pakete.

Einbindung anderer Webseiten

== nicht unter Android ==

Man kann andere Webseiten (sowohl externe als auch solche auf dem AvNav Server) als sogenannte "User Apps" einbetten.
Es gibt eine dialog-basierende Konfiguration dafür auf der User App Konfigurationsseite. Die seiten sind dann auf der User App Seite sichtbar.

Erweiterung der Funktionen der Web App

Mit einigen Zeilen java script code kann man Funktionalität zur Web App hinzufügen. Insbesondere kann man eigene Anzeigen (Widgets) definieren - sowohl textbasiert als auch grafisch. Ausserdem kann man Formatierer für Anzeigewerte ergänzen - oder auch Buttons, die verschiedene Aktionen anstossen.
Dieser java script code wird in einer user.js Datei gespeichert und AvNav enhält einen Editor für diese Datei auf der Files/Download Seite.
Neben dem java script code braucht man typischerweise auch css code um das Aussehen anzupassen, dieser wird in einer user.css Datei gespeichert.

Plugins

== nicht unter Android ==

Plugins erlauben es ti Funktionalität von AvNav sowohl auf der Server-Seite als auch im Display (client) zu erweitern. Es gibt einige Plugins, die AvNav bereits eingebaut mitbringt, andere können als separate Pakete installiert werden. Und man kann eigen plugins schreiben.

In AvNav eingebaute Plugins:

- SignalK
  Verbindungen zu einem SignalK Server. Siehe die detaillierte Beschreibung.
- Canboat
  Diese Plugin stellt einen NMEA 2000 support mit signalK und canboat bereit. Siehe die detaillierte Beschreibung.

Für die Entwicklung von eigenen Plugins und eine Liste der Plugins, die ma als Pakete installieren kann siehe die Plugin Beschreibung.

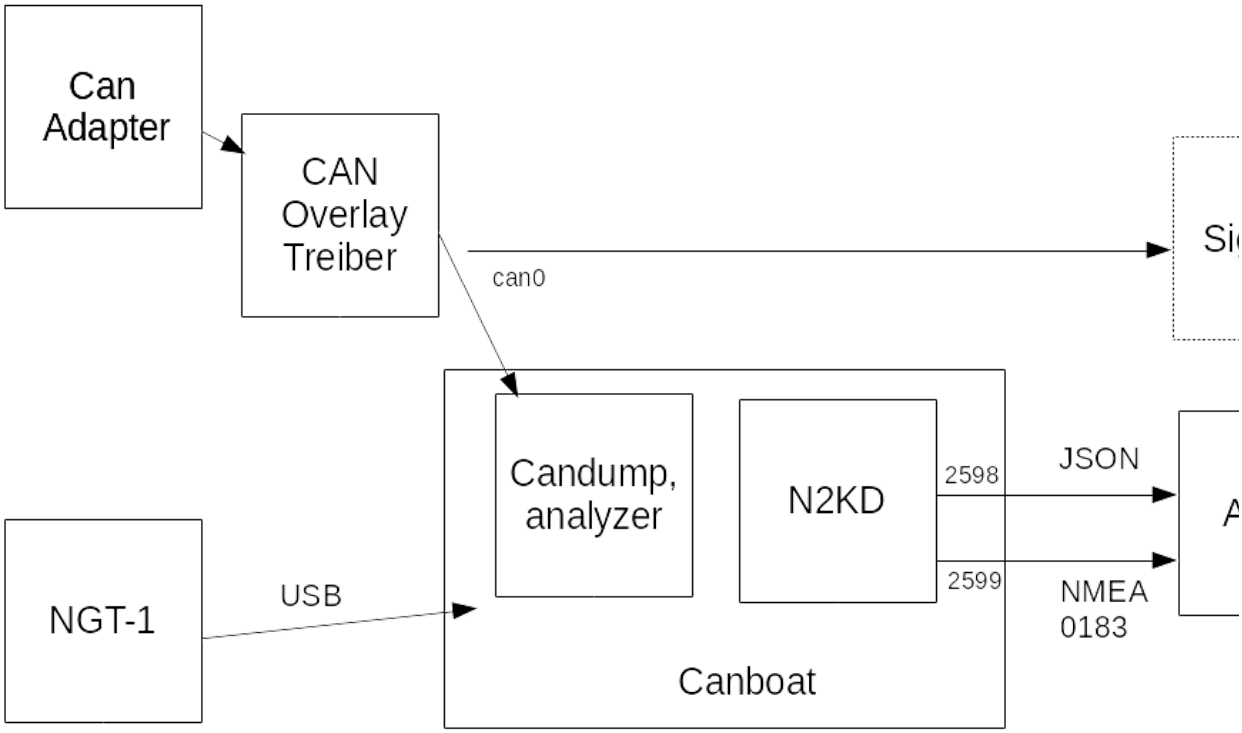
Zusammenwirken mit Canboat und SignalK

Ab Release 20200204 kann AvNav mit canboat (NMEA2000) und SignalK zusammenarbeiten.

Wichtiger Hinweis: Ab Version 20220421 hat sich das Handling für SignalK start verändert.

Canboat (NMEA2000)

Mit canboat können an den Raspberry angeschlossene CAN Adapter (z.B. mit MCP2515 oder ein Waveshare RS485 CAN-HAT) oder per USB angeschlossene Adapter (z.B. Actisene NGT-1) genutzt werden. Für die einfachen CAN-Adapter muss darauf geachtet werden, dass sie 2 Spannungsversorgungen haben (3,3V und 5V) - viele ganz einfache haben das nicht!



Im Bild ist das prinzipielle Setup zu sehen, so wie es von den headless Images bereitgestellt wird.

Für einen per SPI angeschlossenen CAN-Adapter muss meist noch ein Overlay in /boot/config.txt eingeschaltet werden. Für den MCP2515 sind entsprechende Einträge bereits vorbereitet, diese müssen auskommentiert und ggf. muss die Taktfrequenz und der für den Interrupt genutzte GPIO Pin geändert werden.

Dieser CAN-Adapter erscheint dann als Netzwerk-Interface (ggf. muss er noch entsprechend konfiguriert werden - in den Images ist das bereits vorbereitet).

Das Interface sollte mit

```
ifconfig can0
```

sichtbar sein.

Für einen per USB angeschlossenen Actisense NGT-1 siehe die Beschreibung bei canboat.

AvNav kommuniziert mit dem n2kd. Dieser konvertiert empfangene NMEA2000 Daten in NMEA0183 (nicht ganz vollständig). Die Konfiguration für n2kd erfolgt über die Datei

```
/etc/default/n2kd
```

In den Images ist hier eine Verbindung zu can0 vorbereitet, für einen per USB angeschlossenen Adapter muss diese Datei geändert werden. Falls ein solcher USB-Adapter für NMEA2000 angeschlossen wird, sollte er einen Eintrag in der avnav_server.xml bekommen, damit er dort nicht genutzt wird (bei Einstecken die Status-Seite beobachten und die USB-Id von dort kopieren), dann entsprechend eintragen:

```
<AVNUsbSerialReader .....>
<UsbDevice usbid="x:y.z" type="ignore"/>
....
```

Wenn alles korrekt konfiguriert ist, sollten auf den ports 2599 und 2598 NMEA-Daten bzw. json Daten zu sehen sein, wenn auf dem Bus NMEA2000 Datenverkehr vorhanden ist. Kontrolle z.B.

```
nc localhost 2599
```

Sonst den Zustand von canboat mit

```
sudo systemctl status canboat
```

prüfen.

Für AvNav sollten 2 Verbindungen zum n2kd konfiguriert werden. Über eine Verbindung (Port 2599) empfängt er die NMEA0183 Daten und über die andere Verbindung (Port 2598) direkt einige JSON Daten. Das ist notwendig, da n2kd keinen NMEA-Datensatz mit Datum ausgibt (z.B. RMC). Um das Datum zu erhalten, kann AvNav direkt die pgns 126992,129029 lesen, um intern Datum und Zeit zu setzen und kann daraus auch einen RMC Datensatz generieren (wenn über NMEA gültige Positionsdaten empfangen werden).

Dazu sind in der avnav_server.xml einige Konfigurationen nötig. Diese werden bei einer Image Installation automatisch aufgesetzt, sonst sind sie im Template unter /usr/lib/avnav/raspberry/avnav_server.xml zu finden und können von dort kopiert werden.

```
<AVNSocketWriter port="34568" maxDevices="5"
   filter=""read="true" minTime="50"
   name="nmea0183tosignalk"
   blackList="canboatnmea0183,canboatgen"/>
<AVNSocketReader port="2599" host="localhost" filter=""
name="canboatnmea0183"/>
<AVNPluginHandler>
  <builtin-canboat enabled="true" allowKeyOverwrite="true"
```

```
autoSendRMC="30" sourceName="canboatgen"/>
</AVNPluginHandler>
```

Mit dem ersten Eintrag wird ein zusätzlicher Port erzeugt, auf dem AvNav seine NMEA-Daten ausgibt - aber ohne die per canboat empfangenen Daten. Das wird für die Integration mit SignalK genutzt, wenn SignalK die NMEA2000 Daten bereits selbst empfängt.

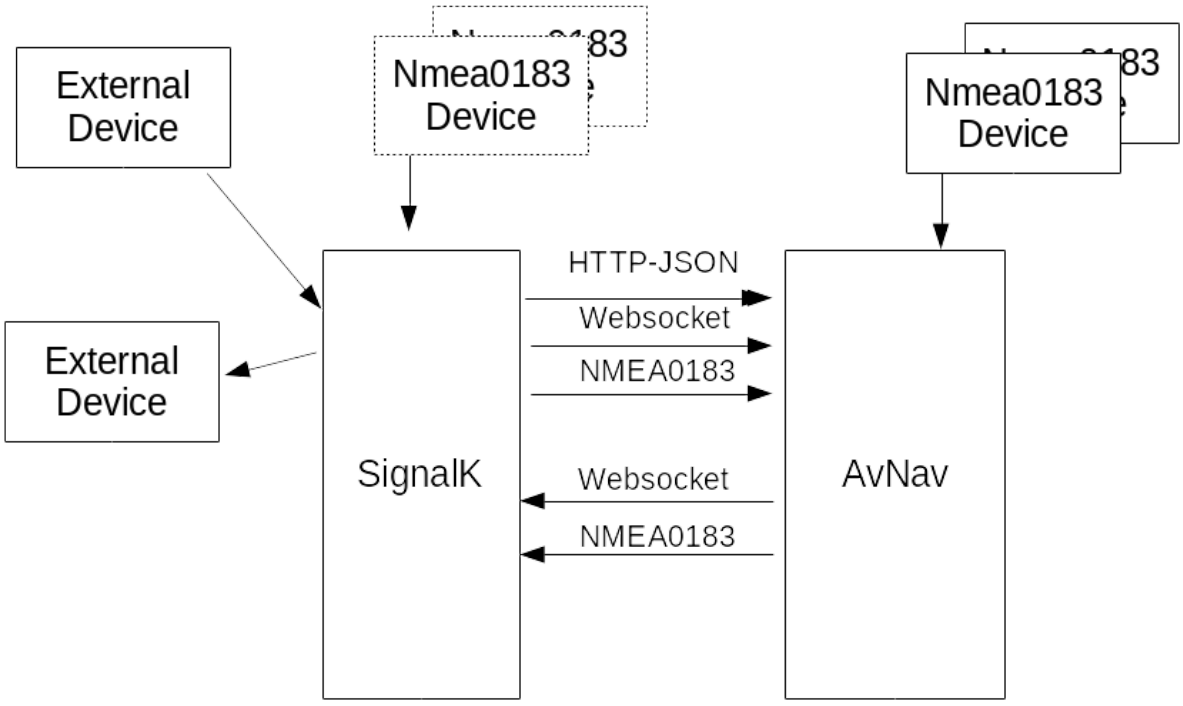
Der Socketreader localhost:2599 empfängt die konvertierten Daten vom n2kd.

