OpenPlotter Documentation

Release 2

Sailoog

Jan 15, 2023

Description


OPENPLOTTER

CHAPTER 1

What is OpenPlotter?

There are people who buy boats but there are also people who build them, why not build your own electronics too? OpenPlotter is a combination of software and hardware to be used as navigational aid on small and medium boats. It is also a complete on-board home automation system. It is open-source, low-cost, low-consumption and it works on ARM computers like the Raspberry Pi or any computer running a Linux Debian derivative. Its design is modular, so you just have to implement what your boat needs. Do it yourself.

CHAPTER 2

Features

Chart plotter Chart a course and track your position using OpenCPN, a concise and robust Chart Plotter Navigation software designed to be used at the helm station of your boat while underway.

Dashboards Build instrument panels to visualize data.

Weather Download and visualization of GRIB files using XyGrib.

NMEA 0183 Connect to your NMEA 0183 devices to receive and send data.

NMEA 2000 Connect to your NMEA 2000 network to receive and send data.

Signal K The free and open source universal marine data exchange format.

Access point Share NMEA and Signal K data with laptops, tablets and phones.

Headless Access to OpenPlotter desktop from the cockpit through your mobile devices.

Compass Heading and heel from an IMU sensor. Tilt compensated. Self-calibration.

Autopilot Build a cheap, accurate and complete autopilot with pypilot.

Sensors Connect multiple sensors for temperature (air, sea, motor, exhaust, fridge...), pressure, humidity, light, gas, smoke, batteries charge, tanks level, wind, opening doors, motion, switches...

IoT Receive or send data to your boat while you are away through Telegram, Mastodon, e-mail, MQTT...

SDR Receive voice or decode AIS using cheap Software Defined Radio receptors.

Hardware Dedicated hardware specially designed for OpenPlotter like the Moitessier HAT.

CHAPTER 3

Examples

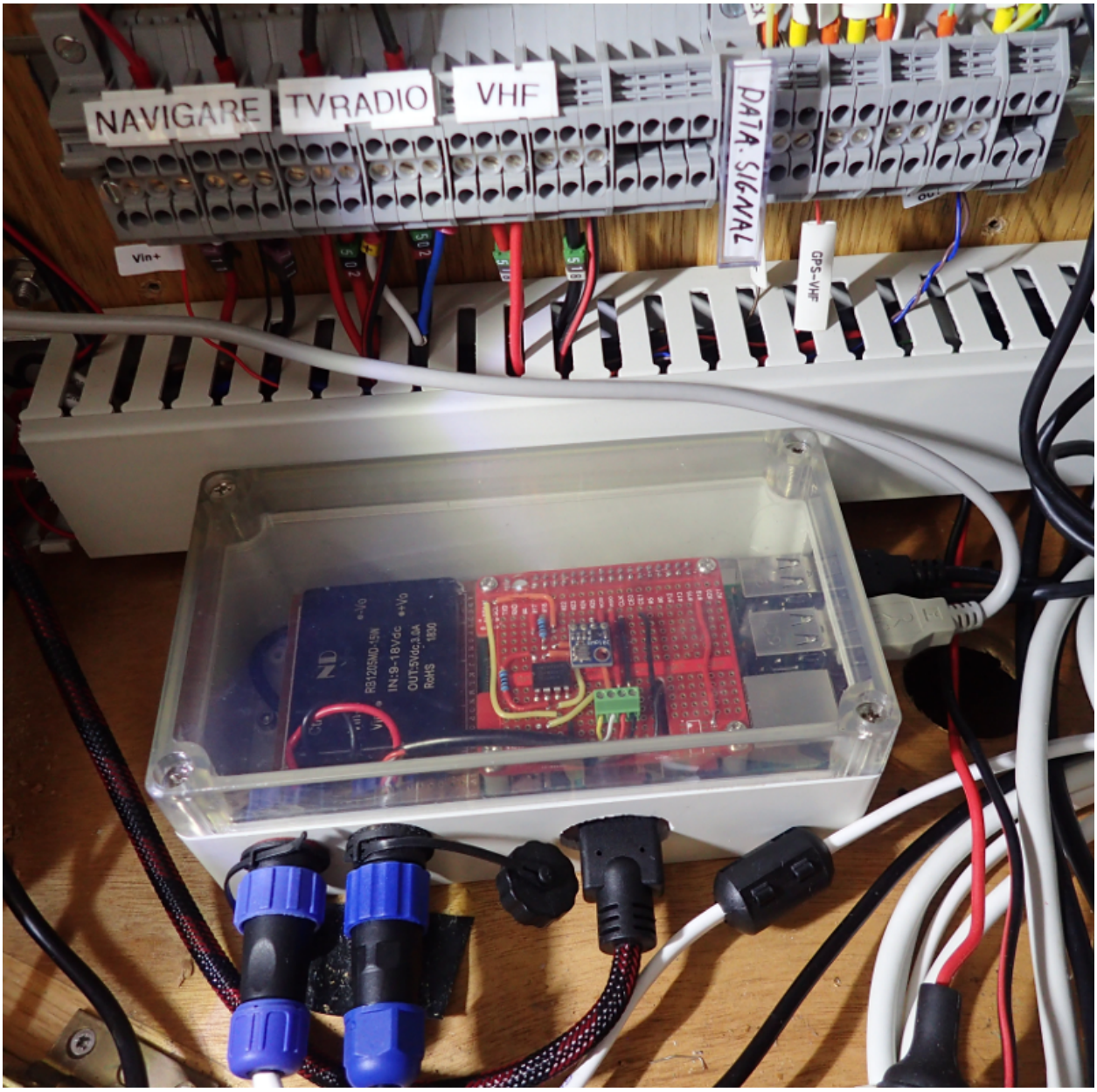
Please send us your projects involving OpenPlotter and we will add them to this Hall of Fame.

3.1 Sigma 33 Build

Description I wanted to build a system that took data from as many instruments on the boat as possible. I started by researching off the shelf systems and I found myself both underwhelmed by their features and appalled by their cost. I figured I must be able to DIY something of at least equal performance for a fraction of the cost. My search for a DIY solution lead me to OpenCPN, the opensource chart plotter software, I was immediately drawn to it's versatility and how it mimicked the user interfaces I was used to on ship ECDIS systems. It didn't take much longer to find and settle on OpenPlotter, a complete linux build incorporating OpenCPN and all the software required to ingest, process, and distribute NMEA data around the boat…

More info https://www.reis-day.com/sailing/openplotter-build

3.2 Yacht server on board



Description The yacht server data system is a Raspberry Pi based system, with main software OpenPlotter and OpenCPN. The design is based on an Internet of Things on Board (IoToB) approach with remote wireless sensors. Most of the server functions are done running OpenPlotter (which contain a SignalK server) and OpenCPN. The SignalK server accept SignalK messages (temperatures, levels etc) from the IoToB nodes around the yacht...

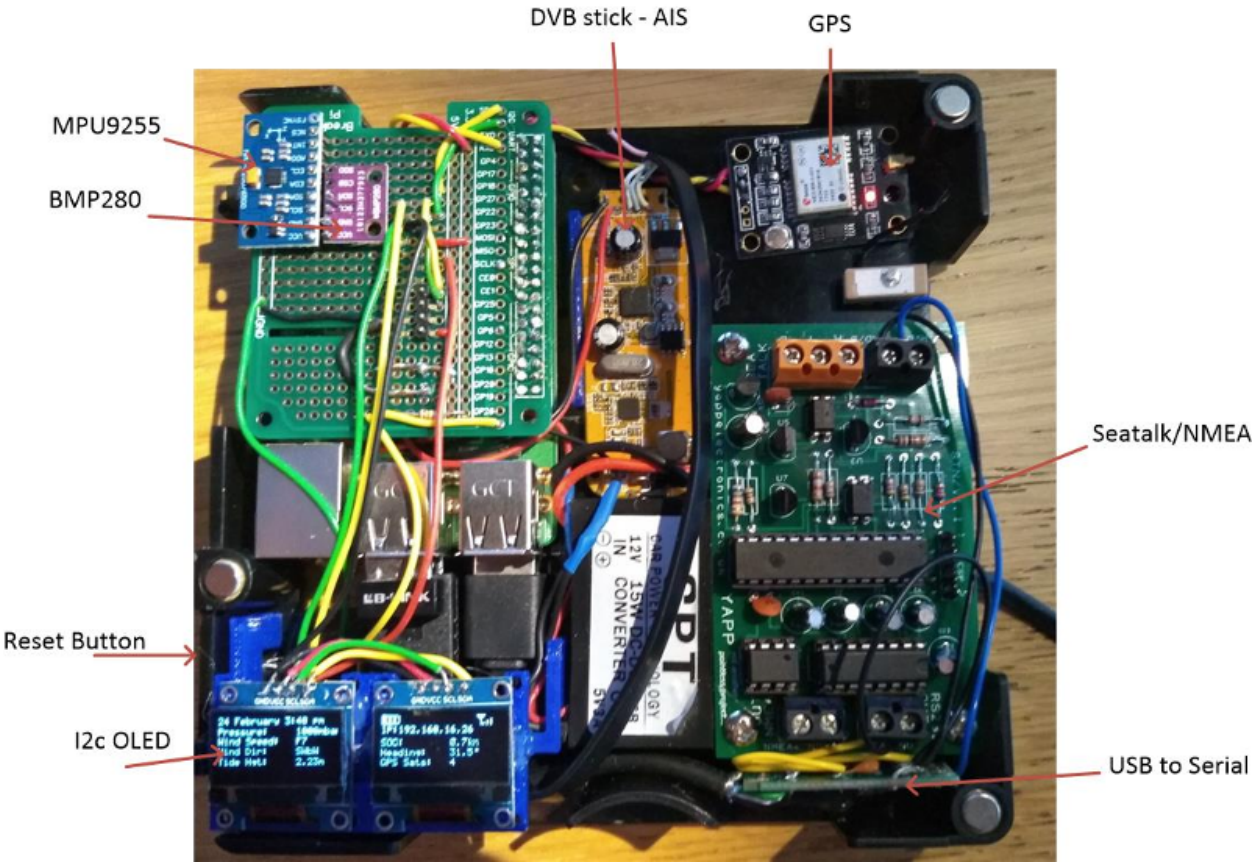
More info https://sites.google.com/site/olewsaa/yacht-server/server-on-board

3.3 Bareboat Necessities

Description DYI project of building a marine computer, a boat LTE/WiFi gateway, and a cockpit chart-plotter display from easily obtainable and affordable components...

More info https://bareboat-necessities.github.io/

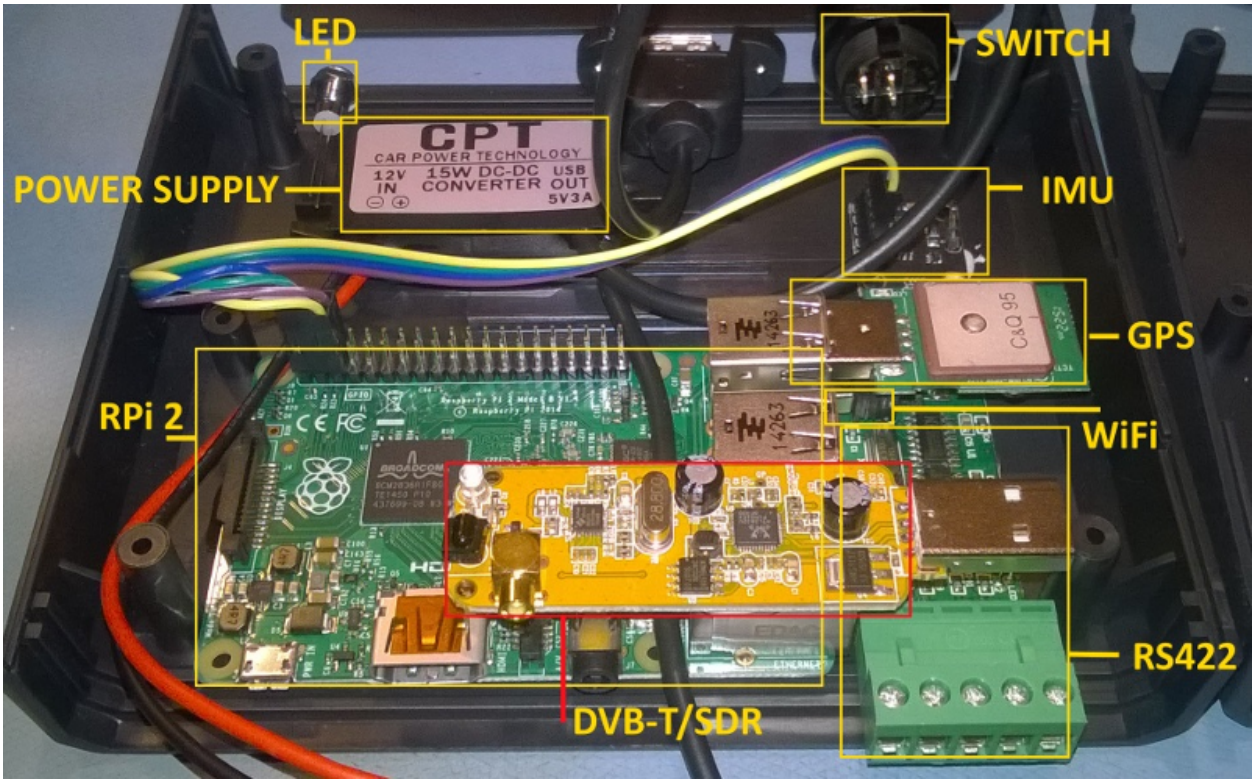
3.4 UK OpenPlotter Build



Description I have finally finished my first build of the my Openplotter computer, I say first build as I would like to build a custom PCB for the interfaces to the Pi, but I wanted to get everything together in one box and tested before I finalise my PCB design. . .

More info http://forum.openmarine.net/showthread.php?tid=2371

3.5 The Boat PC



Description In late 2015 I was doing my usual head-scratching about what gifts to get various family members for the holiday season. My wife mentioned making something electronic for my father-in-laws boat, and after a few hours of collecting thoughts came up with an idea. . .

More info http://labs.domipheus.com/blog/the-boat-pc-a-marine-based-raspberry-pi-project/

3.6 *Uredd II* installation

Description *Uredd* is the boats name, it is Norwegian and translates to *Brave*...

More info http://forum.openmarine.net/showthread.php?tid=99

CHAPTER 4

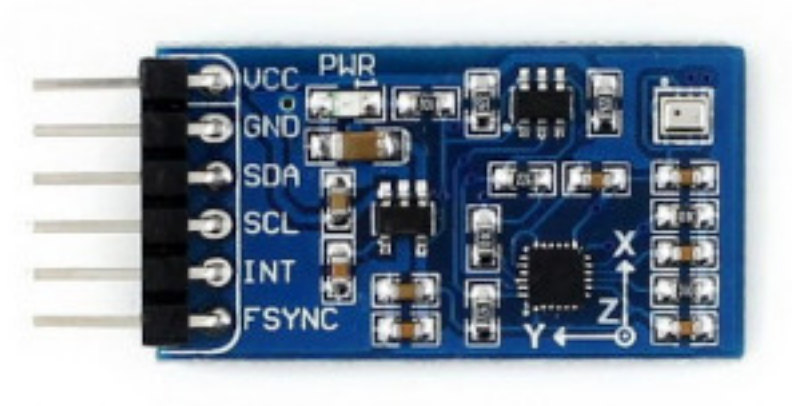
What do you need?

4.1 Basics

To be added (basic Raspberry stuff)

4.2 Extra hardware

4.2.1 IMU sensors



An inertial measurement unit (IMU) is an electronic device that measures and reports a body's specific force, angular rate, and the orientation of the body, using a combination of accelerometers, gyroscopes, and magnetometers.

IMUs in boats are typically used to get Magnetic Heading, Heel and Trim.

Wiring

Configuring *Pypilot compass calibration*.

Recommended

- InvenSense MPU-9250 single chip.
- InvenSense MPU-9255 single chip.

  Providers Pypilot - Waveshare.

- ICM-20948 single chip (coming soon).
- ISM330DHCX + LIS3MDL (coming soon).

Also supported

- InvenSense MPU-9150 single chip.
- InvenSense MPU-6050 plus HMC5883 magnetometer on MPU-6050's aux bus (handled by the MPU-9150 driver).
- InvenSense MPU-6050 gyros + acclerometers. Treated as MPU-9150 without magnetometers.
- STM LSM9DS0 single chip.
- STM LSM9DS1 single chip.
- L3GD20H + LSM303D (optionally with the LPS25H) as used on the Pololu AltIMU-10 v4.
- STM LSM6DS33 + LIS3MDL (optionally with the LPS25H) as used on the Pololu MinIMU-9 v5 and AltIMU-10 v5.
- L3GD20 + LSM303DLHC as used on the Adafruit 9-dof (older version with GD20 gyro).
- L3GD20H + LSM303DLHC (optionally with BMP180) as used on the new Adafruit 10-dof.
- Bosch BMX055 (although magnetometer support is experimental currently).
- Bosch BNO055 IMU with onchip fusion. Note: will not work reliably with Raspberry Pi due to clock-stretching issues.

4.2.2 SDR receivers

Software-defined radio (SDR) is a radio communication system where components that have been traditionally implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded system.

SDR receivers in boats are typically used to get AIS or weather forecasts.

Configuring *SDR-VHF*.

Providers OpenMarine - rtl-sdr.com.

4.2.3 RS-422 converters

NMEA 0183 communication protocol was designed to run over the RS-422 serial interface, which can support a single talker and up to 10 listeners and data rates as high as 10 mbit/sec.

RS-422 converters in boats are typically used to get or send data to your instruments. You can find USB converters or some Raspberry Pi HATs to connect to the GPIO header.

USB

Wiring *Connecting a USB-RS422 converter - Wiring*.

Configuring *Connecting a USB-RS422 converter - Input data*.

Providers DTech.

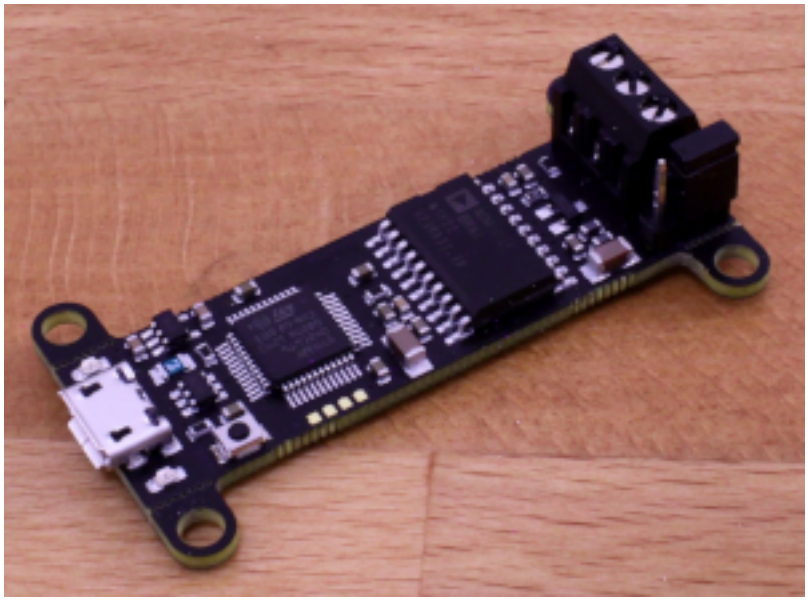
GPIO

Wiring

Configuring

Providers

4.2.4 CAN Bus converters



NMEA 2000, abbreviated to NMEA2k or N2K, is compatible with the Controller Area Network ("CAN Bus") used on road vehicles and fuel engines. Communication runs at 250 kilobits-per-second and allows any sensor to talk to any display unit or other device compatible with NMEA 2000 protocols.

You can find USB converters or some Raspberry Pi HATs to connect to the GPIO header.

USB

Wiring

Configuring

Providers

GPIO

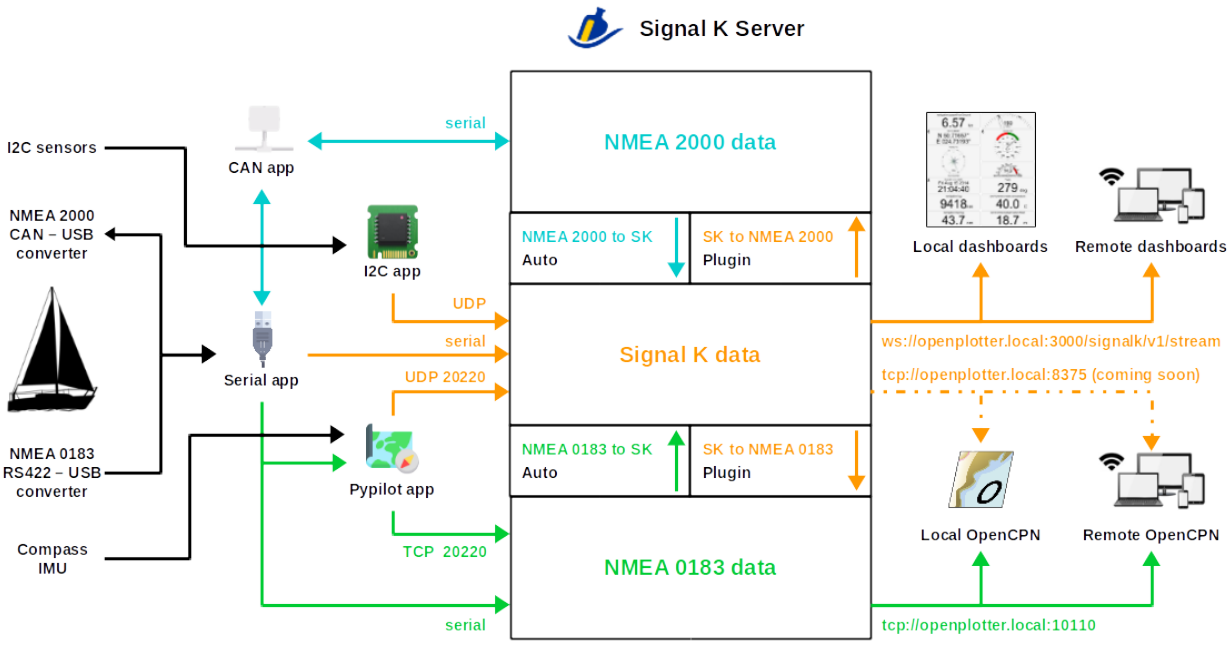
Wiring

Configuring

Providers

CHAPTER 5

How does it work?



OpenPlotter v2 Data Routing

CHAPTER 6

How to collaborate

> Everything takes **time**, **money** and **monkeys**. You need a lot from any two groups and a little from the third. An increase in any one reduces the requirement for the other two. Change occurs when one of those three change.
>
> —Moe's Law (Navigatrix project)

Time

Download and install OpenPlotter and test and test and test . . .

Report bugs and request new features in OpenMarine forums.

Spread the word among your friends in ports and forums.

Money

This project is financed by selling related products or by voluntary contributions.

Monkeys

> Men wanted for hazardous journey. Low wages, bitter cold, long hours of complete darkness. Safe return doubtful. Honour and recognition in event of success.
>
> —Ernest Shackleton

If you have python skills, push your commits to the github repository.

If you have electronics skills, share your work on OpenMarine forums.

6.1 Translations

If you want to help translate the software into your language, create an account on the Crowdin platform and edit the project.

6.2 Documentation

If you want to help us maintain this documentation, let us know what your intentions are in the forum to coordinate.

The best way to do this is to fork this repository GitHub and push your commits.

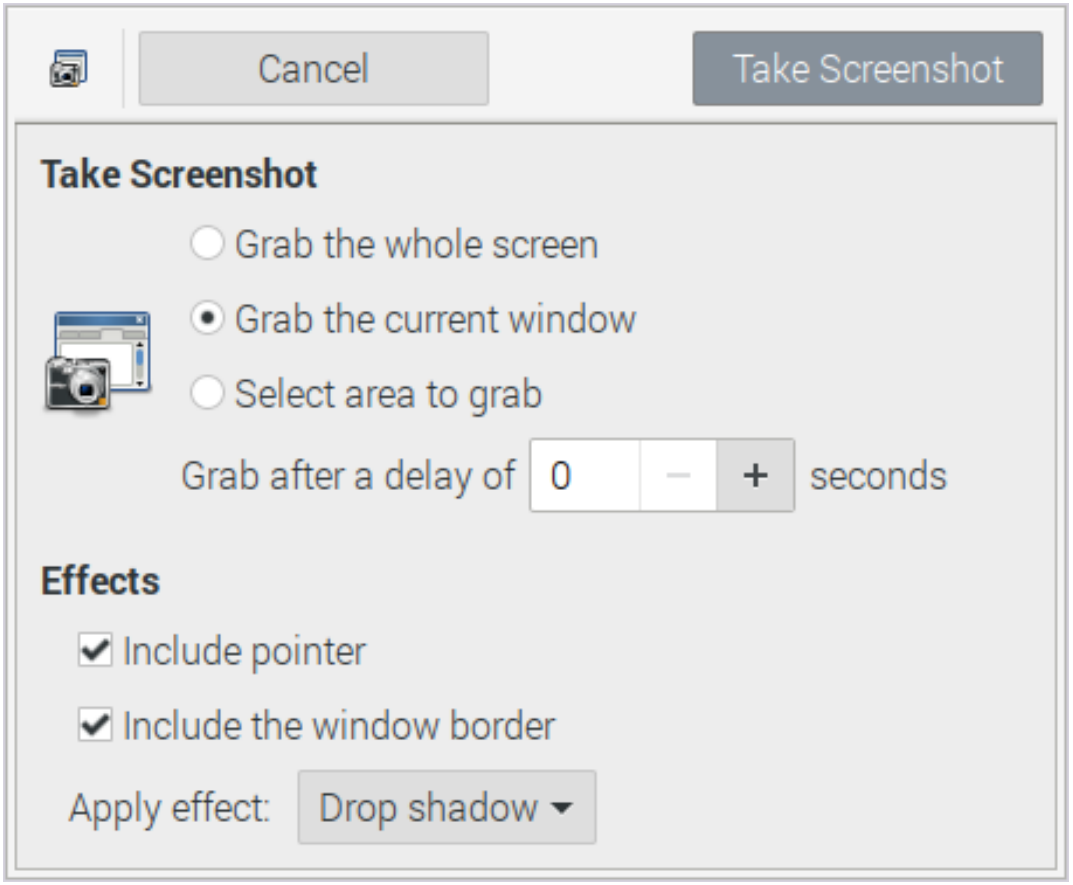
If you are not familiar with GitHub, do not worry, send us your contributions to the forum.

Guidelines

- We will no longer maintain translations, just the source in English. We tried but we failed. This project is too much dynamic and even maintaining this documentation is a hard job. Translators coordination is not an option either. We are going to do what OpenCPN does, schematic and concise documentation just for reference. We do not want manuals, tutorials or detailed "How-to's". That makes people free to generate their detailed manuals, tutorials or videos for newbies in their language. That works for OpenCPN so let's try.
- Remember, this is a reference book, not a tutorial. Be brief and concise.
- A picture is worth a thousand words. In order to keep the same style use this tool to make screenshots:

```
sudo apt install gnome-screenshot
```

with these settings:



- English is not our main language and there will be many grammatical mistakes, help us solve it too.

CHAPTER 7

Downloading

Important: The latest models of Raspberry 4 with the latest firmware will not boot with these OpenPlotter 2 images of the **Basic** method. If this is your case, you have to use the **Advanced** method until we finish OpenPlotter 3.

Level	Platform	Download
Basic	• Raspberry Pi 3 • Raspberry Pi 4 *	• `OpenPlotter Starting` • `OpenPlotter Headless` • `OpenPlotter Moitessier HAT` • OpenPlotter À la Carte (under construction)
Advanced	• Raspberry Pi 3 • Raspberry Pi 4 * • Debian derivative 32-bit • Debian derivative 64-bit *	`OpenPlotter Settings deb`
Expert	• Raspberry Pi 3 • Raspberry Pi 4 *	`OpenPlotter pi-gen`

* Recommended

What to choose?

Some OpenPlotter features are exclusive for Raspberry Pi but if you do not need them you can install OpenPlotter in any computer running a Linux Debian derivative like *Ubuntu*, *Linux Mint* or even *Raspberry PI OS*. That is the first

question you should ask yourself.

We try to provide solutions for everyone, from newbies to experts. The second question would be, *what do I know about Linux?* or even, *I am a Linux expert but I feel lazy, should I choose a ready-to-use option?* The text below will help you answer these questions.

7.1 Basic

This is the easiest and fastest way of having OpenPlotter working. Our OpenPlotter distributions are based on Rasbian. We publish different editions according to the most demanded uses containing all the required apps installed and preconfigured. Just plug and sail!

OpenPlotter Starting - All required apps to fulfill most OpenPlotter marine features.

Installed apps Settings, OpenCPN installer, Signal K installer, Xygrib, Network, Dashboards, Serial, CAN, Docs

OpenPlotter Headless - *Starting* edition apps ready to be used remotely without monitor.

Installed apps *Starting* edition.

Settings WiFi access point enabled, ssh enabled, remote desktop enabled.

OpenPlotter Moitessier HAT - *Starting* edition apps plus required apps to use the Moitessier HAT out of the box.

Installed apps *Starting* edition, I2C, Pypilot.

Settings GNSS reception, AIS reception, compass, hell, pitch, pressure.

OpenPlotter À la Carte (under construction) - *Starting* edition apps plus any app of your election. You will be able to customize some settings too.

Installed apps

Customizable settings

You do not need previous knowledge of Linux to install and use these OpenPlotter distributions. Follow the *basic manual* to install them on your SD card.

7.2 Advanced

Important: OpenPlotter 2 only works in Debian 10 buster 32-bit. You need to download and install Raspberry Pi OS (Legacy) from the offcial site: https://www.raspberrypi.com/software/operating-systems

We are already working on OpenPlotter 3 for Debian 11 bullseye 32-bit and 64-bit, but it is still in beta and not recommended for production. Go here to test and help with development: https://forum.openmarine.net/showthread.php?tid=3878

You can install OpenPlotter from scratch in any computer running your favourite Debian derivative distribution, and of course in *Raspberry PI OS*. Hovewer, if your distribution is not *Raspberry PI OS* and your computer is not a Raspberry Pi, you will not be able to install some apps.

Common apps Settings, Docs, OpenCPN installer, Xygrib, Signal K installer, Dashboards, Network, Serial, CAN, IoT, Signal K filter, Kplex, SDR VHF

Raspberry apps Pypilot, Moitessier HAT, I2C sensors, GPIO

You need basic knowledge of Linux to install OpenPlotter from scratch. Follow the *advanced manual* to install OpenPlotter from scratch.

7.3 Expert

Pi-gen is the tool used to create the official *Raspberry PI OS* images. We use a fork of pi-gen to create OpenPlotter images. Use the *openplotter* branch of our repository to create your own OpenPlotter flavor.

You need good knowledge of Linux to create your own OpenPlotter distributions. Follow instructions in README file.

CHAPTER 8

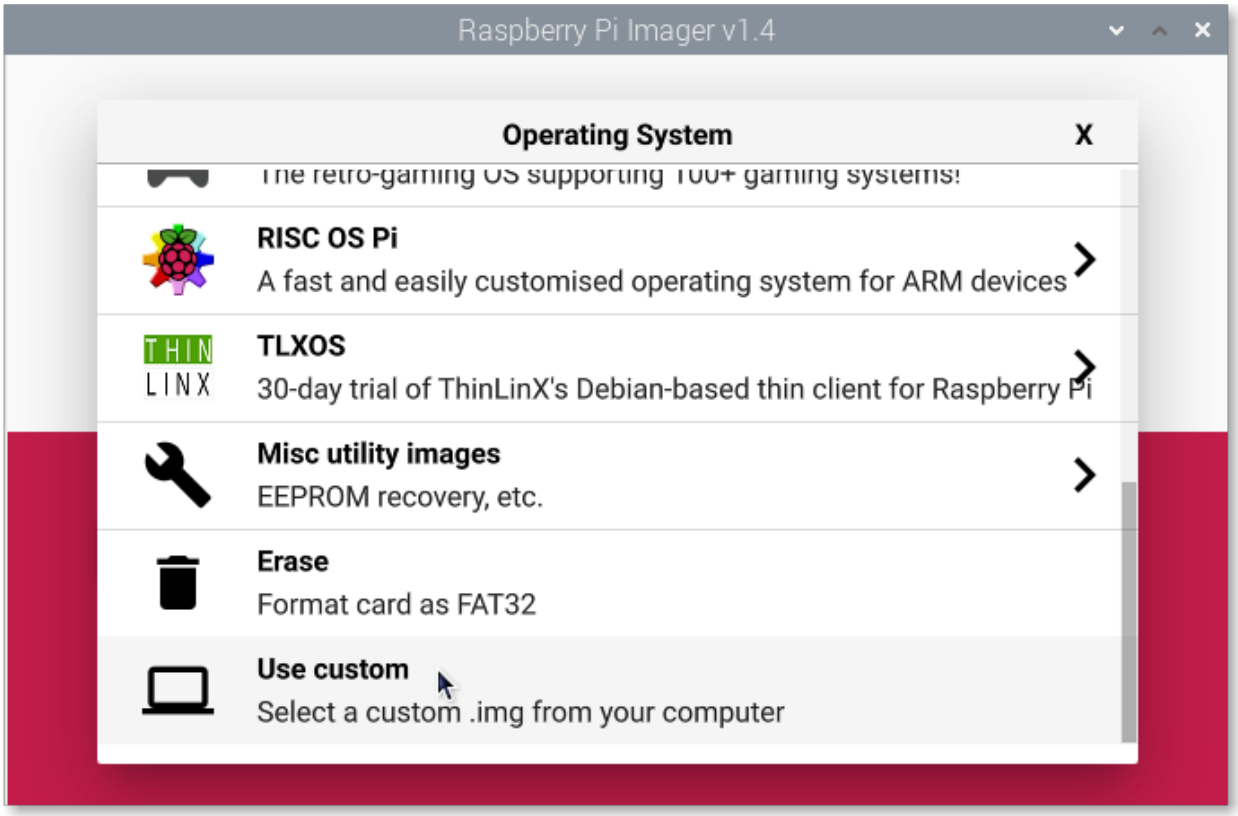
Installing

Important: The latest models of Raspberry 4 with the latest firmware will not boot with these OpenPlotter 2 images of the **Basic** method. If this is your case, you have to use the **Advanced** method until we finish OpenPlotter 3.

8.1 Basic

This is the easier and most common way to have OpenPlotter working on a Raspberry Pi in few minutes. You only need a micro SD card (minimum 8GB, recommended 16GB) and a computer with an SD card reader.

- Download and unzip your preferred OpenPlotter edition from the **Basic** section in *downloading* chapter.
- Download and install `Raspberry Pi Imager`.
- Put the SD card you will use with your Raspberry Pi into the reader and run *Raspberry Pi Imager*.
- Click on `CHOOSE OS` and then on `Use custom`:



- Select the .img file of your preferred OpenPlotter edition.
- Click on `CHOOSE SD CARD` and select your SD card.
- Click on `WRITE` and take a coffe.
- Remove the SD card from the reader, insert it into the raspberry and you are done.

After the first boot you can customize and localize your system changing some important settings like password or system language. You can also change these settings later in *Main → Preferences → Raspberry Pi configuration*.

Danger: You MUST change the default password for the user *pi*. Otherwise, any user will be able to access your system easily.

If you are using the **OpenPlotter Headless** edition, you should see the SSID of the access point after a few seconds of inserting the SD into the Raspberry and turning it on.

These are the access data to connect remotely to OpenPlotter when you use this headless edition:

Access Point	**SSID** openplotter **Password** 12345678
IP	**IP** 10.10.10.1 **Address** openplotter.local
SSH	**Command** ssh pi@openplotter.local **Password** raspberry
Remote desktop	**Address** openplotter.local **Port** 5900 **User** pi **Password** raspberry

Danger: You must change the default access point password in `OpenPlotter Network` app. Otherwise, any user will be able to access your system easily.

8.2 Advanced

Important: OpenPlotter 2 only works in Debian 10 buster 32-bit. You need to download and install Raspberry Pi OS (Legacy) from the offcial site: https://www.raspberrypi.com/software/operating-systems

We are already working on OpenPlotter 3 for Debian 11 bullseye 32-bit and 64-bit, but it is still in beta and not recommended for production. Go here to test and help with development: https://forum.openmarine.net/showthread.php?tid=3878

To Install OpenPlotter on any computer running a Linux Debian derivative system (*Raspberry PI OS*, *Ubuntu*, *Linux Mint*...) you have to install the dependencies.

Open a terminal and type:

```
sudo apt update
sudo apt install python3-wxgtk4.0 python3-ujson python3-pyudev whois vlc
```

Now you have to install the main app `openplotter-setting` from the .deb file you can download from the *Advanced* section in *downloading* chapter.

After downloading the .deb file, you can install it by double click or typing this in a terminal:

```
sudo dpkg -i openplotter-settings_x.x.x-stable.deb
```

Every time OpenPlotter needs to perform an action that requires administrator permission, it will ask for the password. To avoid having to continuously enter your administrator password you can add your user to the *sudoers* list. Do this only if you know what you are doing:

```
sudo visudo
```

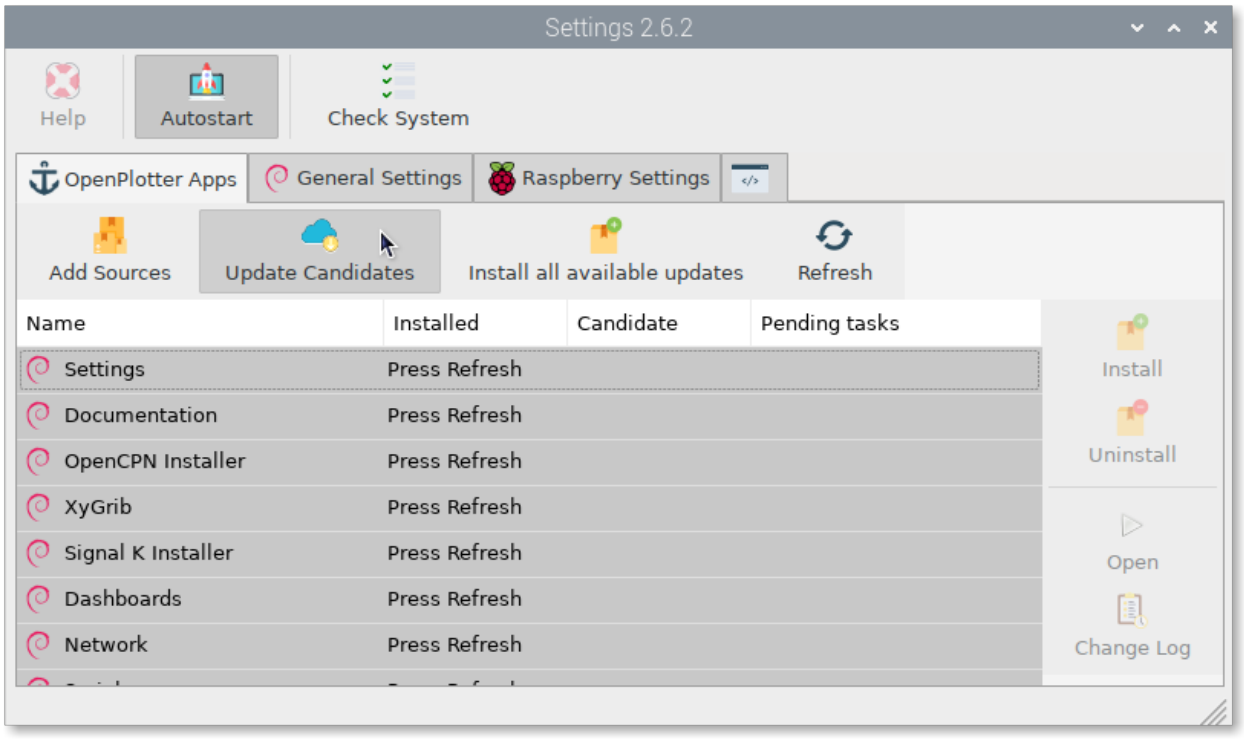
Add this line to the end of the document and save:

```
myuser ALL=(ALL) NOPASSWD: ALL
```

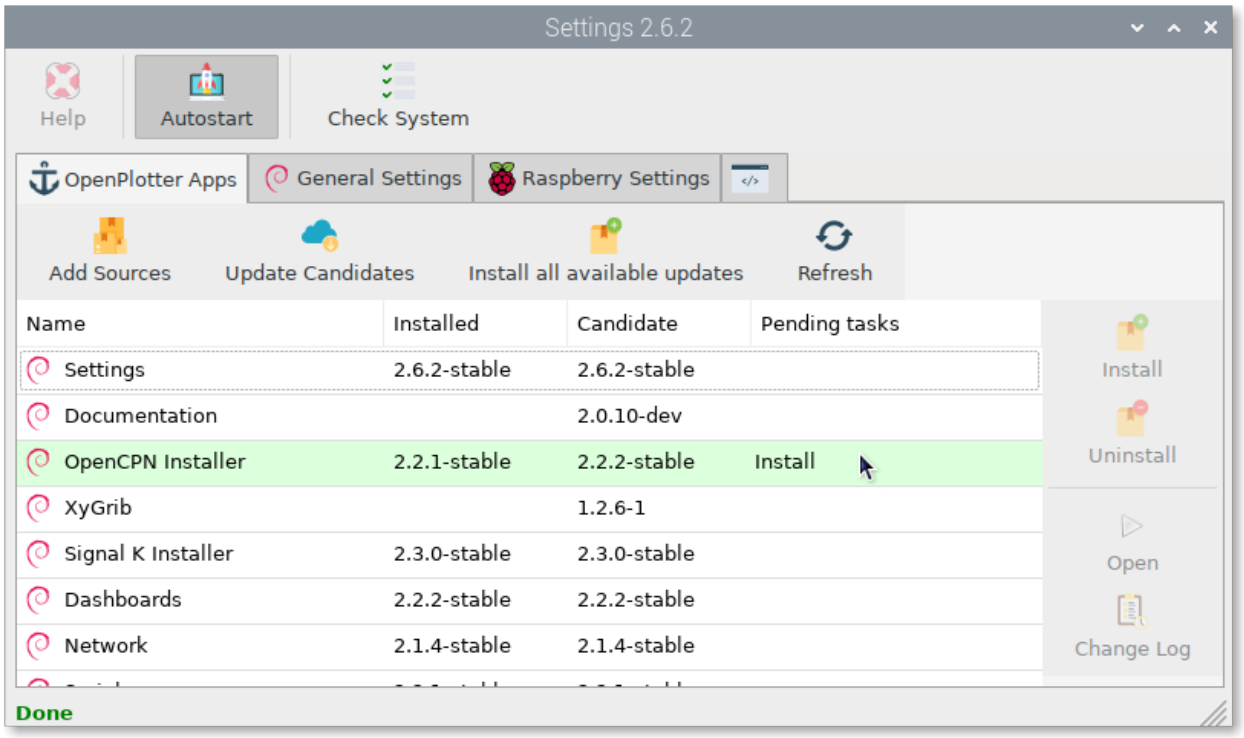
CHAPTER 9

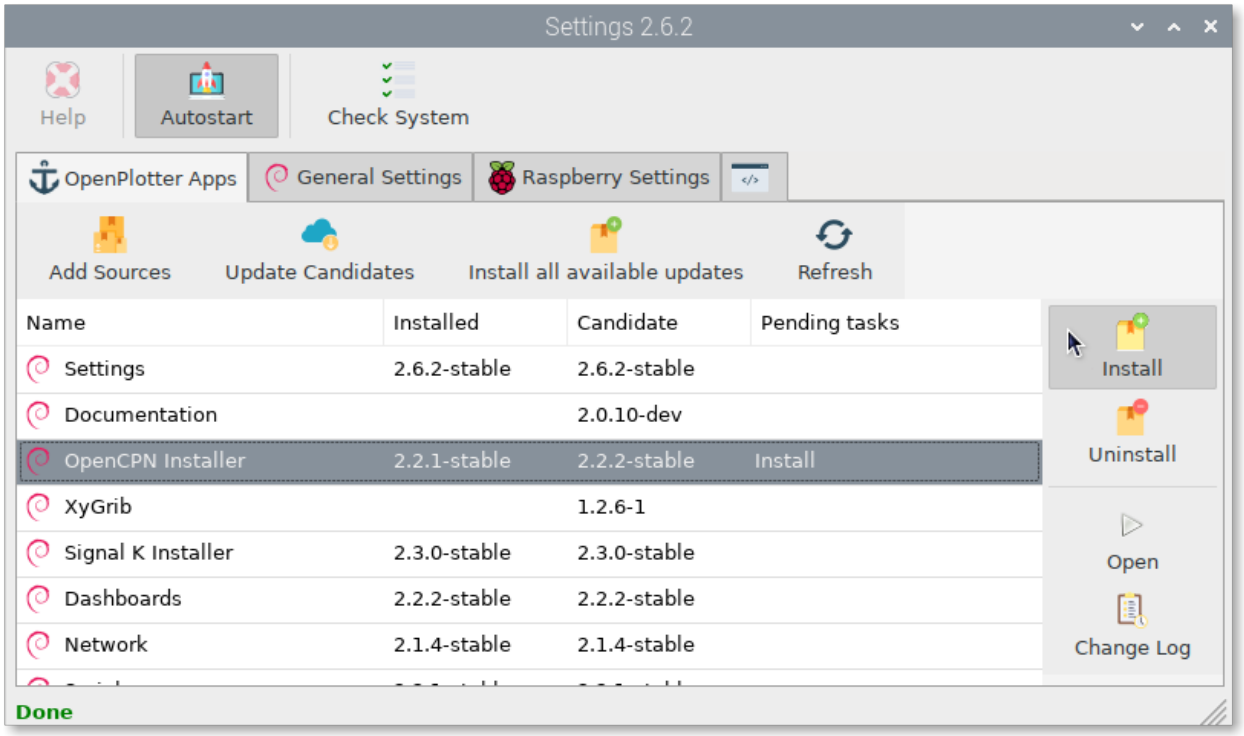
Updating

Occasionally, you should check if there are new versions of OpenPlotter apps to enjoy new features and correct errors. Click on `Update Candidates`



Available updates will be marked with green background. Select the app you want to update and click on `Install`

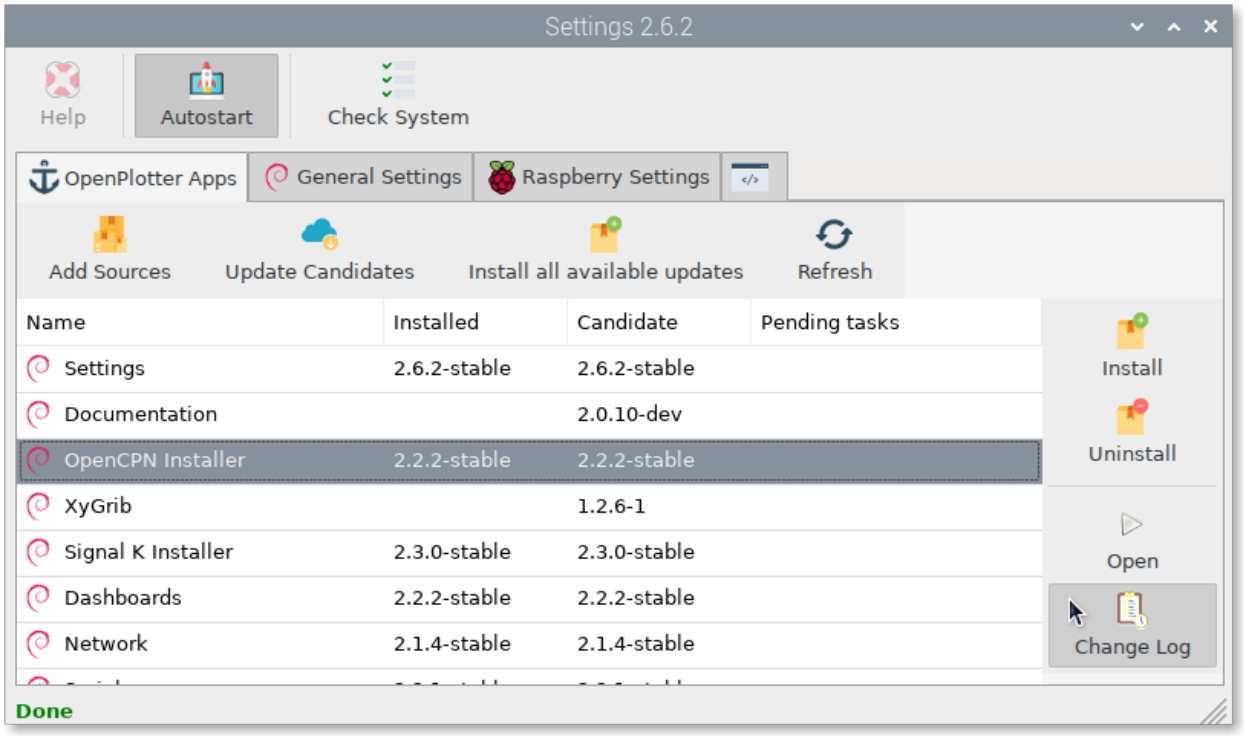




If the updated apps were running while the installation, you will have to close and open them again to see changes.

9.1 Change Log

If you want to know the changes in old and latest versions, select any app and click on `Change Log`



9.2 Version numbering

OpenPlotter apps versions consist of 3 digits separated by periods (a.b.c), a code name and a state:

a This is the OpenPlotter version the app belongs. This value will change only when a new Debian version is released.

b This value will change when major updates like new features have been added.

c This value will change when minor updates like fixed bugs or translations have been added.

codeName Name to identify the OpenPlotter version (a).

state dev, beta or stable.

CHAPTER 10

Backup

CHAPTER 11

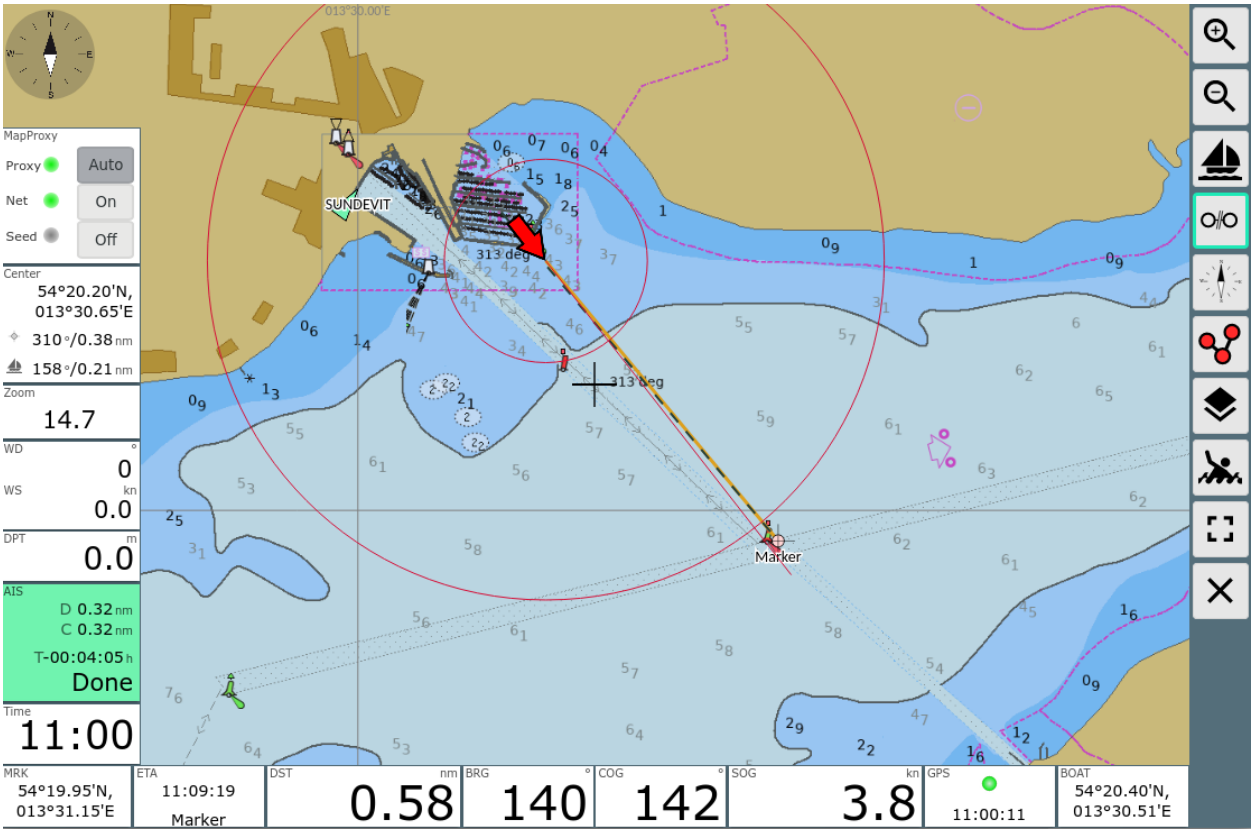
OpenPlotter Settings

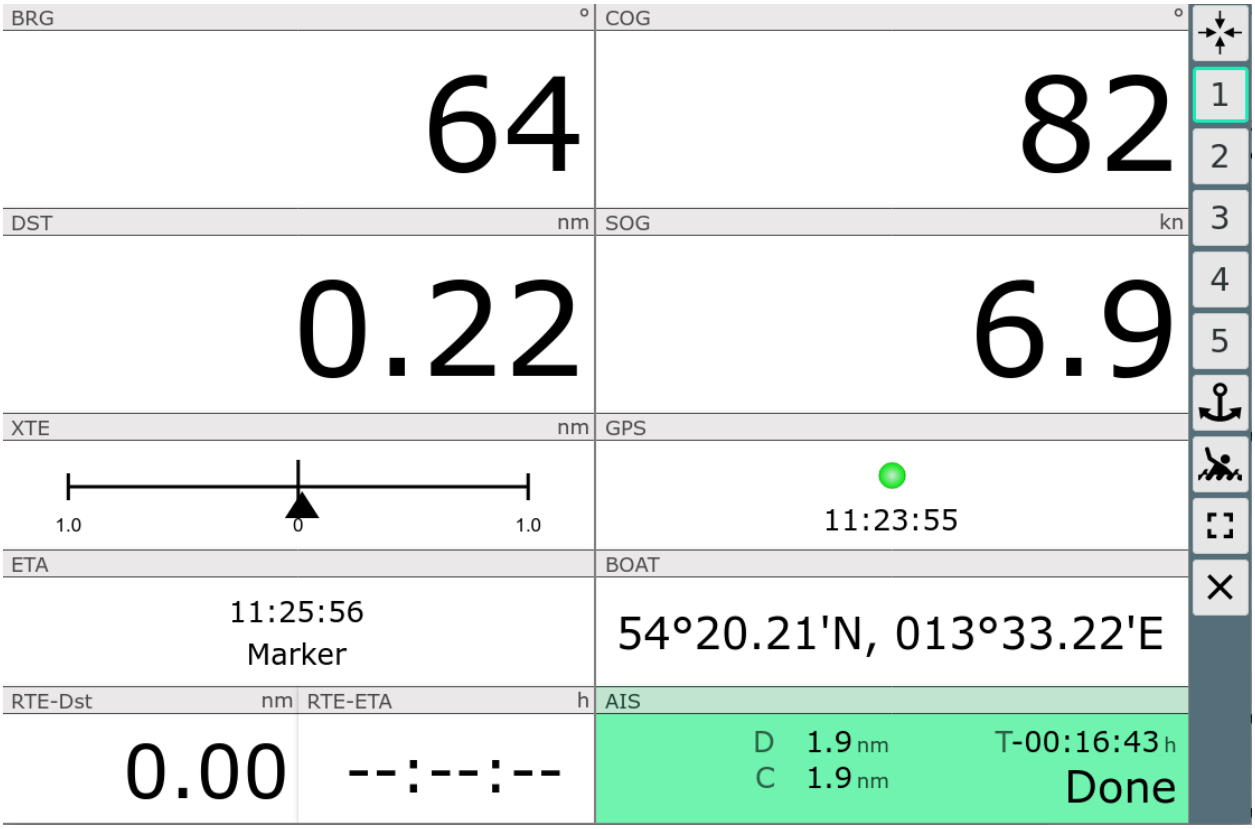
CHAPTER 12

OpenCPN Installer

CHAPTER 13

AvNav Installer





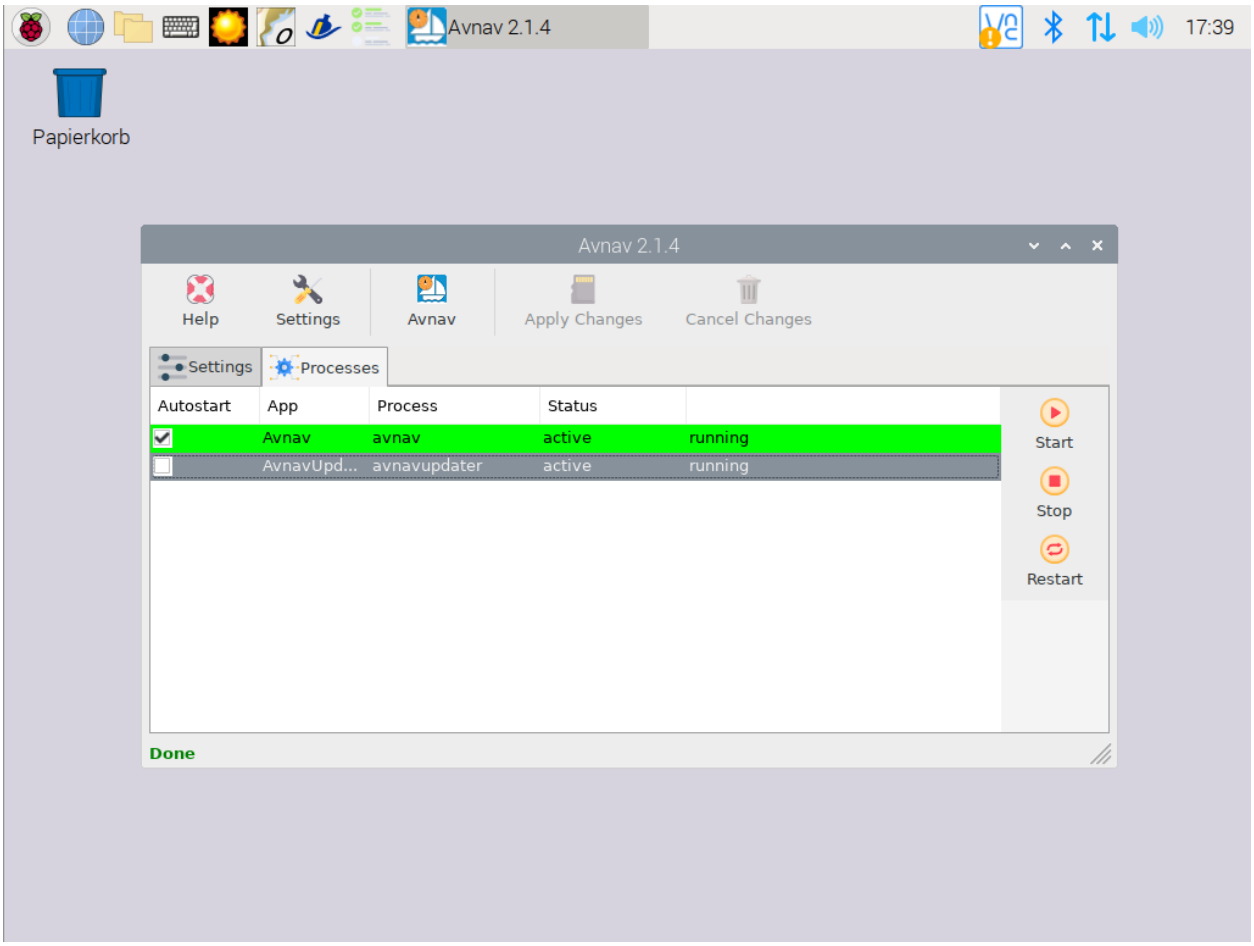
13.1 AvNav Features

- client server based navigation solution
- server runs on OpenPlotter, client can be any browser
- optimized for touch devices
- integration with SignalK and stand alone
- raster charts and vector charts from o-charts
- AIS display and CPA computation
- creating, editing, using, importing and exporting of routes
- simple waypoint routing
- chart overlays (multiple chart layers, gpx, kml, geojson. . .)
- MOB alarms
- anchor watch and alarm
- chart display with configurable info displays
- multiple configurable dashboard pages
- night mode
- multiple displays all being synced
- plugins (history, mapproxy, update,. . .)

- can be adapted/customized with css, java script and python

For further details refer to the documentation or have a look at some videos.

13.2 Installer



- will download and install the AvNav software and the necessary plugins
- shows the status of the server part and allows to start and stop it
- allows for some basic port settings (others directly within the AvNav app itself)
- set up connectivity between SignalK and AvNav
- NMEA traffic is configured to run from SignalK to AvNav

13.2.1 Processes and Ports

AvNav has a main process ("avnav") that handles all the NMEA data and server functions. It has one port for the web access (default: 8080). To handle o-charts there is an additional process (controlled and started by avnav) that has an own port for the access to those charts (default 8082). To run updates from the AvNav web app there is an additionaly avnavupdater process that is running independently from avnav and again has an own port for the web access (default: 8085). You can modify the ports at the settings tab if they interfere with others applications you have installed. OpenPlotter will check this and warn you accordingly. aha

CHAPTER 14

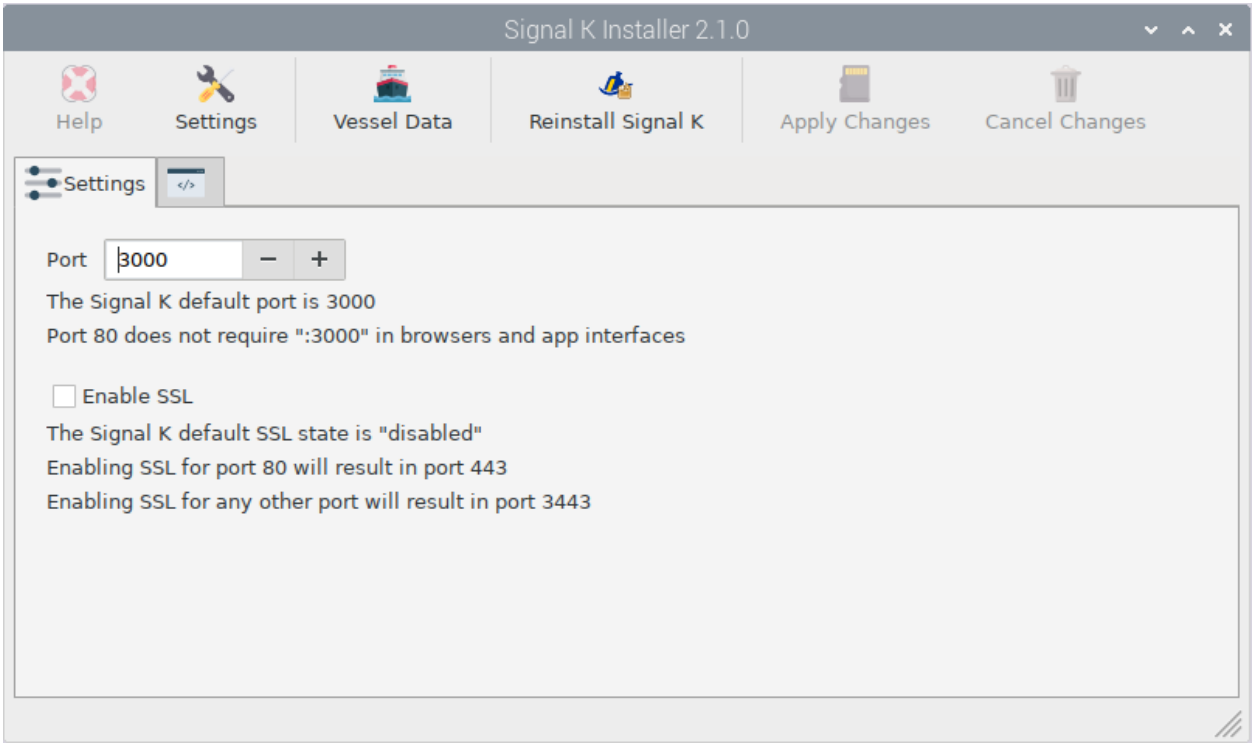
Signal K Installer

This app helps you to:

- Reinstall Signal K
- To change the default port
- To enable/disable SSL

Usually this shouldn't be necessary.

- There is a link into the Signal K admin ui to the Vessel data (Name, MMSI, Length, . . .)



CHAPTER 15

Signal K Security

When you enter the Signal K admin ui, it will ask you for a name and a password to create an administrator account. Once you do that you will be offered the login page.

The last menu item in the ui is security. You can add/delete users and change passwords . . .

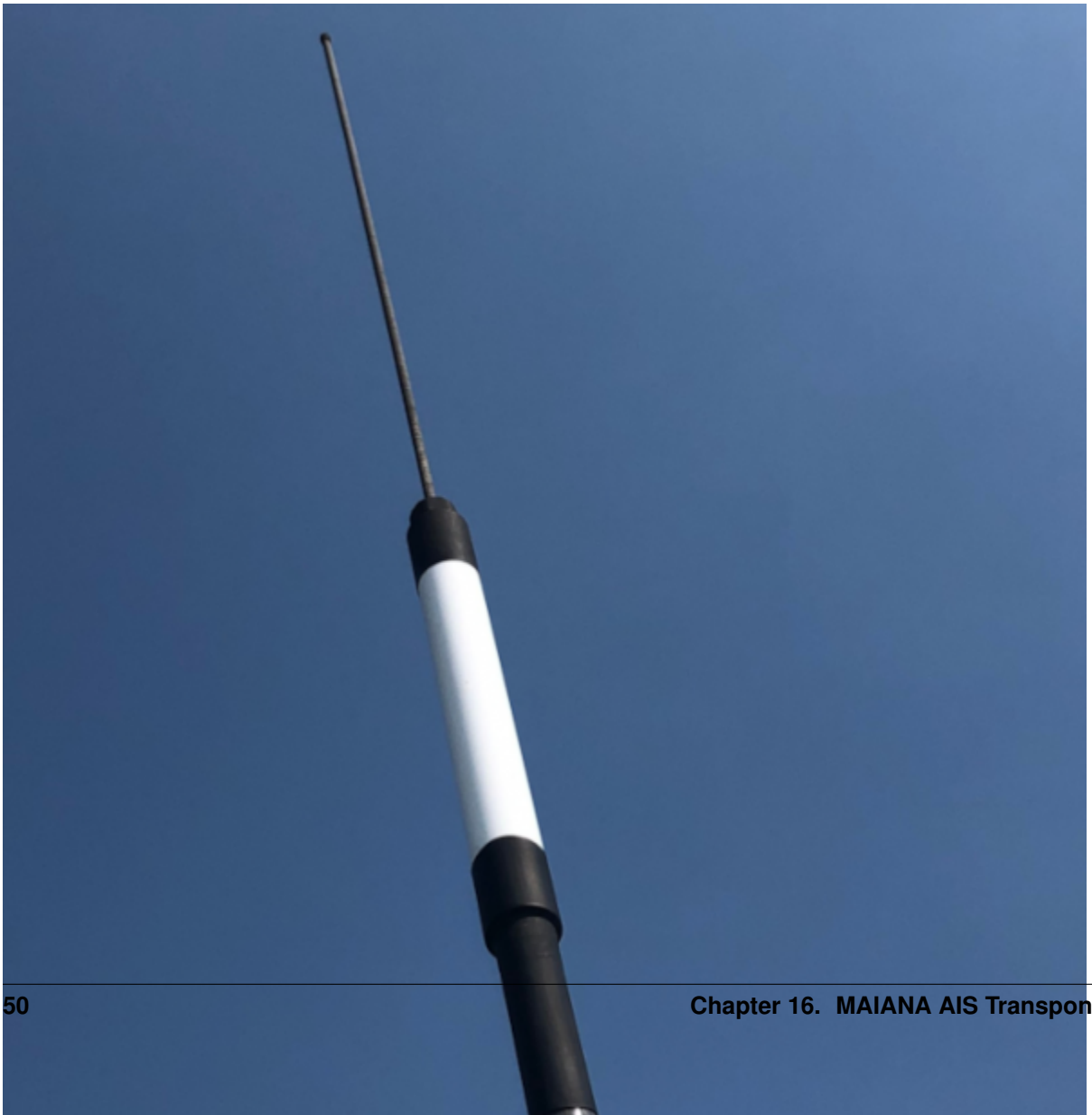
15.1 If you have lost the password

You can reset Signal K security by:

1) Open a terminal
2) Delete the existing file: /home/pi/.signalk/defaults.json
3) Run the setup sudo signalk-server-setup. Accept the update option rather than configuring from scratch. Select no for port 80 and SSL
4) Navigate to the login page with a browser. Note that the option is now not to login but instead to create an administrator account. Once you do that you will be offered the login page.

CHAPTER 16

MAIANA AIS Transponder



MAIANA™ is the first Open Source AIS transponder.

The main difference between MAIANA and all commercial AIS devices is that it is a self-contained unit, all AIS and GNSS circuits are located in the antenna housing. MAIANA receives GNSS and AIS data on both channels and can be enabled as a class B transponder. The transponder outputs just over 2 Watts (+33dBm). It has a verified range of over 20 nautical miles on a masthead and 10+ miles on a pushpit.

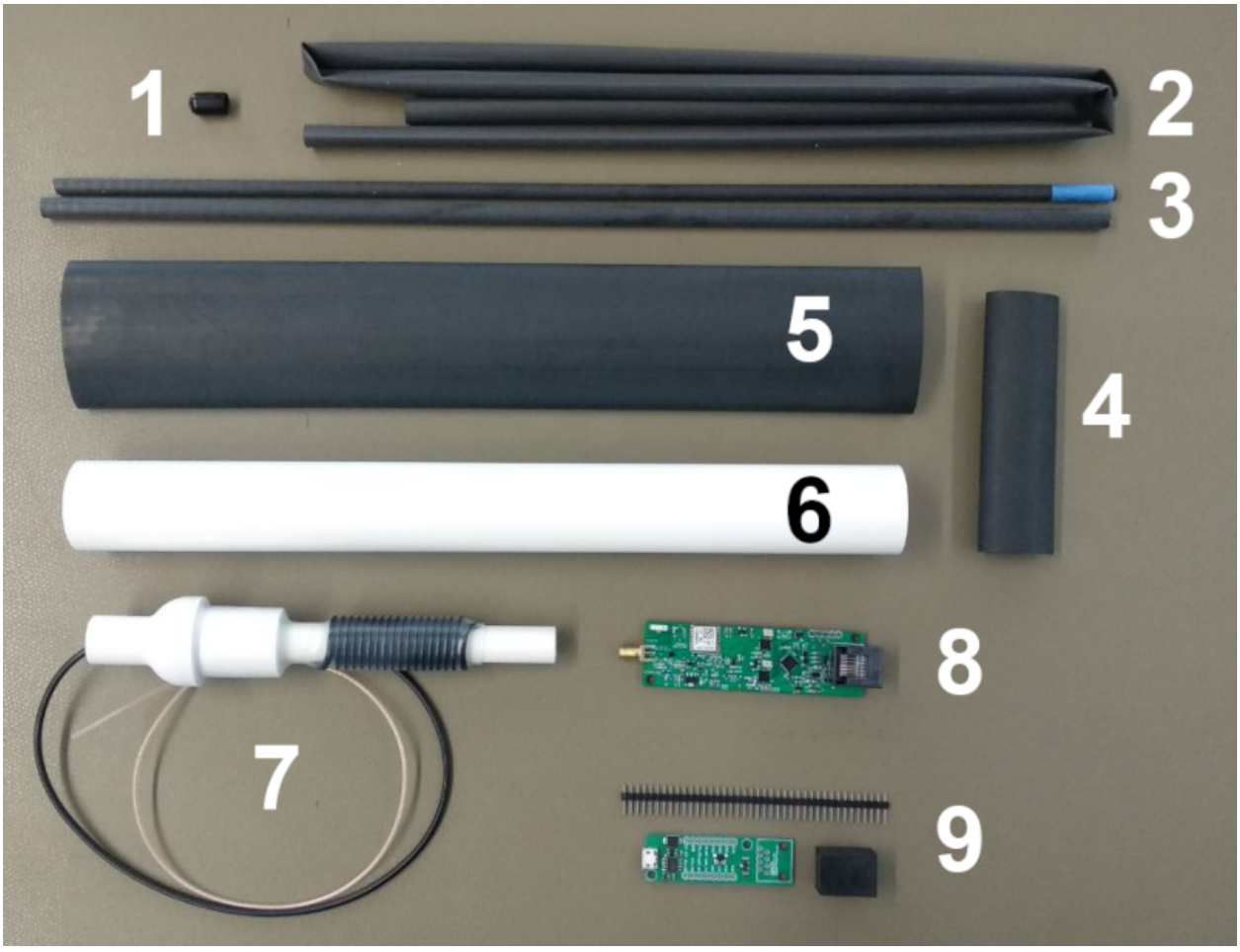
Note: This product is available in the OpenMarine Shop.

On the official page you will find the full specification and a better option for US/Canada users to get a kit.

Disclaimer

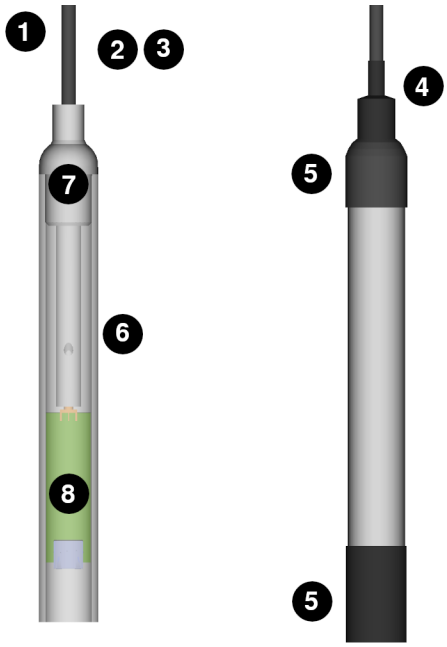
MAIANA™ devices are not finished products, they need to be assembled by end users to function. MAIANA™ devices are distributed for research and development purposes. MAIANA™ devices are delivered with the ability to transmit disabled by default and have not been tested for compliance with regulations governing transmission of radio signals. End users are legally responsible for using the MAIANA™ device for transmission. We do not recommend to rely solely on MAIANA™ devices for navigation and collision avoidance.

Assembly



1. The vinyl end cap for the antenna tube.
2. 48” of ” 2:1 heat shrink tube, folded.
3. The antenna tube (two telescopic sections).

4. 6” of ¾” 4:1 heat shrink tube, black (enough for 2 builds).
5. 12” of 1.5” wide heat shrink tubing, black (enough for 2 builds).
6. The main case (high-UV resistance PVC).
7. The antenna core (coiled and stripped coax with SMA male on one end).
8. The main PCBA (in ESD envelope).
9. The unsoldered breakout board (in ESD envelope).



Easy assembly. To complete the installation you will need:

- A pair of strong scissors for cutting (thick) heat shrink tubing.
- A heat gun for the heat shrink tubing. You will need this both on your workbench for the initial assembly, as well as on your boat for the final installation. Alternatives: How to Use Heat Shrink Tubing Without a Hot Air Gun. Beware of anything that projects a flame, as it can easily melt the PVC enclosure!
- 1” OD steel railing or a similar diameter fiberglass mast on your boat. This is the preferred way to mount the unit. You may, of course, use your own mechanism, but then you are responsible for sealing the (bottom) cable end from moisture.
- A Cat5 cable for connecting the main unit to the breakout board in the cabin. This should be a regular *patch* cable and not a *crossover* cable. Pick one with appropriate length and flexibility to suit your installation. The exact configuration of the cable (568A or 568B) is not important.
- Some kind of instant glue to secure the end cap of the antenna tube.
- If you are using the included breakout board you will need a soldering iron for the RJ45 connector and “break-away” pin headers. All other optional adapters are mounted and soldered.

Note: Download the official assembly manuals.

CHAPTER 17

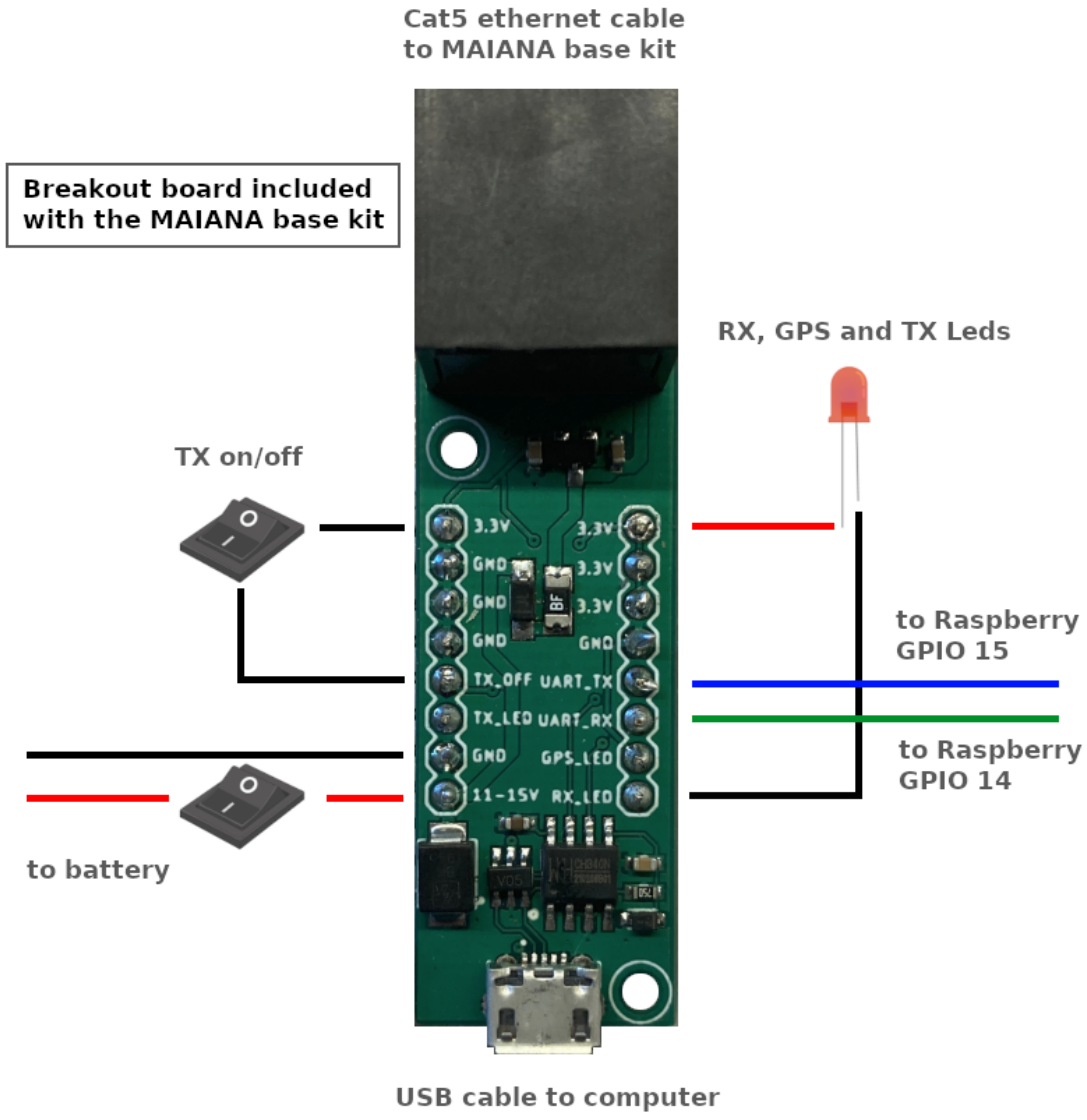
Connecting the base kit

1. MAIANA base kit.
2. Cat5 cable (ethernet cable with RJ45 connectors). Not included with the MAIANA base kit.
3. Breakout board. Included with the MAIANA base kit.
4. USB + UART adapter. Not included with the MAIANA base kit.
5. USB + RS422 Adapter. Not included with the MAIANA base kit.
6. USB + CAN Adapter. Not included with the MAIANA base kit.