Die direkte Abfrage der NMEA2000 Daten erfolgt über ein Plugin, daher muss ein Eintrag unter AVNPluginHandler gemacht werden. Mit den Settings im Beispiel wird das Plugin aktiviert, allowKeyOverwrite erlaubt das Überschreiben der internen Zeit durch das Plugin und autoSendRMC=30 sorgt dafür, das (wenn 30s kein RMC im NMEA Datenstrom aufgetaucht ist) im Intervall 1s ein RMC geschrieben wird. Für die Parameter des Plugins siehe den source code.

Ein Senden von Daten über NMEA2000 ist bisher nicht vorgesehen, dass kann ggf. über SignalK konfiguriert werden.

SignalK

Mit der Version 20220421 ist die Integration von AvNav mit SignalK stark erweitert worden.



Für die Integration zwischen AvNav und SignalK ist es zunächst wichtig, zu entscheiden wie die Daten fliessen sollen.
Dabei gibt es 2 grundsätzliche Möglichkeiten:

1. NMEA Daten landen zunächst in AvNav und werden von dort zu SignalK weiter geleitet.
   Dieses Set Up wird in den AvNav Headless Images genutzt. Die SignalK Daten können (per HTTP-Json und websocket) wieder zu AvNav geschickt werden und dann dort auch angezeigt werden.
   Die Daten die AvNav zur Navigation nutzt (inklusive der AIS Daten) werden hier direkt von AvNav aus den NMEA Daten dekodiert.
   Ausnahme: NMEA2000 Daten die über Canboat kommen müssen parallel auch zu SignalK geschickt werden.
2. NMEA Daten landen zunächst in SignalK und können von dort per HTTP-Json und websocket zu AvNav weiter geleitet werden.
   Das ist das Set Up, was z.B. in OpenPlotter verwendet wird.

Für beide Varianten kann AvNav auch eigene Daten an SignalK schicken. Im Moment sind das die Routing Daten zum nächsten Wegepunkt (entweder als RMB/APB NMEA0183 Daten oder als SignalK update - s.u.).

Ausserdem können Notifications (Alarme) von SignalK gelesen und dort hin gesendet werden.

Mit der Version 20220421 wird das Handling nicht mehr durch ein Plugin von AvNav erledigt sonder durch einen eigenen "Handler" AVNSignalKHandler. Die Konfiguration muss daher an diesem erfolgen.

1. NMEA zu AvNav und von dort zu SignalK

Die Konfiguration ist in den AvNav Headless Images vorbereitet.

Für diese Konfiguration wird in AvNav ein AVNSocketWriter vorgesehen (Standard: port 34568) der empfangene NMEA Daten weiterleitet. Über blacklist Einträge werden Daten von canboat (NMEA2000) nicht mit ausgesendet.

```
<AVNSocketWriter port="34568" maxDevices="5" filter="" read="true"
minTime="50" name="nmea0183tosignalk"
blackList="canboatnmea0183,canboatgen"/>
```

In SignalK muss dazu eine entsprechende data connection für NMEA0183, TCP client angelegt werden.

Der AVNSignalKHandler ist per default so konfiguriert, das er SignalK über localhost:3000 erreicht und alle Daten von vessels.self liest. Diese werden dann unter gps.signalk,... in AvNav abgespeichert und können so in Anzeigen verwendet werden.
Dabei wird eine Mischung aus polling per HTTP-Json und einer Websocket Vebindung genutzt. Das Polling sorgt für eine sichere Aktualisierung, die Websocket-Verbindung für ein zeitnahes Update.

Auf der obigen SocketWriter Verbindung schickt AvNav auch seine Routing Daten als RMB bzw. APB Sätze zu SignalK.
Zusätzlich kann beim SignalKHandler noch die Integration von Alarmen (SignalK: Notifications) aktiviert werden.

Daten die über andere Wege direkt in SignalK ankommen, werden zwar wie beschrieben zu AvNav geleitet, **sind dort jedoch nicht direkt für die Navigation nutzbar**.

Wenn man möchte, das Daten zunächst in SignalK ankommen und von dort zu AvNav weiter gehen, sollte man überlegen, ob der andere Signalfluss (2) nicht besser geeignet ist.

Vorteil an Signalfluss 1 (NMEA erst zu AvNav) ist , das auch wenn SignalK nicht verfügbar ist oder Probleme macht, die Navigationsfunktionen von AvNav noch arbeiten können.

2. NMEA Daten zuerst zu SignalK

In diesem Fluss, der unter OpenPlotter der Default ist (ab AvNavInstaller Version xxxx) werden NMEA Daten zunächst in SignalK empfangen und gespeichert. Dazu müssen in SignalK die

entsprechenden data connections konfiguriert werden.

In AvNav wird der AVNSignalKHandler so konfiguriert, das er die Daten von SignalK in einer Kombination von HTTP-Json und einer Websocket Verbindung abholt.
Es sind die Flags "decodeData" und "fetchAis" (siehe Konfiguration) gesetzt. Damit werden empfangene Daten intern in AvNav für die Navigation gespeichert. Zusätzlich werden sie wie bei (1) noch einmal unter gps.signalk.... gespeichert, damit die Anzeigen genauso funktionieren.
Für das Mapping der Daten siehe [Mapping].

Im AVNSignalKhandler ist ausserdem das Senden von Daten aktiviert ("sendData"). Es werden die Daten für den aktuellen Wegepunkt sowie Alarme zu SignalK gesendet.
Ausserdem werden Notifications von SignalK empfangen.
Für das Schreiben von Daten zu SignalK muss ein unter SignalK verfügbarer Nutzer mit Schreibrechten konfiguriert werden (siehe Konfiguration).

Die in früheren Versionen nötigen NMEA Verbindungen von SignalK (port 10110) zu AvNav und zurück sind mit dieser Version nicht mehr nötig. Es ist auf SignalK Seite auch kein Plugin zum Erzeugen von NMEA Daten erforderlich.

Wenn man ein update von einer älteren Version macht, kann man die NMEA Verbindungen zu SignalK einfach deaktivieren und am AVNSignalKHandler die neuen Einstellungen vornehmen.

Auswahl des Datenflusses

Für die Entscheidung ob der Datenfluss 1 (erst zu AvNav) oder der Datenfluss 2 (erst zu SignalK) genutzt werden soll, kann man zunächst von den defaults ausgehen, je nachdem auf welcher Basis man aufsetzt.

Man sollte nur dann davon abweichen, wenn es gute Gründe dafür gibt. Durch Konfiguration auf der  Status/Server Seite kann man jeden dieser Flüsse einstellen, es sind sogar Mischformen möglich.

Man muss dabei nur darauf achten, das man keine Schleifen erzeugt - also z.B. Daten von AvNav per NMEA0183 zu SignalK schickt und diese dann wieder von dort zurückholt.
AvNav versucht solche Probleme mit einer "sourcePriority" an jeder Verbindung zu vermeiden. Alle NMEA Verbindungen haben per default eine Priority von 50, der AVNSignalKHandler 40 - damit "gewinnen" direkt empfangene NMEA Daten immer gegenüber den von SignalK geholten Daten.

Konfiguration

Der AVNSignalK Handler hat die folgenden Konfigurationen.

Name	Beschreibung	Default

name	Ein Name für den Handler. Wird auf der Status-Seite angezeigt und im Log verwendet.	leer (signalk)
enabled	Wenn ausgeschaltet, wird die SignalK Integration deaktiviert. Ggf. konfigurierte NMEA Verbindungen sind davon nicht betroffen.	ein
decodeData	Wenn eingeschaltet, werden die empfangenen Daten von SignalK auch für die Navigationsfunktionen in AvNav verwendet. Siehe Mapping.	aus (ein auf OpenPlotter)
fetchAis	Wenn eingeschaltet, werden alle 10s (aisQueryPeriod) die AIS Daten von SignalK geholt und in AvNav als AIS Daten gespeichert.	aus (ein auf OpenPlotter)
priority	Die Priorität der dekodierten SignalK Daten	40
port	Der SignalK HTTP Port.	3000
host	Der SignalK server hostname/die IP Adresse.	localhost
aisQueryPeriod	Intervall (in s) für die Abfrage der AIS Daten	10
period	Periode in ms für die HTTP-Json Abfrage auf SignalK. Wenn die python websocket bibliotheken verfügbar sind (der Normalfall), wird dieses Intervall auf nahezu die expiryTime der Daten in AvNav vergrößert.	1000
fetchCharts	Hole die Informationen über die bei SignalK installierten Karten. Dazu muss dort der signalk-chart-provider installiert sein.	ein
chartQueryPeriod	Intervall (in s) für die Abfrage der Karten.	10
chartProxyMode	Wenn SignalK auf einem anderen Rechner läuft als AvNav, kann es sein, das der Browser diesen anderen Rechner nicht direkt erreichen kann. Daher besteht die Möglichkeit, das das Laden der Karten über einen Proxy in AvNav erfolgt. Das erzeugt allerdings etwas zusätzliche Last	sameHost

	und kann daher abgeschaltet werden. *sameHost*: nur Proxy wenn SignalK nicht auf dem gleichen Server läuft wie AvNav *never*: kein Proxy (kann genutzt werden, wenn auch vom Browser aus der SignalK Server unter der hier eingetragenen Adresse erreicht werden kann) *always*: Immer proxy. Kann genutzt werden, wenn der SignalK Port (3000) nicht direkt von ausserhalb erreichbar ist.	
ignoreTimestamp	Im Normalfall wertet der Handler den Timestamp der SignalK Daten aus und ignoriert Daten, die zu alt sind (expiryPeriod in AVNConfig). Eine potentielle Zeitdifferenz zwischen dem eigenen Server und dem SignalK Server wird dabei berücksichtigt. SignalK nutzt allerdings manchmal nicht seine lokale Zeit als Basis für diesen Zeitstempel sondern niimmt den Zeitstempel aus empfangegen Daten. Der kann insbesondere bei der Verwendung von Simulationsdaten weit in der Vergangenheit liegen - und AvNav würde solche Daten ignorieren. Durch Setzen dieses Flags ignoriert AvNav diese Zeitstempel und trägt als Zeitstempel seine lokale Zeit der letzten Änderung eines Wertes ein. Das ist natürlich nicht so genau, wie der originale Zeitstempel in SignalK, führt aber dazu, das auch ältere Simulationsdaten genutzt werden könen.	aus
sendData	Sende Daten an SignalK. Es kann jeweils noch konfiguriert werden, on Alarme oder/und Wegepunkt-Daten gesendet werden können. Diese Einstellungen werden aber erst sichtbar, wenn sendData aktiviert wurde. Das steuert **nicht das Senden von NMEA Daten** zu SignalK!	aus (ein auf OpenPlotter)

userName	Der Name eines SignalK Nutzers mit Schreibrechten auf dem SignalK server. Dieser Nutzer muss vorher mit den entsprechenden rechten bei SignalK angelegt worde sein. Leider hat SignalK kein Interface um direkt prüfen zu können, ob ein bestimmter Nutzer die gewünschten Pfade schreiben kann, nur für "localhost" wird geprüft, ob Schreibrechte vorliegen.	admin
password	Das Passwort für den konfigurierten Nutzer. Dieses Passwort ist im Normalfall für eine SignalK Installation auf dem gleichen Server nicht erfordelich (sofern die Konfiguration auf dem Standardpfad $HOME/.signalk/security.json liegt). Wenn in der Status-Anzeige unter "authentication" ein Fehler auftritt, kann u.U. auch lokal das Setzen des Passwortes das Problem lösen. Achtung: Das Passwort wird im Klartext in der avnav_server.xml gespeichert.	<leer>
sendWp	Sende Wegepunkt-Daten zu SignalK. Siehe auch Mapping.	ein
sendNotifications	Sende AvNav Alarme als Notifications zu SignalK - siehe Mapping.	ein
receiveNotifications	Empfange Notifications von SignalK als Alarme.	aus
notifyWhiteList	Eine Komma-separierte Liste von SignalK notifications, die emfangen werden sollen. Anzugeben sind jeweils die Pfade ohne "notification." - als z.B. navigation.arrivalCircleEntered,mob,fire,sinking. Wenn die Liste leer ist, werden alle Notifications empfangen - aber es wird noch die notificationBlacklist betrachtet,	<leer>
notifyBlackList	Eine Komma-separierte Liste von SignalK notifications, die nicht empfangen werden	server.newVersion

	sollen.	
webSocketRetry	Interval (in s) in dem versucht wird, eine erneute Websocket-Verbindung aufzubauen, wenn die vorige geschlossen wurde.	20

Einige der Parameter werden erst sichtbar, wenn der jeweils "übergeordnete" Parameter aktiviert wurde.

Mapping

Die SignalK Pfade werden wie folgt auf AvNav Datenpfade gemappt:

/vessels/self/... => gps.signalk....

Diese Pfade werden in AvNav intern nicht verwendet, können aber angezeigt werden.

Wenn "decodeData" aktiviert ist, wird wie folgt gemappt (siehe im code).

Anmerkung: Kurse/Winkel werden intern in AvNav in ° gespeichert, in SignalK in rad. Wenn mehrere SignalK Pfade angegebn sind, wird der jeweils erste bei SignalK vorhandene genutzt.

SignalK unter /vessels/self/	AvNav
navigation.headingMagnetic	gps.headingMag
navigation.headingTrue	gps.headingTrue
environment.water.temperature	gps.waterTemp
navigation.speedThroughWater	gps.waterSpeed
environment.wind.speedTrue oder environment.wind.speedOverGround	gps.trueWindSpeed
environment.wind.speedApparent	gps.windSpeed
environment.wind.directionTrue environment.wind.directionGround	gps.trueWindAngle
environment.wind.angleApparent	gps.windAngle

navigation.position.latitude	gps.lat
navigation.position.longitude	gps.lon
navigation.courseOverGroundTrue	gps.track
navigation.speedOverGround	gps.speed
environment.depth.belowTransducer	gps.depthBelowTransducer
environment.depth.belowSurface	gps.depthBelowWaterline
environment.depth.belowKeel	gps.depthBelowKeel
navigation.datetime	gps.time
navigation.gnss.satellitesInView.count	gps.satInview
navigation.gnss.satellites	gps.satUsed

AIS Daten werden wie folgt gemappt ("fetchAis" ein):

SignalK vessels/*/	AvNav ais	Bemerkung
mmsi	mmsi	nur wenn mmsi gesetzt ist, werden die Daten übernommen
name	shipname	
navigation.speedOverGround	speed	
navigation.courseOverGroundTrue	course	
communication.callsignVhf	callsign	
design.aisShipType	shiptype	
navigation.position.longitude	lon	
navigation.position.latitude	lat	

navigation.destination	destination	
sensors.ais.class	type	class A -> type 1 class B -> type 18 other -> type other

Wenn "sendWp" aktiv ist, werden die Daten wie folgt gemappt:

AvNav	SignalK vessels/self/
currentLeg.to.lon	navigation.courseGreatCircle.nextPoint.position.longitude
currentLeg.to.lat	navigation.courseGreatCircle.nextPoint.position.latitude
currentLeg.from.lon	navigation.courseGreatCircle.previousPoint.position.longitu
currentLeg.from.lon	navigation.courseGreatCircle.previousPoint.position.latitude
currentLeg.distance	navigation.courseGreatCircle.nextPoint.distance
currentleg.dstBearing	navigation.courseGreatCircle.nextPoint.bearingTrue
currentLeg.xte	navigation.courseGreatCircle.crossTrackError
currentLeg.approachDistance	navigation.courseGreatCircle.nextPoint.arrivalCircle
currentLeg.bearing	navigation.courseGreatCircle.bearingTrackTrue

Für Notifications gibt es einige wenige Mappings, ungemappte Notifications werden in AvNav mit ihrem Name und "sk:" vorangestellt gehandelt. Also z.B. notifications.sinking wird zu sk:sinking.

AvNav	SignalK vessels/self/notifications	Value
mob	mob	'state':'emergency', 'method':['visual','sound'], 'message':'man overboard'
waypoint	arrivalCircleEntered	'state': 'normal', 'method': ['visual','sound'],

		'message': 'arrival circle entered'
anchor	navigation.anchor	'state':'emergency', 'method': ['visual','sound'], 'message': 'anchor drags'

SignalK Notifications ohne ein direktes Mapping werden basierend auf ihrem state einer Kategorie zugeordnet (darüber kann im AVNCommandHandler das auszuführende Kommand und der Sound definiert werden).

emergency -> critical
normal -> normal

SignalK - Karten

Ab Version 202011xx ist auch der SignalK chart provider integriert. Karten, die von dort angeboten werden, können auf der Einstiegsseite ebenfalls ausgewählt werden. Dazu muss natürlich innerhalb von SignalK das entsprechende Plugin installiert und aktiviert sein - und Karten müssen dort verfügbar sein.

Im Normalfall werden durch AvNav nur die Informationen über die Karten bereitgestellt, der Zugriff auf die Karten erfolgt direkt vom Browser zu SignalK.

Falls das z.B. durch Firewall Einstellungen verhindert wird, kann man auch alle Karten-Zugriffe über AvNav leiten (Proxy) - siehe Konfiguration.

User Spezifischer Java Script Code

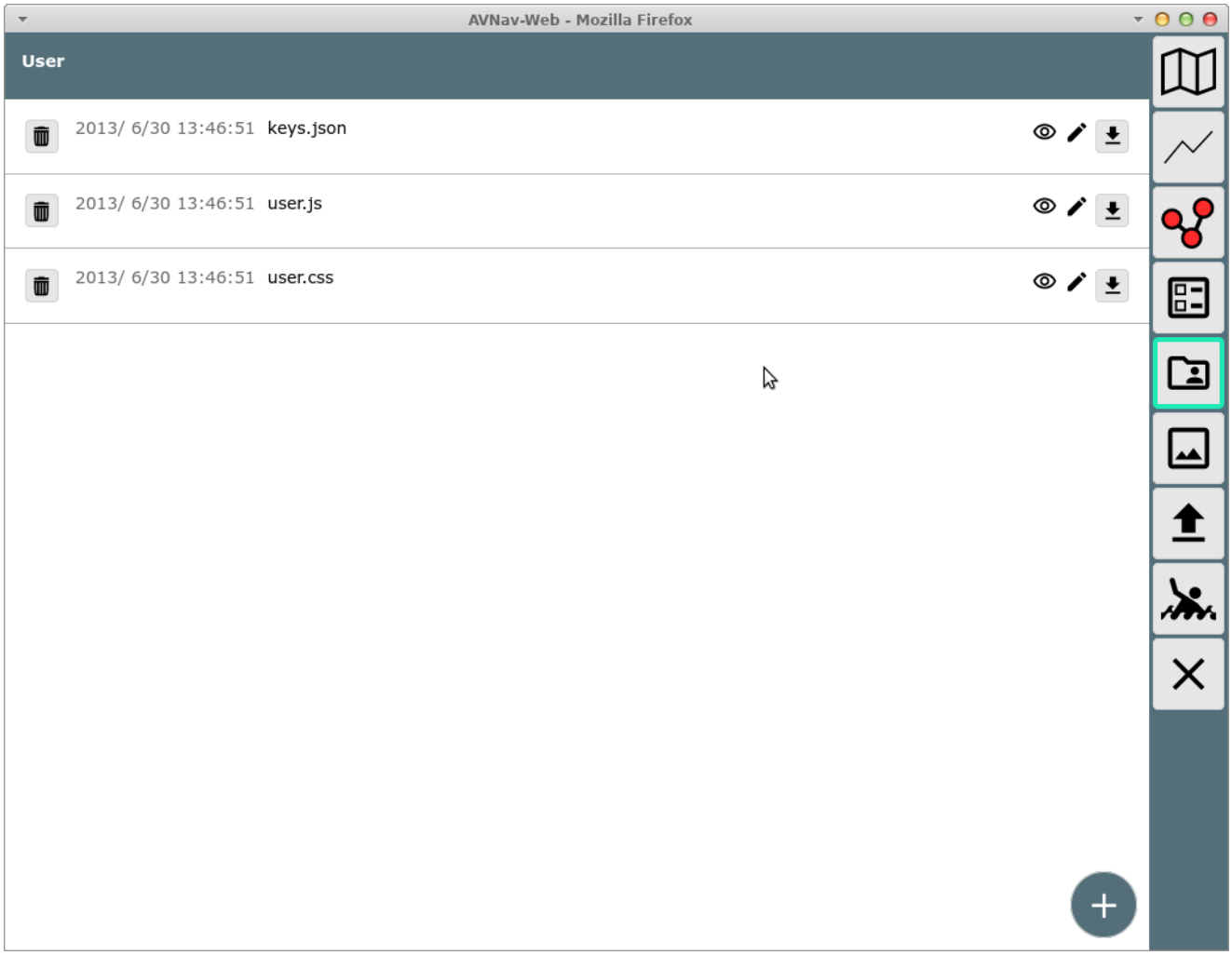
Um eine einfache Möglichkeit zu bieten, AvNav an seine Bedürfnisse anzupassen, kann man mit ein wenig Java Script Code AvNav relativ einfach erweitern.