After the MAIANA base kit assembly process, you need to connect it to your boat via the Cat5 ethernet cable to power it, to get AIS/GNSS data and to configure the device. You have 4 options: the Breakout board or the USB+UART, USB+RS422 and USB+CAN adapters.

17.1 Breakout board

This board is included with the base kit and it is designed so that you can incorporate MAIANA into your projects. Both USB and UART connections are used to get AIS/GNSS data, configure the device or update the firmware.



If you want a power and/or TX switch, you can use simple rocker switches. A 1A-rated SPST can simply interrupt the main 12V supply.

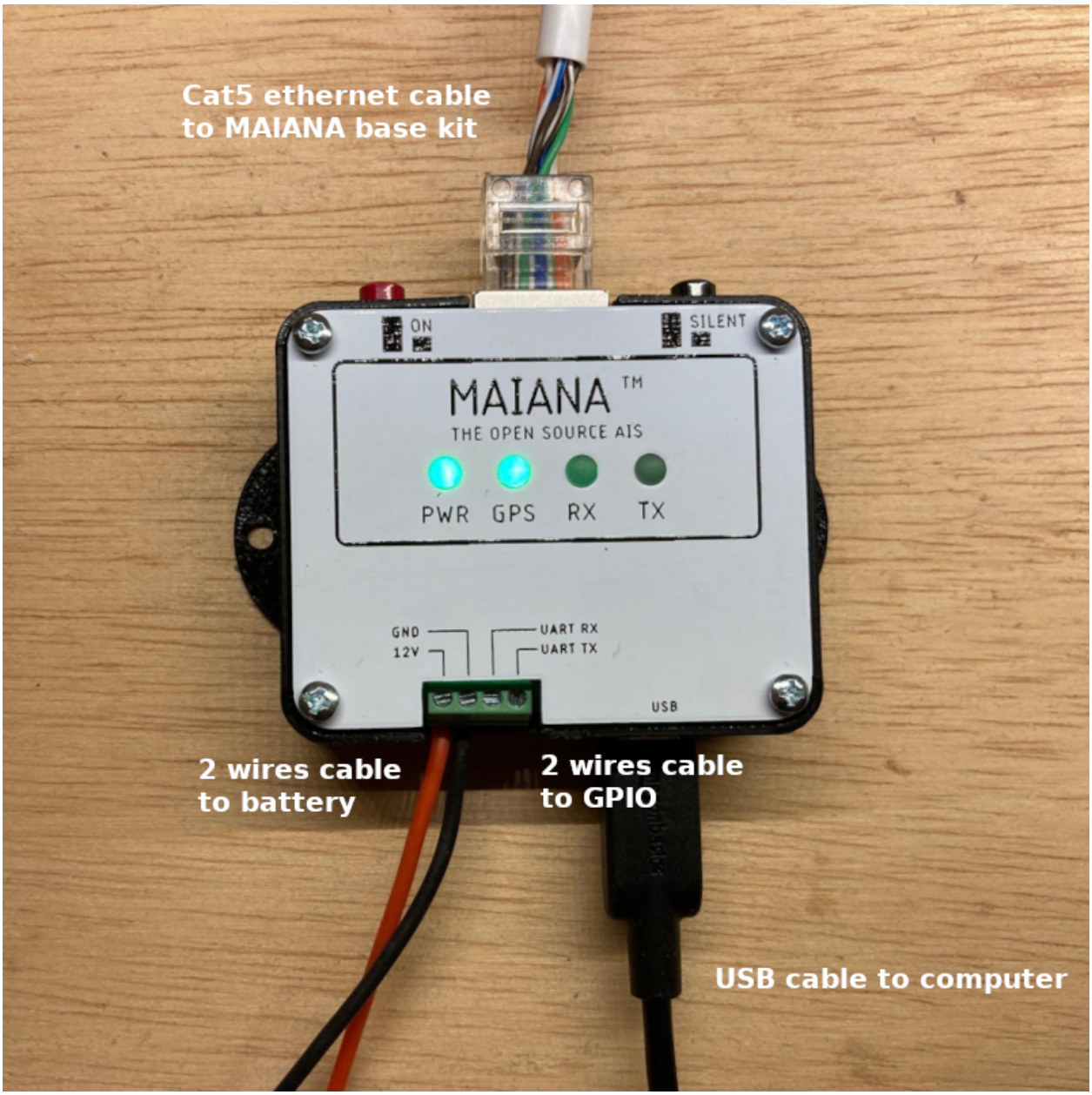
If you want a hardware switch for *silent mode*, you need to remember that transmission is disabled if the TX_OFF signal is driven to a logic *high* (above 2V), so wire it as shown in the picture.

The LED signals are open drain outputs. Rather than supplying a voltage, they pull the cathode of the LED to GND via a built-in 100 Ohm resistor. The voltage you apply to the anode is flexible (up to 30V tolerated), but the breakout supplies 3.3V so take advantage! That said, some LEDs may still draw too much current and will need an extra resistor added in series. You can wire that on either the anode or the cathode side.

17.2 USB + UART adapter

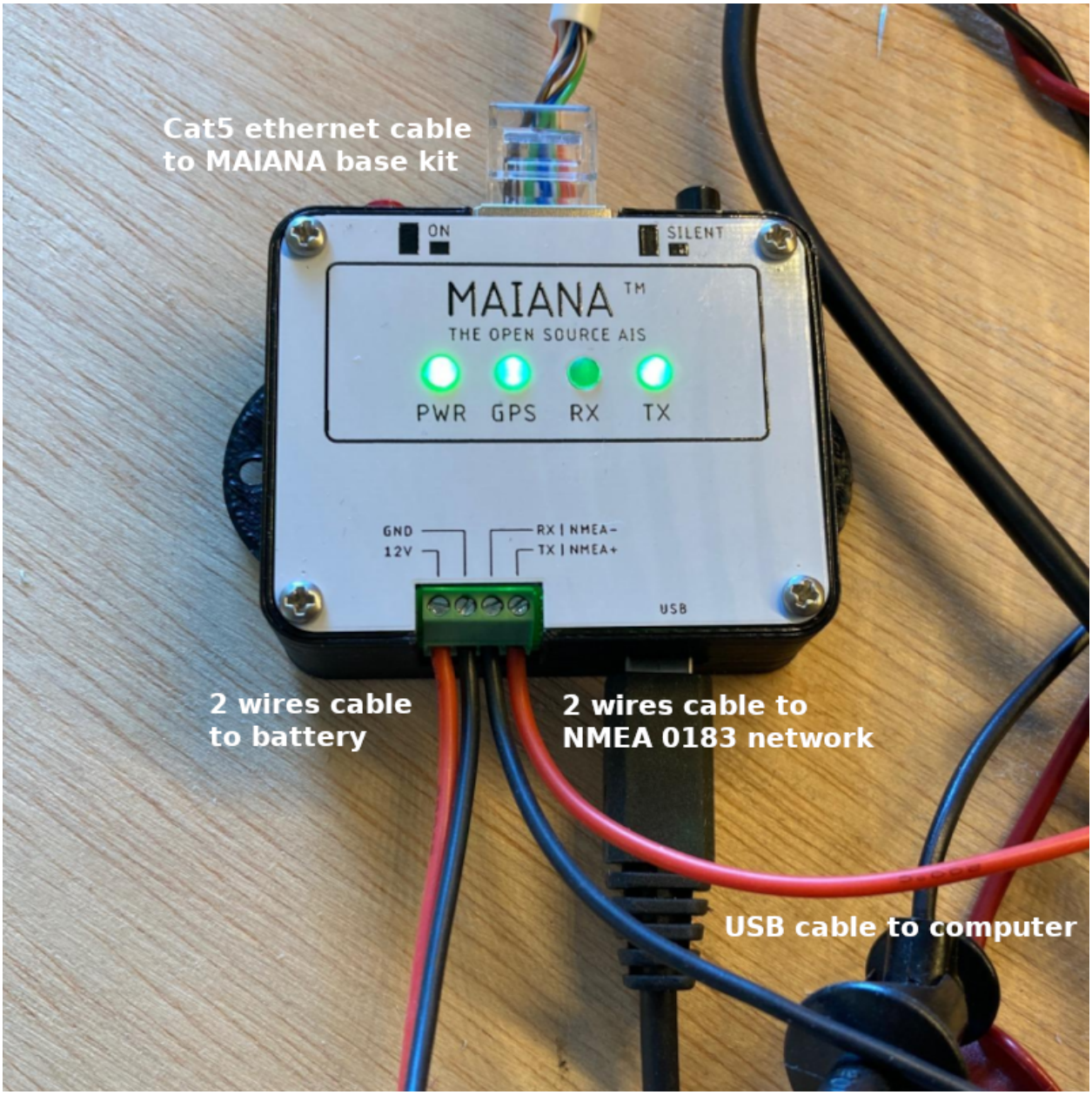
Both USB and UART connections are used to get AIS/GNSS data, configure the device or update the firmware. This adapter has the same functions as the included *breakout board* but incorporates status LEDs, switches and is soldered

and assembled.



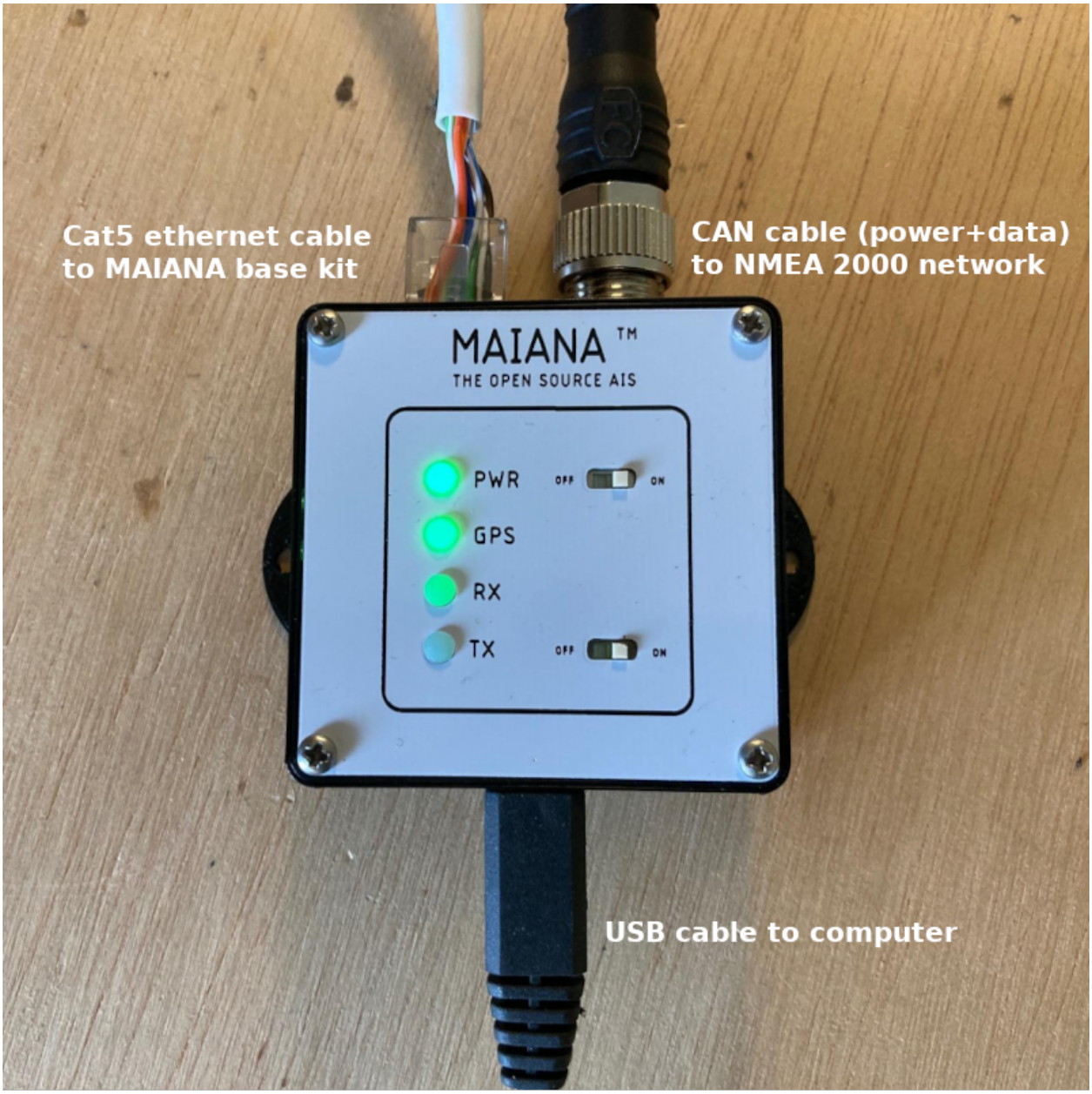
17.3 USB + RS422 Adapter

USB connection is used to get AIS/GNSS data, configure the device or update the firmware. RS422 connection is used only to send AIS/GNSS data to the NMEA 0183 network of your boat. This adapter incorporates status LEDs, switches and is soldered and assembled.



17.4 USB + CAN Adapter

USB connection is used to get AIS/GNSS data, configure the device or update the firmware. CAN connection is used only to send AIS/GNSS data to the NMEA 2000 network of your boat and power the device. This adapter incorporates status LEDs, switches and is soldered and assembled.


Cat5 ethernet cable
to MAIANA base kit
CAN cable (power+data)
to NMEA 2000 network
MAIANA ™
THE OPEN SOURCE AIS
PWR
OFF
ON
GPS
RX
TX
OFF
ON
USB cable to computer

CHAPTER 18

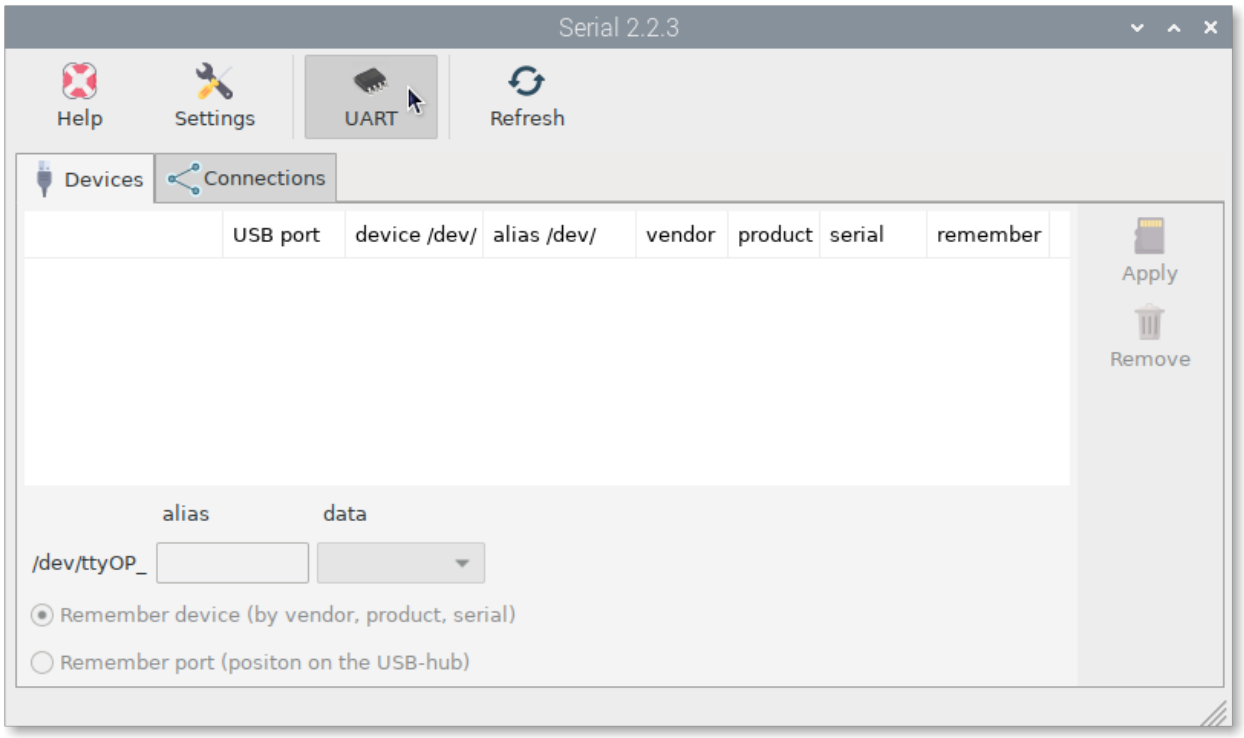
Configuring OpenPlotter

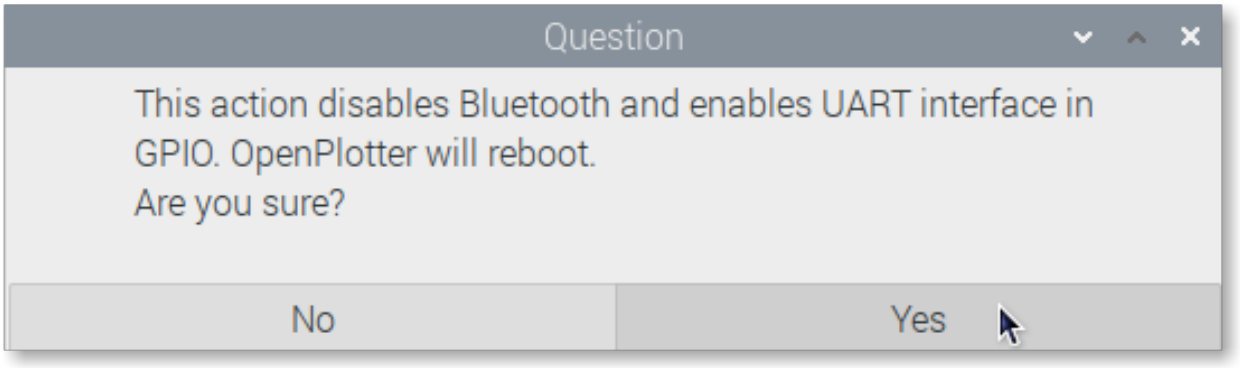
You can configure OpenPlotter to get AIS and GNSS data from a MAIANA transponder with just a few clicks. You will also learn how to enable transmission, configure the device, and update the firmware.

18.1 Getting AIS and GNSS data

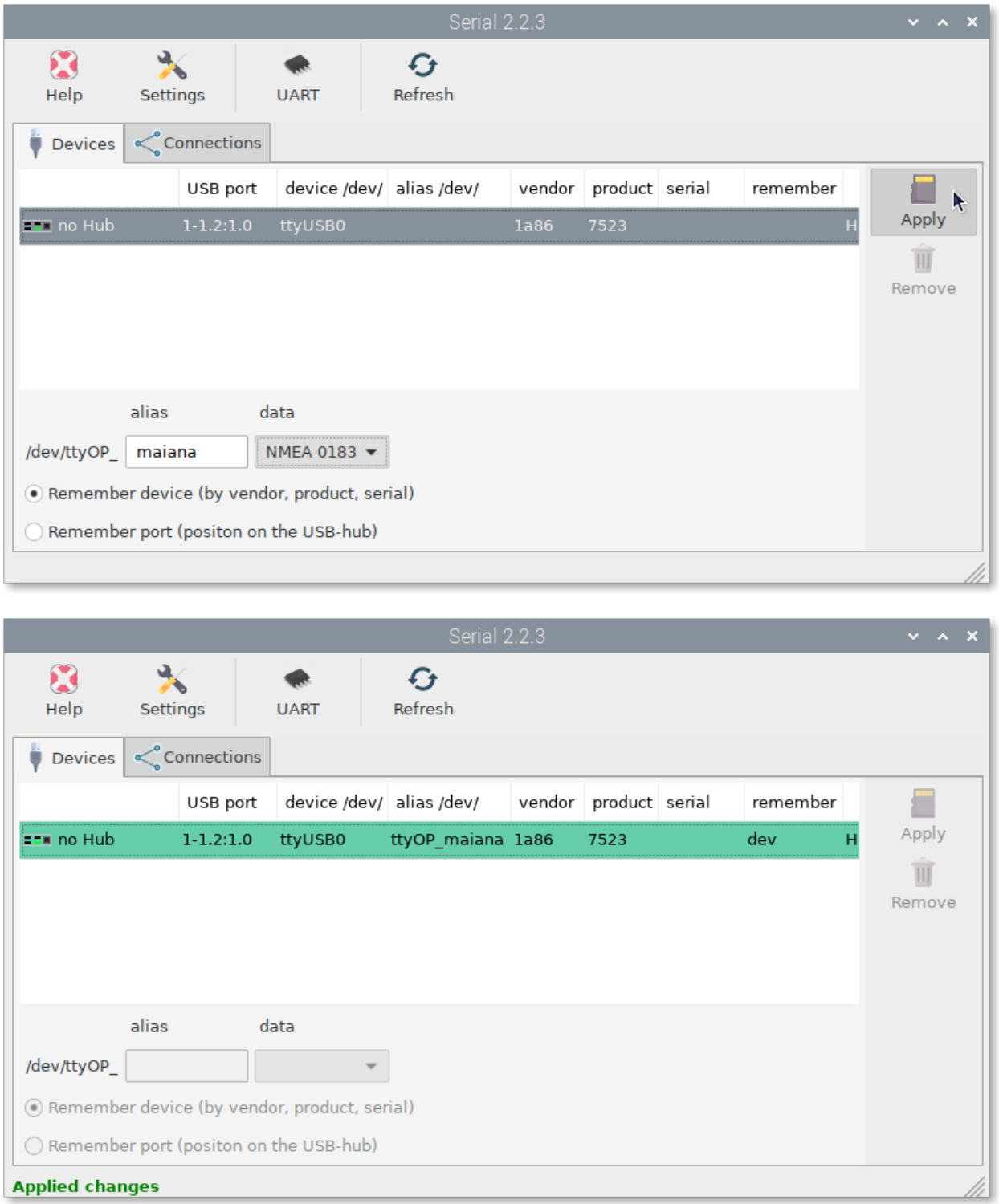
MAIANA is ready to rceive and send AIS and GNSS data out of the box, just power on the device and connect by USB or UART to OpenPlotter. We want to send MAIANA data to the Signal K server so that any program like OpenCPN can access AIS and GNSS data. We will do it easily using the *OpenPlotter Serial* app.

If you are connected by UART, first of all you need to enable the UART interface of your Raspberry Pi. Click `UART` and then click `Yes`. Remember that enabling the UART interface will disable Bluetooth. If you are connected by USB, skip this step.

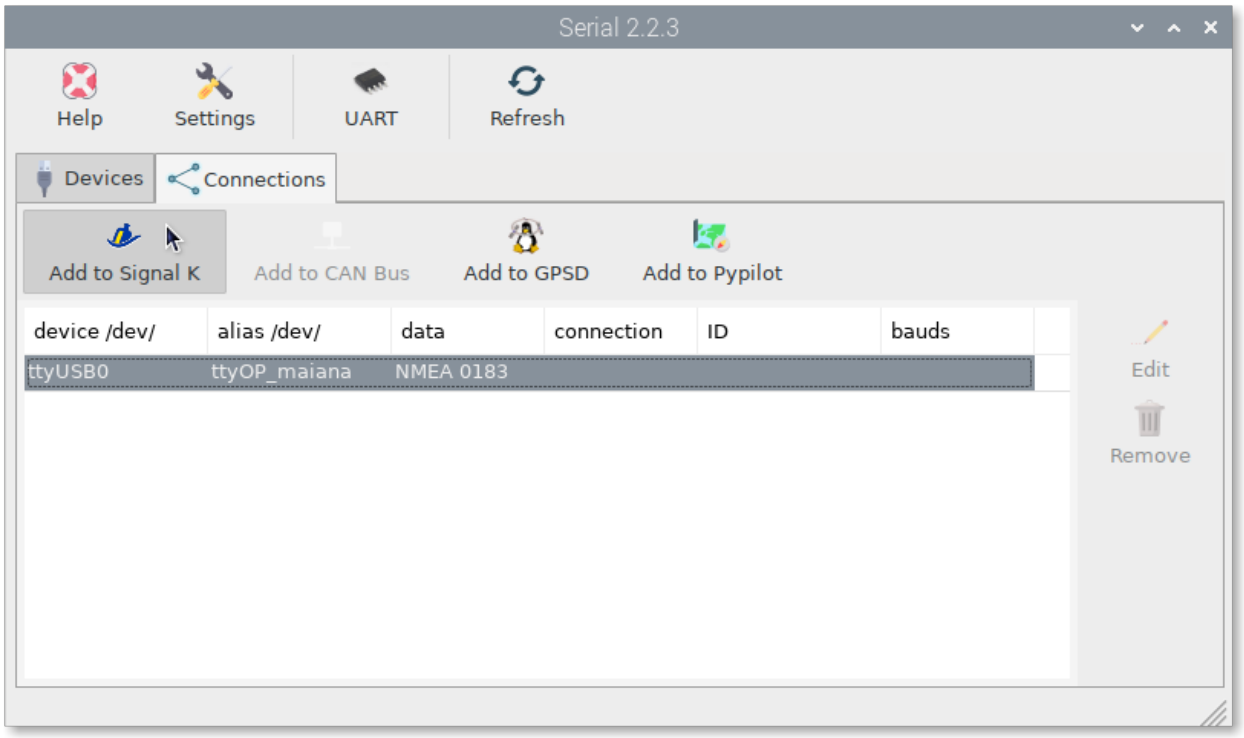


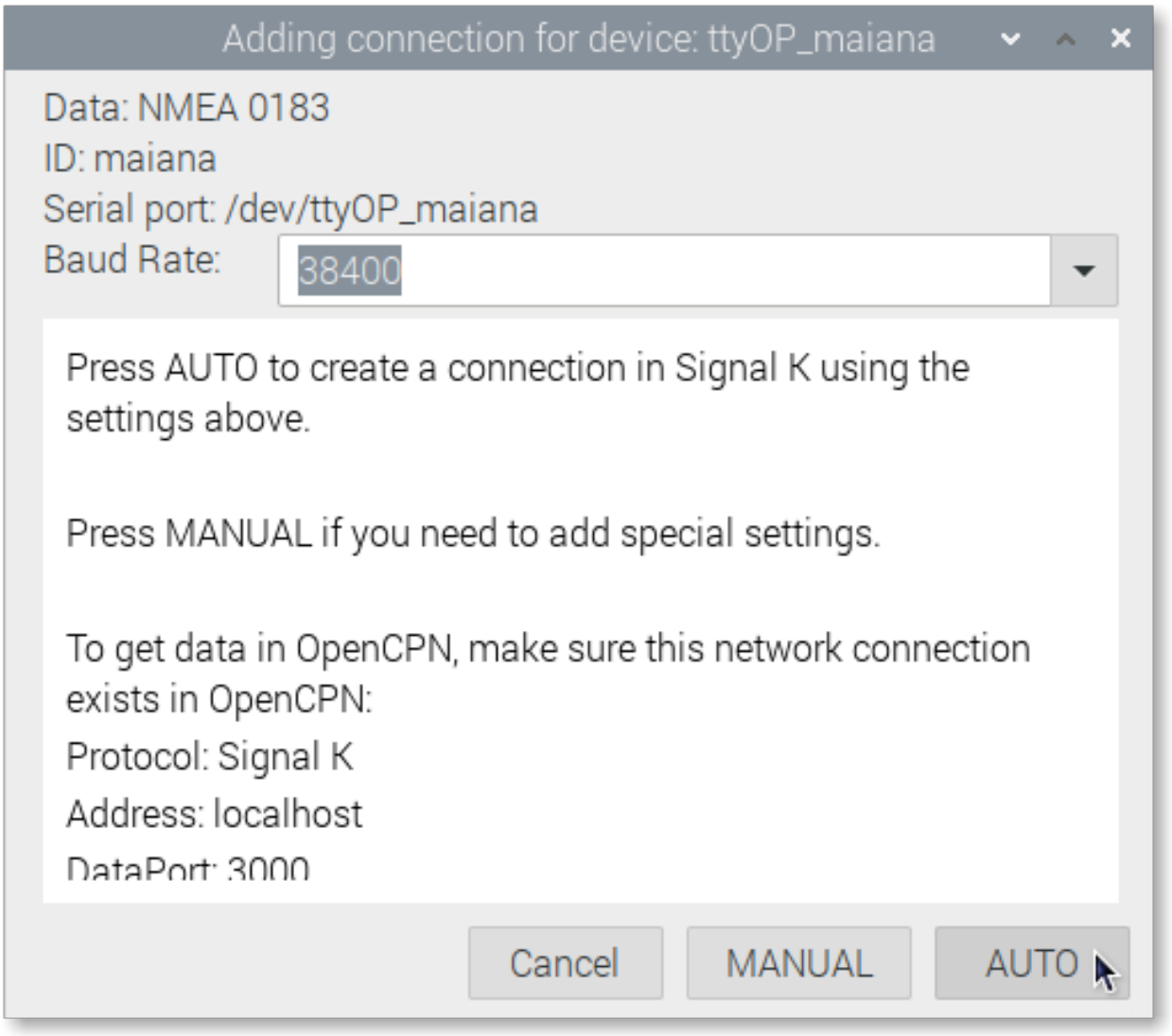


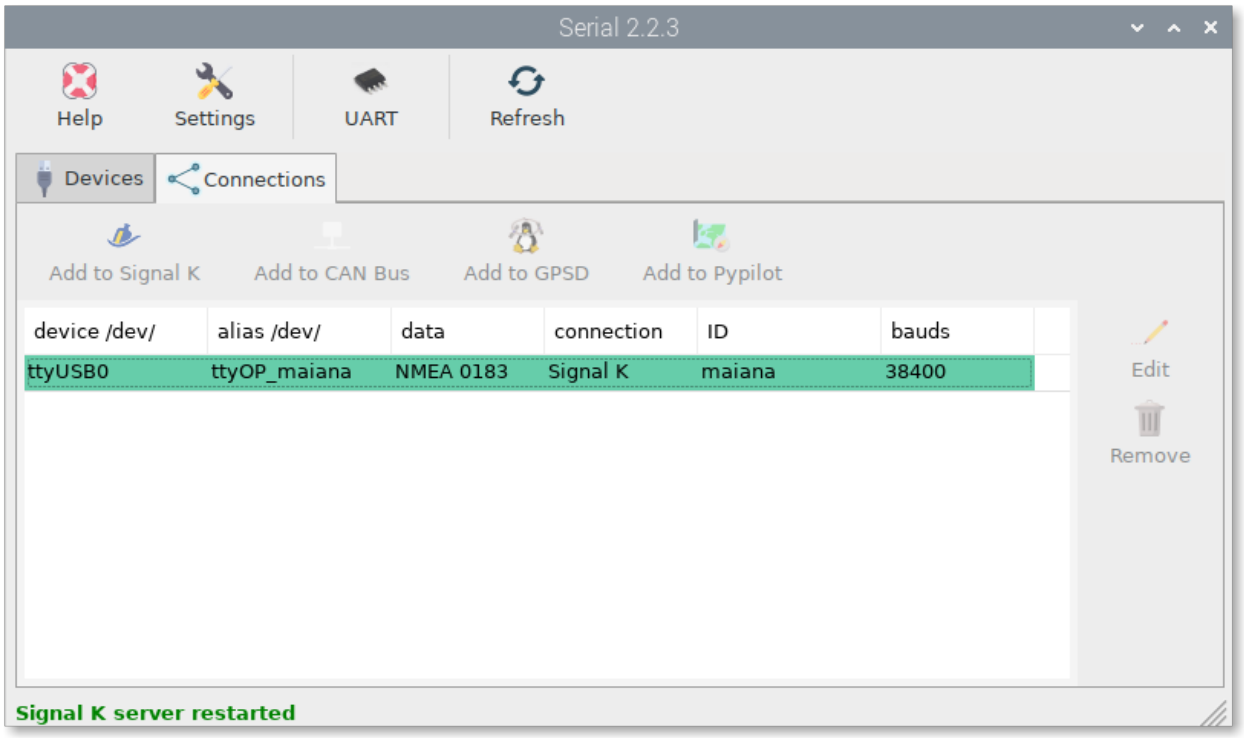
After enabling UART or just plugging in the USB and clicking `Refresh`, you will see a new device listed. Select this new device and provide a short name for the *alias* and select NMEA 0183 under *data*. If it is connected by USB check *Remember device* and if it is connected by UART check *Remember port*. Click `Apply` when done.



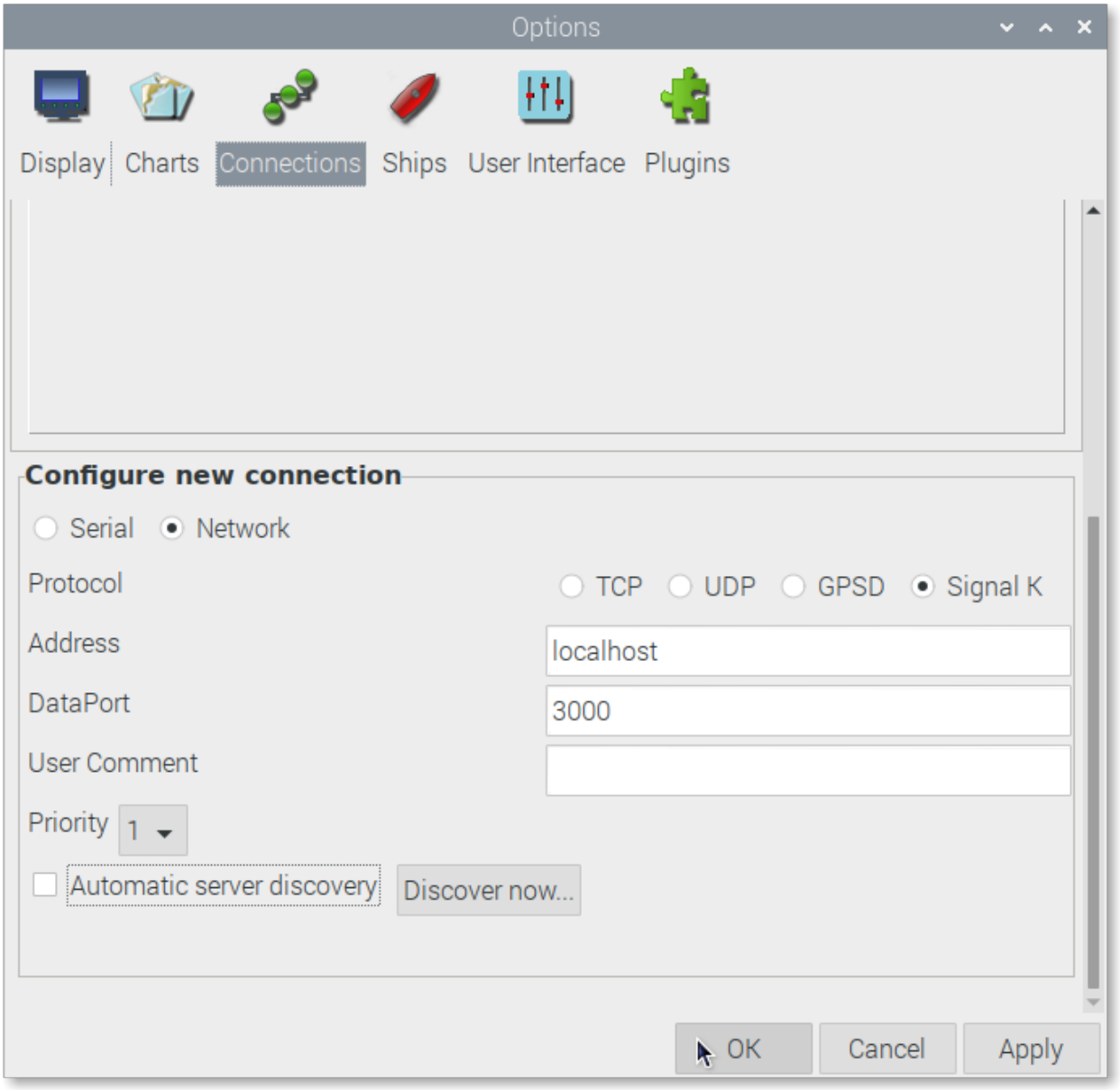
Go to the *Connections* tab and select the new device you just created. Click `Add to Signal K` and then click `AUTO`. A connection will be created on the Signal K server for your device.