Dabei ist es zunächst einmal möglich, neue Anzeigen zu definieren, die dann im Layout Editor ausgewählt werden können. Prinzipiell kann man dort natürlich beliebigen Java Script Code ausführen - muss dabei aber natürlich zusehen, die Funktionen von AvNav nicht zu stören.

Der Java Script Code liegt in der Datei user.js im Verzeichnis BASEDIR/user/viewer.
BASEDIR ist z.B. auf dem pi /home/pi/avnav/data.

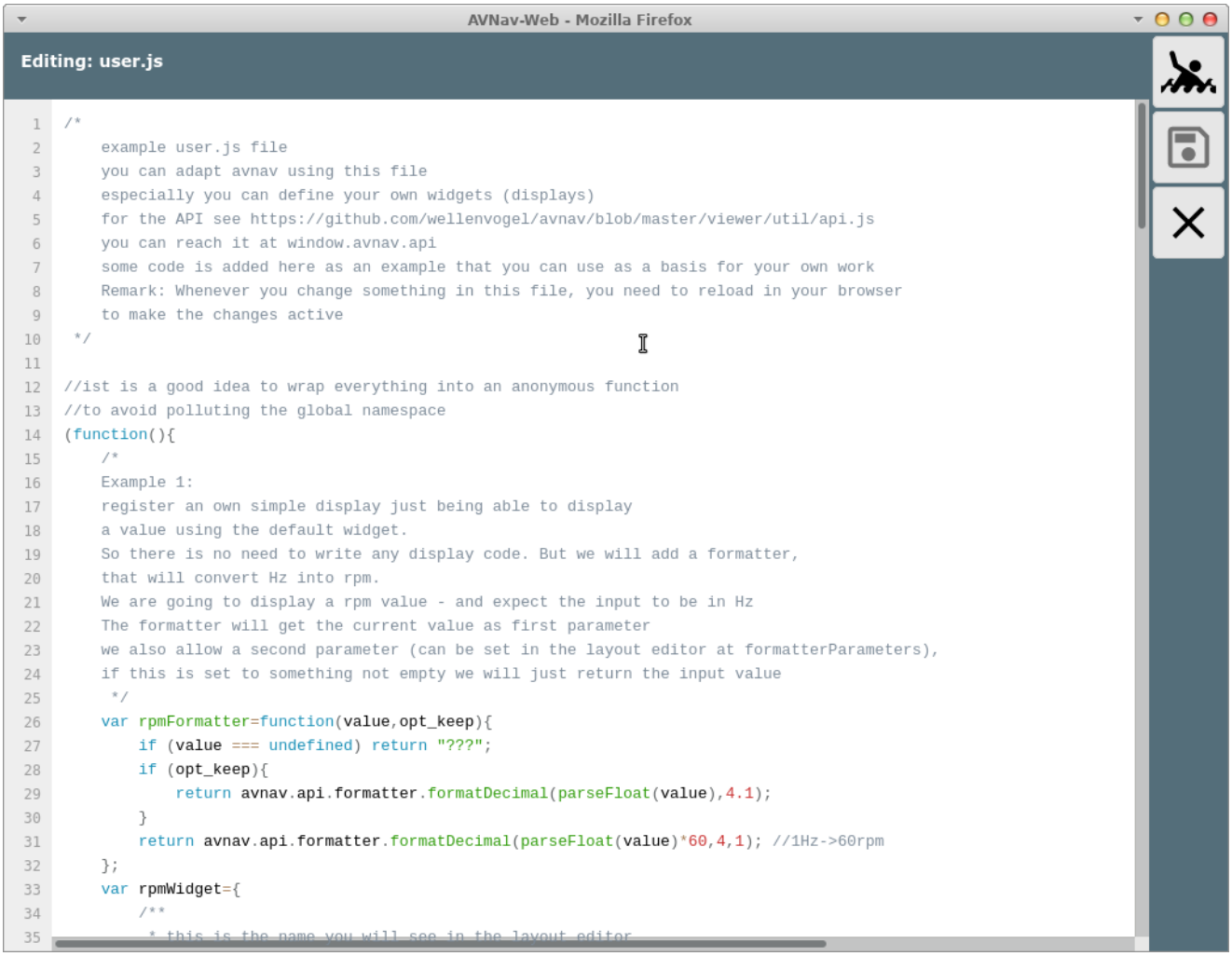
Bearbeitung

Um die Bearbeitung zu erleichtern, kann man über die Files/Download Seite und die Unterseite auf die Dateien in diesem Verzeichnis zugreifen.



Wie im Bild zu sehen, existiert dort bereits eine Datei user.js. Diese wird beim erstmaligen Start aus einem Template erzeugt.

Durch einen Klick auf die Datei und die Auswahl von "Edit" kann man die Datei direkt bearbeiten.



In der Datei sind bereits Beispiele vorhanden für die Möglichkeiten der Anpassung. Nach dem Bearbeiten die Datei speichern und AvNav neu laden.

Es empfiehlt sich, die Datei in regelmäßigen Abständen nach dem Bearbeiten herunterzuladen und noch einmal irgenwo zu speichern - es gibt keine Versionsverwaltung in AvNav.

Das aktuelle Template kann man auch auf github finden.

Anzeigen (Widgets)

Man kann im Wesentlichen die folgenden Arten von Anzeigen hinzufügen:

- Widgets mit eigenem Formatter (und ggf. festen Werten) basierend auf dem Default Widget (Beispiel 1 - user.js: rpmWidget, testPlugin: testPluing_simpleWidget)
- Anpassung/Erweiterung der Grafik Widgets mit canvas gauges (Beispiel 2 - user.js: rpmGauge)
  Hiermit ist es auch möglich, Parameter zugänglich zu machen, die in den bisher vorhandenen Widgets nicht enthalten sind
- Widgets mit eigenem HTML code (Beispiel 3 - user.js: userSpecialRpm, TestPlugin: testPlugin_courseWidget)

- Widgets mit Grafik in einem Canvas Element (Beispiel im TestPlugin: testPlugin_courseWidget)
- Widgets mit eigenem HTML, die mit dem Server Teil eines Plugins interagieren (TestPlugin: testPlugin_serverWidget)
- Widgets, die Grafiken auf der Karte darstellen (type: map) - ab 20220819 z.B. SailInstrument

Das Interface, über das mit AvNav kommuniziert wird, findet sich auf github bzw. im Template Code.
Für map Widgets kann über das Api auf die zugrunde liegenden Bibliotheken für geografische Berechnungen zugegriffen werden (Funktion LatLon und Dms).

Canvas Gauges

Für Canvas Gauge Widgets können bestimmte Parameter (siehe Canvas Gauges Beschreibung) entweder auf feste Werte gesetzt werden (dann müssen sie in der Widget Definition vorhanden sein - siehe die Werte im Beispiel ab Zeile 134) oder sie können für den Nutzer im Layout Editor einstellbar gemacht werden (dann müssen sie als editable WidgetParameter gesetzt werden - im Beispiel ab Zeile 156).

Ausserdem kann eventuell ein eigener Formatter definiert werden und als default für das Widget gesetzt werden.

Wenn man für bestimmte vordefinierte Parameter vermeiden möchte, das sie im Layout Editor für den Nutzer sichtbar werden, müssen sie in den editable Parameters mit false angegeben werden.

```
var rpmGaugeUserParameter = {
      ...
      formatter: false,
      formatterParameters: false
      };
```

Für jedes gauge widget muss der Parameter "type" angegeben werden - entweder "radialGauge" oder "linearGauge".
Ausserdem haben sie den zusätzlichen Parameter

```
drawValue (boolean)
```

Dieser Parameter steuert, ob auch eine numerische Anzeige des Wertes erfolgen soll. Der originale "valueBox" Parameter der canvas gauges wird ignoriert!

Neben den Parametern kann auch noch eine translateFunction definiert werden. Diese erhält als Parameter ein Objekt mit den aktuellen Werten aller Parameter und kann dieses

modifizieren, bevor sie an canvas gauges übergeben wird (Beispiel ab Zeile 104). Diese Funktion muss "zustandsfrei" sein, d.h. die Werte der Rückgabe dürfen nur von den übergebenen Werten (und anderen unveränderlichen Werten) abhängig sein. Andernfalls werden potentiell Änderungen nicht in der Anzeige sichtbar.

Eigene Widgets

Für ein selbst geschriebenes Widget können die folgenden Funktionen/Eigenschaften implementiert werden:

Name	Typ	Nutzbar bei Typ	Beschreibung
name	String	alle	der Name des Widgets
type	String (optional)	alle	Bestimmt, welches Widget erzeugt werden soll. Werte: radialGauge, linearGauge, map Wenn der Typ nicht gesetzt ist, wird entweder das default widget genutzt (keine Funktion renderHtml und keine Funktion renderCanvas angegeben) - oder ein nutzer definiertes Widget (userWidget)
renderHtml	Funktion (optional)	userWidget	Diese Methode muss einen String zurückgeben, der dann als HTML code in das Widget eingebaut wird. Falls eventHandler für Elemente genutzt werden sollen, müssen diese vorher registriert werden (siehe initFunction) und werden im HTML code einfach mit `<button onclick="myHandler">Click! </button>` angegeben (das ist keine exakte HTML Syntax, da nur der Name des event handlers angegeben wird, kein java script code).