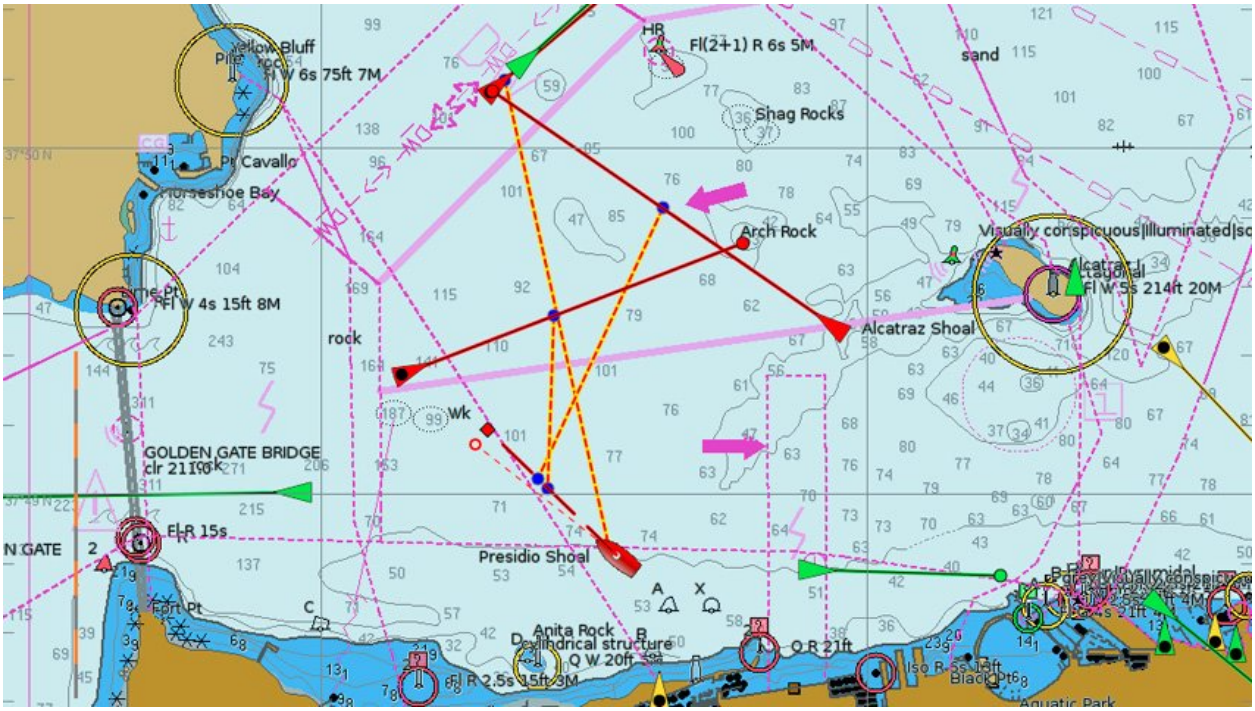

Serial 2.2.3
Help
Settings
UART
Refresh
Devices
Connections
Add to Signal K
Add to CAN Bus
Add to GPSD
Add to Pypilot
device /dev/
alias /dev/
data
connection
ID
bauds
ttyUSB0
ttyOP_maiana
NMEA 0183
Edit
Remove


Adding connection for device: ttyOP_maiana
Data: NMEA 0183
ID: maiana
Serial port: /dev/ttyOP_maiana
Baud Rate:
38400
Press AUTO to create a connection in Signal K using the settings above.
Press MANUAL if you need to add special settings.
To get data in OpenCPN, make sure this network connection exists in OpenCPN:
Protocol: Signal K
Address: localhost
DataPort: 3000
Cancel
MANUAL
AUTO



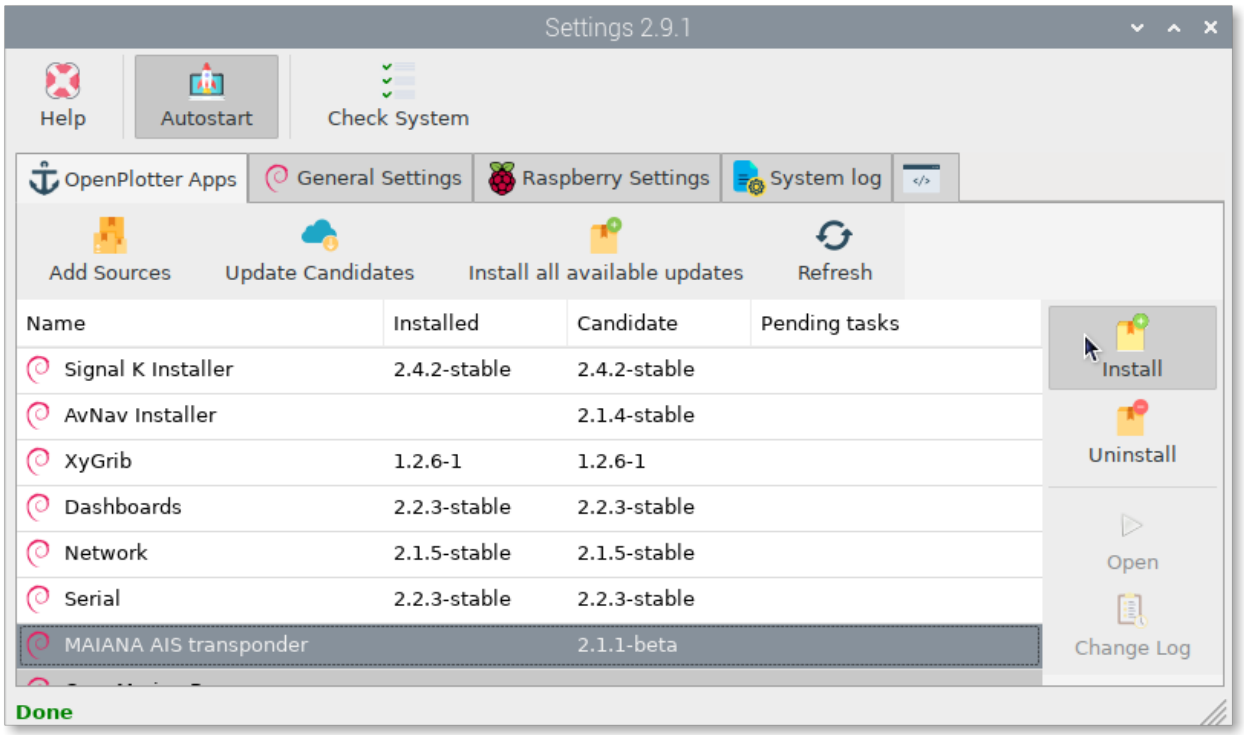
Make sure there is an OpenCPN enabled connection to the Signal K server and your are done.


Options
Display
Charts
Connections
Ships
User Interface
Plugins
Configure new connection
Serial
Network
Protocol
TCP
UDP
GPSD
Signal K
Address
localhost
DataPort
3000
User Comment
Priority
1
Automatic server discovery
Discover now...
OK
Cancel
Apply

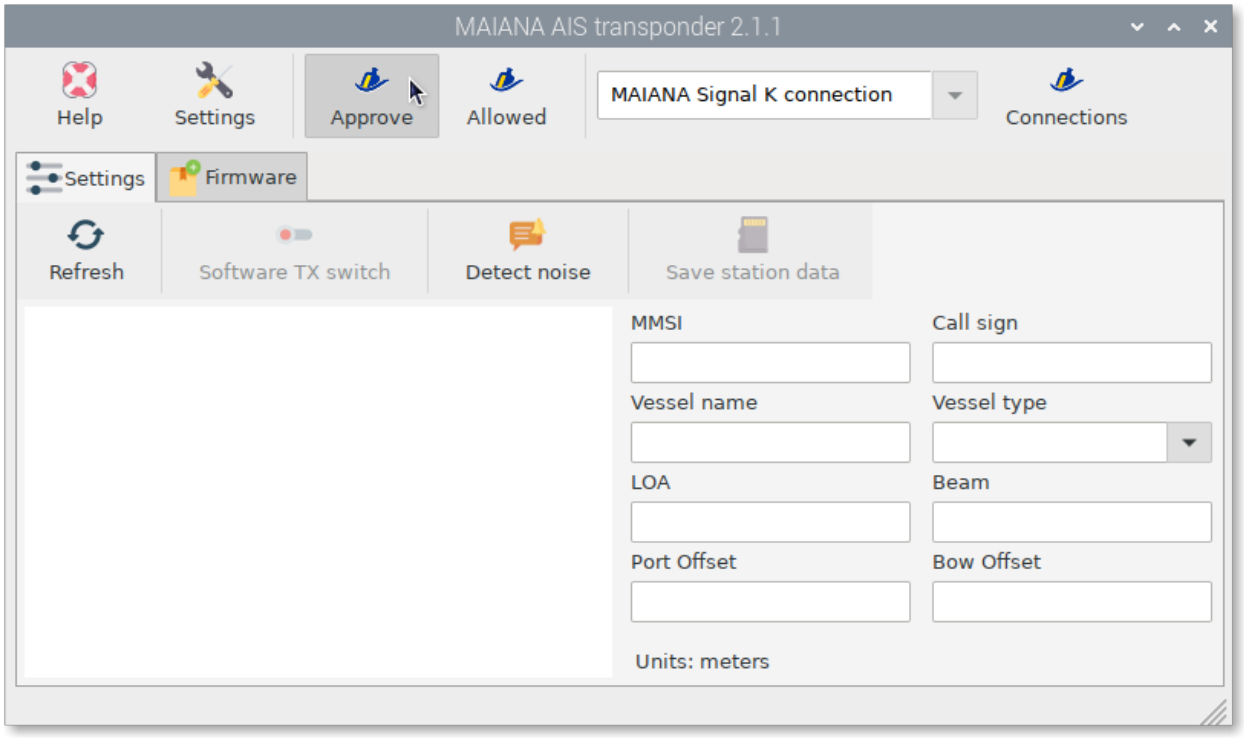


18.2 Connecting to MAIANA

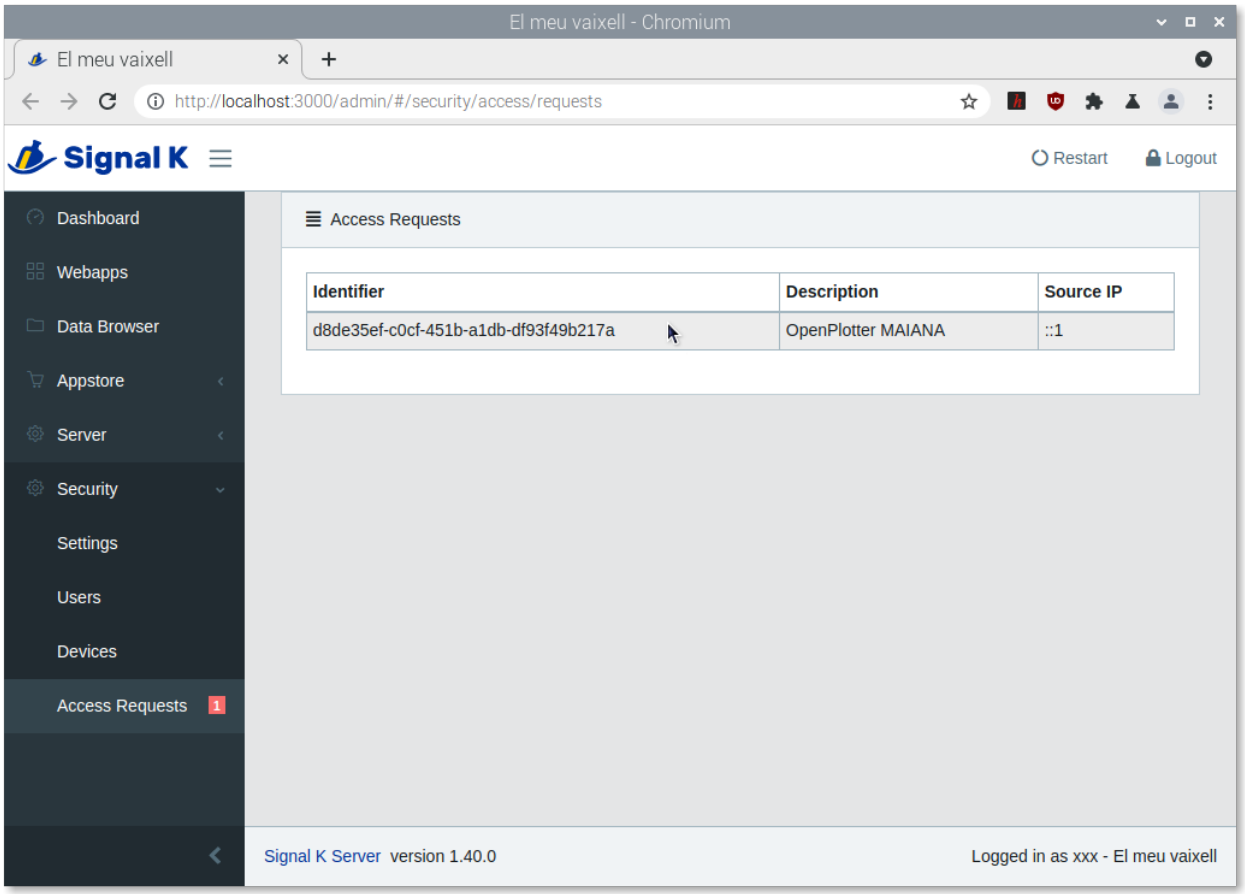
Using the *OpenPlotter MAIANA AIS transponder* app you can manage all the settings of your device. Open *OpenPlotter Settings* app, select this app and click `Install`.

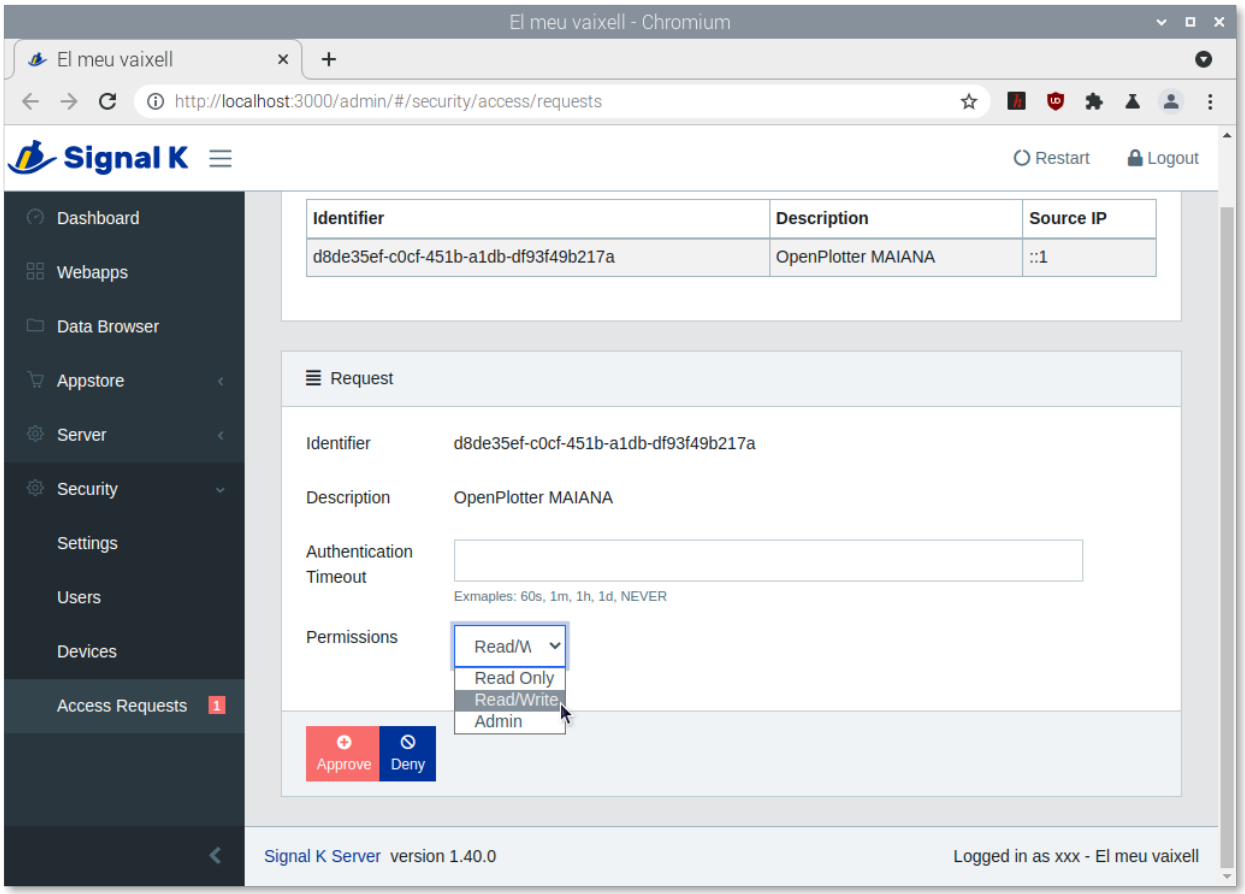


Once the *OpenPlotter MAIANA AIS transponder* app is installed, we have to create a connection between this app and the Signal K server. Open the app and a connection request will automatically be sent to the Signal K server. Click `Approve` to access the administrator of the Signal K server:

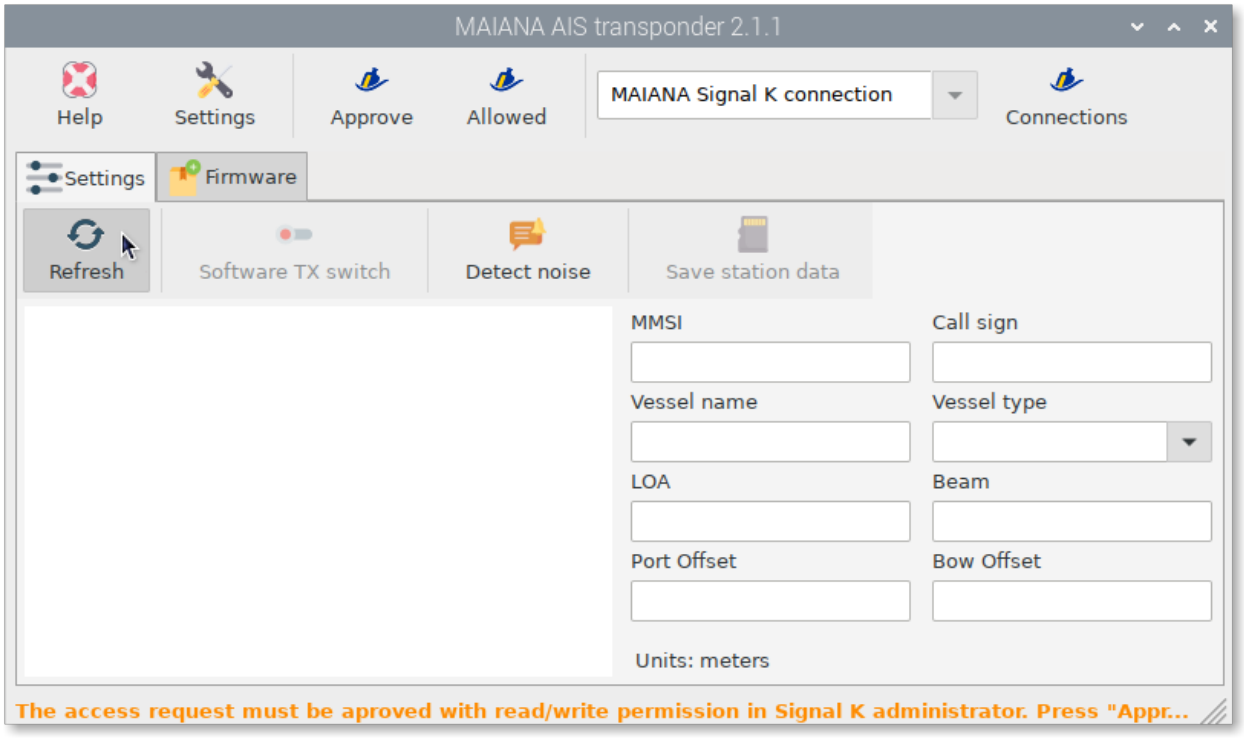


Select the new request and then select *Read/Write* in *Permissions* and click `Approve`:

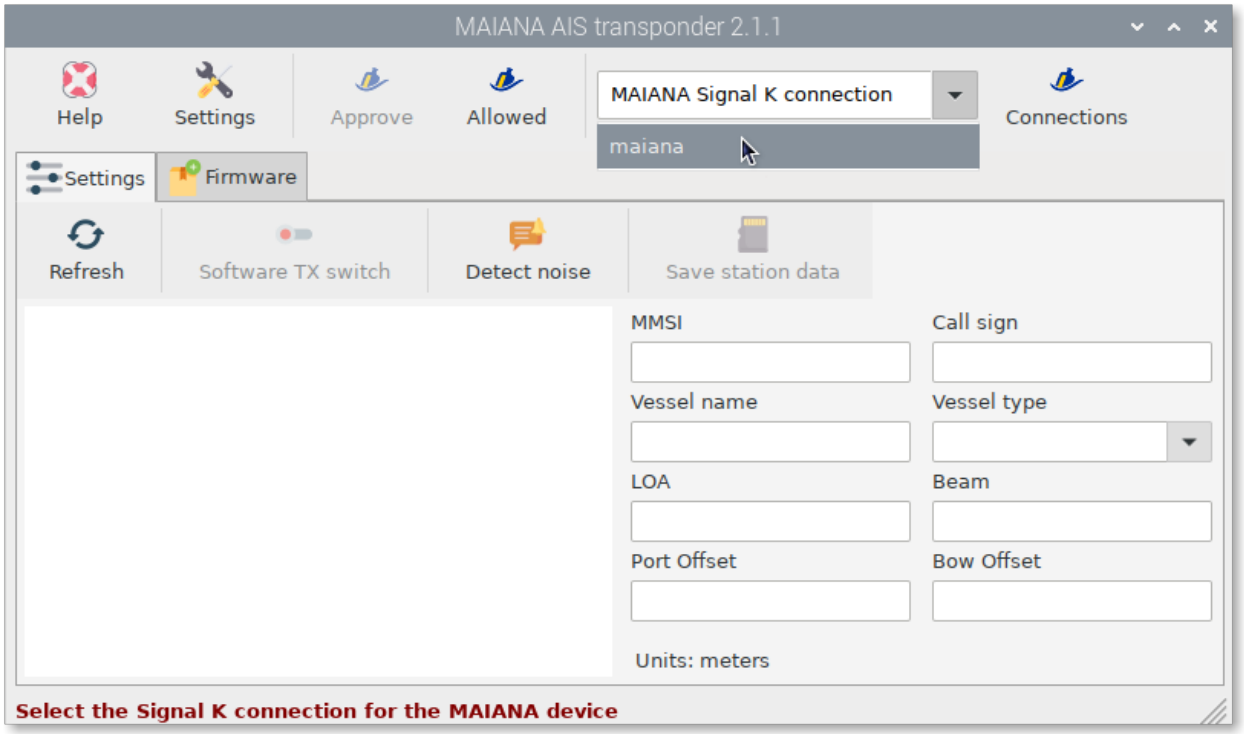

El meu vaixell - Chromium
El meu vaixell
http://localhost:3000/admin/#/security/access/requests
Signal K
Restart
Logout
Dashboard
Webapps
Data Browser
Appstore
Server
Security
Settings
Users
Devices
Access Requests
1
Access Requests
Identifier
Description
Source IP
d8de35ef-c0cf-451b-a1db-df93f49b217a
OpenPlotter MAIANA
::1
Signal K Server version 1.40.0
Logged in as xxx - El meu vaixell



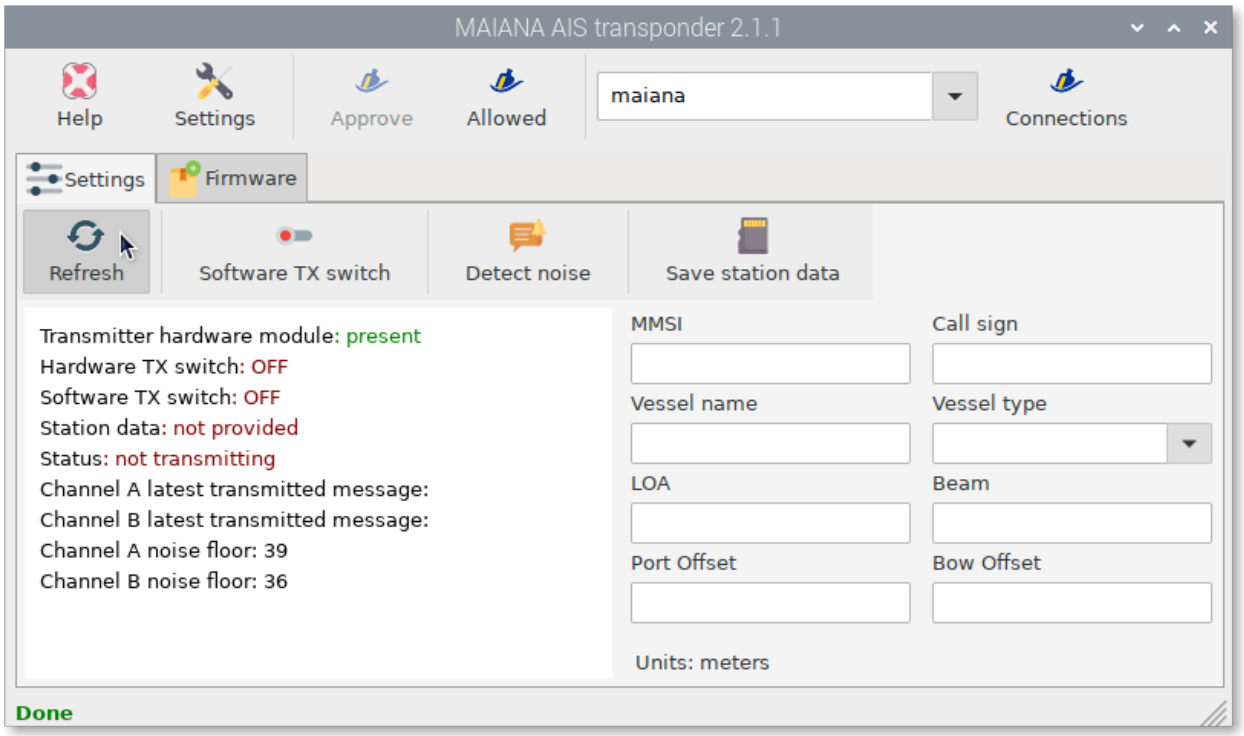
Go back to the *OpenPlotter MAIANA AIS transponder* app and click `Refresh`:



Now we have to select the connection we previously configured with the *OpenPlotter Serial* app by clicking on the *MAIANA Signal K connection* field:



And that's it. All connections have been made and you will be able to communicate with MAIANA every time you open the *OpenPlotter MAIANA AIS transponder* app and the device is turned on. If you can not get a connection the first time, try again by clicking `Refresh`.



18.3 Enabling transmission

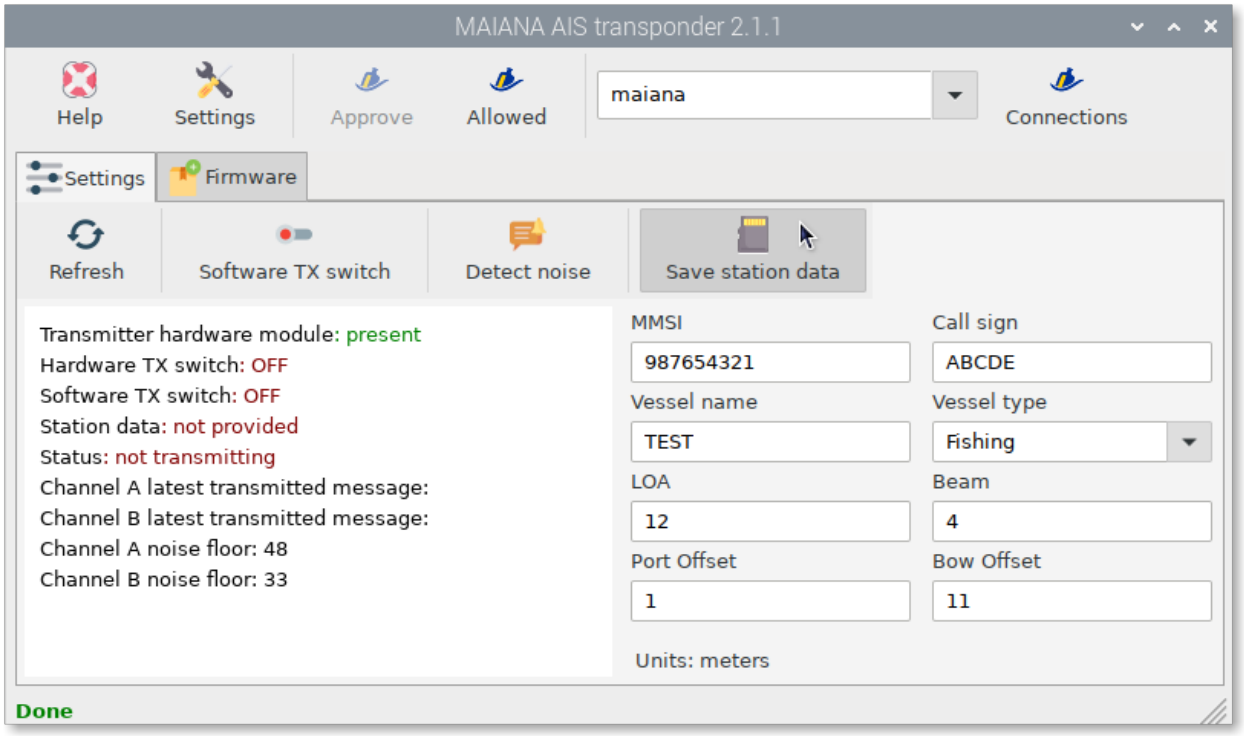
If we want to enable transmission, we must provide the station data. Complete the form using this syntax for each field:

- MMSI (you should have one for your boat already)
- Boat name (up to 20 alphanumeric characters, no punctuation. Use all caps)
- Call sign (may be empty if you don't have one)
- Type (this is the numeric type of the vessel, see below)
- Length in meters (integer only)
- Beam (width) in meters (integer only)
- Port offset (meters from the port side where the unit is located)
- Bow offset (meters from the bow where the unit is located)

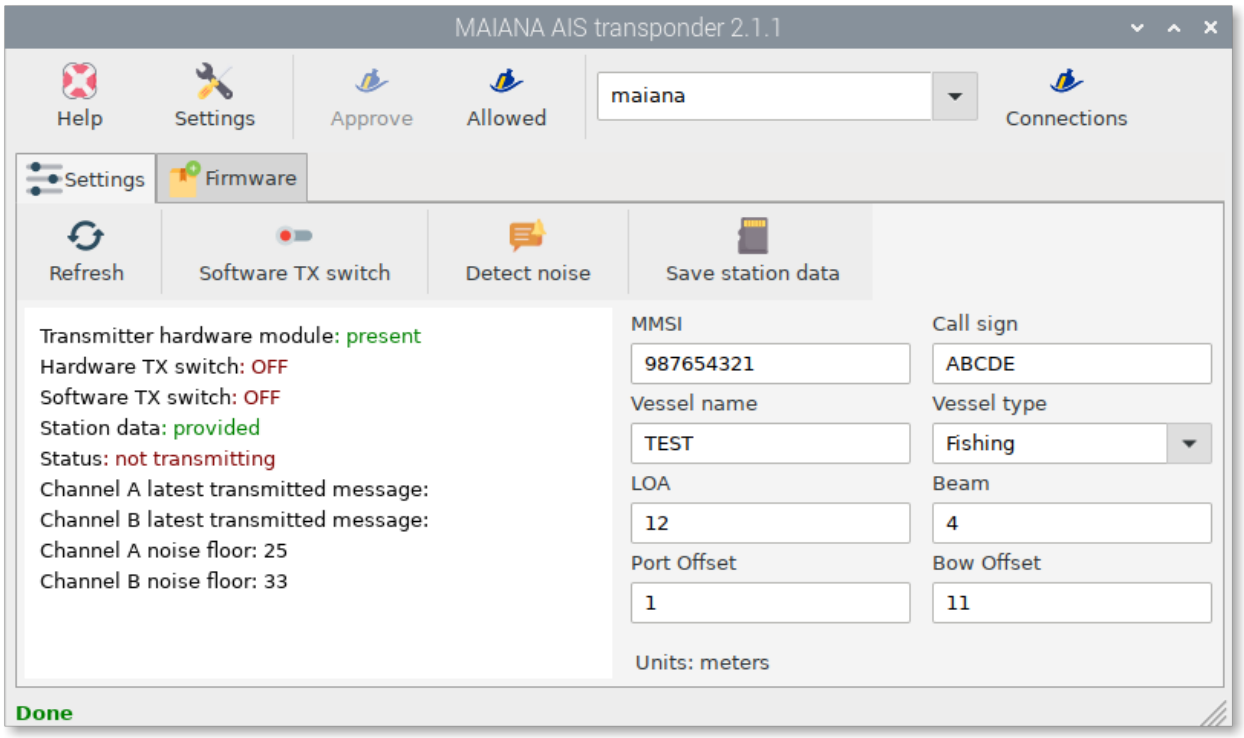
For vessel type, here are some numeric values that apply to class B transponders:

- 30 - Fishing
- 34 - Diving
- 36 - Sailing
- 37 - Pleasure craft

Click `Save station data` when you are done:



You will see that the value of *Station data* has changed to *provided* in green:



There are 2 switches to turn on/off transmission:

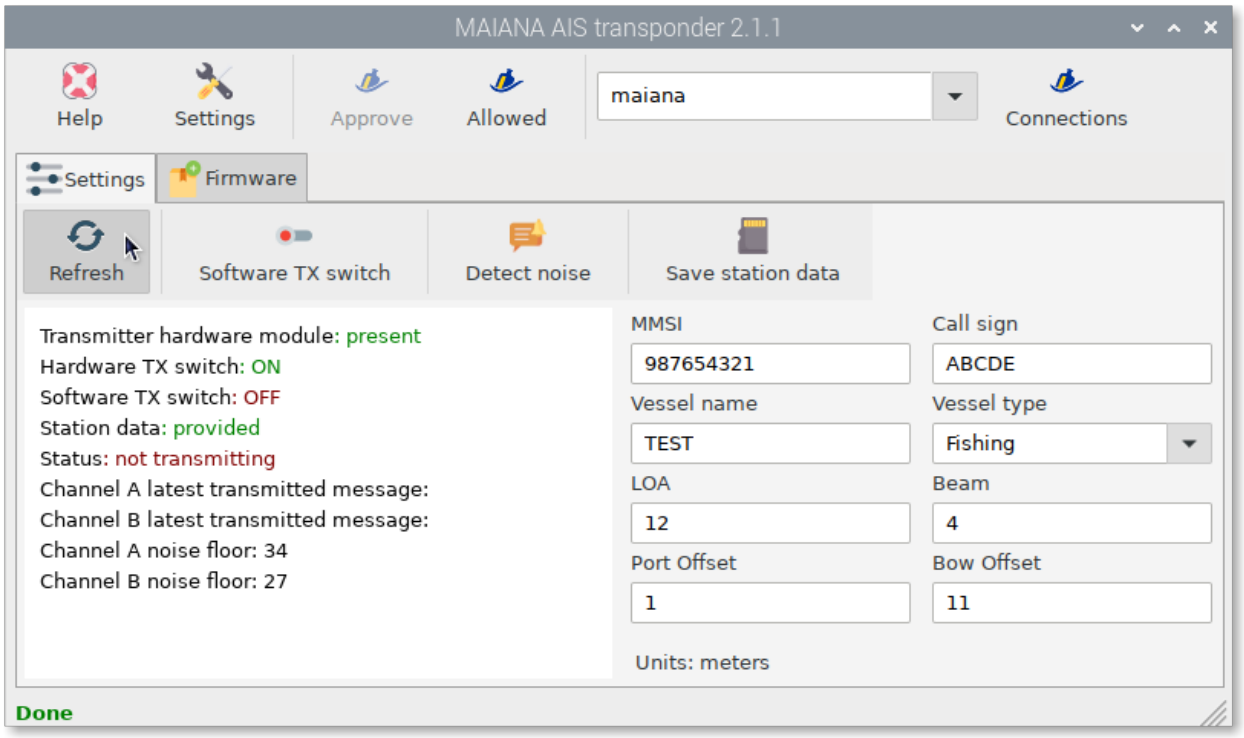
Hardware There is a physical switch on all adapters. The breakout board also has a pin for this. This switch has priority over the Software switch.

Software You will find a button *Software TX switch* in *OpenPlotter MAIANA AIS transponder* app.

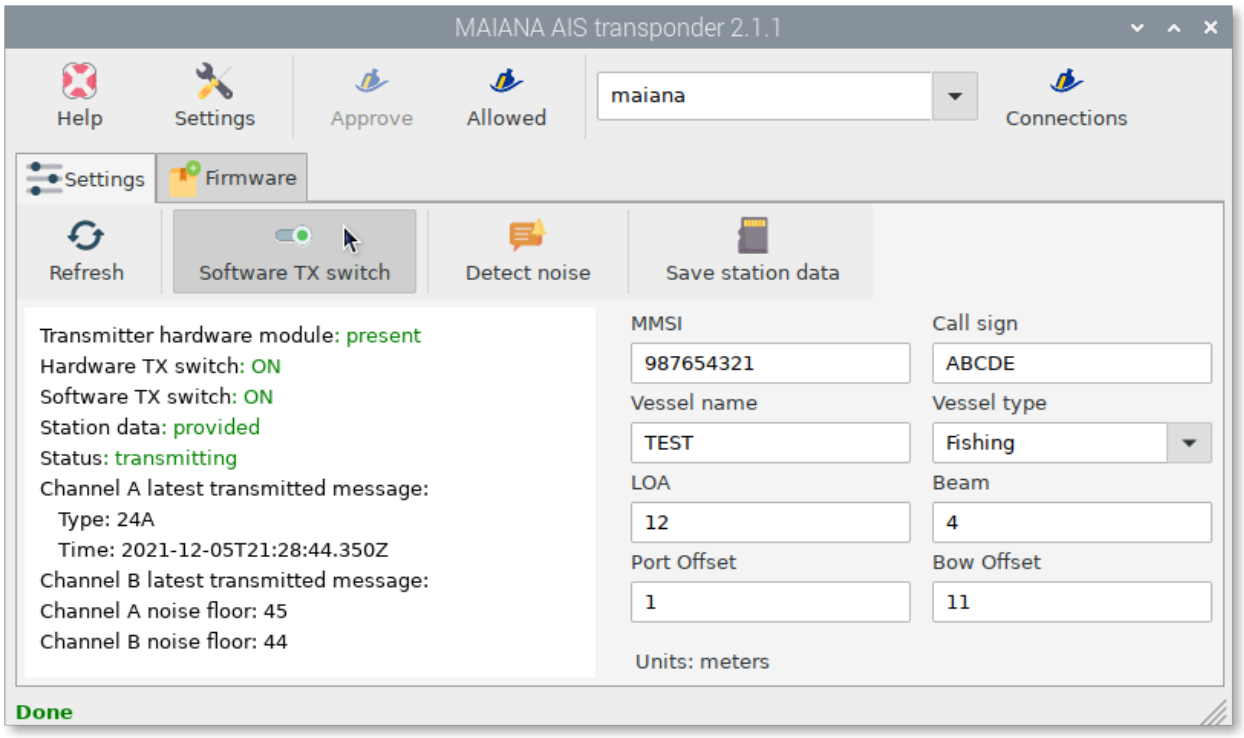
This is the relation between the two states of these switches:

Hardware	Software	TX
ON	ON	ON
ON	OFF	OFF
OFF	X	OFF

Turn on your Hardware switch and you will see that the value of *Hardware TX switch* has changed to *ON* in green:



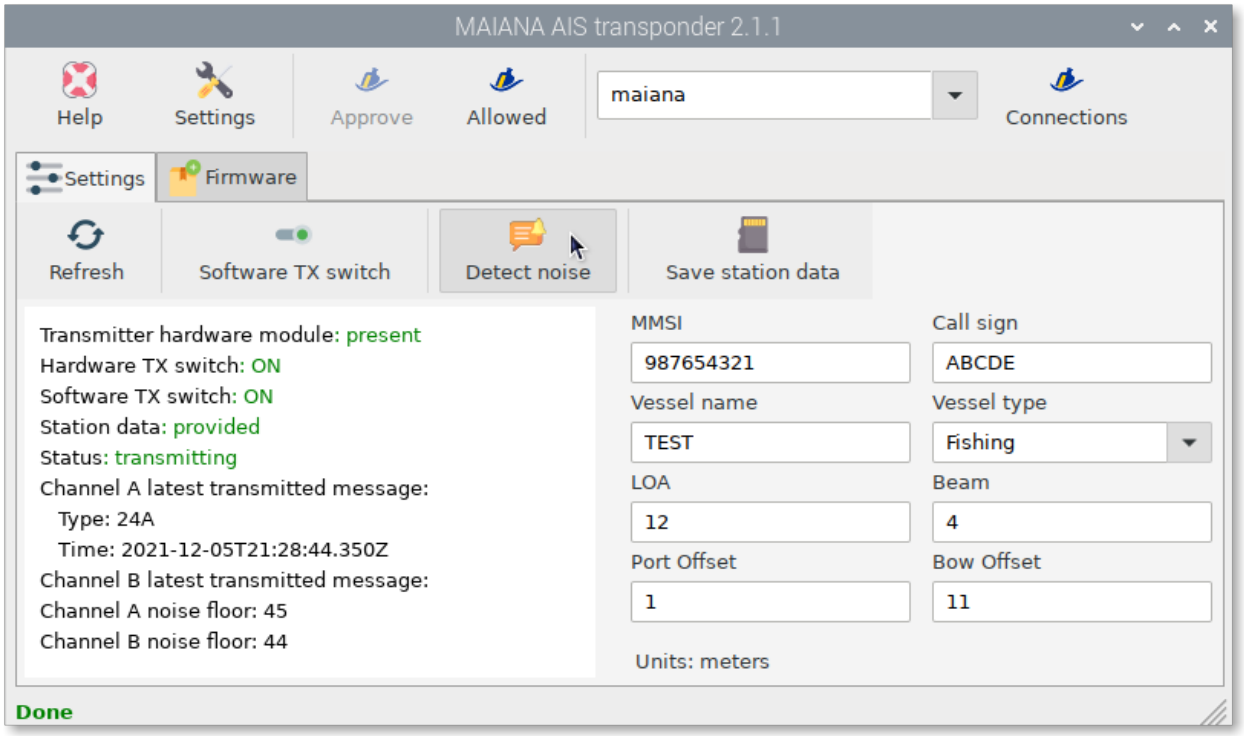
Now click `Software TX switch` and you will see that the value of *Software TX switch* has changed to *ON* in green and the value of *Status* has changed to *transmitting* in green:



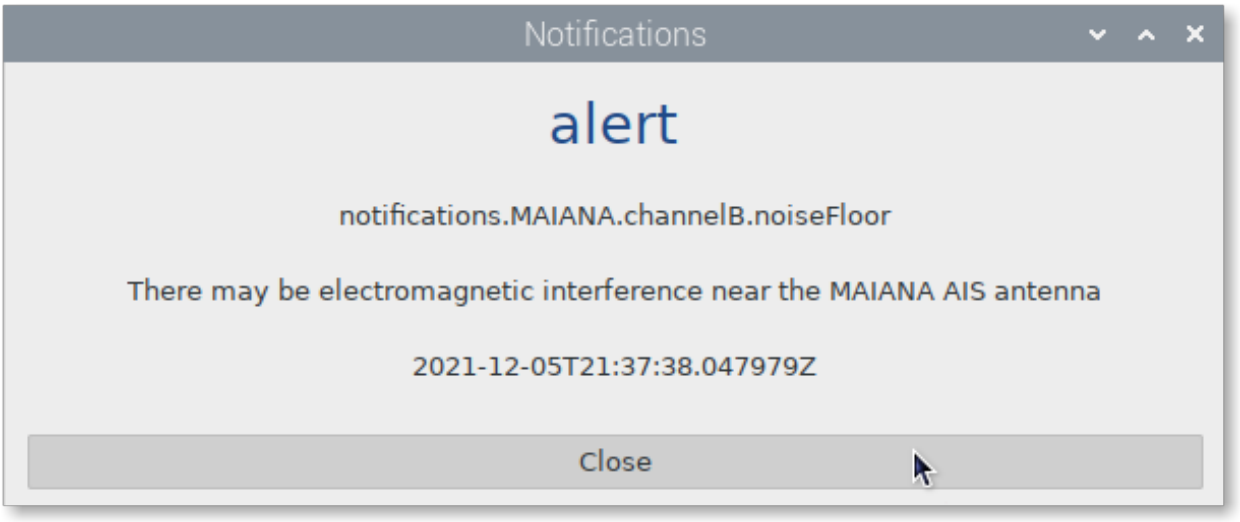
Congratulations, you are already transmitting!

18.4 Detecting EMI

MAIANA constantly checks for noise floor on both channels to detect any electromagnetic interference (EMI) near your device. If you enable `Detect noise` and the noise level is higher than 64, an alert notification will be sent to the Signal K server.

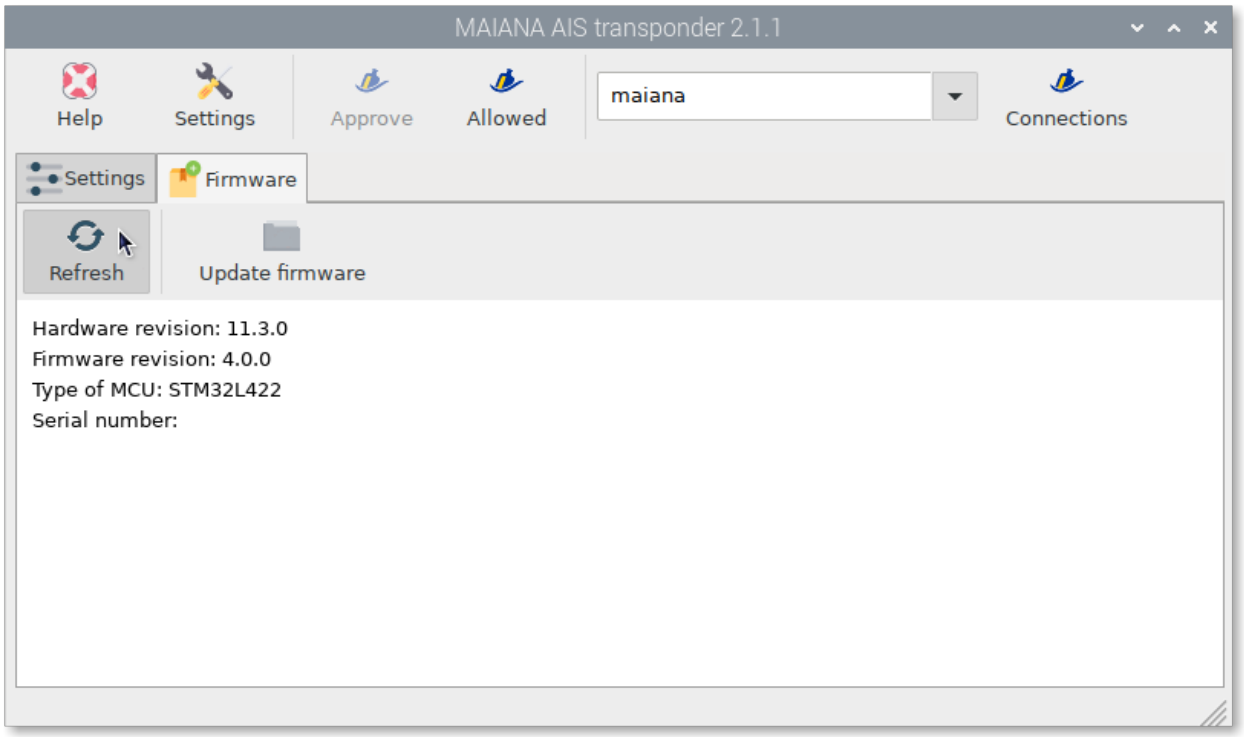


If you have the *OpenPlotter Notifications* app installed, you will see an alert window like this one:

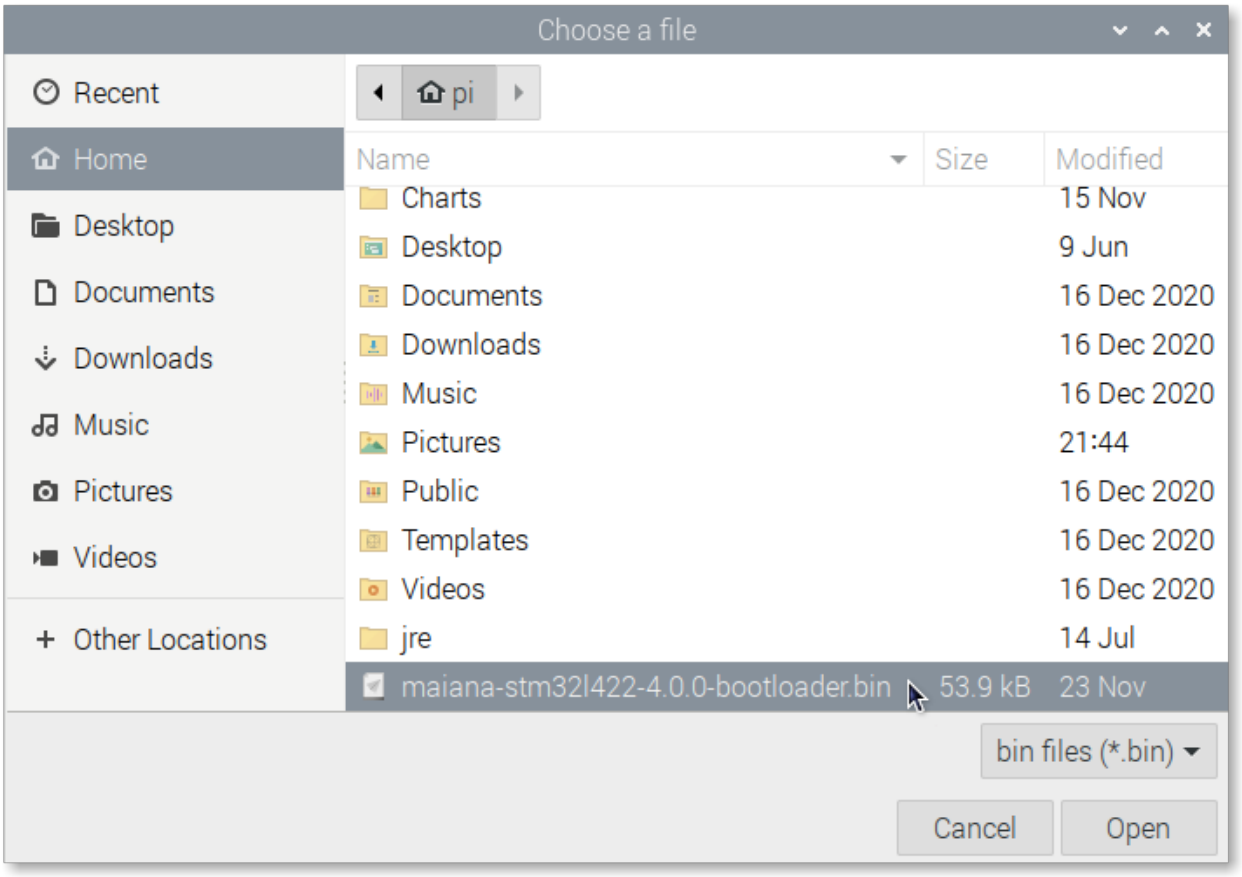


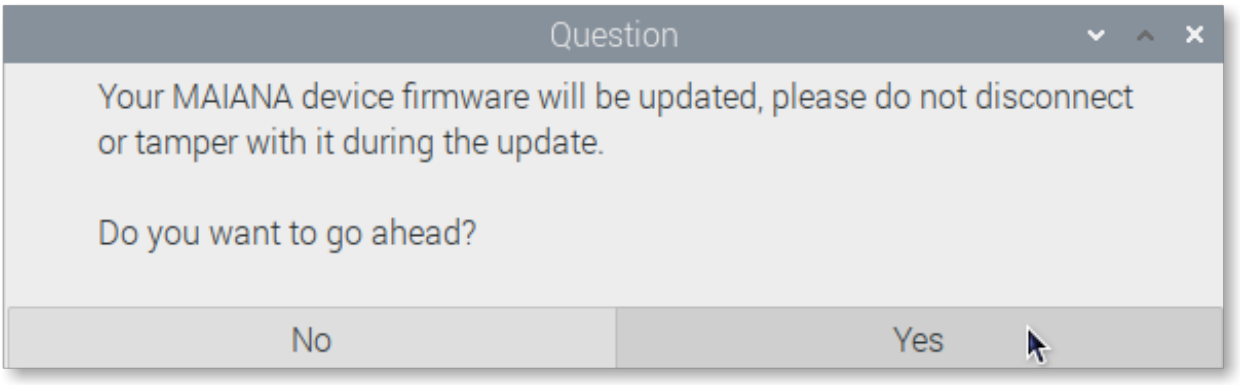
18.5 Updating firmware

You will receive your MAIANA base kit with the latest stable firmware installed. Go to the *Firmware* tab and click `Refresh` to see the version of your device:

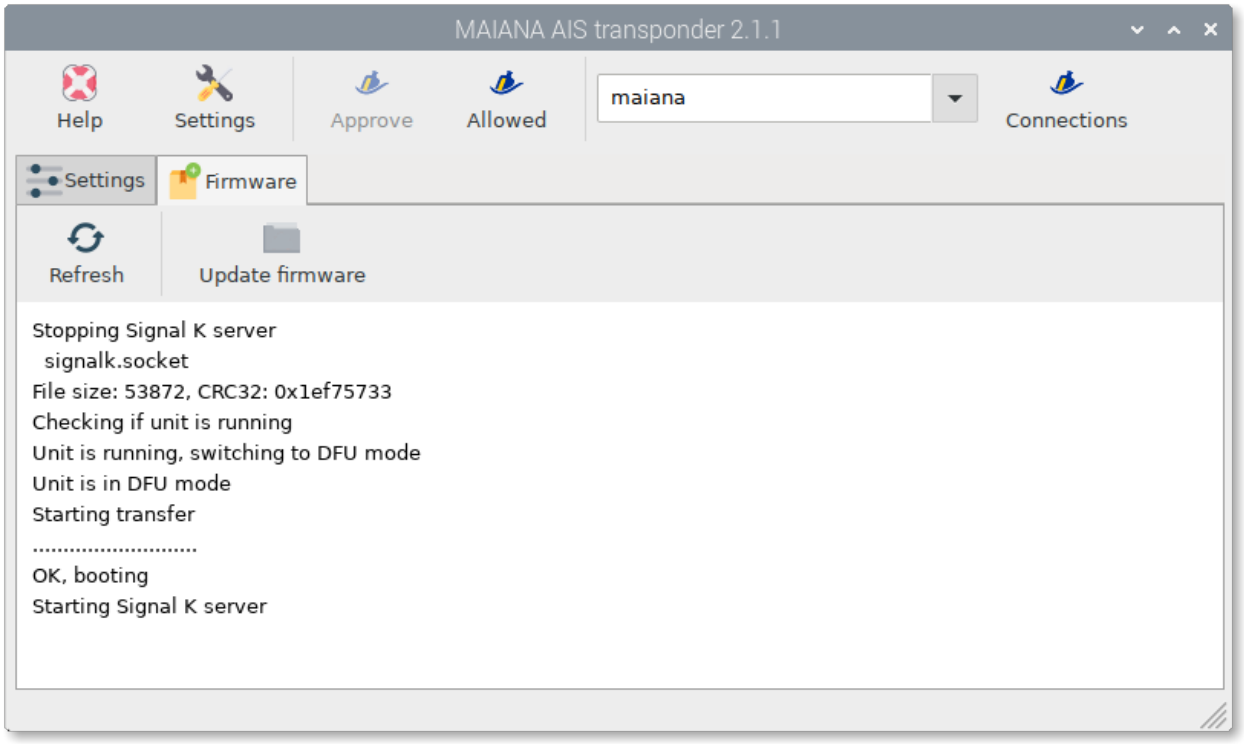


If a new version of the firmware is released by MAIANA developers, download the bin file and click `Update firmware`. Select the file, click `Open` and finally `Yes`:





The system will stop the Signal K server to make sure it can take control of the device and load the new firmware. When done, both the Signal K server and the device will reboot:

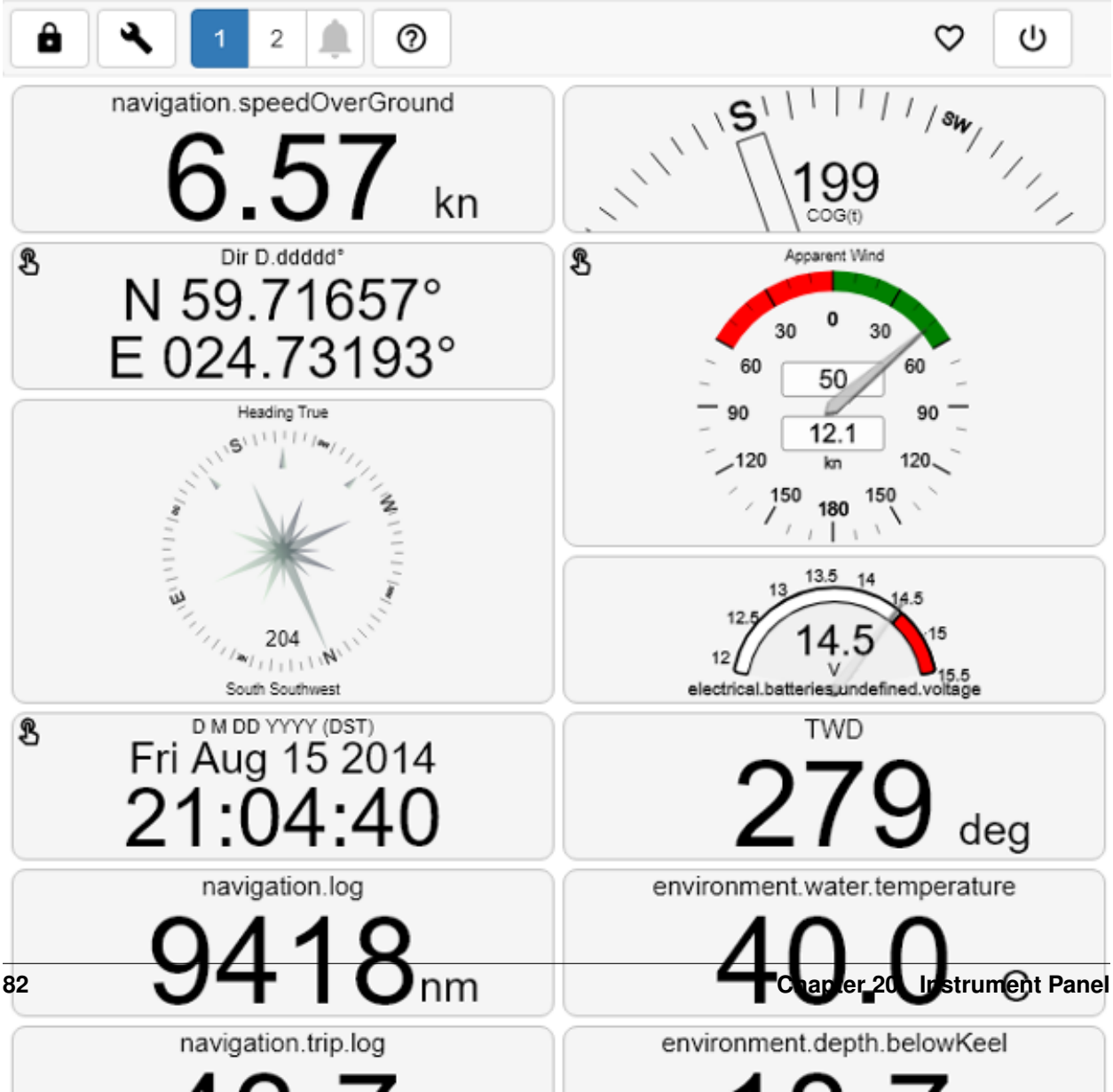

MAIANA AIS transponder 2.1.1
Help
Settings
Approve
Allowed
maiana
Connections
Settings
Firmware
Refresh
Update firmware
Stopping Signal K server
signalk.socket
File size: 53872, CRC32: 0x1ef75733
Checking if unit is running
Unit is running, switching to DFU mode
Unit is in DFU mode
Starting transfer
.....
OK, booting
Starting Signal K server

CHAPTER 19

Dashboards

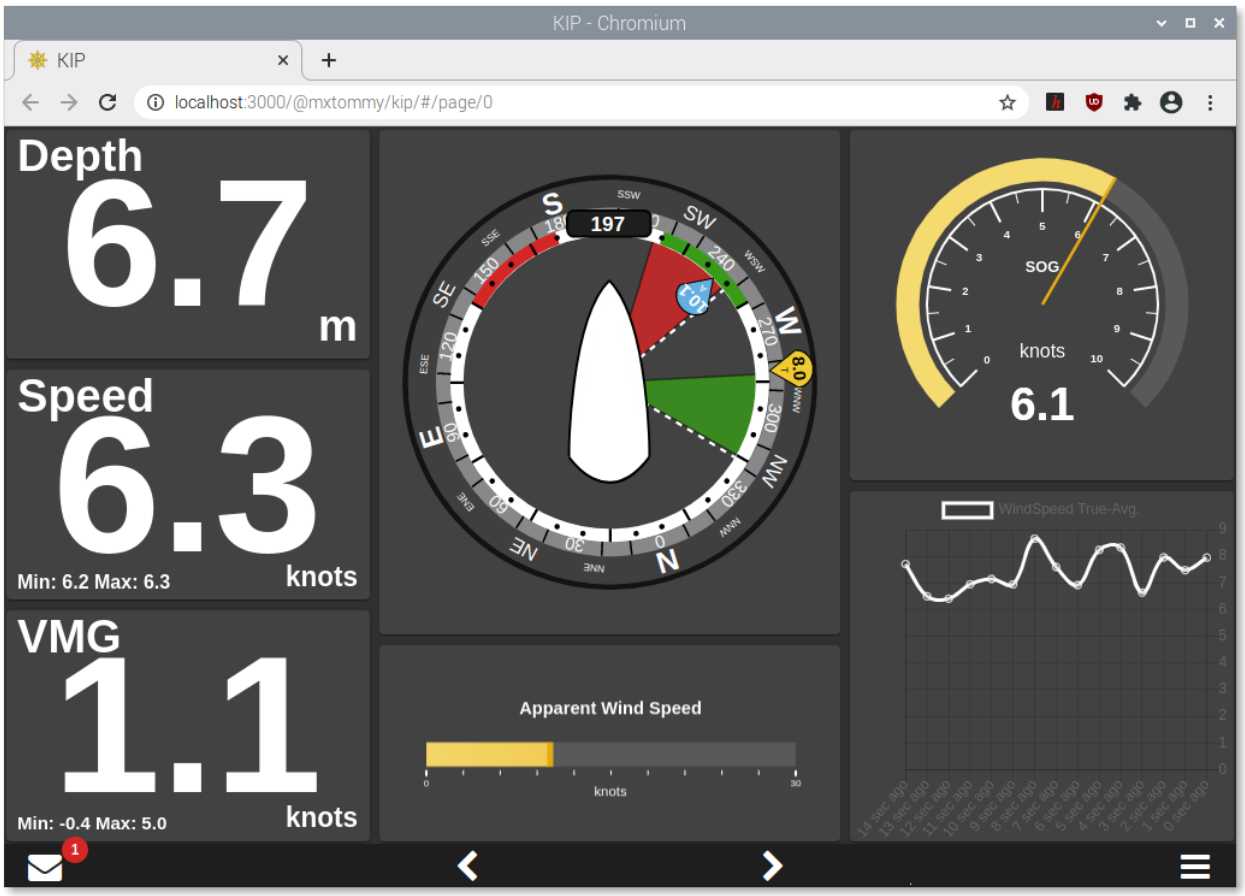
CHAPTER 20

Instrument Panel



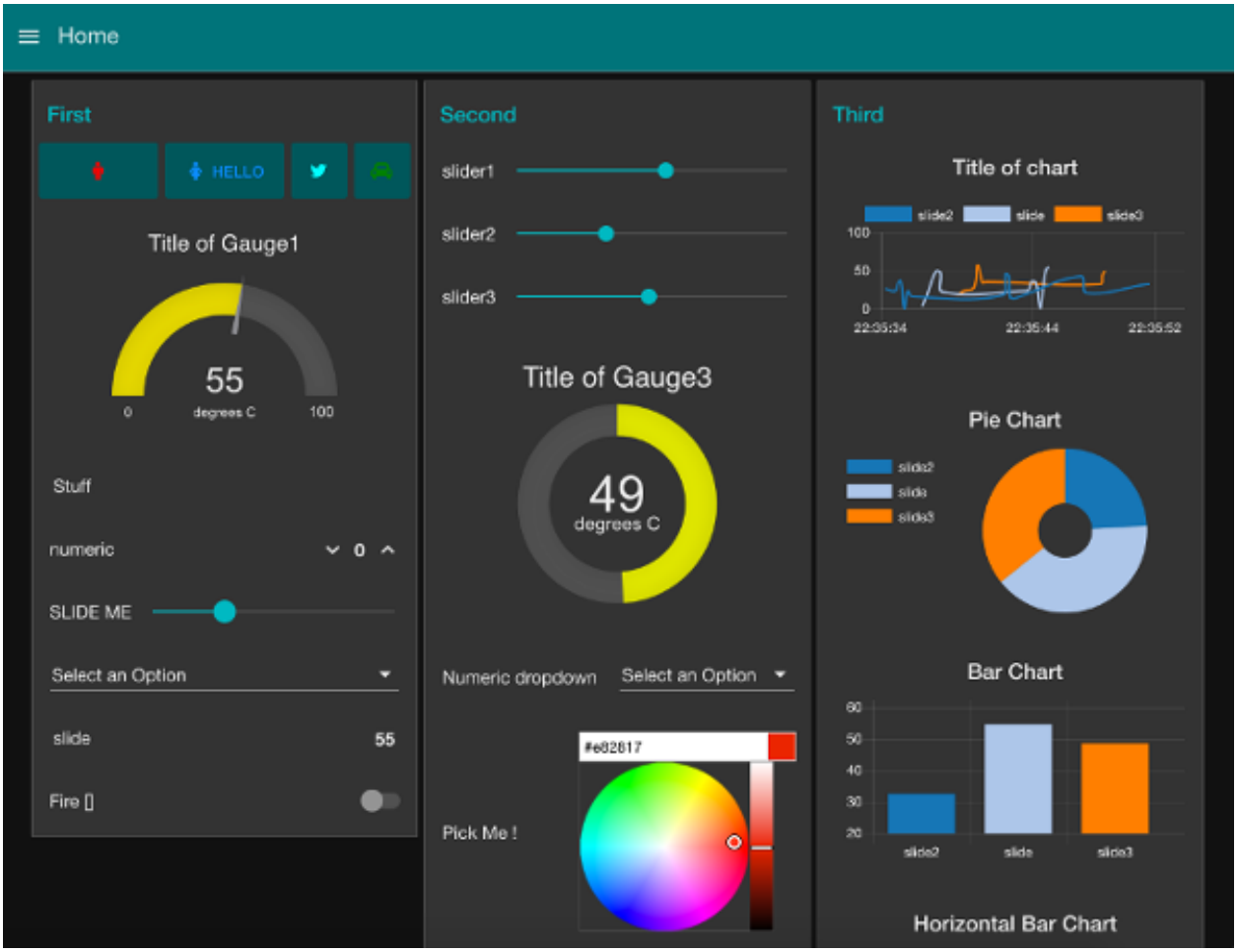
CHAPTER 21

KIP



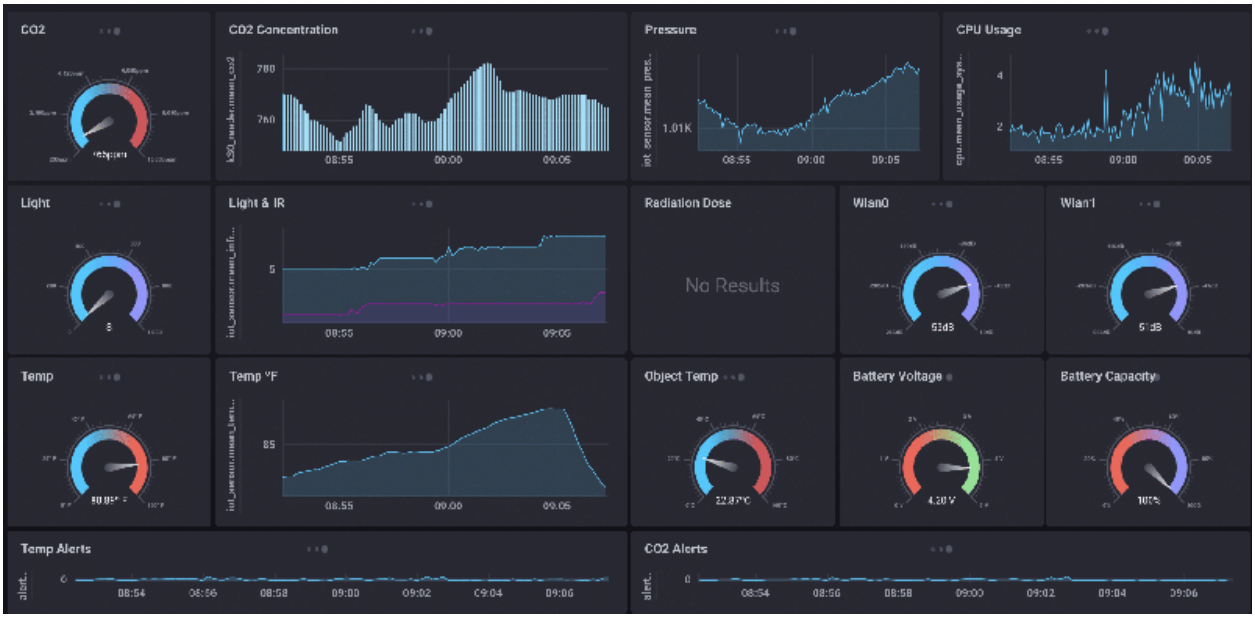
CHAPTER 22

Node-Red Dashboard



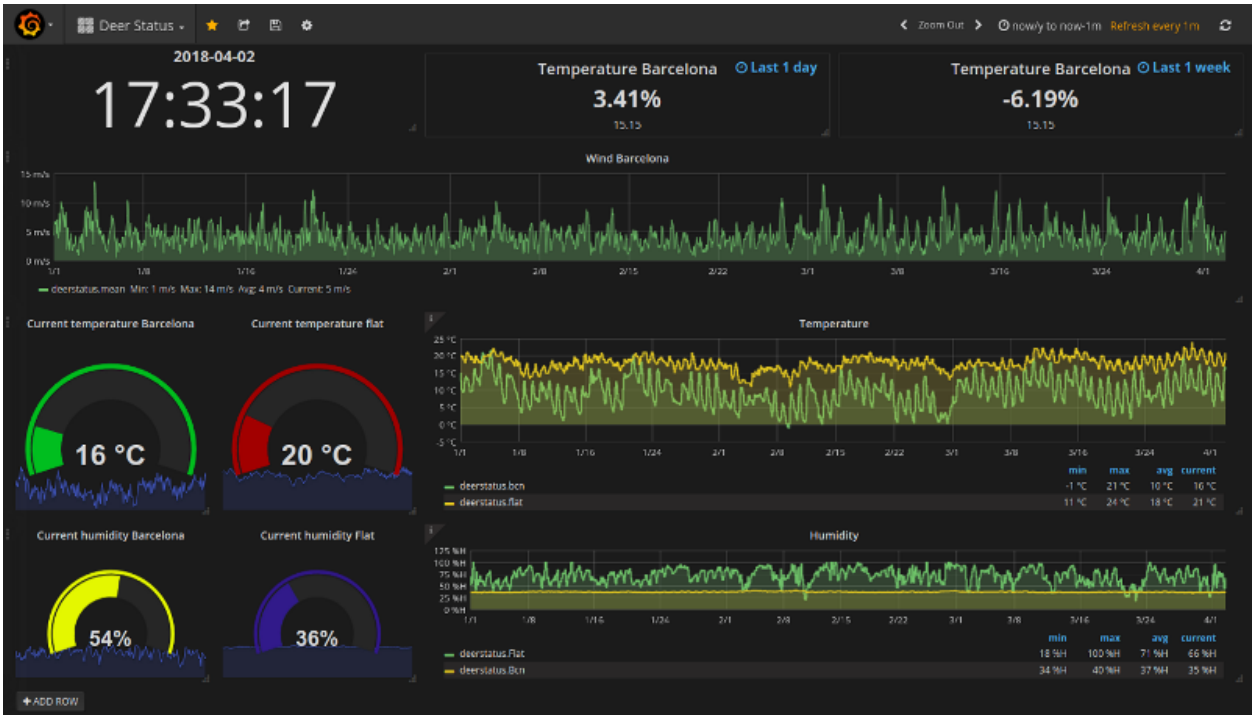
CHAPTER 23

Influxdb 1



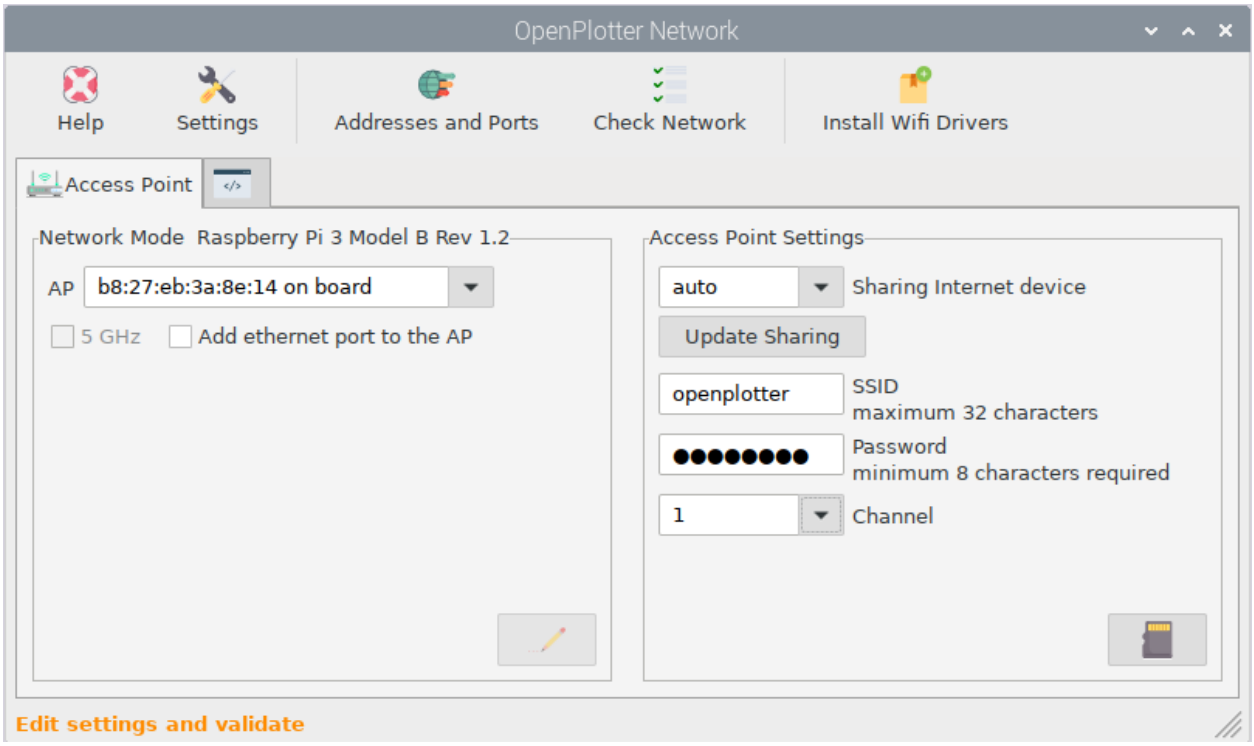
CHAPTER 24

Grafana



CHAPTER 25

Network



Picture: network app

25.1 Features

- Access Point (AP)
- AP 2.4 GHz or 5 GHz

- Router functionality (eth0 and AP on the same subnet)
- AP and station mode (not recommended, not stable)
- Zeroconf functionality
- Driver collection for special Wifi devices
- Individual settings
- Android USB tethering and reverse tethering
- IOS USB tethering when IOS is connected to the internet with GSM

The Network management is based on the native management system of Debian. You can connect to a hotspot as you do without openplotter-network.

25.2 Access Point (AP)

You can configure the Linux system to act as an AP (wifi-hotspot). Then you can connect to OpenPlotter with wifi devices like notebooks, tablets, smartphones, e-book reader,...

To setup a hotspot you first select the wifi device you want to act as AP (see picture network app on the left side behind AP). *We recommend to use the Raspberry on board wifi.* (In V2.x the AP will always have the interface name wlan9)

25.3 AP 2.4 GHz or 5 GHz

If your AP and the devices can work with 5 GHz you can select it.

25.4 Router functionality

On your internet router at home you can communicate on the same subnet between your ethernet connected devices and your wifi connected devices. If you wish that Linux treats the AP and ethernet port this way choose Add ethernet port to the AP.

Warning: When setup as router don't connect the ethernet port to a router! The router can malfunction, including communication paralyzes.

25.5 AP and station mode

The Raspberry built-in wlan device can act as AP and connect as a standard client (station mode) to the AP of the marina simultaneously. This does work but isn't stable. We recommend using an USB-wlan device to connect to the AP of the marina. Or connect an Android smartphone to the Raspberry with an USB cable. Connect the smartphone to the marina wifi and turn on USB-tethering.

25.6 The Zeroconf functionality

With Zeroconf devices can connect to each other without having a dns-server or knowing the ip-address. Use your hostname and add the postfix ".local" to it.

25.7 Driver collection for special Wifi devices

If your wifi stick isn't recognized by Linux, you can try to install an extra drivers by pushing the "Install Wifi Drivers" button.

25.8 Individual settings

You have to give your AP a SSID (a name). This will pop up on smartphones etc. When they search for a wifi AP (hotspot). Give your wlan a secure "Password"

Remember to set your wifi country correct (Settings->Raspberry-Pi-Konfiguration) .

Note: Not all selectable channels will work depending on the country setup.

You can choose a channel you like.

Note: In every marina the wlan traffic can be different. If you have network issues you can sometimes solve them by changing the channel.

Sharing Internet device is only important for any device that is connected to the Raspberry and needs internet connection. An incorrect setting makes no difference for the Raspberry itself. *We recommend setting "Sharing Internet device" to auto.* Because Linux changes often wlan0 and wlan1.

The button *Update Sharing* changes the internet connection to the first internet gateway in the list of the command *route*. This is important during the journey when switching the internet connection between devices.

If you have more than one internet connection. Linux will only use the one with the highest priority (lowest metric). This depends on the interface type and the number. In other words disconnect other internet connections and press the *Update Sharing* button

25.9 Android USB tethering and reverse tethering

In the section AP and station mode we already spoke about using the Android feature tethering. But you can do more with tethering. You can use the realvnc app to remote control the Linux desktop with your smartphone or tablet also if there is a wifi issue. The Raspberry will listen on the ip-address 192.168.42.10 . The original USB-tethering idea (providing a GSM internet connection) can also be used.

(The interface name will be USB0)

Note: Android USB-tethering: Every time you restart Linux or reconnect the cable you have to switch on USB-tethering again (on some devices).

25.10 IOS USB tethering when IOS is connected to the internet with GSM

On IOS you can use the remote control with the vnc viewer only if you have a gsm internet connection. Go to settings switch personal hotspot and connect the USB-cable. The rapberry will listen on the ip-address 172.20.10.3 . (The interface name will be eth1)

Advantages of USB-tethering

- Emergency replacement if your display is broken
- If your mouse or keyboard does not work
- If your network does not work
- In headless use

25.11 Here are some examples how to configure the network with OpenPlotter

Use Raspberry as router to connect a notebook or a plotter with an ethernet cable. Use the internal wlan as AP and station.

Picture 1: the RPi works like a router (AP) for your tablet or smartphone and gets Internet

Advantage

- Less power consumption.
- A free USB port.

Disadvantage

- unstable
- Lower download performance.
- The Raspberry must be in a good place to get a good internet connection (unrealistic).

Picture 2: Same as Picture 1 but with a second WIFI device. *recommended*

Picture 3: Same as picture 2 but with an Android smartphone connected by USB as a replacement for the WIFI device. The smartphone can be connected to the marina WIFI or to GSM internet. USB tethering must be activated.