			Die "this" variable innerhalb von renderHtml zeigt auf ein Objekt, das spezifisch für das Widget ist (Kontext). Wenn der EventHandler aufgerufen wird, zeigt this ebenfalls auf diesen Kontext. Das als Parameter an renderHtml übergebene Objekt enthält die unter storeKeys definierten Werte. Die Funktion wird jedesmal erneut aufgerufen, wenn sich die Werte geändert haben.
renderCanvas	Funktion (optional)	userWidget, map	Mit dieser Funktion kann in das übergebene Canvas Objekt gezeichnet werden. Das als zweiter Parameter an renderCanvas übergebene Objekt enthält die unter storeKeys definierten Werte. Die Funktion wird jedesmal erneut aufgerufen, wenn sich die Werte geändert haben. Die "this" variable innerhalb von renderCanvas zeigt auf ein Objekt, das spezifisch für das Widget ist (Kontext). Für map widgets ist dieser Canvas ein Overlay, das über die Karte gelegt wird. Am Widget Kontext stehen Funktionen zur Umrechnung von Koordinaten in Canvas Pixel bereit. Es ist wichtig den Canvas korrekt mit save/restore zu beschreiben, da sich alle map widgets den gleichen Canvas teilen.
storeKeys	Object	alle	Hier müssen die Daten angegeben werden, die aus dem zentralen Speicher gelesen und als Parameter

			den renderXXX Funktionen mitgegeben werden sollen
caption	String (optional)	alle	Eine default Beschriftung
unit	String (optional)	alle	Eine default Einheit
formatter	Funktion (optional)	defaultWidget, radialGauge, linearGauge	Ein Formatierer für den Wert. Für das defaultWidget muss diese Funktion angegeben werden.
translateFunction	Funktion (optional)	alle ausser map	Diese Funktion wird mit den aktuellen Werten als Parameter aufgerufen (so wie bei storeKeys angegeben) und muss die daraus berechneten Werte zurückgeben. Falls keine eigene renderXXX Funktion genutzt werden soll, kann hier vor dem Rendern eine Umrechnung von Werten erfolgen - siehe Beispiel.
initFunction	Funktion (optional)	userWidget, map	Falls vorhanden, wird diese Funktion einmalig aufgerufen, wenn das Widget erzeugt wird. Als Parameter (und als this) ist der Widget Context vorhanden. Dieses Objekt hat eine eventHandler Eigenschaft - hier müssen die im renderHTML genutzten eventHandler eingetragen werden. Mit der Funktion triggerRedraw am Widget Kontext kann ein erneuter Aufruf der renderXXX Funktionen erzwungen werden, Ab Version 20210422 erhält die initFunction einen 2. Parameter, der die Eigenschaften des Widgets enthält. Das sind insbesondere auch alle editierbaren Widget Parameter, die definiert wurden.

finalizeFunktion	Funktion (optional)	userWidget, map	Falls vorhanden, wird diese Funktion aufgerufen, bevor das Widget nicht mehr genutzt wird. Die "this" Variable zeigt wieder auf den Widget Kontext. Ausserdem ist der Kontext auch als erster Parameter vorhanden - wie bei der initFunction.

Der java script code erhält folgende globale Variablen:

Name	plugin.js/user.js	Beschreibung
AVNAV_BASE_URL	beide	die URL zum Verzeichnis, aus dem die Java script Datei geladen wurde. Diese kann z.B. verwendet werden, um weitere Elemente von dort zu laden. Für die user.js können Dateien aus dem images Verzeichnis über AVNAV_BASE_URL+"../images" erreicht werden. Für plugins kann über AVNAV_BASE_URL+"/api" die Kommunikation mit dem Python Anteil erreicht werden.
AVNAV_PLUGIN_NAME	plugin.js	Der Name des Plugins.

Nach der Definition muss das Widget bei AvNav bekannt gemacht werden (avnav.registerWidget).

Widget Context

User Widgets und Map Widgets bekommen einen WidgetContext. Dieser wird für jedes Widget erzeugt und den Funktionen:

- initFunction (this und erster Parameter)
- finalizeFunction (this und erster Parameter)
- renderHtml (this)
- renderCanvas (this)

übergeben.

Damit der Kontext als this Parameter genutzt werden kann, müssen die Funktionen "klassisch" mittels function definiert werden und nicht als "arrow function".

Richtig:

```
let userWidget={
        renderHtml: function(context,props){
                return "<p>Hello</p>";
        }
}
```

Im WidgetContext können Nutzerdaten gespeichert werden, die in aufeinanderfolgenden Aufrufen benötigt werden.
Ausserdem enthält er einige Funktionen, die vom Widget Code aufgerufen werden können.

Name	Widget	Parameter	Beschreibung
eventHandler	userWidget	---	eventHandler ist keine Funktion sondern ein array. Falls im renderHtml event Handler angegeben werden (z.B. <button onclick="clickHandler"/>), dann muss in der initFunction eine Funktion clickHandler hier registriert werden: this.eventHandler.clickHandler=function(ev){...} Siehe TestPlugin.
triggerRedraw	userWidget	---	Diese Funktion muss gerufen werden, wenn das Widget (z.B. nach einer Kommunikation mit dem Server) möchte, das es neu gezeichnet wird. Siehe TestPlugin.
lonLatToPixel	map	lon,lat	Konvertiert die Koordinaten in pixel Koordinaten für das Zeichnen in renderCanvas. Gibt ein array mit x,y Koordinate zurück.
pixelToLonLat	map	x,y	Berechnet aus den Canvas-Koordinaten x,y longitude und latitude. Gibt ein array mit lon,lat zurück.

getScale	map	---	Gibt den Scaling Faktor für das Display zurück. Hochauflösende Display haben einen scaling Factor > 1. Gezeichnete Objekte (besonders Text) sollten in ihren Dimensionen angepasst werden.
getRotation	map	---	Gibt die Drehung der Karte (in radian!) zurück
getContext	map	---	Gibt den renderingContext2D des Canvas zurück (nur aktiv innerhalb der renderCanvas Funktion)
getDimensions	map	---	gibt die Größe des Canvas zurück [Breite,Höhe]
triggerRender	map	---	gleiche Funktion wie triggerRedraw beim user Widget

Widget Parameter

Neben der Widget Definition können hier noch Parameter angegeben werden, die dann im Layout Editor für das Widget angezeigt werden.

Beispiele sind im user.js Template zu finden. Die Werte, die im Layout Editor für diese Parameter angegeben werden, stehen später in den renderHtml und renderCanvas Funktionen zur Verfügung (Ausnahme: Typ KEY, hier wird der aus dem Speicher gelesene Wert zur Verfügung gestellt).
Für jeden Parameter kann man die folgenden Werte angeben:

Name	Type	Beschreibung
	key	Der Name des Parameters so wie er im Layout Editor angezeigt werden soll, und wie er den renderXXX Funktionen zur Verfügung stehen soll.
type	String	STRING, NUMBER, KEY, SELECT, ARRAY, BOOLEAN, COLOR Der Typ für den Parameter. Je nach Typ wird er dem Nutzer unterschiedlich angezeigt.

		Für COLOR eine Farb-Auswahl, für SELECT eine AuswahlListe und für KEY die Liste der momentan verfügbaren Werte im Store. Für ein Array kann eine durch Komma getrennte Liste angegeben werden.
default	je nach type	Der default Wert. Für COLOR eine color css Property - also z.B. "rgba(200, 50, 50, .75)"
list	Array (nur für type SELECT)	Ein Array von Strings oder von Objekten {name:'xxx',value:'yyy'} - diese Werte werden zur Auswahl angezeigt.

Es gibt eine Reihe von vordefinierten Parametern für den Layout Editor. Bei diesen wird zur Beschreibung kein Objekt mit Eigenschaften angegeben, sonder nur true oder false (das zeigt, ob sie zum Ändern angeboten werden sollen oder nicht).

Das sind:

- caption (STRING)
- unit (STRING)
- formatter (SELECT)
- formatterParameters (ARRAY)
- value (KEY)
- className (STRING)

Ein Beispiel für eine Definition:

```
var exampleUserParameters = {
    //formatterParameters is already well known to avnav, so no need
for any definition
    //just tell avnav that the user should be able to set this
    formatterParameters: true,
    //we would like to get a value from the internal data store
    //if we name it "value" avnav already knows how to ask the user
about it
    value: true,
    //we allow the user to define a minValue and a maxValue
    minValue: {type: 'NUMBER', default: 0},
```

```
    maxValue: {type: 'NUMBER', default: 4000},
   };
```

Formatierer (Formatter)

Neben den eigentlichen Anzeigen können auch eigene Formatierer geschrieben werden, die die Werte für die Anzeige aufbereiten.
Im System sind bereits eine Reihe von Formatierern vorhanden - siehe Layout Editor.

Ab Version 20210106 können eigene Formatierer bei AvNav registriert werden und stehen dann allen Widgets zur Verfügung. Ein Formatierer ist eine Funktion, die als ersten Parameter den zu formatierenden Wert übergeben bekommt und als Ergebnis einen String zurück liefern muss.
Der String sollte dabei unabhänging vom momentanen Wert immer die gleiche Länge haben (ggf. Leerzeichen voranstellen) um die Größenanpassung auf den Dashboard-Seiten nicht zu stören.

Eine Formatierer-Funktion kann zusätzliche Parameter akzeptieren, um die Umwandlung zu steuern. Diese werden über die Widget Eigenschaft formatterParameters typischerweise im Layout Editor gesetzt.

Beispiel:

```
const formatTemperature=function(data,opt_unit){
    try{
        if (! opt_unit || opt_unit.toLowerCase().match(/^k/)){
            return avnav.api.formatter.formatDecimal(data,3,1);
        }
        if (opt_unit.toLowerCase().match(/^c/)){
            return
avnav.api.formatter.formatDecimal(parseFloat(data)-273.15,3,1)
        }
    }catch(e){
        return "-----"
    }
}
formatTemperature.parameters=[
    {name:'unit',type:'SELECT',list:
['celsius','kelvin'],default:'celsius'}
]

avnav.api.registerFormatter("mySpecialTemperature",formatTemperature);
```

Falls ein Formatierer mit dem gleichen Namen schon existiert, wirft registerFormatter eine Exception.

Jede Formatter Funktion sollte eine Property "parameters" bekommen. Diese beschreibt die im Layout-Editor sichtbaren Parameter für die Funktion. Die Werte in dieser Definition haben die gleiche Syntax wie die editierbaren Widget-Parameter.