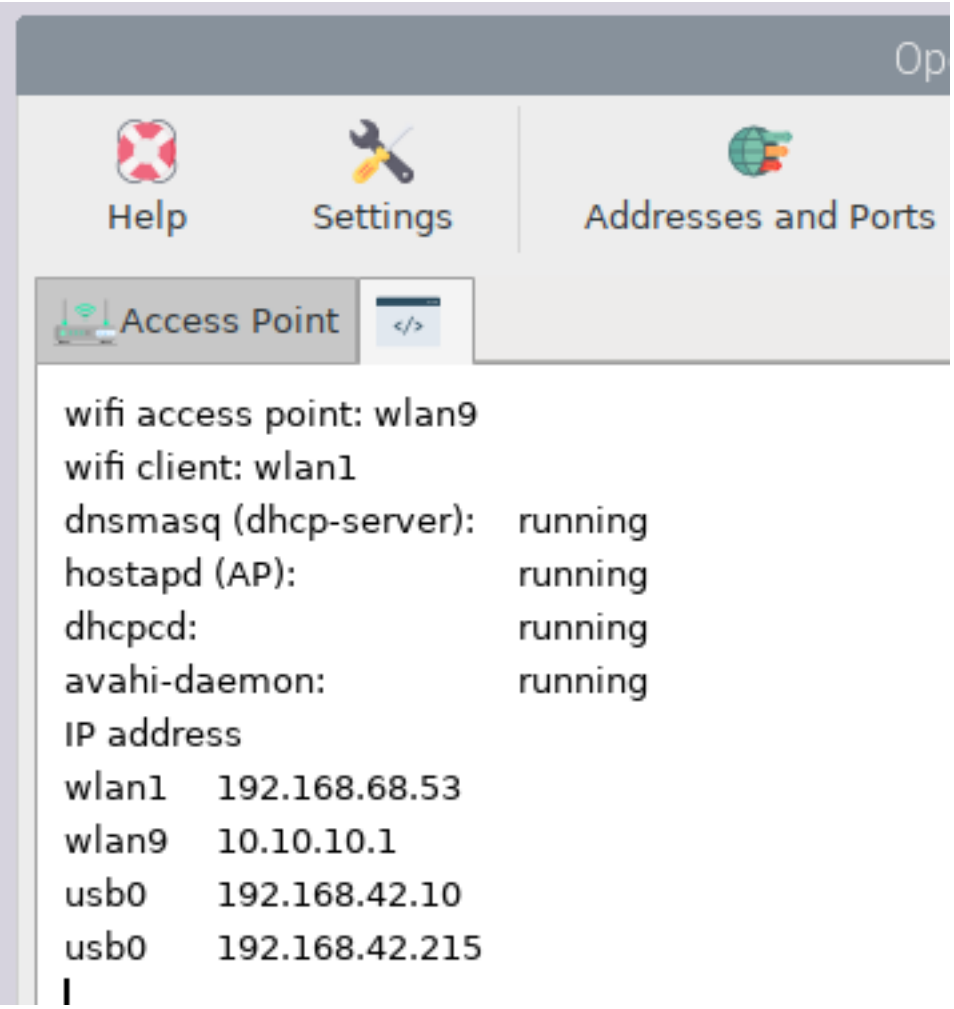
Advantage

- You do not have to change the network mode when you get to a marina or leave it (Android switches automatically to GSM if it looses the wifi connection).
- One device for gsm and wifi
- Some marina AP aren't Linux friendly but will work good with Android

Disadvantage

- Tethering doesn't start automatically on some devices
- The USB-cable disturbs
- It can eat up your complete mobile internet volume

25.12 Check Network



Picture 4: Shows the network status (how it should look if everything is okay + usb0 USB-tethering Android).

CHAPTER 26

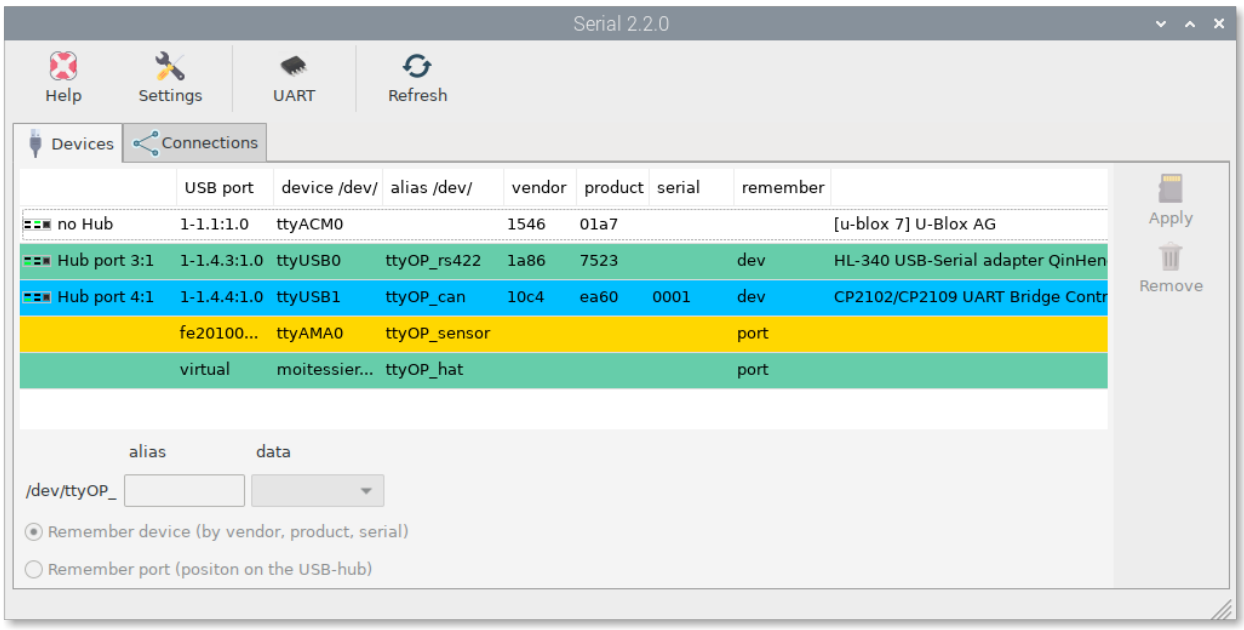
Setting devices

When you connect a USB device or any serial device to Linux, it is named as: /dev/ttyUSB0, /dev/ttyACM0, /dev/ttyS1... If any program needs to get data from this device, you have to provide this name in the settings of the program. But there is a problem, this name is not tied to your physical device, so it could be that the system gives it a different name on the next reboot and your program points to a wrong name.

This `Serial` app allows you to define an alias for your device that will always be tied to it and will facilitate the configuration of some programs to obtain data from it.

This app will detect any serial device connected to the system. Press `Refresh` when connecting or disconnecting a device to update the list of detected devices.

In the image below you can see some devices in different colors:



white not set

green set as NMEA 0183

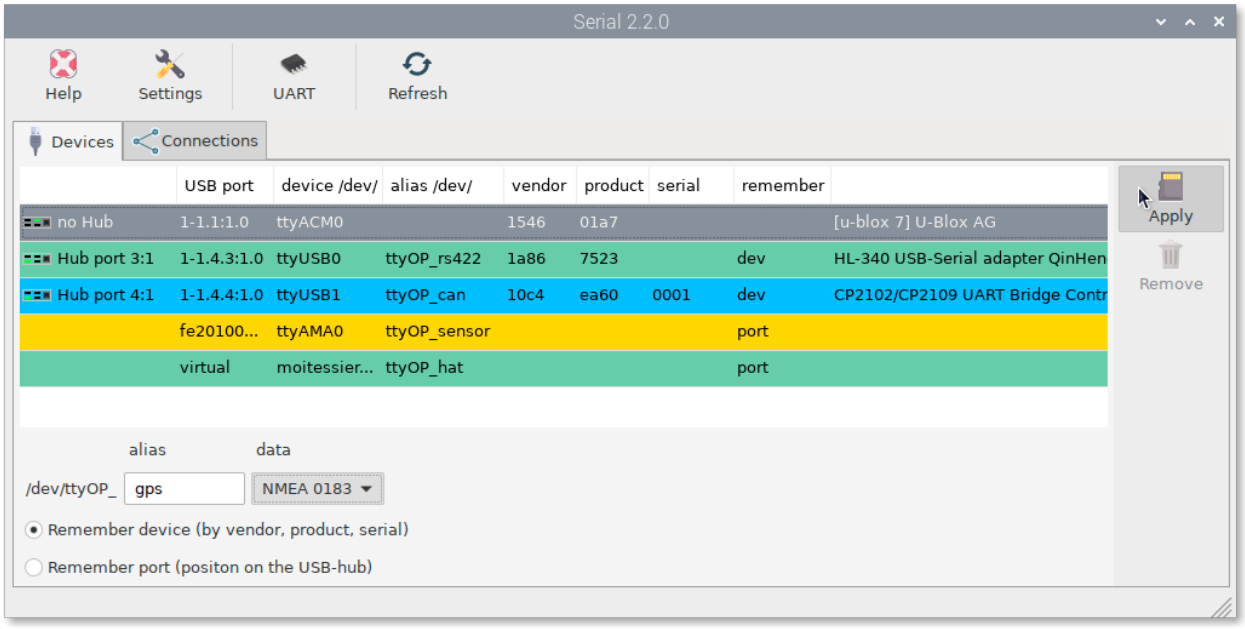
blue set as NMEA 2000

yellow set as Signal K

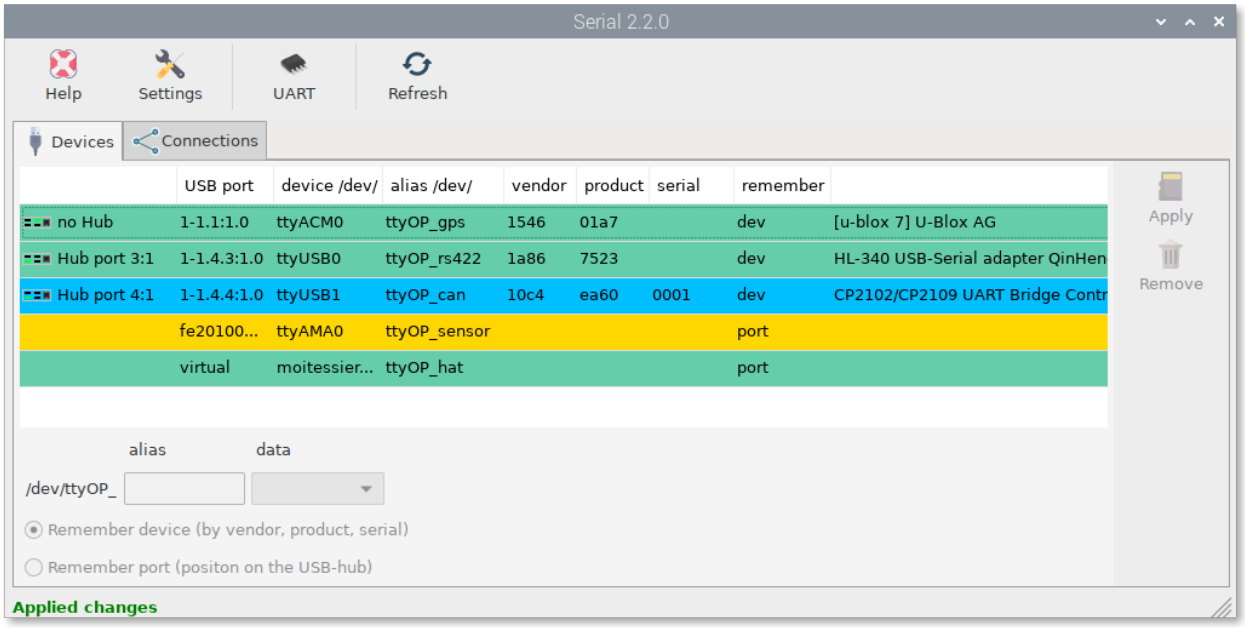
red the device is missing

To see how this works we are going to configure a USB GPS receiver. Select the device and enter a name for it in the `alias` field. Select the type of data that flows through the device (NMEA 0183 in this case) and finally select whether the system should remember the device or the position of the USB port where the device is connected.

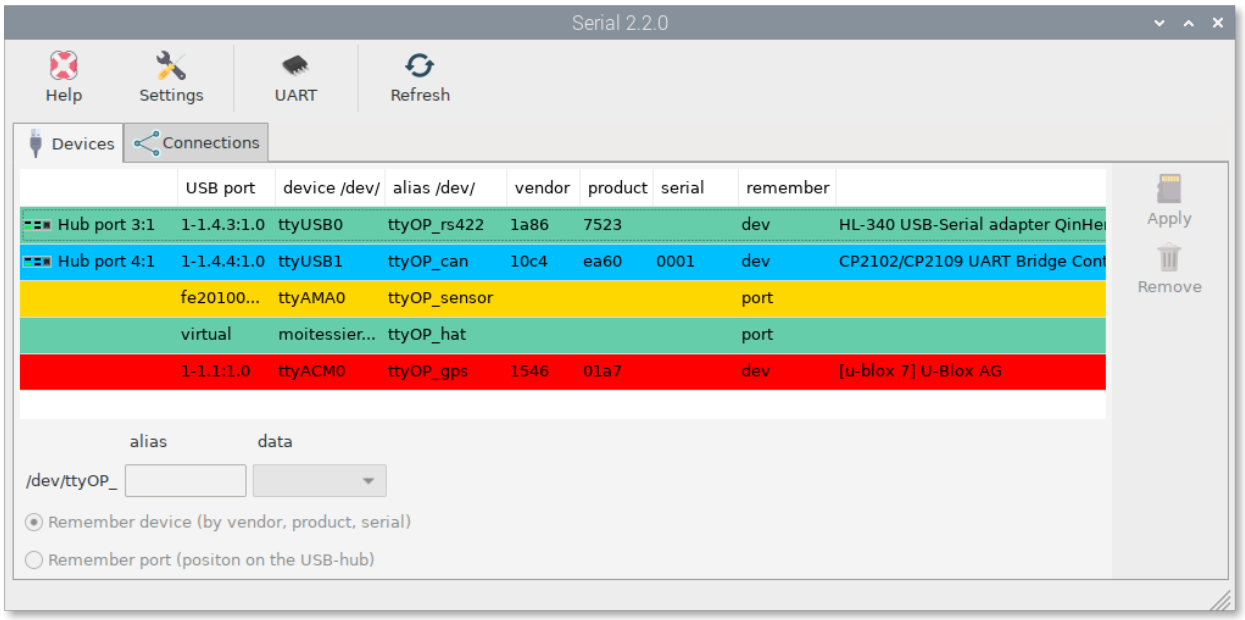
You should use `Remember port` only if 2 or more of your devices have the same vendor, product and serial or if they do not have any of these identifiers at all. For Raspberry Pi, the first column in the list will show you which USB port your device is connected to and if you are using a HUB.



Press `Apply` when done and the device will be marked green:



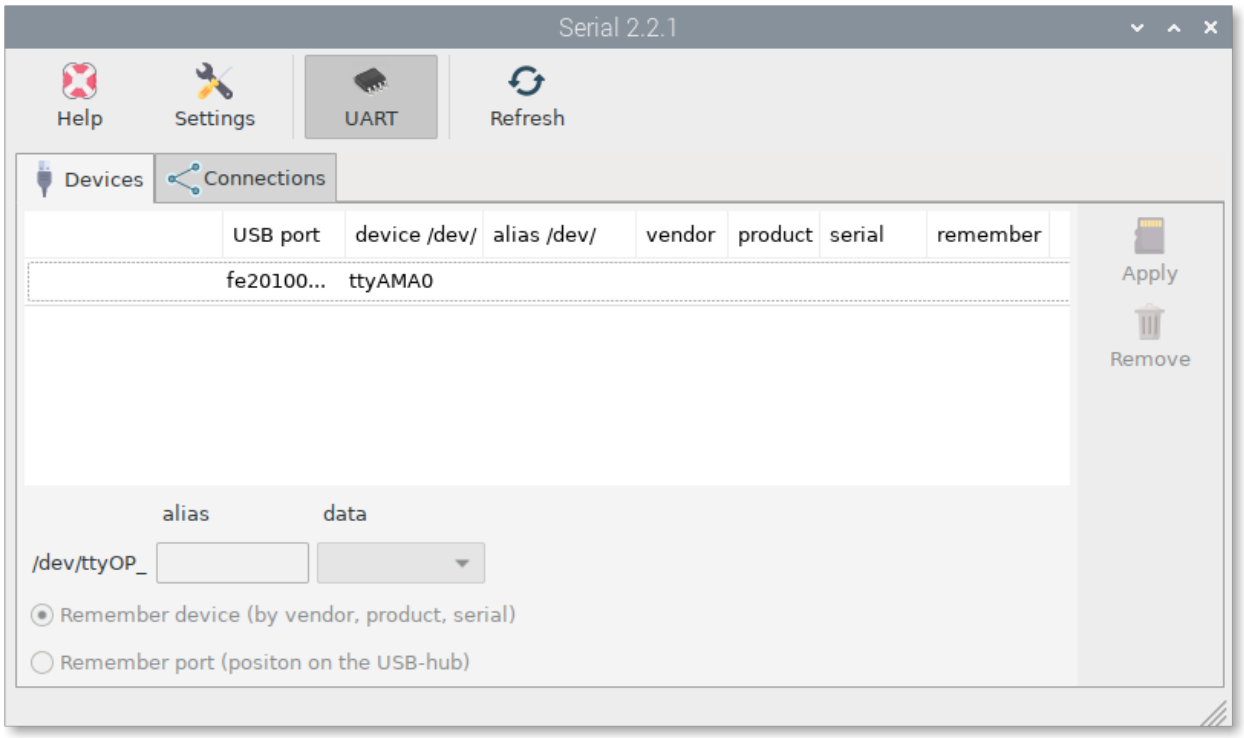
Unplug the device and press `Refresh` to check if the system detects the lost device:



Plug the device back in, press `Refresh` and you are ready to configure any program using your device's alias and be sure it will always work. The next chapter will teach you how to configure the devices in some programs automatically.

UART

In Raspberry Pi 3 and 4 the Bluetooth interface and the UART interface share GPIO pins (GPIO14 for TXD0 and GPIO15 for RXD0). Bluetooth is enabled and UART is dissabled by default. If you want to connect a serial device to UART you need to disable Bluetooth and enable UART. Press `UART` and after reboot, UART interface will be enabled, Bluetooth will be disabled and you will see a new `ttyAMA0` device:

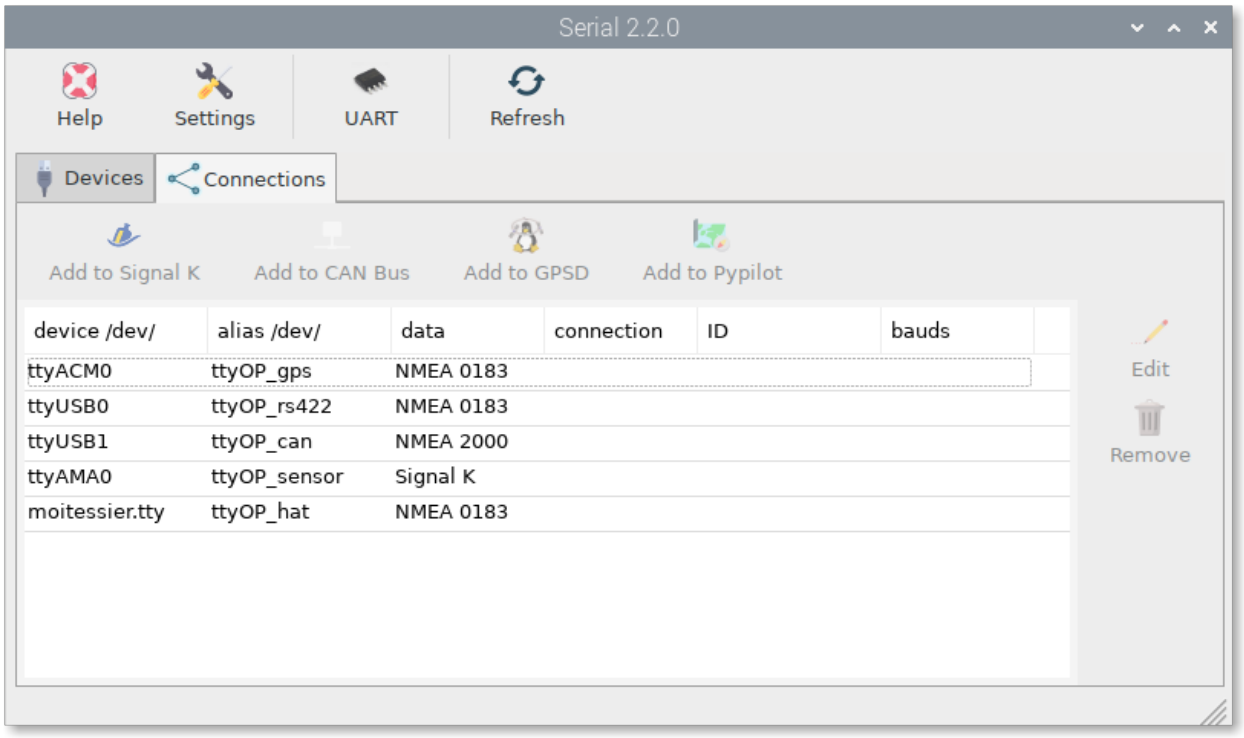


Note: If you want to connect a Pypilot motor controller, you do not need to set an alias for `ttyAMA0` because Pypilot will detect the controller automatically. See *Pypilot* for more info.

CHAPTER 27

Connecting devices

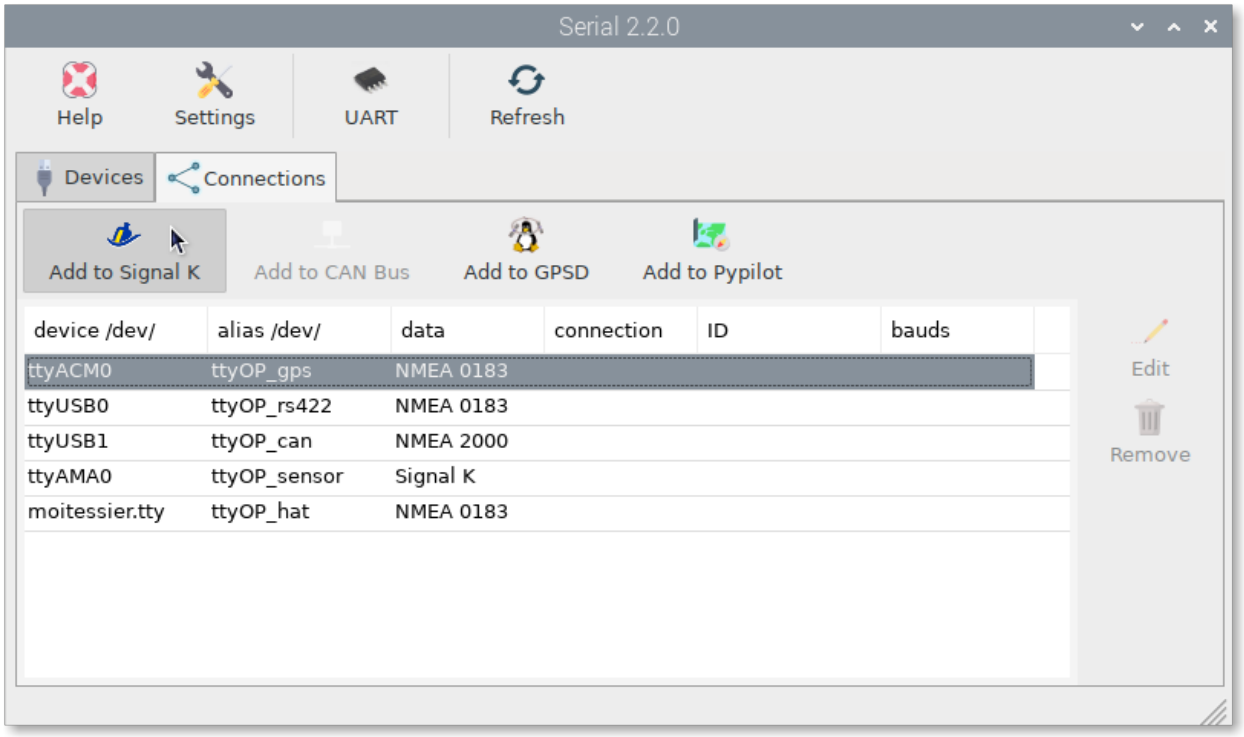
Using the `Connections` tab you can easily configure some programs to obtain data from your device:



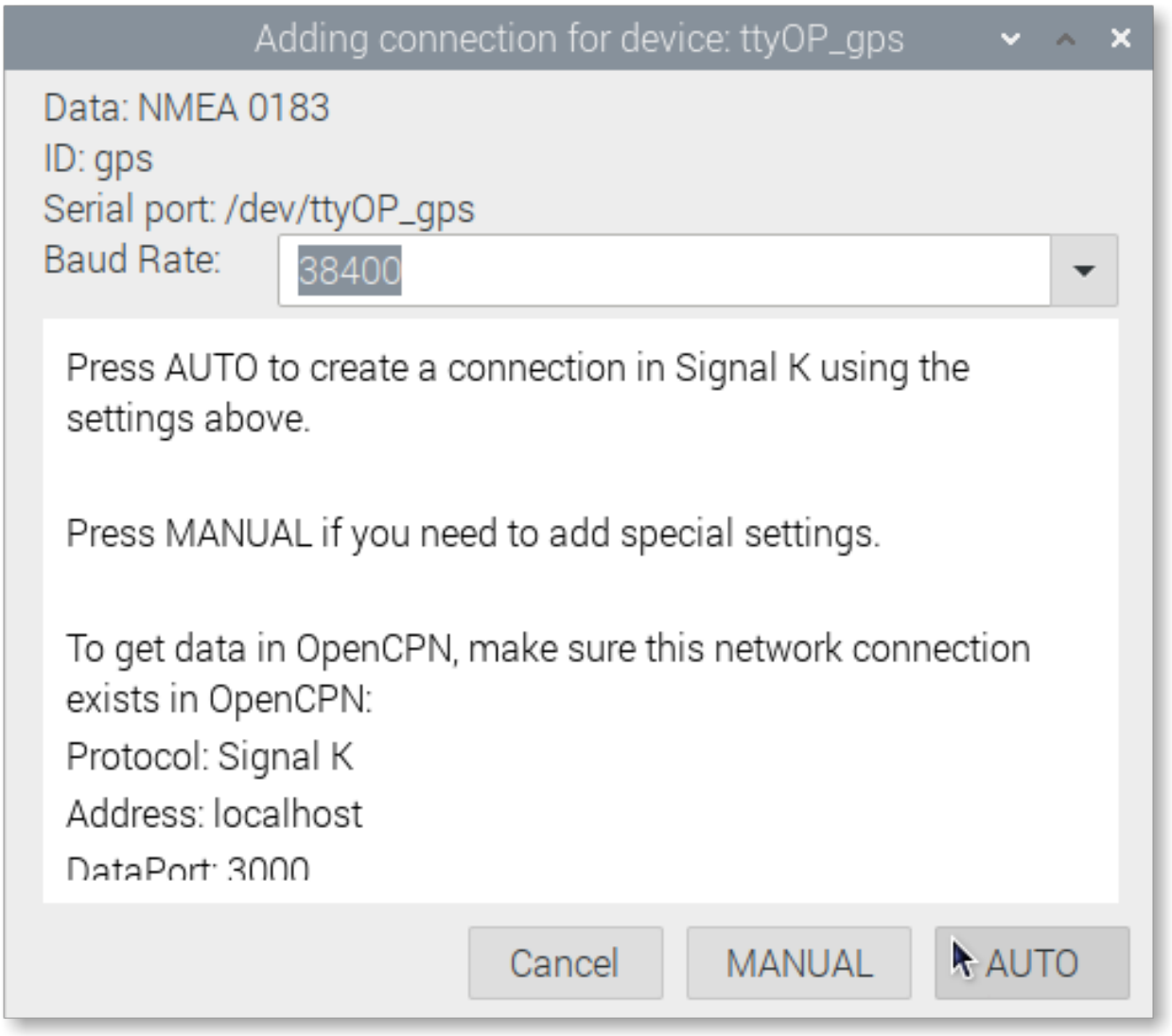
Depending on the type of data you set when defining the alias, some supported programs will be enabled in the toolbar when selecting devices. To see how this works we are going to configure our USB GPS receiver. Select the `ttyOP_gps` device and press `Add to Signal K`:

Note: Select `Add to Pypilot` only if you are using a Pypilot controller. See *Pypilot* for more info.

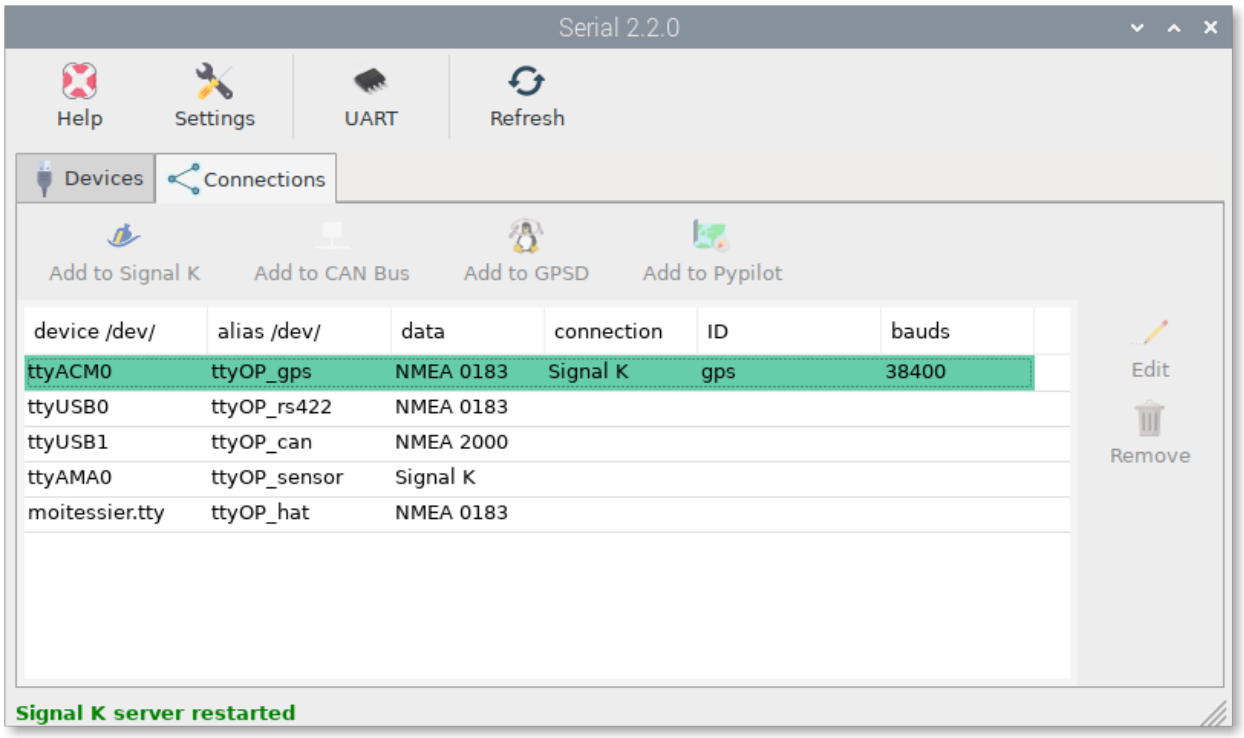
Select `Add to GPSD` only if you want GPSD to manage your GPS/AIS device. All GPSD and Signal K settings will be created automatically.



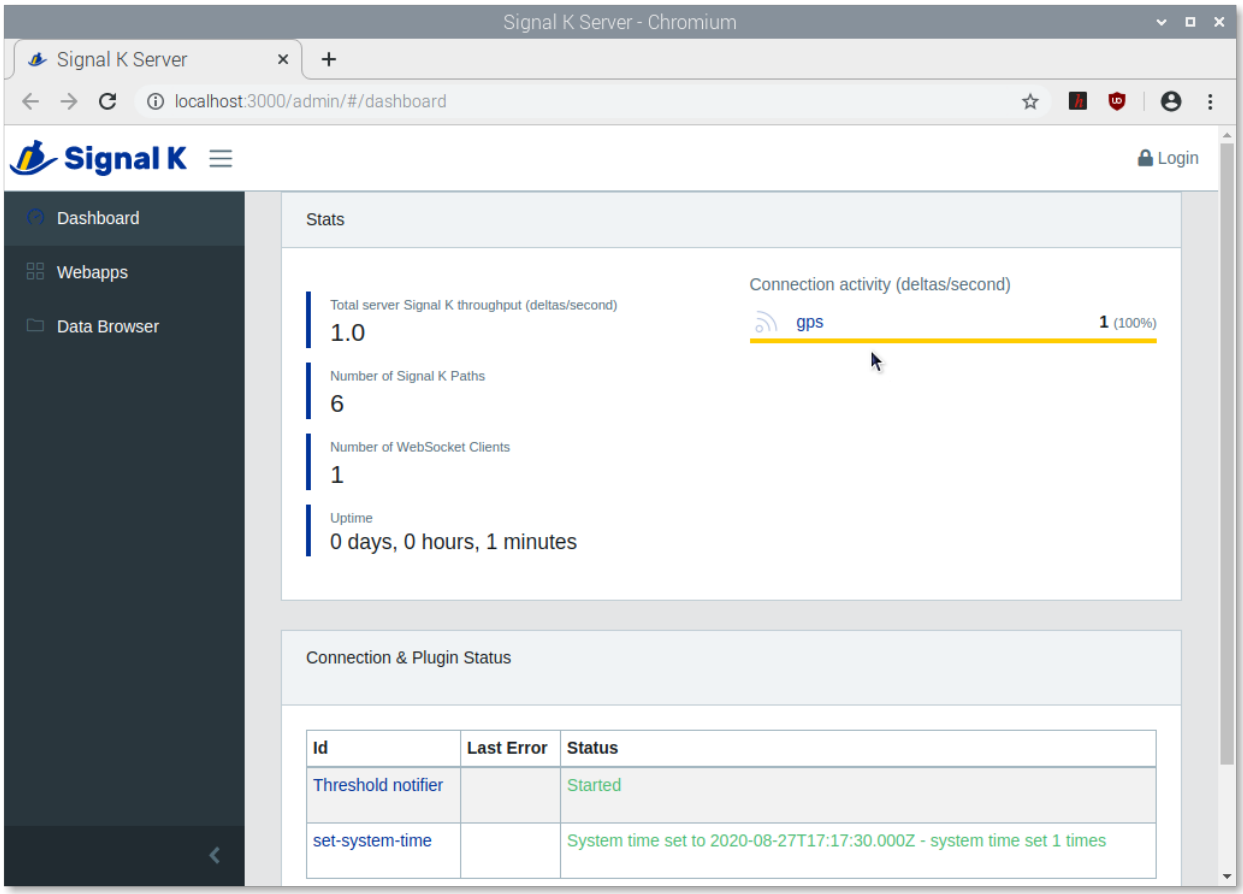
Then select the `Baud Rate` required by your device and press `AUTO`:



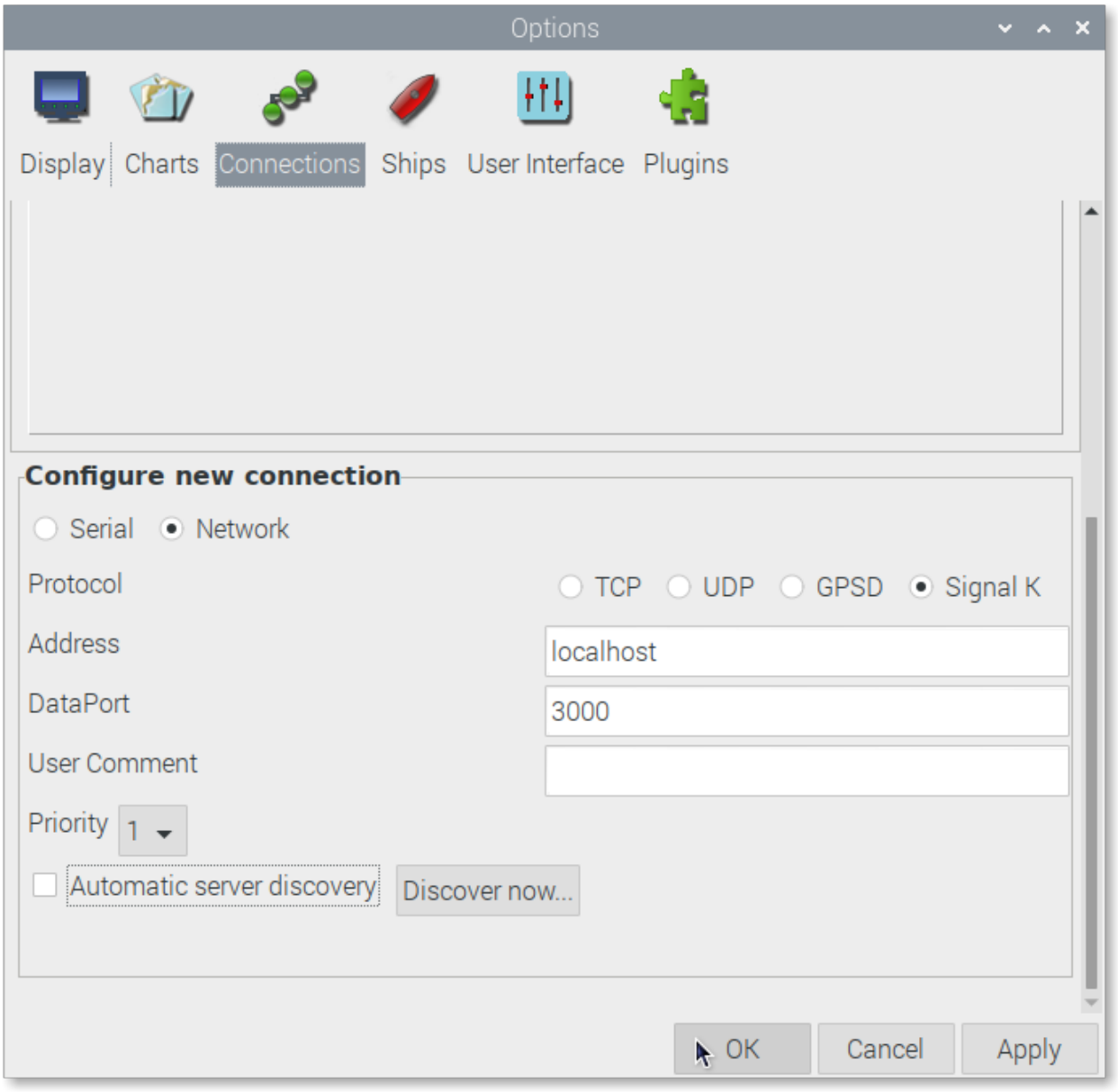
The signal K server will restart and the connection will be marked green:



And you are done. Check in Signal K server the new connection:



And check OpenCPN to make sure there is a connection to the Signal K server:



CHAPTER 28

Other examples

28.1 Connecting a USB-RS422 converter

You probably still have some devices onboard that use the old NMEA 0183 protocol. Most commercial plotters collect data from all onboard devices and send it through an RS422 output. To connect these devices to OpenPlotter, you need any inexpensive USB-RS422 converter.

28.1.1 Wiring

Typical RS422 device looks like the one below:

There are normally 4 or 5 connections: TX+, TX-, RX+, RX-, GND.