Bibliotheken und Bilder

Falls der eigene Java Script code auf libraries oder images zugreifen soll, können diese in das gleiche Verzeichnis hochgeladen werden - Images auch in das Images Verzeichnis.

Das Einbinden von Bibliotheken kann z.B. so erfolgen

```
var fileref=document.createElement('script');
fileref.setAttribute("type","text/javascript");
fileref.setAttribute("src", AVNAV_BASE_URL+"/my_nice_lib.js");
document.getElementsByTagName("head")[0].appendChild(fileref)
```

Es empfiehlt sich, für alle Widgets css Klassen zu vergeben, damit man diese dann mit nutzerspezifischem CSS anpassen kann. IDs sollten nicht verwendet werden, da die Elemente potentiell mehrfach auf der Seite auftauchen können.

Falls Daten vom Server geladen werden sollen, empfiehlt sich die Verwendung von fetch. Alle Dateien im user Verzeichnis (oder im plugin Verzeichnis für plugin.js) sind nach dem Schema AVNAV_BASE_URL+"/"+name abrufbar.

Falls im User-Verzeichnis z.B. eine weitere Text-, Html- oder andere Datei angelegt werden soll (ohne eine hochzuladen), kann man das auch direkt mit dem "+" Button unten rechts erledigen - die Datei kann dann natürlich ebenfalls direkt bearbeitet werden.

Feature Formatierer (featureFormatter)

Ab Version 20210114 gibt es die Möglichkeit, eigene Funktionen zu registrieren, die die Anzeige von Daten aus Overlays aufbereiten.
Solche Funktionen können in der user.js oder in Plugins implementiert werden.

Mit

```
avnav.api.registerFeatureFormatter('myHtmlInfo',myHtmlInfoFunction);
```

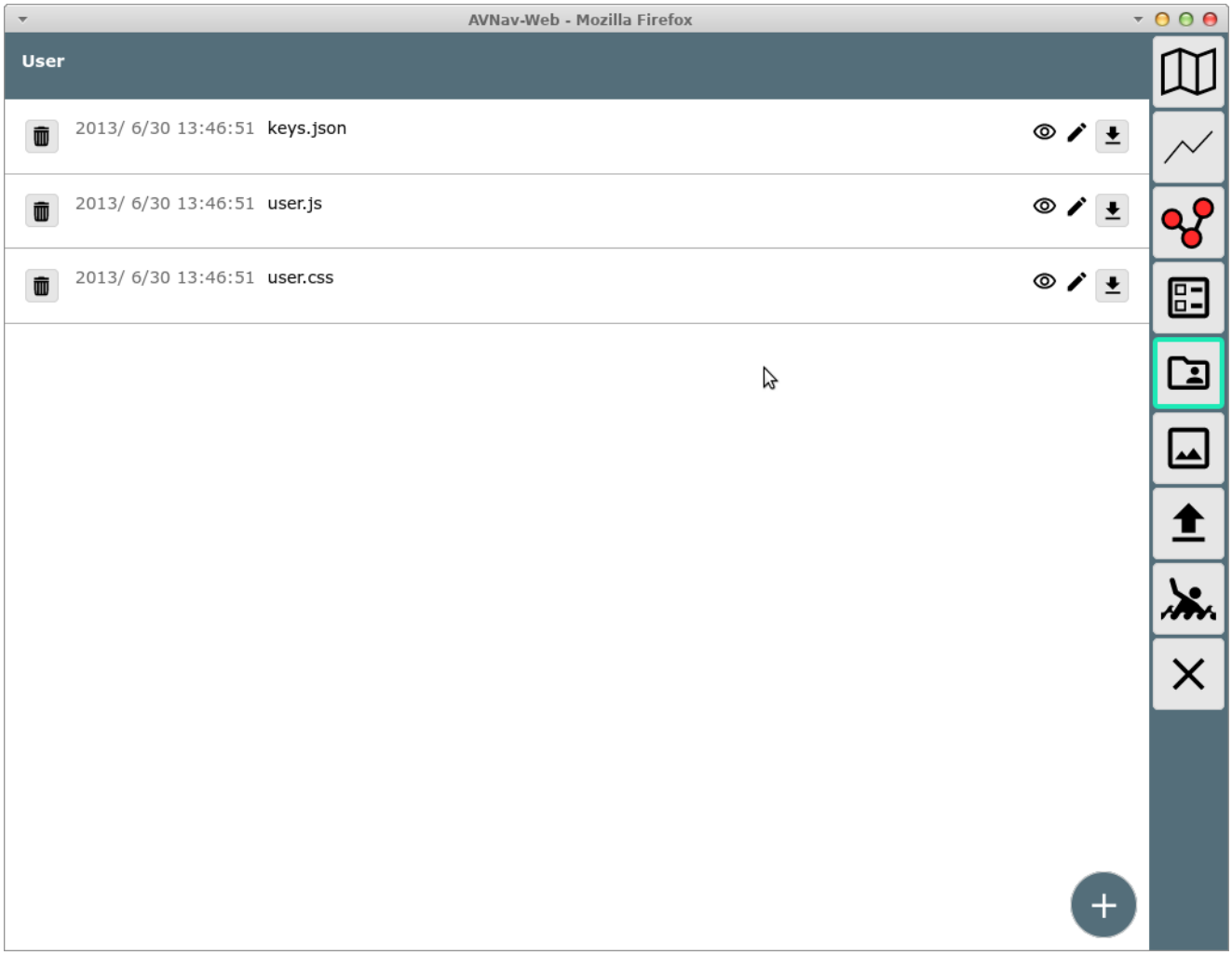
werden sie registriert. Für Details siehe Overlays.

Anpassung mit css

Da AvNav eine Web-Anwendung ist (single page app mit reactjs), kann man das Aussehen weitgehend mit CSS anpassen.

Die dafür vorgesehene Datei user.css befindet sich (wie die Datei user.js) im BASEDIR/user/viewer Verzeichnis (BASEDIR ist /home/pi/avnav/data auf dem Pi).

Über die Files/Download Seite und die Unterseite kann man die Datei direkt bearbeiten.



Durch Klick auf die Datei und Auswahl von "Edit" kann die Datei bearbeitet werden. Bei der Installation wurde ein Template eingebracht, das ein Beispiel enthält.

Ab Version 20210619 wird der Name des aktuellen Layouts als CSS Klasse an der Applikation gesetzt. Dabei werden alle Sonderzeichen durch einen "_" ersetzt. Das Layout user.default führt somit zu einer CSS Klasse user_default. Man kann damit CSS Einstellungen vornehmen, die spezifisch für ein bestimmtes Layout sind (damit kann man für unterschiedliche Geräte auch unterschiedliche CSS Einstellunegn nutzen).

```
.user_default .widget .COG {
  color: green;
}
.user_special .widget .COG {
  color: red;
}
```

Eine sinnvolle Vorgehensweise ist die Nutzung der Developer Tools (Chrome, Firefox,...), die Auswahl des Elementes, das angepasst werden soll und ein Test der Änderungen in den Tools.

Anschliessend kann man die Werte in die user.css Datei übernehmen und durch Neu-Laden der AvNav Seite testen.

Avnav Plugins

=== nicht für Android ===

Um die Funktionalität von AvNav erweitern zu können gibt es Plugins. Plugins können mit Python code den Server erweitern und sie können mit Java Script und Css die WebApp erweitern.

Jedes Plugin muss sich in einem eigenen Verzeichnis befinden. Dessen Name ist gleichzeitig der Plugin Name. Es gibt 2 Wurzelverzeichnisse, die AvNav nach Plugins durchsucht:

- "systemdir" - ein Verzeichnis für Plugins, die für alle Nutzer auf einem System installiert werden (z.B. als Pakete). Dieses ist /usr/lib/avnav/plugins.
- "userdir" - ein Verzeichnis für Plugins eines einzelnen Nutzers. Das befindet sich unterhalb des "datadir" - also auf dem pi /home/pi/avnav/data/plugins, sonst unter Linux $HOME/avnav/plugins.

Daneben gibt es noch ein internes Plugin Verzeichnis (builtin).

Grundsätzlich können die Server-Anteile an verschiedenen Stellen Daten aus AvNav lesen oder hineinschreiben. Die WebApp Anteile können im Allgemeinen dazu dienen, diese Daten dann z.B. anzuzeigen. Daneben können sie aber auch einfach weitere Anzeigen einbringen oder das Aussehen anpassen.

In einem Plugin Verzeichnis kann es bis zu 3 Dateien geben, die von AvNav beachtet werden (daneben natürlich weitere Dateien, die das Plugin selbst benötigt).

Diese sind:

- plugin.py - die Serveranteile des Plugins, optional
- plugin.js - die Java Script Anteile des Plugins, optional
- plugin.css - die CSS Anteile des Plugins, optional

Ein Beispiel für ein Plugin findet man auf GitHub.

Installation

Um ein eigenes plugin zu erzeugen kann man entweder eine zip Datei oder ein debian Paket bereitstellen.
Wenn man eine zip Datei bereitstellt, sollte diese auf der obersten Ebene genau einen Ordner mit dem Namen des plugins enthalten ("plugin" oder "avnav" sollten nicht Bestandteil dieses Namens sein).
Ein Nutzer des Plugins muss dieses dann im Unterverzeichnis "plugins" des AvNav Datenverzeichnisses auspacken (z.B. /home/pi/avnav/data/plugins auf einem Raspberry Pi).
Auf diese Weise wird das plugin zu einem "user plugin".

Wenn man ein debian Paket bereitstellt, sollte es dem Namensschema avnav-xxx-plugin folgen. Der Inhalt sollte in das Verzeichnis /usr/lib/avnav/plugins/<pluginName> entpackt werden.
Ein solches plugin wird zu einem "system plugin".
Plugin Pakete sollten den Namen des plugins (d.h. denb Verzeichnis-Namen) als Metadaten Feld mit dem Namen packages should contain the name of the plugin (i.e. the directory name) as a meta data field "avnav-plugin" enthalten.
Beispiel:

```
avnav-plugin: system-obp-plotterv3
```

Das unterstützt den AvNav Updater dabei festzustellen, ob ein Plugin-Paket als Installationskandidat angezeigt werden soll oder nicht.
Wenn man ausserdem das Metdatenfeld "avnav-hidden" auf true setzt , wird das Paket im Updater nur angezeigt, wenn das Plugin explizit enabled wurde.