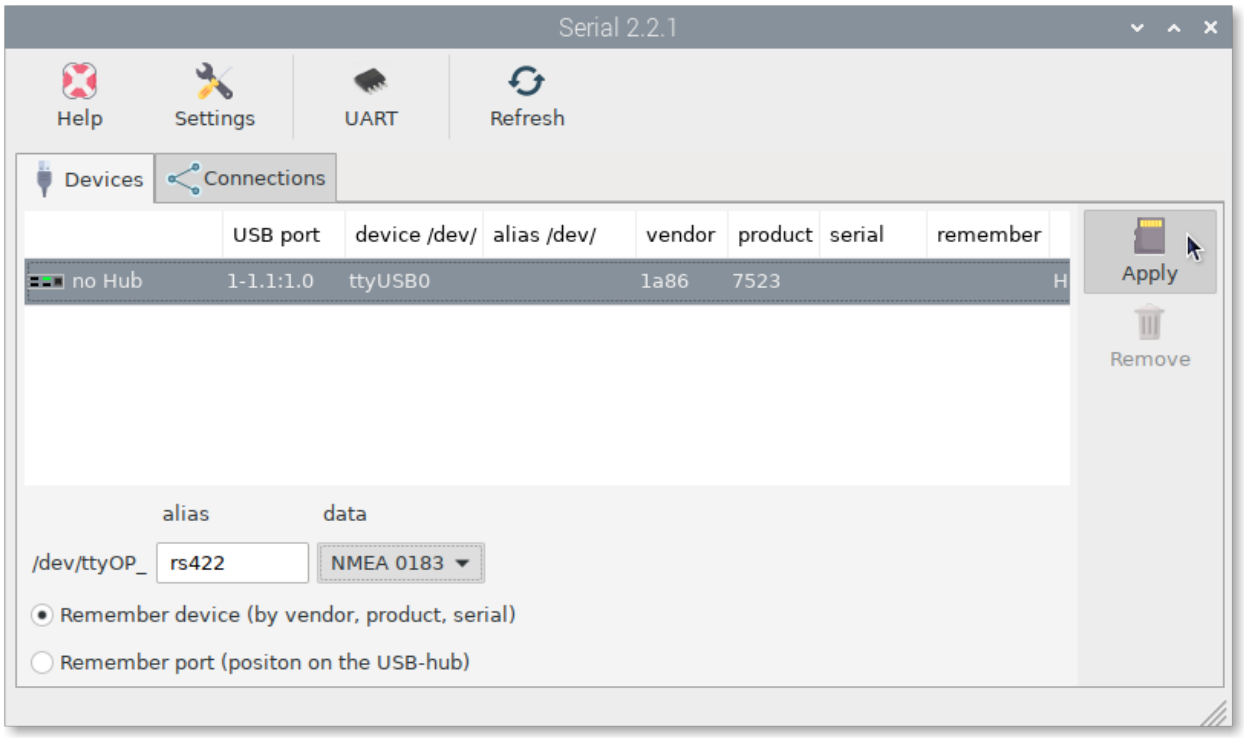
Important: Normally you do not need GND and you would connect TX of the chart plotter/VHF etc to the RX of the RS422 to USB device and vice versa. However, there is little consistency between different devices as to what is possitve and what is negative - so if the TX+ connected to the RX+ does not work, try connecting to the RX-.

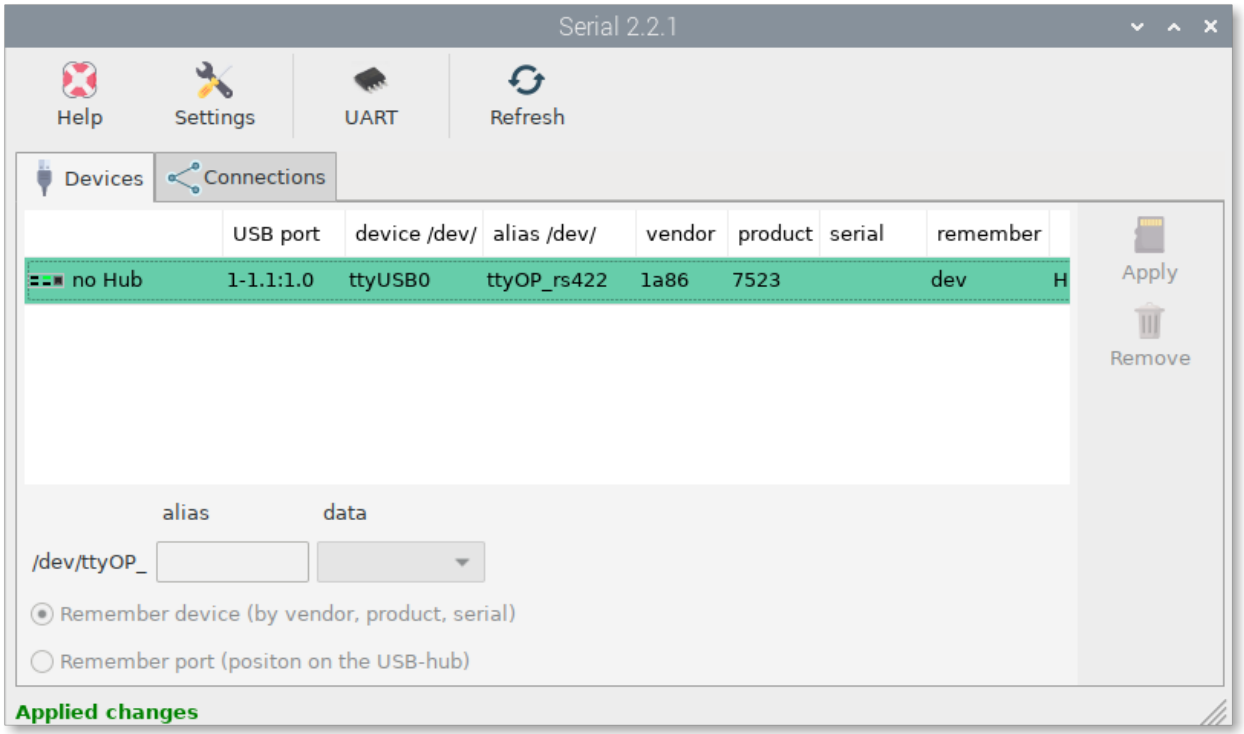
Consult your device manual to find the baud rate, if you can not find the baud rate then usually, if the device is older and pre-AIS the baud rate may be 4800, later devices that may have or accept AIS will be 38400. Use an absolute value if Auto does not work

28.1.2 Input data

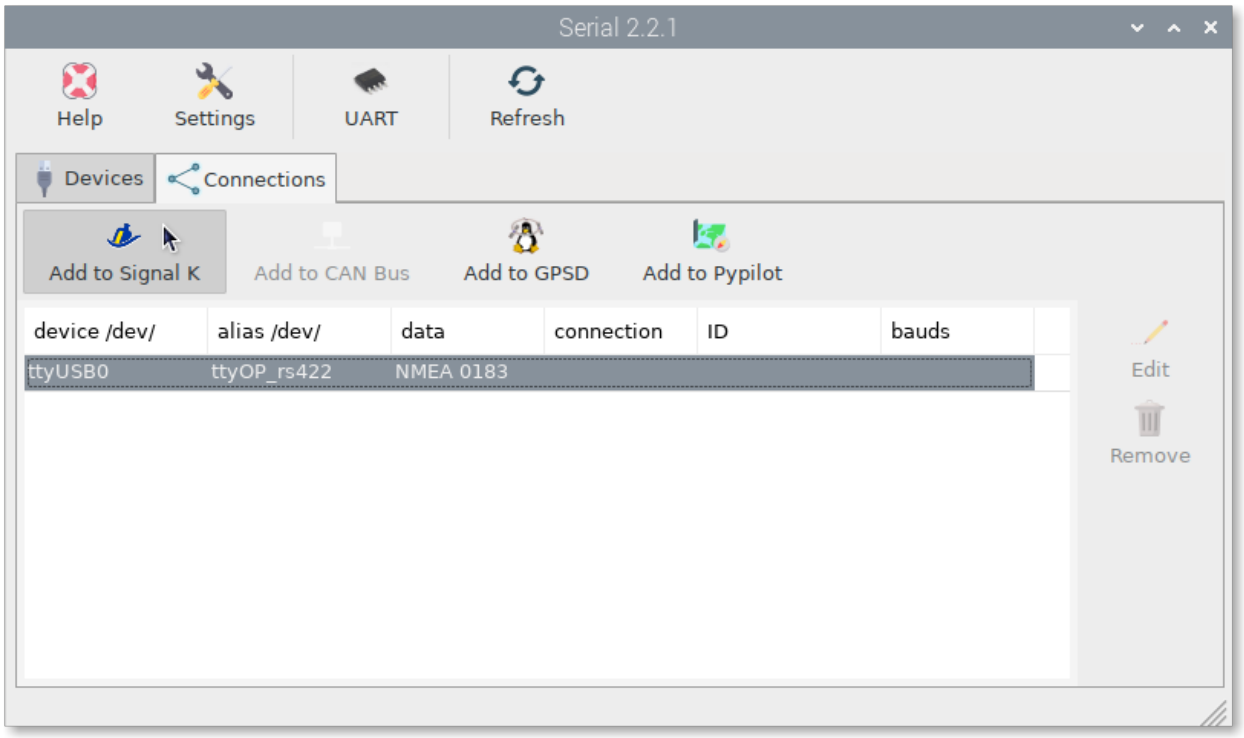
To obtain data from these converters, follow the same steps as for connecting the GPS in the examples of the previous sections of this chapter. Below are the summarized steps.

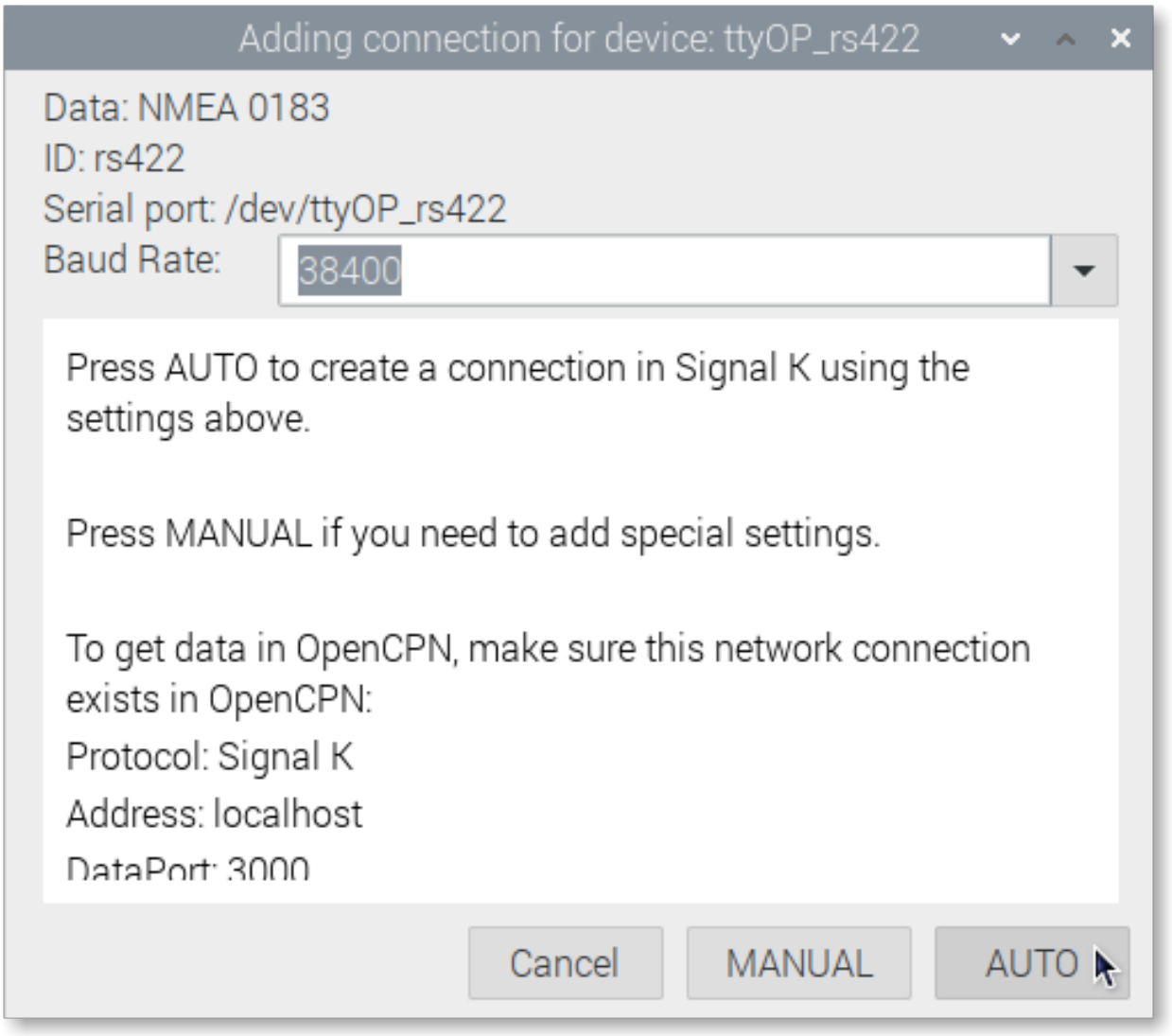
Enter an `alias` and select the type of `data`:

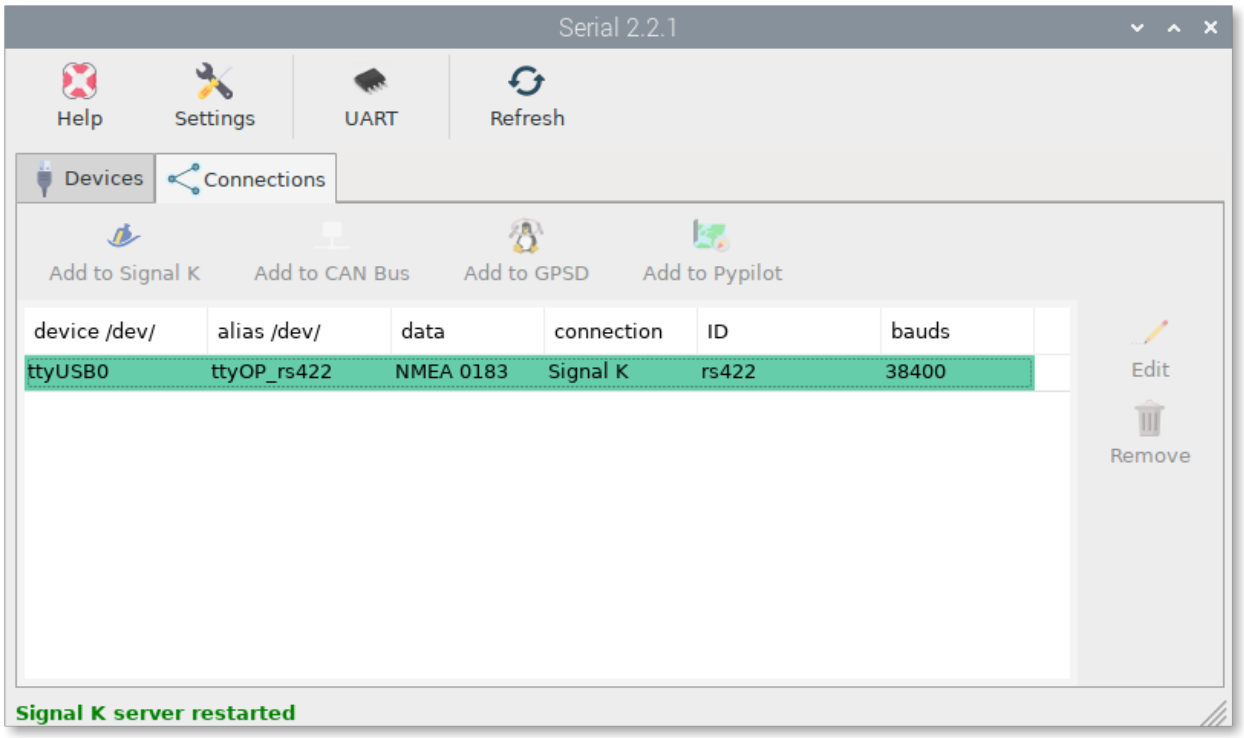




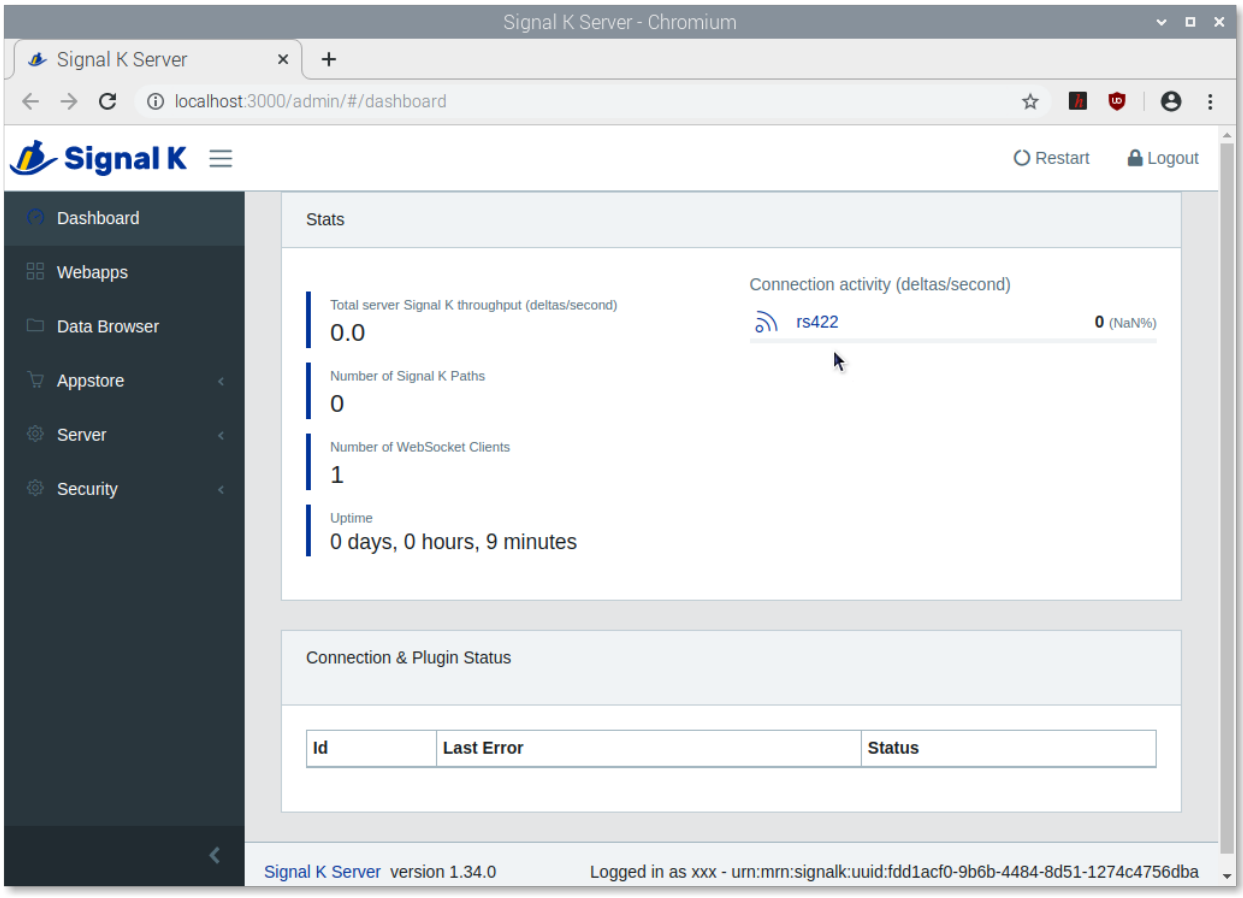
Create a signal K connection:


Serial 2.2.1
Help
Settings
UART
Refresh
Devices
Connections
Add to Signal K
Add to CAN Bus
Add to GPSD
Add to Pypilot
device /dev/
alias /dev/
data
connection
ID
bauds
ttyUSB0
ttyOP_rs422
NMEA 0183
Edit
Remove

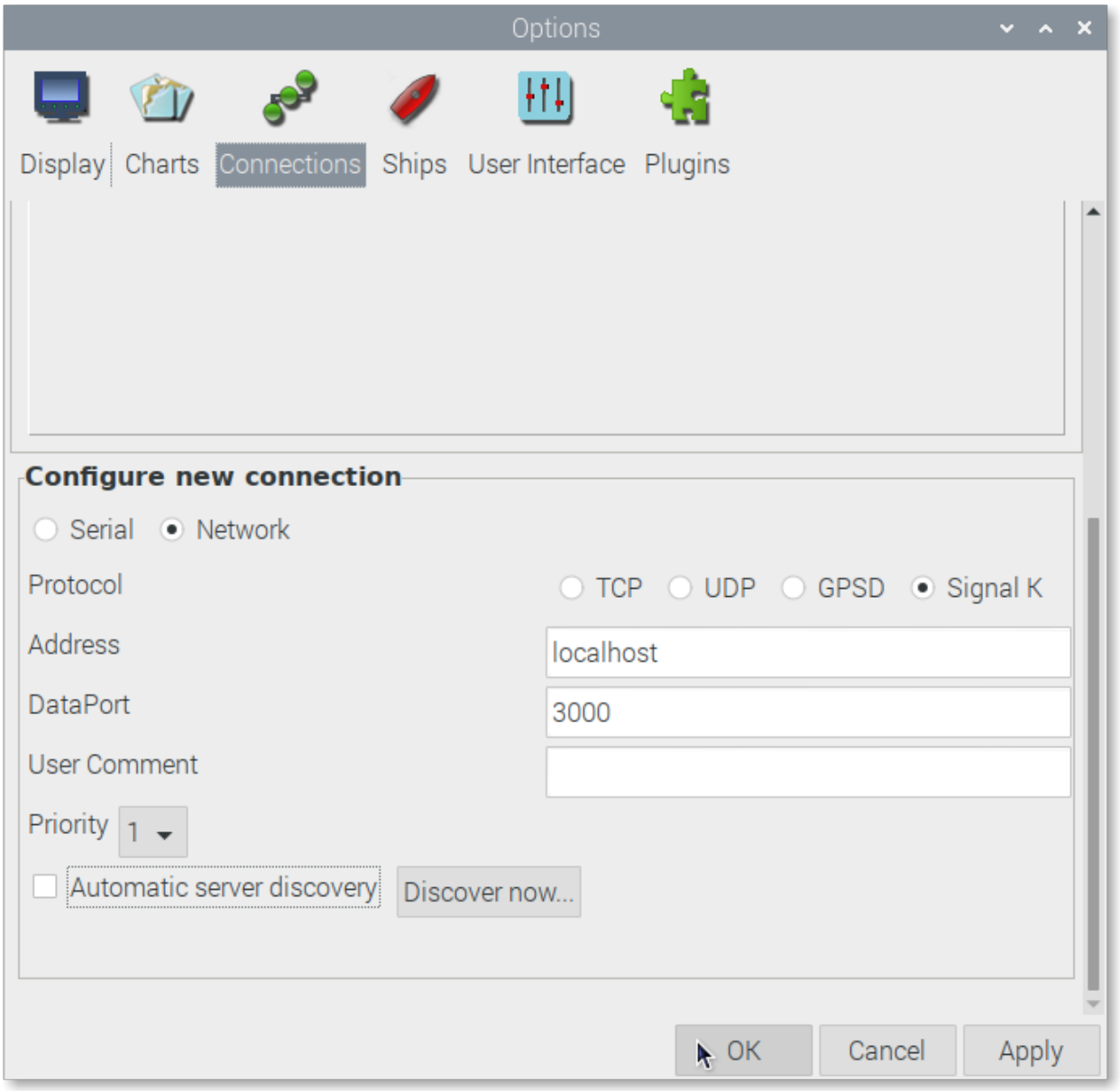

Adding connection for device: ttyOP_rs422
Data: NMEA 0183
ID: rs422
Serial port: /dev/ttyOP_rs422
Baud Rate:
38400
Press AUTO to create a connection in Signal K using the settings above.
Press MANUAL if you need to add special settings.
To get data in OpenCPN, make sure this network connection exists in OpenCPN:
Protocol: Signal K
Address: localhost
DataPort: 3000
Cancel
MANUAL
AUTO



Check the Signal K connection has been made:



And check OpenCPN to make sure there is a connection to the Signal K server:

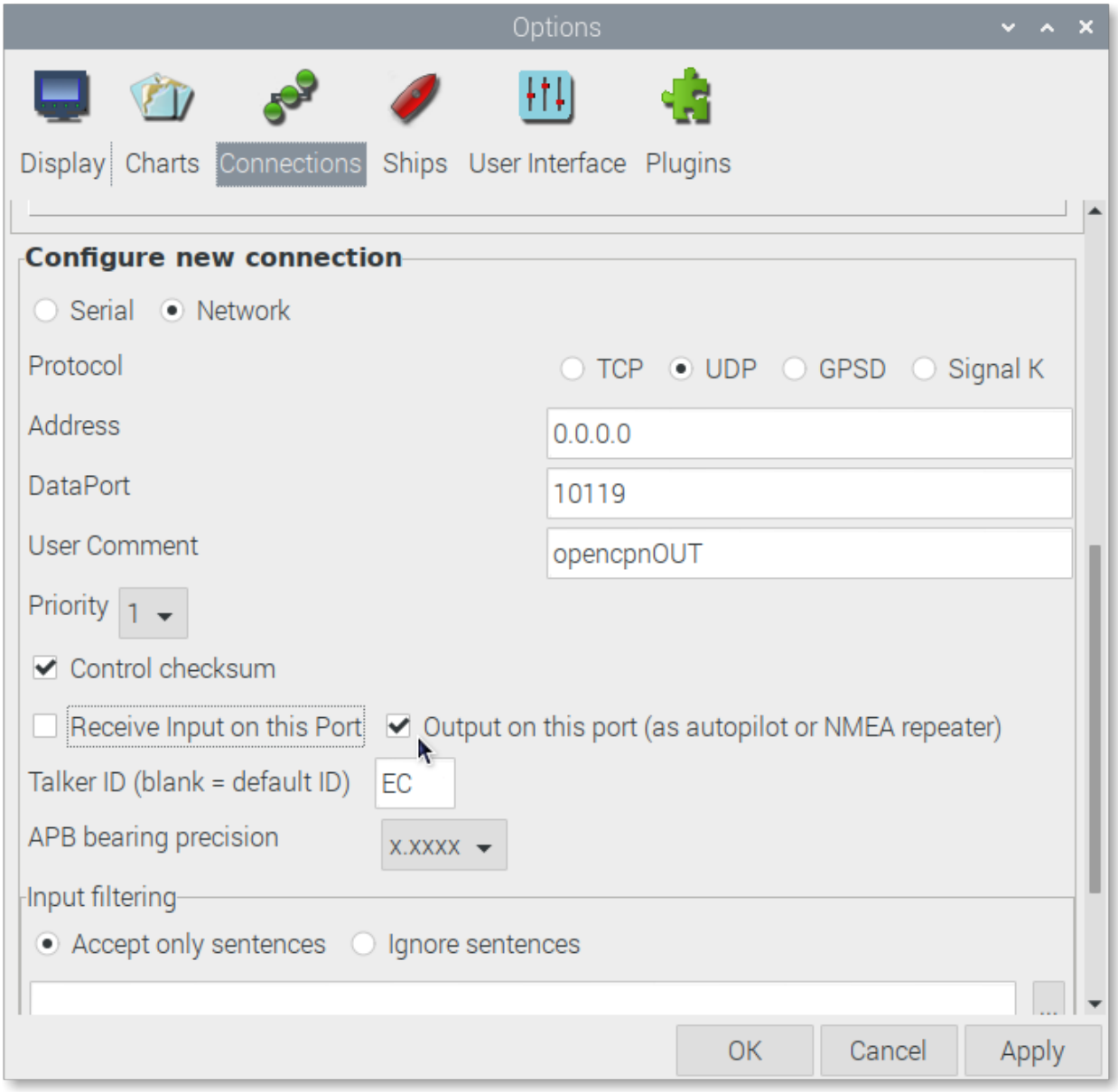


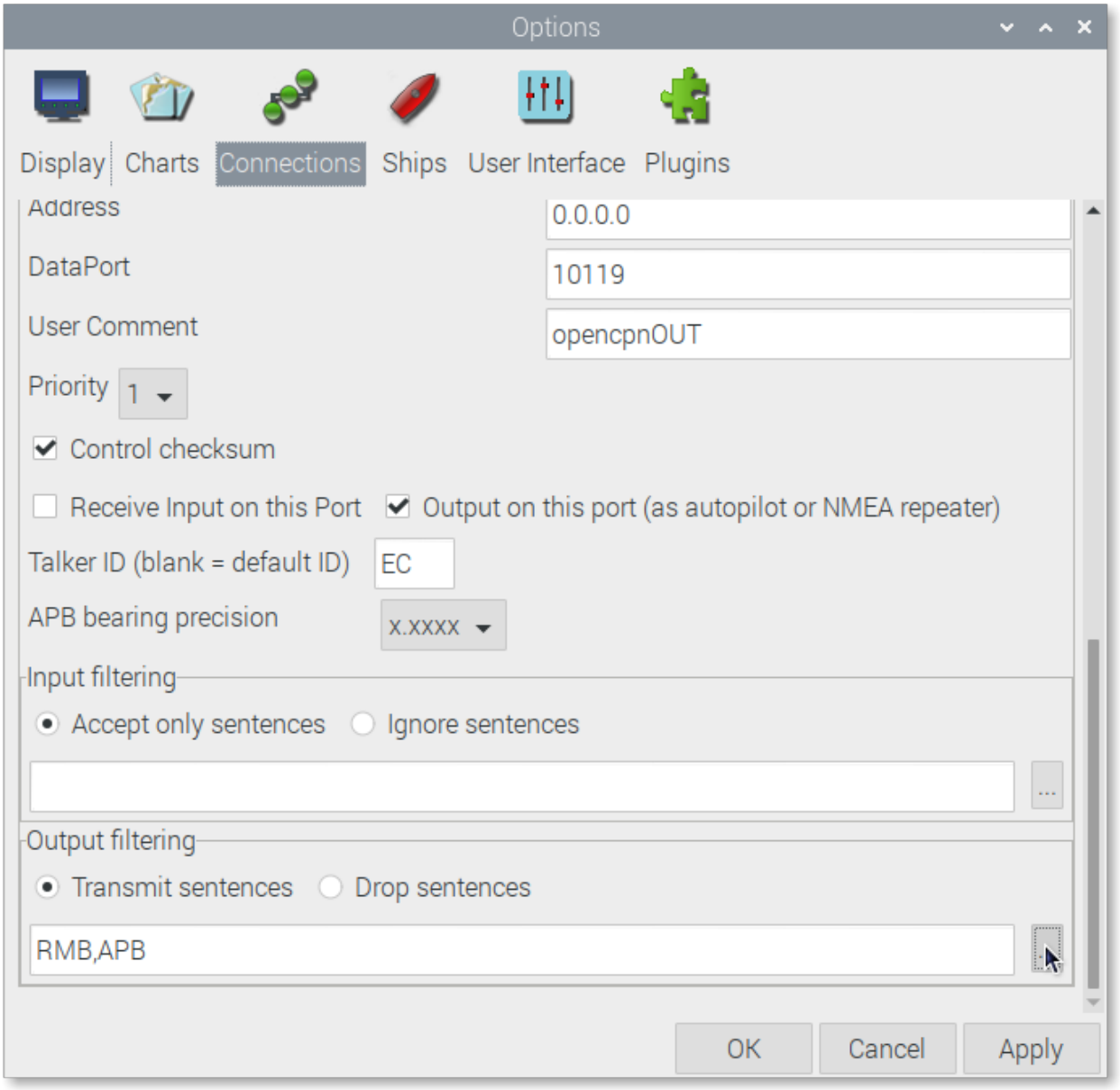
You should now be ready to get NMEA 0183 data from your boat.

28.1.3 Input + output data

Now that you are getting NMEA 0183 data from your boat, you may also want to send some NMEA 0183 data generated in OpenPlotter to your boat. The classic case is to let openplotter control your autopilot. Let's see how to configure your route in OpenCPN and send data to the autopilot using the same USB-RS422 converter.

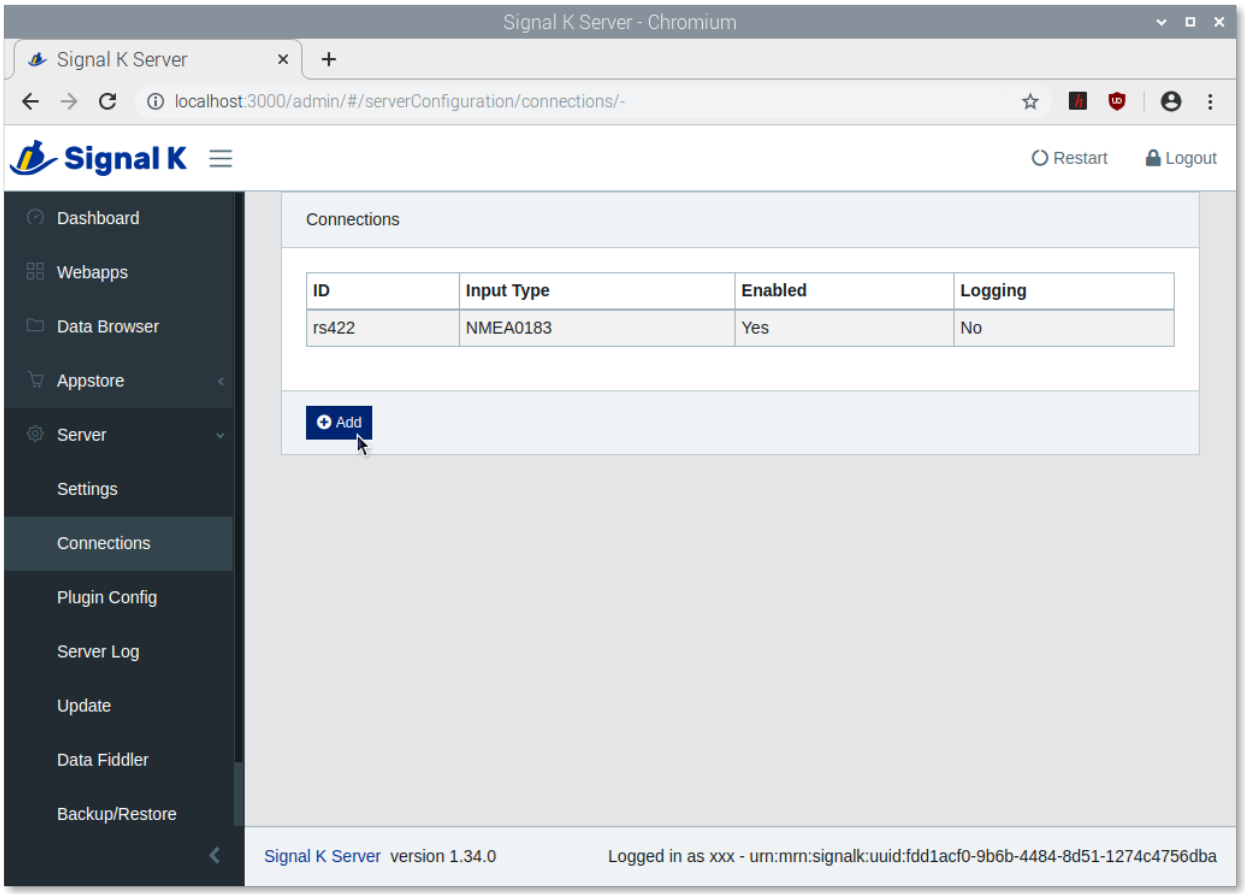
You have to create a UDP connection in OpenCPN to send data to Signal K server. Select `Network`, `Protocol`: UDP, `Address`: 0.0.0.0, `DataPort`: 10119 (or any unused UDP port on your system), `User Comment`: opencpnOUT, uncheck `Receive input on this Port`, check `Output on this port`, transmit only sentences RMB and APB in `Output filtering`:


Options
Display
Charts
Connections
Ships
User Interface
Plugins
Configure new connection
Serial
Network
Protocol
TCP
UDP
GPSD
Signal K
Address
0.0.0.0
DataPort
10119
User Comment
opencpnOUT
Priority
1
Control checksum
Receive Input on this Port
Output on this port (as autopilot or NMEA repeater)
Talker ID (blank = default ID)
EC
APB bearing precision
x.xxxx
Input filtering
Accept only sentences
Ignore sentences
OK
Cancel
Apply

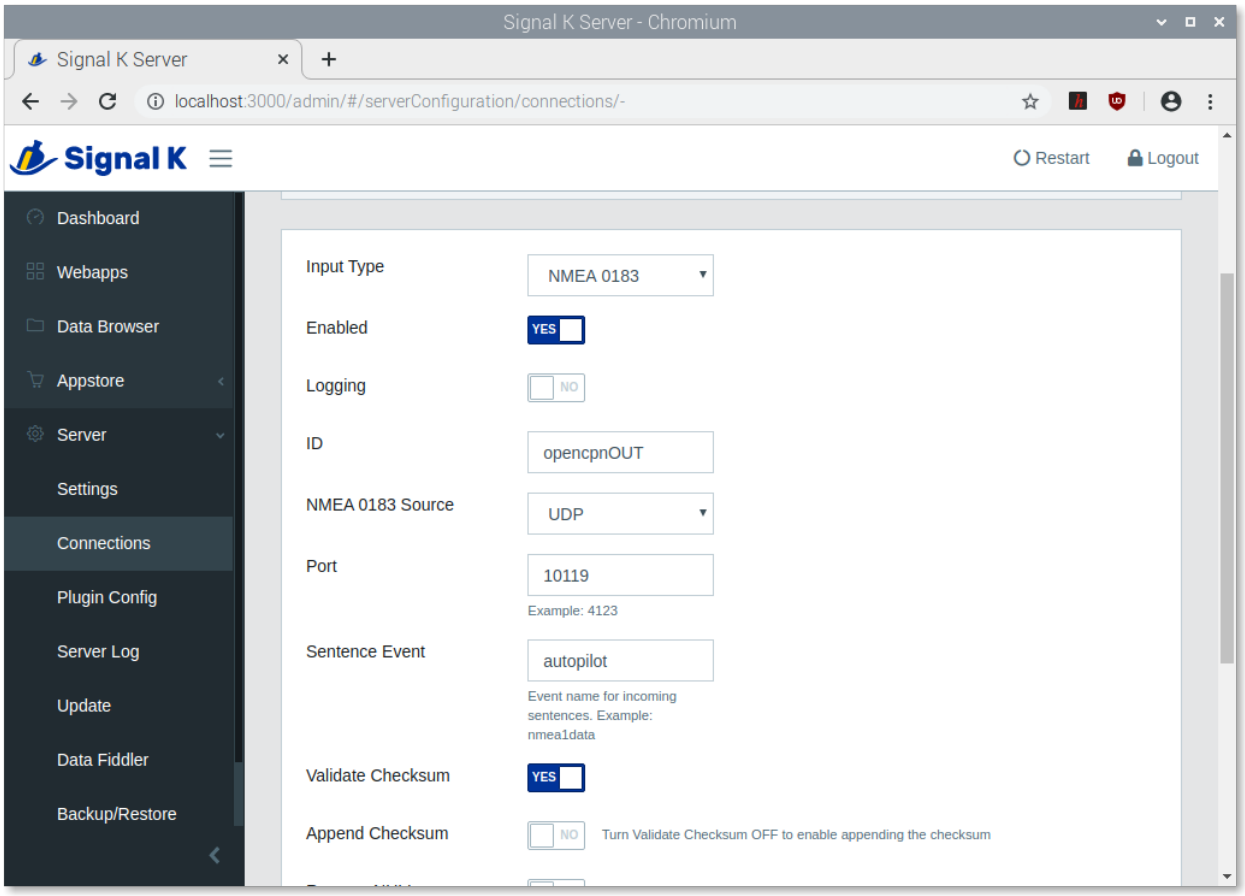


Warning: Allowing only RMB and APB sentences in the output is important to avoid data loops in your system.