Liste von Plugins

- Seatalk Remote - in Zusammenspiel mit der Fernbedienung von AK-Homberger
- History - Datenspeicherung und Anzeige
- Update - Update von AvNav (und den dazugehörigen Paketen) ohne die Kommandozeile nutzen zu müssen.
  Konfig-Editor und Log-Viewer für AvNav
- MapProxy - integriert MapProxy für Zugriff und Download verschiedener online Kartenquellen
- Obp-RC-Remote - plugin für die Nutzung der Fernbedienung von Christian
- More-NMEA-Plugin - Dekodierung und Berechnung von weiteren Kurs- und Winddaten
- rudder-angle - Anzeige des Ruderwinkels (über SignalK)
- Obp-PlotterV3 - Spezialfunktionen für den Open Boat Projects 10 Zoll Plotter (V3)

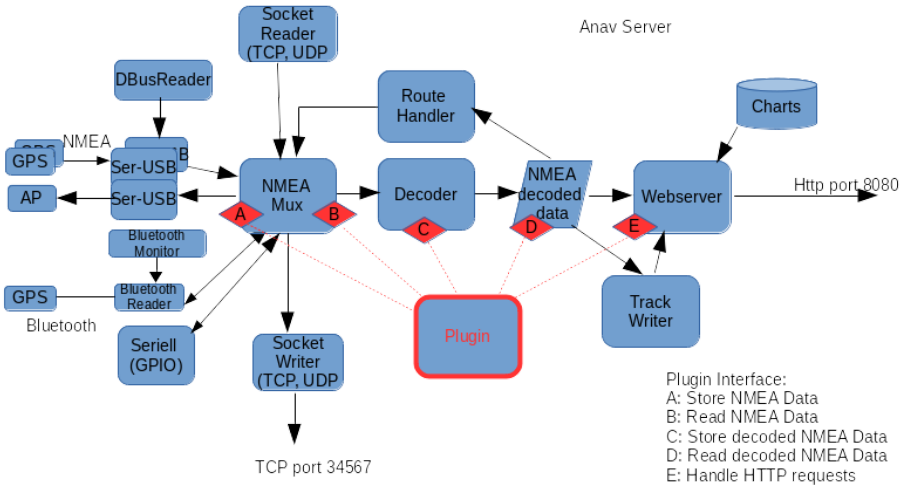
plugin.js

Im Java script code sind genau die gleichen Funktionen verfügbar wie unter nutzerspezifischer Java Script code beschrieben.

plugin.css

Im CSS code sind die gleichen Möglichkeiten vorhanden wie unter nutzerspezifisches CSS beschrieben.

plugin.py



Die Zeichnung gibt einen groben Überblick über die interne Struktur des AvNav Servers und die Punkte, an denen ein Plugin Daten auslesen oder einspeisen kann.

Punkt	Funktion	Beispiel
A	Einspeisen von NMEA Daten in die interne Liste. Diese stehen dann an allen Ausgängen zur Verfügung. Hinweis: Solche Daten stehen zunächst nicht für die WebApp zur Verfügung,	Auslesen eines Sensors und Erzeugen des passenden NMEA0183 Datensatzes.

	solange es keinen Dekoder für diesen Datensatz gibt.	
B	Auslesen von empfangenen NMEA Daten. Hier können (ggf. mit einem Filter) alle in AvNav durchlaufenden NMEA Daten gelesen werden.	In Zusammenspiel mit Punkt "C" Dekodieren von NMEA Datensätzen
C	Einspeisen von Daten in den internen Speicher von AvNav. Die Daten im internen Speicher sind in einer Baumstruktur abgelegt. Jedes Element ist durch einen Schlüssel der Form "a.b.c...." adressiert. Beispiel: "gps.lat". Alle Schlüsselwerte, die mit "gps." starten, werden automatisch an die WebApp übertragen und sind dann dort unter "nav.gps...." verfügbar. (siehe Layout Editor und nutzerspezifisches Java Script). Schlüsselwerte müssen vorher durch das Plugin angemeldet werden, es ist nicht möglich, bereits im System genutzte Schlüssel zu überschreiben. Ausnahme: Der Nutzer konfiguriert für das Plugin den Wert "allowKeyOverride" auf true.	Einspeisen eines von einem Sensor gelesenen Wertes - z.B. gps.temperature.outside oder von dekodierten NMEA Daten.
D	Auslesen von Daten aus dem internen Speicher.	Berechnung neuer Daten und Einspeisung unter "C" - oder Weiterreichen an eine externe Verbindung.
E	Bearbeiten von HTTP Requests	Die Java script Anteile können einen HTTP request senden, der im python code bearbeitet werden kann. Anworten typischerweise in Json

Ein Beispiel für eine plugin.py findet sich auf GitHub.

Damit das Plugin von AvNav erkannt wird, müssen folgende Voraussetzungen eingehalten werden:

1. In plugin.py muss mindestens eine Klasse vorhanden sein (der Name sollte Plugin sein)
2. Die Klasse muss eine statische Methode (@classmethod) mit dem Namen pluginInfo haben, die ein dictionary zurückgibt.

   ```
   * description (mandatory)
   * data: list of keys to be stored (optional)
        * path - the key - see AVNApi.addData, all pathes starting
   with "gps." will be sent to the GUI
        * description
   ```

   Ein Beispiel könnte so aussehen:

   ```
   @classmethod
   def pluginInfo(cls):
     return {
       'description': 'a test plugins',
       'data': [
           {
           'path': 'gps.test',
           'description': 'output of testdecoder',
           }
         ]
       }
   ```

3. Der Konstruktor der plugin Klasse muss einen Parameter erwarten.
   Beim Aufruf wird hier eine Instanz des API übergeben, über das die Kommunikation mit AvNav erfolgt.
4. Die Klasse muss eine run Methode (ohne Parameter) besitzen.
   Diese wird in einem eigenen Thread aufgerufen, nachdem die Initialisierung abgeschlossen ist.
   Typischerweise wird diese Methode eine Endlosschleife enthalten, um die Plugin-Funktion zu realisieren.

Für das Plugin können in der avnav_server.xml Parameter konfiguriert werden, diese sind dann über das API (getConfigValue) abrufbar.

Plugin API

Am API stehen die folgenden Funktionen zur Verfügung

Funktion	Beschreibung
log,debug,error	Logging Funktionen. Es werden Zeilen in die AvNav log Datei geschrieben. Man sollte für log und error vermeiden, solche Einträge in grosser Zahl zu schreiben, da sonst im Log potentiell wichtige Informationen verloren gehen (also z.B. nicht jede Sekunde ein Fehlereintrag...)
getConfigValue	lies einen config Wert aus der avnav_server.xml.
fetchFromQueue	Interface B: lies Daten aus der internen NMEA Liste. Ein Beispiel ist im API code vorhanden. Der filter Parameter funktioniert wie in der avnav_server.xml.
addNMEA	Interface A: schreibe einen NMEA Datensatz in die interne Liste. Man kann AvNav die Prüfsummenberechnung überlassen und man kann auch eine Dekodierung in AvNav verhindern. Der Parameter source ist ein Wert, der in blackList parametern genutzt werden kann.
addData	Interface C: schreibe einen Wert in den internen Speicher. Es können nur Werte geschrieben werden, deren Schlüssel in der Rückgabe der pluginInfo Methode vorhanden waren.
getSingleValue	Interface D: lies einen Datenwert aus dem internen Speicher. Zur Zusammenfassung mehrerer solcher Lesevorgänge existiert die Funktion getDataByPrefix
setStatus	Hier sollte der aktuelle Zustand des Plugins gesetzt werden. Das ist der Wert, der auf der Statusseite angezeigt wird.
registerUserApp	Ein Plugin kann eine User App registrieren. Dafür nötig ist eine URL und eine Icon Datei. Die Icon Datei sollte mit im Plugin Verzeichnis liegen. In der URL kann $HOST verwendet werden, das wird dann durch die korrekte IP Adresse des AvNav Servers ersetzt. Beispiel im signalk Plugin.
registerLayout	Falls das Plugin z.B. eigene Widgets mitbringt, ist es u.U. hilfreich ein vorbereitetes Layout mitzuliefern, das der Nutzer dann auswählen kann. Das Layout dazu nach der Erstellung mit dem Layout Editor herunterladen und im

	Plugin Verzeichnis speichern. Beispiel wieder im signalk Plugin.

registerSettingsFile (since 20220225)	Registrierung einer eigenen Einstellungsdatei (die vorher von der Settingsseite aus exportiert werden kann).
	Der Dateiname (zweiter Parameter) ist relativ zum Plugin-Verzeichnis. Der Name (erster Parameter) wird dem Nutzer angezeigt.
	Within this file you can use $prefix$ in the layout name if you want to refer to a layout that you register from the same plugin.
	`...`
	`"layoutName": "$prefix$.main"`
	`....`
	This will refer to a layout that you registered with the name "main".

getDataDir	Das Verzeichnis, in dem AvNav Daten ablegt
registerChartProvider	Falls das Plugin Karten bereitstellt, wird hier ein callback registriert, der eine Liste der Karten zurückgibt.
registerRequestHandler	Falls das Plugin HTTP requests bearbeiten soll (Interface E) muss hier ein callback registriert werden, der den Request behandelt. Die url für den Aufruf ist: <pluginBase>/api Dabei ist pluginBase der unter getBaseUrl zurückgegebene Wert. Die java script Anteile können die API url mit der Variable AVNAV_BASE_URL bilden: AVNAV_BASE_URL+"/api" Im einfachsten Fall kann die aufgerufene callback-Funktion ein dictionary zurückgeben, dieses wird als Json zurück gesendet.
getBaseUrl	gib die Basis URL für das Plugin zurück
registerUsbHandler (ab 20201227)	registriert einen Callback für ein USB Gerät. Mit dieser Registrierung wird AvNav mitgeteilt, dass es das USB Gerät nicht beachten soll. Der Callback wird mit dem Device-Pfad für das Gerät aufgerufen, wenn das Gerät erkannt wurde. Die USB-Id kann am einfachsten durch Beobachten der

	Status-Seite beim Einstecken des Gerätes ermittelt werden. Siehe auch AVNUsbSerialReader. Damit kann ein Plugin selbst einfach das Handling für ein spezielles Gerät übernehmen, Ein Beispiel findet sich auf GitHub.
getAvNavVersion (ab 20210115)	Aktuelle AvNav Version (int)
saveConfigValues (ab 20210322)	Speichere config Werte für das Plugin in avnav_server.xml. Der Parameter muss ein dictionary mit den Werten sein. Das Plugin muss sicherstellen, dass es später mit diesen Werten wieder starten kann.
registerEditableParameters (ab 20210322)	Registriert eine Liste mit config Werten, die zur Laufzeit geändert werden können. Der erste Parameter ist eine Liste von dictionaries mit den Parameter Beschreibungen, der zweite ein callback, der bei Änderungen mit den geänderten Werten aufgerufen wird (wird typischerweise saveConfigValues rufen). Die Syntax für die Parameter-Liste ist im Source Code beschrieben.
registerRestart (ab 20210322)	Registriere einen Stop Callback. Damit kann das Plugin disabled (deaktiviert) werden.
unregisterUserApp (ab 20210322)	Deregistriere eine User App.
deregisterUsbHandler (ab 20210322)	Deregistriere eine usb device id (siehe registerUsbHandler)
shouldStopMainThread (ab 20210322)	Kann in der Hauptschleife genutzt werden, um zu prüfen, ob das Plugin gestoppt werden soll. In jedem anderen Thread wird immer True zurück gegeben.
sendRemoteCommand (ab 20230426)	Sende ein Fernsteuerungskommando, siehe den Source Code für Details.
registerSettingsFile (ab 20230426)	Mache eine Datei mit gespeicherten Einstellungen bekannt. Diese kann vom Nutzer dann geladen werden.

registerCommand (ab 20230426)	Registriere ein Kommando, das von AvNav ausgeführt werden kann. Dieses kann z.B. dafür genutzt werden ein bereits vorhandenes Kommando zu ersetzen. Auch neue Kommandos sind möglich. Siehe den Source Code oder die AVNCommandHandler Konfiguration für Details.

Aktivieren und Verbergen von System Plugins

(ab 20230426)

Um plugins "unsichtbar" zu machen, die mit debian Paketen installiert wurden, gibt es ein Script /usr/lib/avnav/plugin.sh.
Als root kann man dieses Script aufrufen um die Sichtbarkeit von system plugins zu steuern und default Parameter zu setzen.
Ein Aufruf ohne Parameter bringt eine Hilfe mit den Aufruf-Optionen.

Spezialfunktionen für den Raspberry Pi

(ab 20230426)

Plugins die als debian Pakete für den Raspberry Pi erzeugt werden (system plugins) können ein Shell Script "plugin-startup.sh" bereitstellen.
Dieses Script ermöglich es plugins, Systemparameter zu konfigurieren.
Es wird immer während des Bootprozesses des Systems aufgerufen.
Ob und mit welchen Parametern ein solches Script aufgerufen wird, hängt von einem Parameter in der Datei /boot/avnav.conf ab (siehe Image Vorbereitung). Der Parametername ist:
AVNAV_<PLUGIN>
Dabei ist <PLUGIN> der Pluginname (d.h. der Name seines Verzeichnisses) übersetzt in Großbuchstaben (und ohne alle Zeichen ausser 0-9 und a-z).
Wenn dieser Parameter auf "yes" gesetzt ist, wird das Pluginscript gerufen.

Es gibt 3 Aufruf-Varianten:

plugin-startup.sh enable

Dieser Aufruf findet beim ersten Boot mit dem Parameter in der avnav.conf auf "yes" statt.
Das Plugin sollte jetzt alle notwendigen Änderungen am System vornehmen (wenn möglich so, das sie auch später wieder rückgängig gemacht werden können).
Typischerwe betrifft das Anpassungen /boot/config.txt oder anderen Konfigurationsdateien.
Das Script sollte 1 zurückgeben, wenn ein Reboot nötig ist, sonst 0 oder < 0 bei Fehlern.

Es gibt einige Helper Funktionen die im Script genutzt werden können. Diese Helper Funktionen bindet man mit

```
. "$AVNAV_SETUP_HELPER"
```

in das Script ein.
Die Umgebungsvariable AVNAV_SETUP_HELPER ist gesetzt, wenn das Script aufgerufen wird.
Ein Beispiel findet man im obp-plotterv3-plugin.

plugin-startup.sh disable

Dieser Aufruf wird ausgeführt, wenn der Parameter in der avnav.conf von yes auf einen anderen Wert geändert oder entfernt wird. Das Script sollte die am System gemachten Änderungen - soweit möglich - wieder zurücknehmen.
Anmerkung: Da das Handling im Normalfall nur dazu vorgesehen ist, einmalig bei der ersten Nutzung eines Images stattzufinden, ist es kein grosses Problem, wenn Änderungen nicht zurückgenommen werden.

plugin.startup.sh [keine Parameter]

Dieser Aufruf erfolgt bei jedem boot. In diesem Falls sollten keine Einstellungen im System geändert werden - es könnte sonst sehr überraschend für den Nutzer sein, wenn bei einem beliebigen Startvorgang plötzlich Systemeinstellungen geändert werden. Es können aber beispielsweise notwendige Initialisierungen von Hardware vorgenommen werden.

Mobile Atlas Creator Mapsources

2022/04/17
Anpassungen an mobac 2.2.2

2021/08/22
Anpassungen an mobac 2.2x, Anpassung an neue BSH Layer

2020/01/26
Wieder mal ein paar kleine Modifikationen an den Map-Sources, damit der Zugriff auf die BSH Server wieder funktioniert.

Sourcen

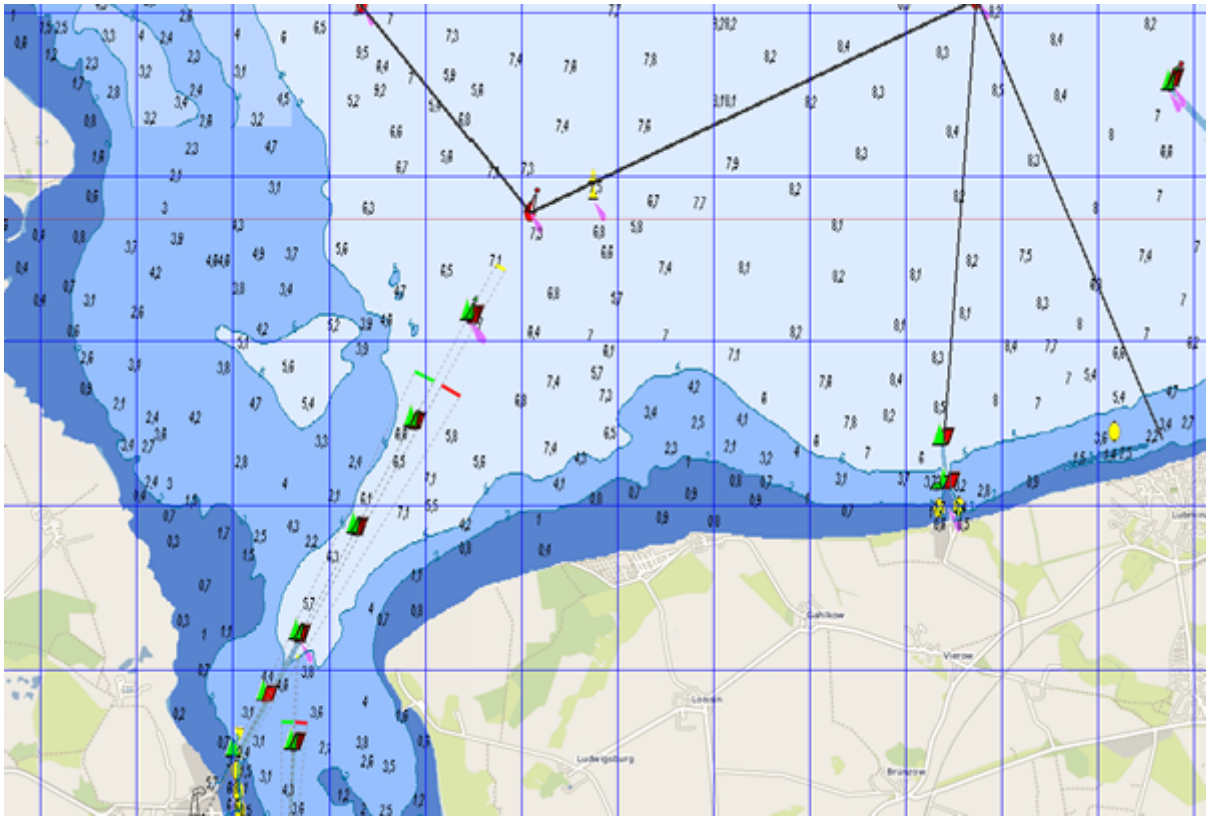
Für den Mobile Atlas Creator habe ich einige Map-Sources erzeugt, die es erlauben, etwas flexibler per xml den Zugriff auf Kartendienste zu definieren. Dazu die Datei avnav-mapsources.zip im Verzeichnis "mapsources" des Mobile Atlas Creator entpacken.
Für Mobac Version 2.2.1 bitte die Datei avnav-mapsources-before222.zip nutzen.
Für Mobac Versionen < 2.2.1 bitte die Datei avnav-mapsources-before22.zip nutzen.
Dann erhält man u.a. ein "mashUp" aus den BSH-Kartendiensten (siehe auch bsh-viewer) und OpenSeaMap ("BSH OpenSeaMap 2021 Extended"). Außerdem noch BSH alleine ("BSH 2021 Extended") oder OpenSeaMap + OpenStreetMap ("OWS OpenSeaMap 2021"). Wenn jemand "spielen" möchte, kann man die .exml entsprechend anpassen. Spannend sind insbesondere die Layer für die BSH-Abfrage. Die kann man mit meinem bsh-viewer ausprobieren (jeweils rechts in der Quelle editieren). Außerdem kann man bei Bedarf die Farben noch etwas anpassen - ich habe mich bemüht, etwas mehr Kontrast zu erzeugen. Wenn man etwas ändern will - eine der Karten z.B. mit paint.net öffnen und dann die Hex-Werte für die Farben aussuchen und in der .exml Datei eintragen.

Der Download dauert meist ziemlich lange - oft ist der BSH-Server sehr langsam oder stürzt auch gerne mal ab. Dann einfach wieder neu versuchen (dazu im Mobac die cache Einstellungen so setzen, dass er die Karten z.B. 1 Monat im Cache lässt) - irgendwann sind sie alle heruntergeladen. Als Format jetzt immer "OsmdroidGEMF" wählen (das kann man übrigens auch in anderen Programmen nutzen...).

Wenn man in den exml Dateien ändert (insbesondere bei den Layern) muss man allerdings unter Tilestore die enstprechenden Caches löschen - sonst werden die Änderungen nicht wirksam.

Source files auf github.

Das Ergebnis sieht z.B. so aus (das ist die Einfahrt nach Greifswald):



Hier nochmal die Dateien:

- avnav-mapsources.zip (die Mapsources BSH, BSH+OpenSeaMap)