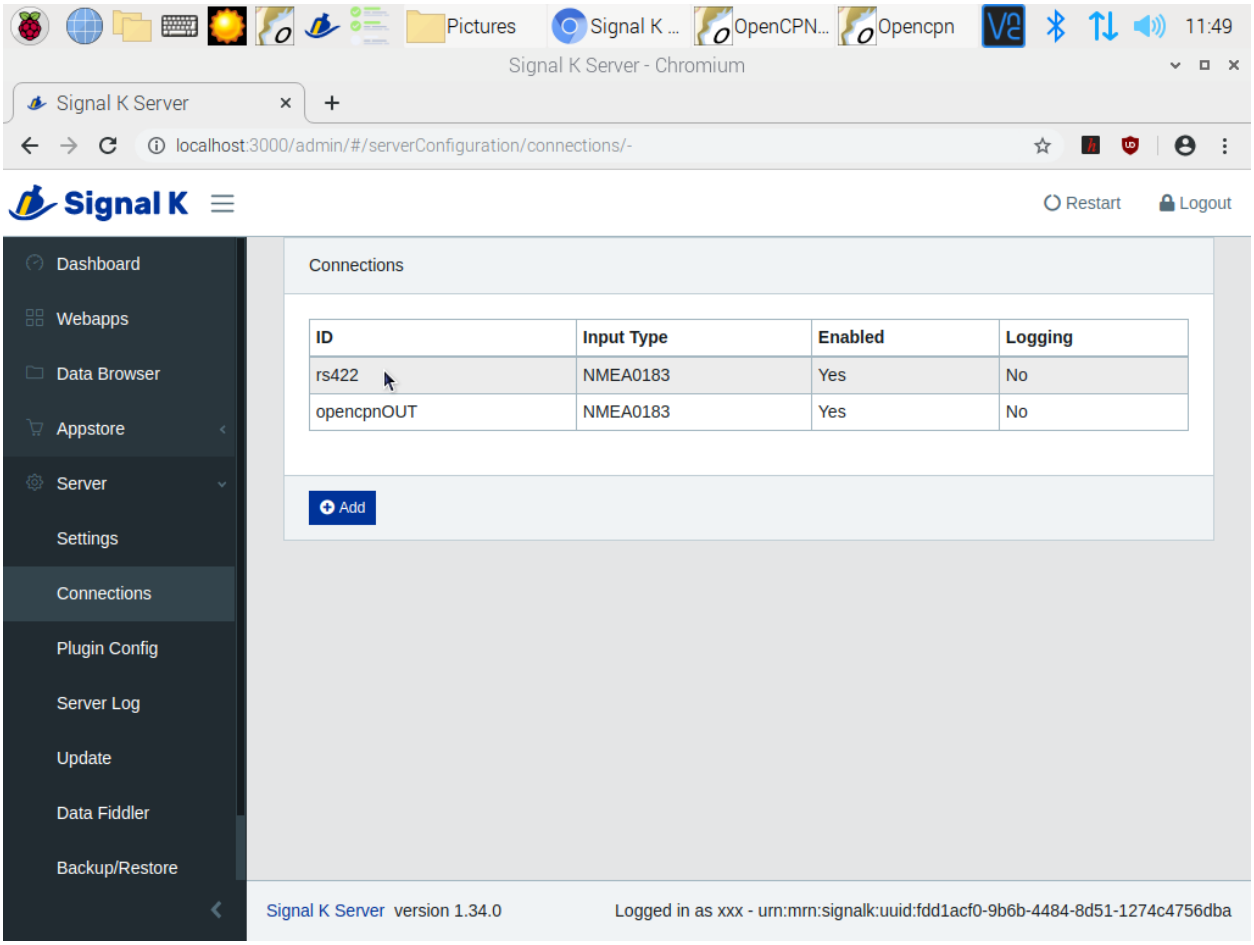
Now you have to create a connection in Signal K server to get data from OpenCPN. Login to the Signal K server, go to *Server* → *Connections* and click on `Add`:



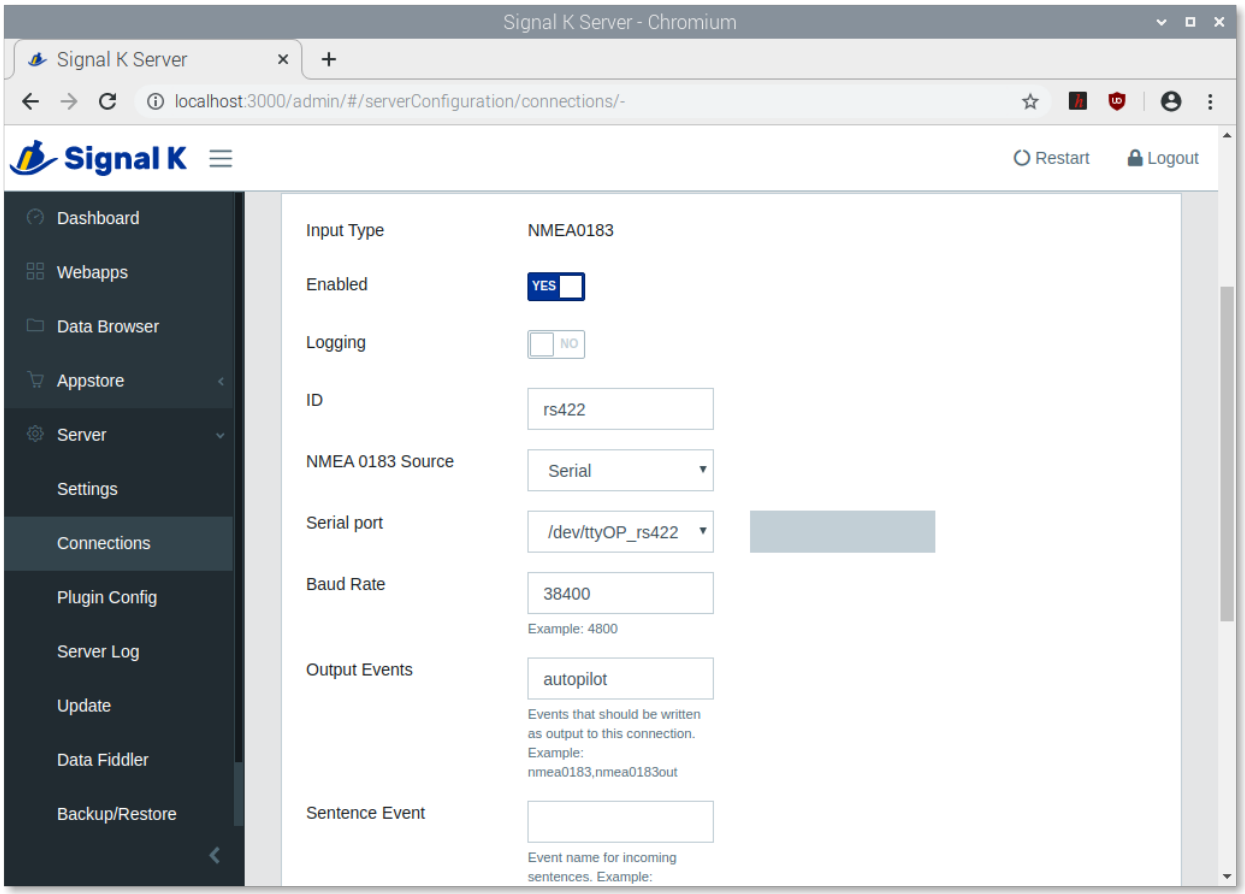
Set `Input Type`: NMEA 0183, `ID`: opencpnOUT, `NMEA 0183 Source`: UDP, `Port`: 10119 (or whatever you have set in OpenCPN), `Sentence Event`: autopilot and Click `Apply`:



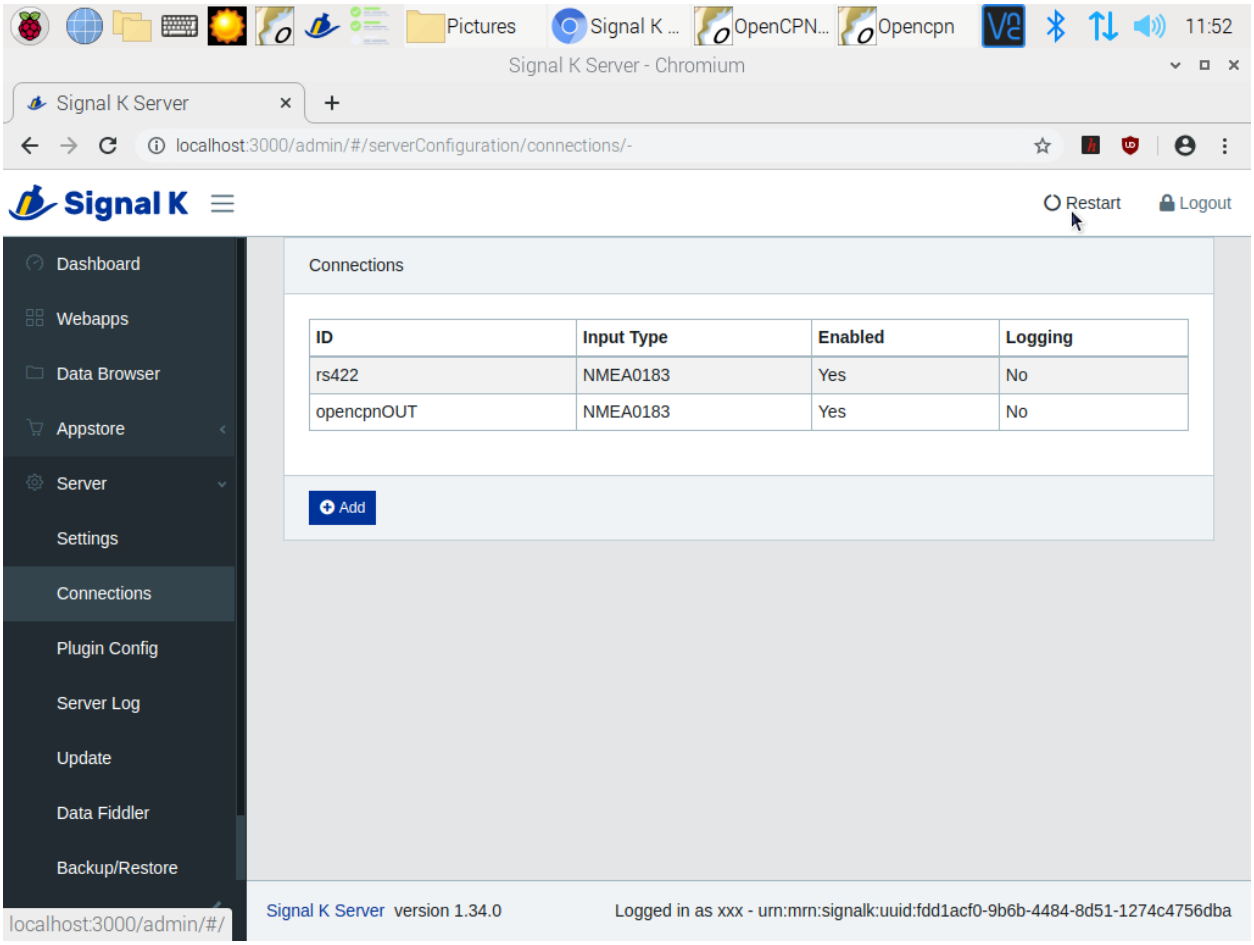
Finally, you have to edit your device connection to specify what data should be sent to your boat via the USB-RS422 converter. Go to *Server* → *Connections* and click on your device connection, in this case `rs422`:

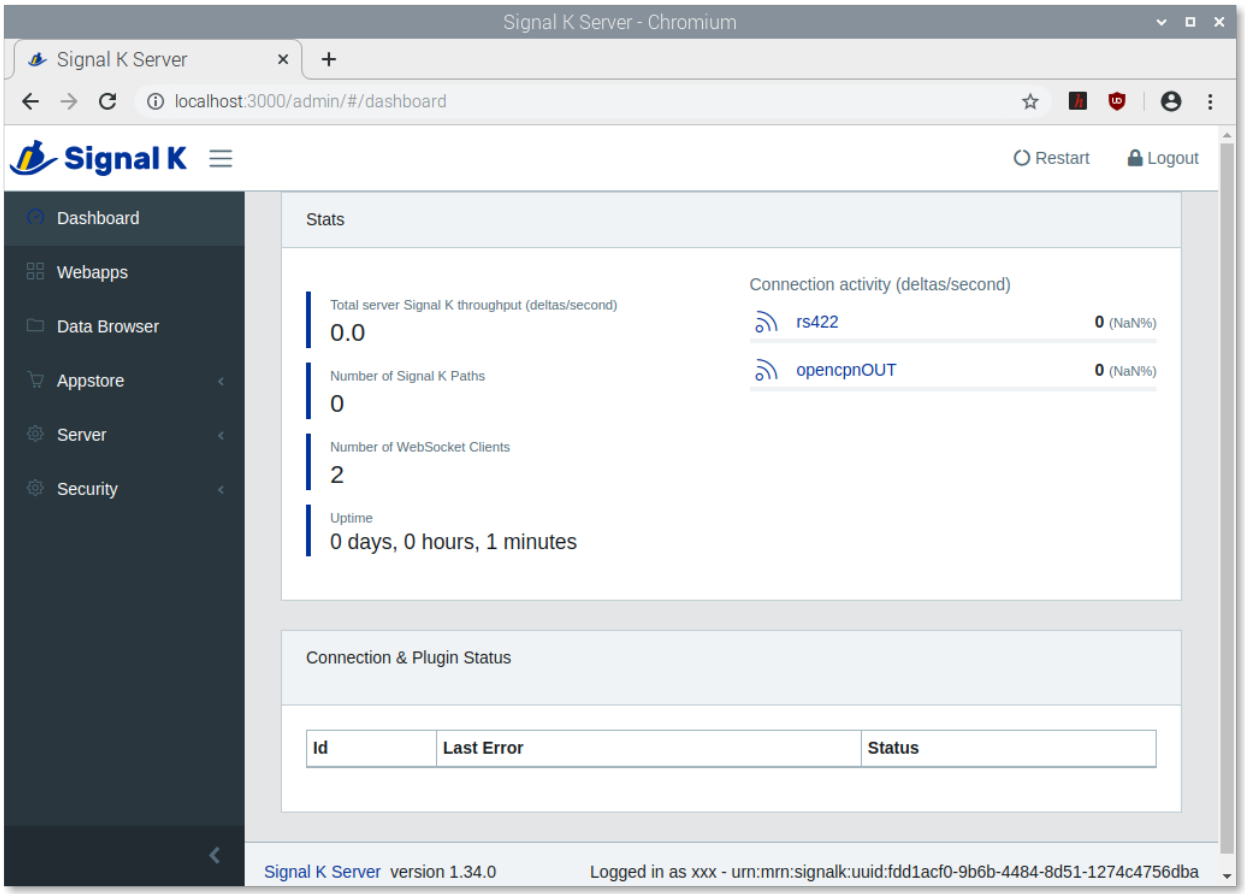


Set `Output Events`: autopilot and Click `Apply`:

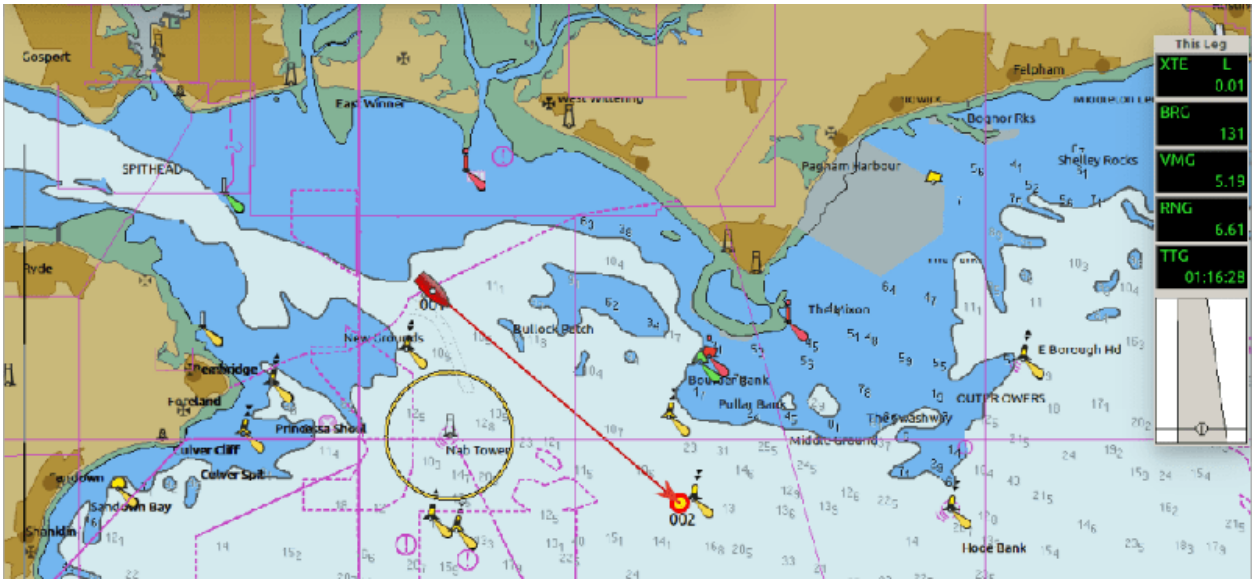


`Restart` Signal K server and you are done:


Pictures
Signal K ...
OpenCPN...
Opencpn
11:52
Signal K Server - Chromium
Signal K Server
localhost:3000/admin/#/serverConfiguration/connections/-
Signal K
Restart
Logout
Dashboard
Webapps
Data Browser
Appstore
Server
Settings
Connections
Plugin Config
Server Log
Update
Data Fiddler
Backup/Restore
Connections
ID
Input Type
Enabled
Logging
rs422
NMEA0183
Yes
No
opencpnOUT
NMEA0183
Yes
No
Add
localhost:3000/admin/#/
Signal K Server version 1.34.0
Logged in as xxx - urn:mrn:signalk:uuid:fdd1acf0-9b6b-4484-8d51-1274c4756dba



Activate a route in OpenCPN and you will start sending data to your autopilot.

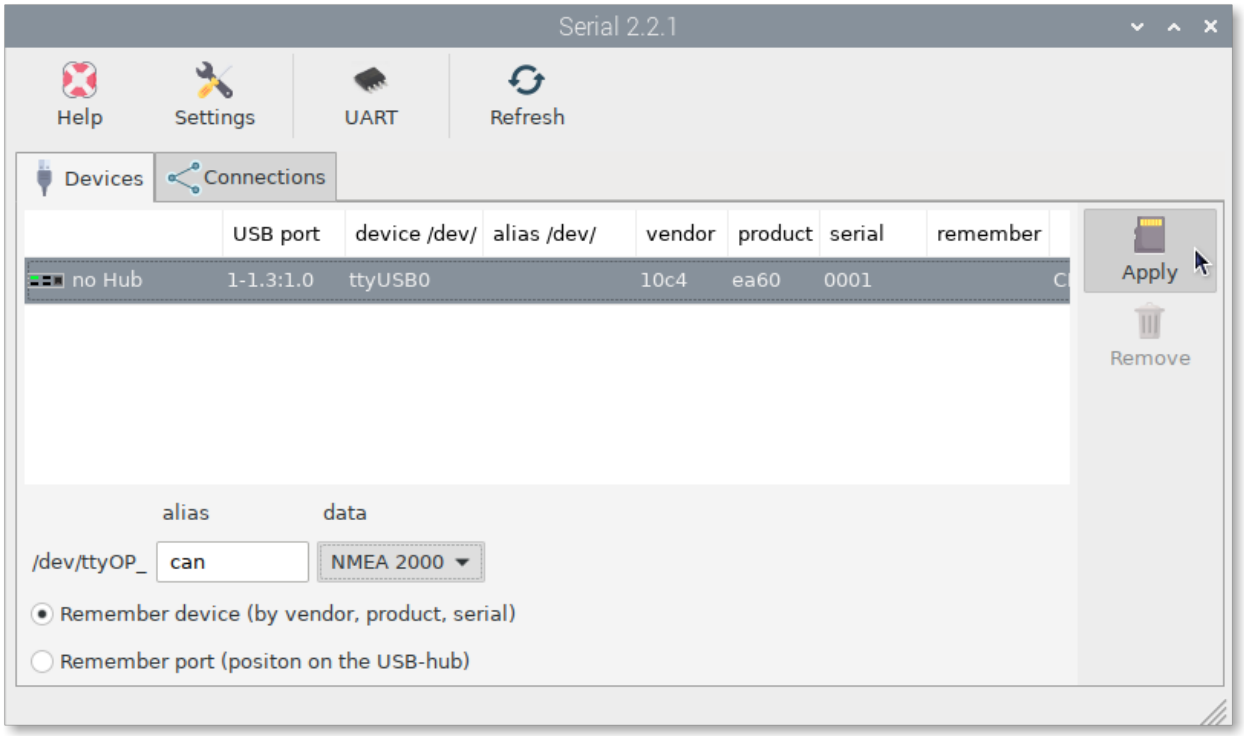


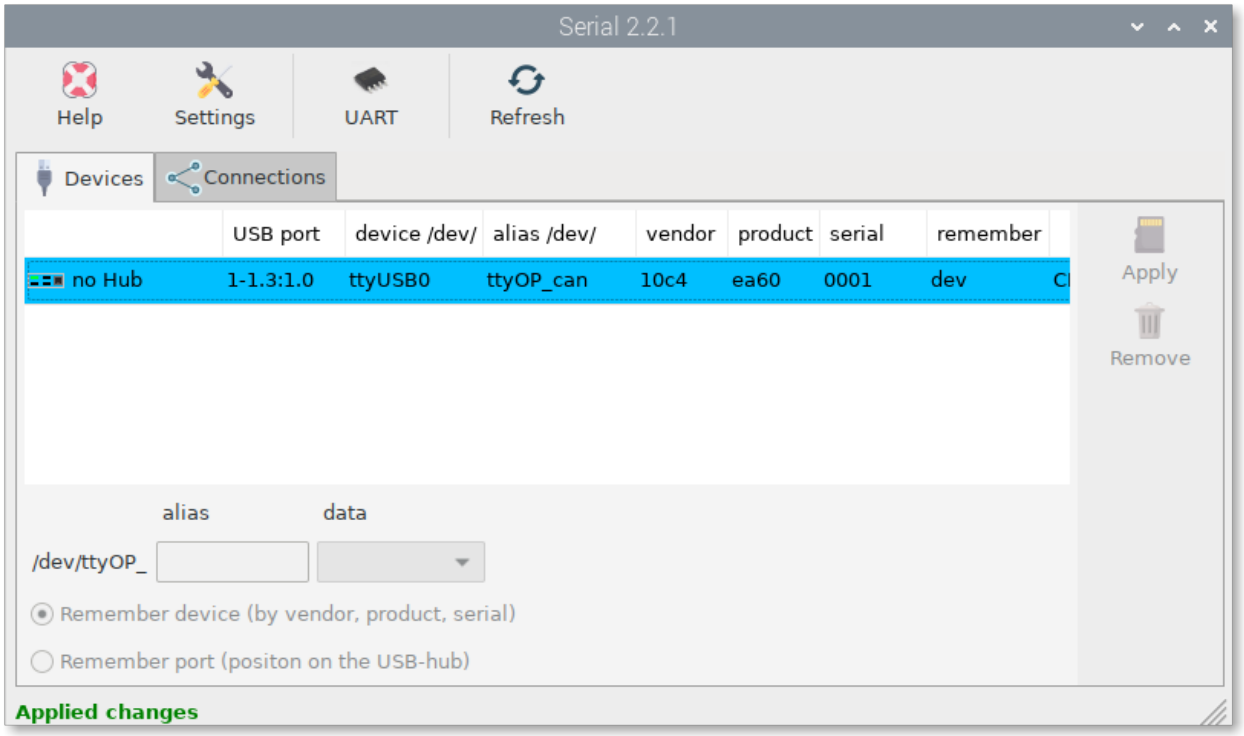
28.2 Connecting a USB-CAN converter

This tutorial is for the *Actisense NGT-1*, the *OpenMarine CAN-USB Stick* (discontinued) and the *CANable* devices.

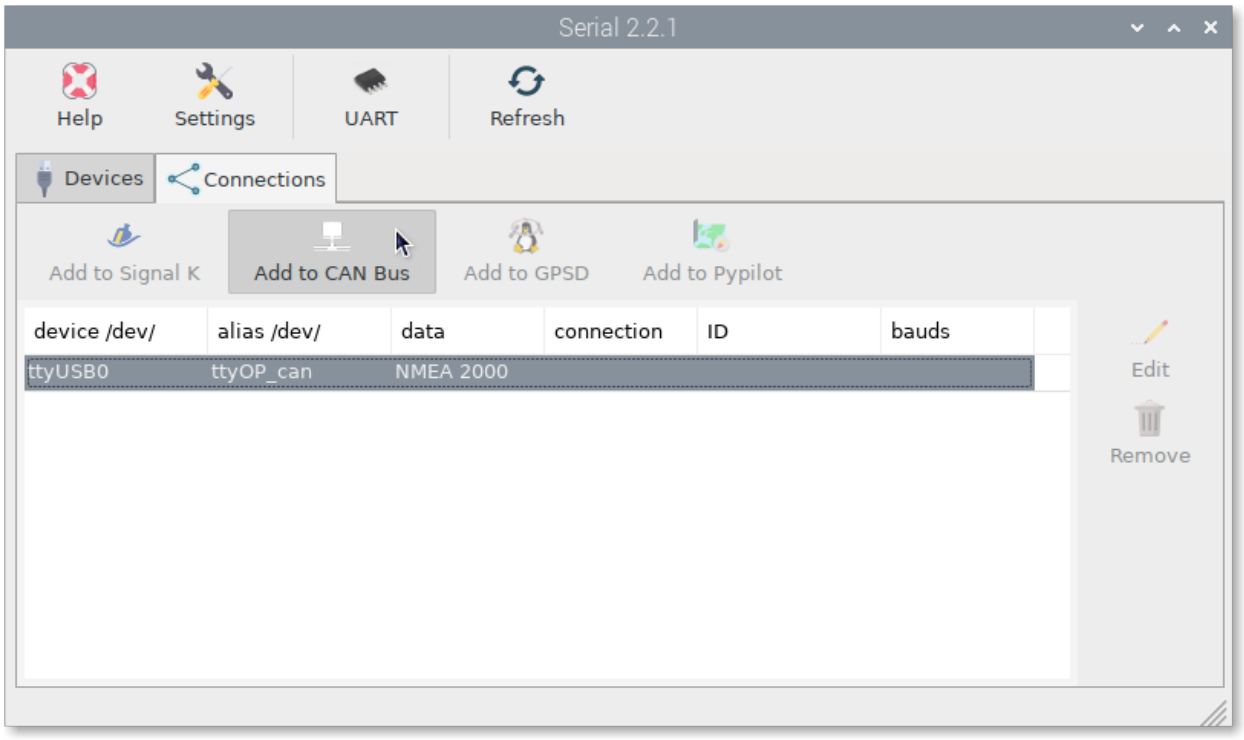
28.2.1 Input data

To get data from your NMEA 2000 network you have to select the device, enter an `alias` and select NMEA 2000 in `data` field. Finally press `Apply` and the device will be marked blue:



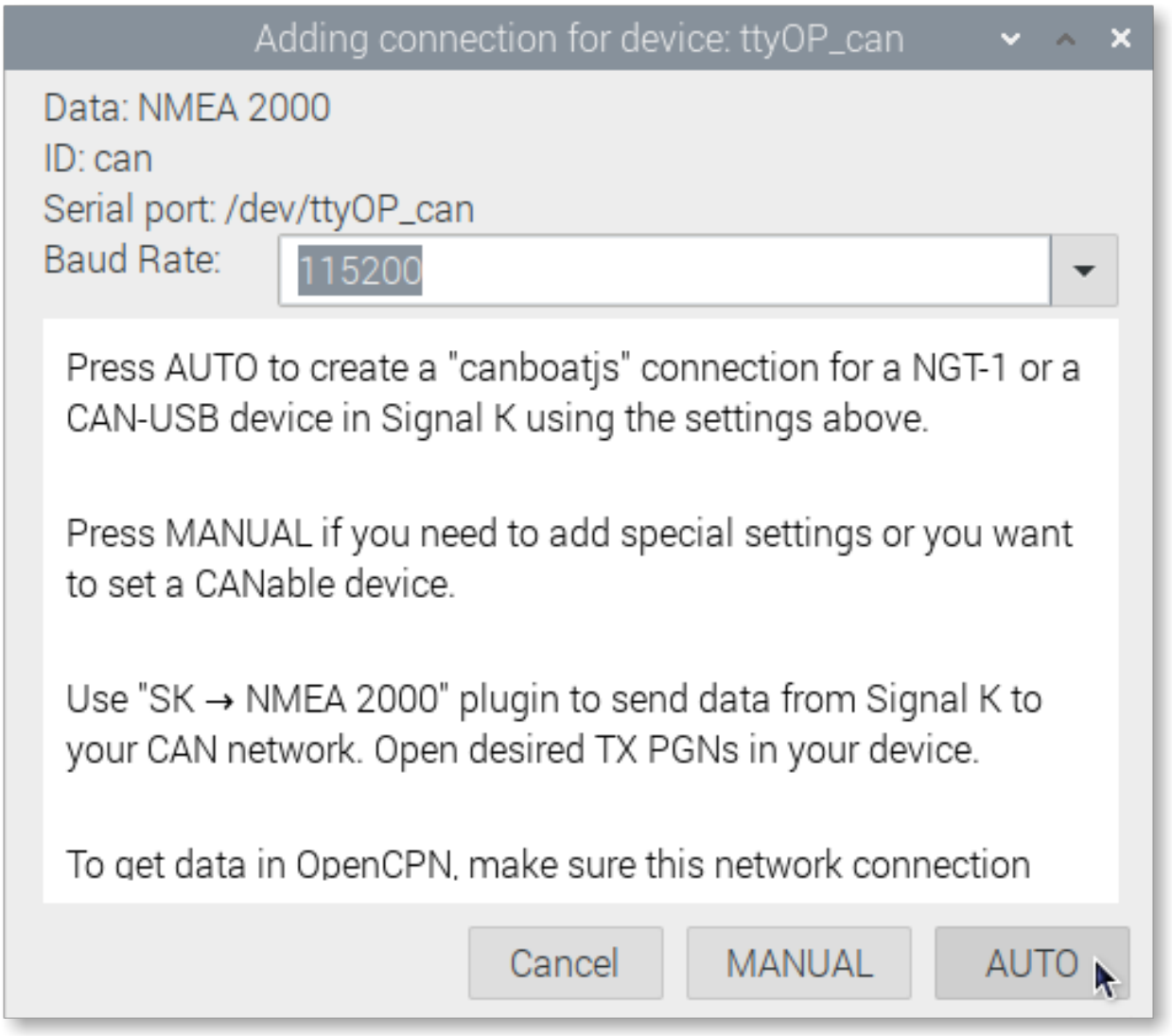


Then go to `Connections` tab, select the device and click on `Add to CAN Bus`:

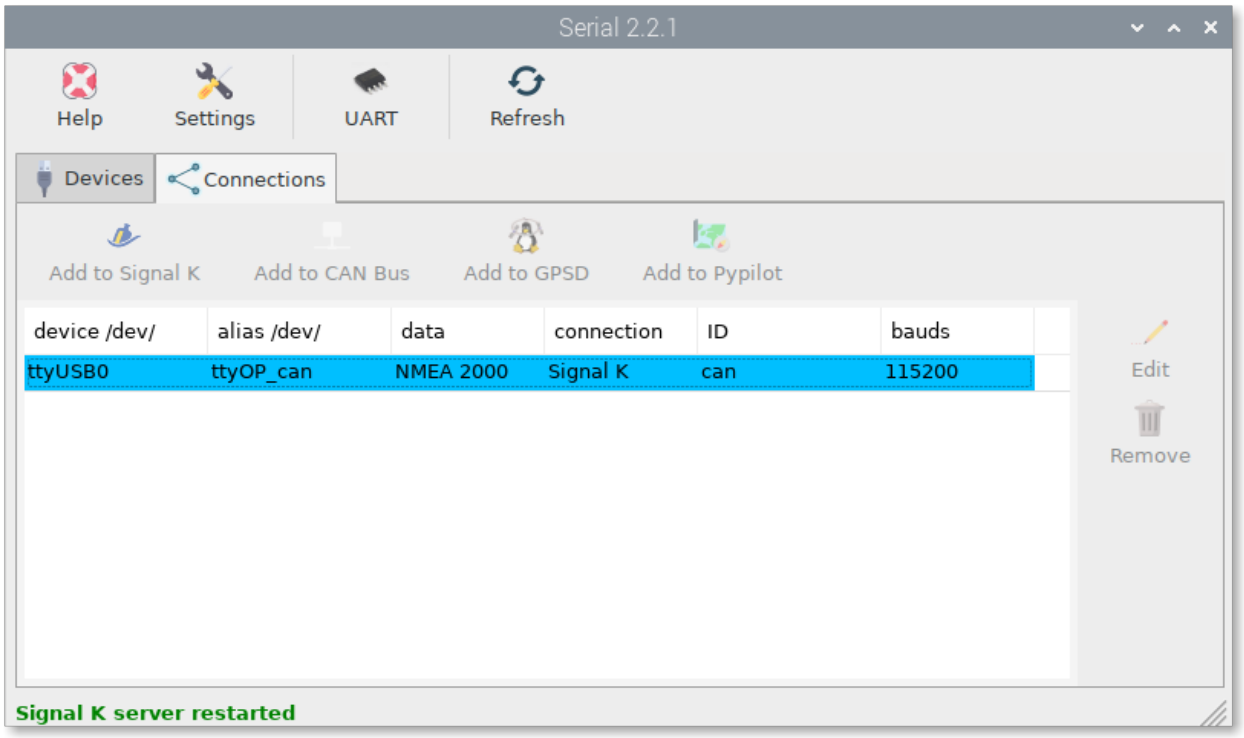


If you are using a *CANable* device click on `MANUAL` and go to *CAN Bus* chapter to learn how to configure this device.

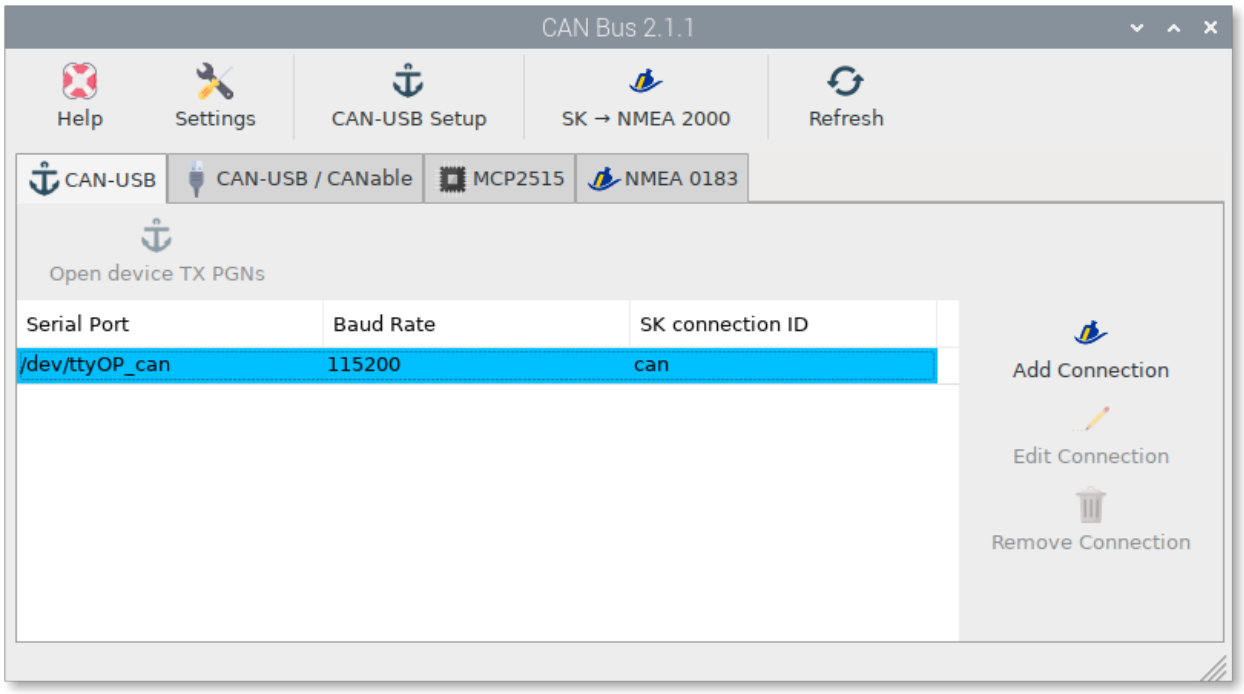
If you are using an *Actisense NGT-1* or an *OpenMarine CAN-USB Stick* (discontinued) device, select the `Baud Rate` (usually 115200) and click on `AUTO`.



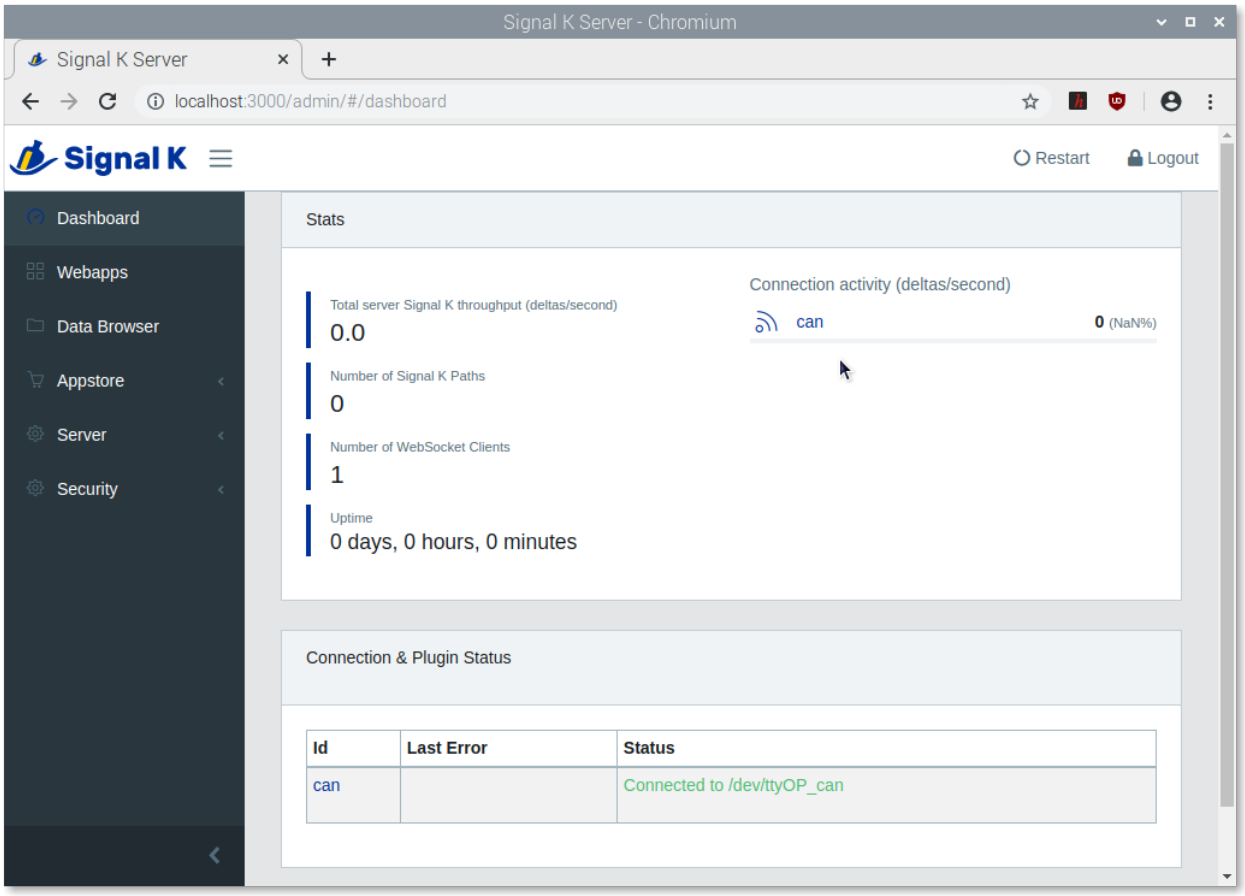
The device will be marked blue and you are done:



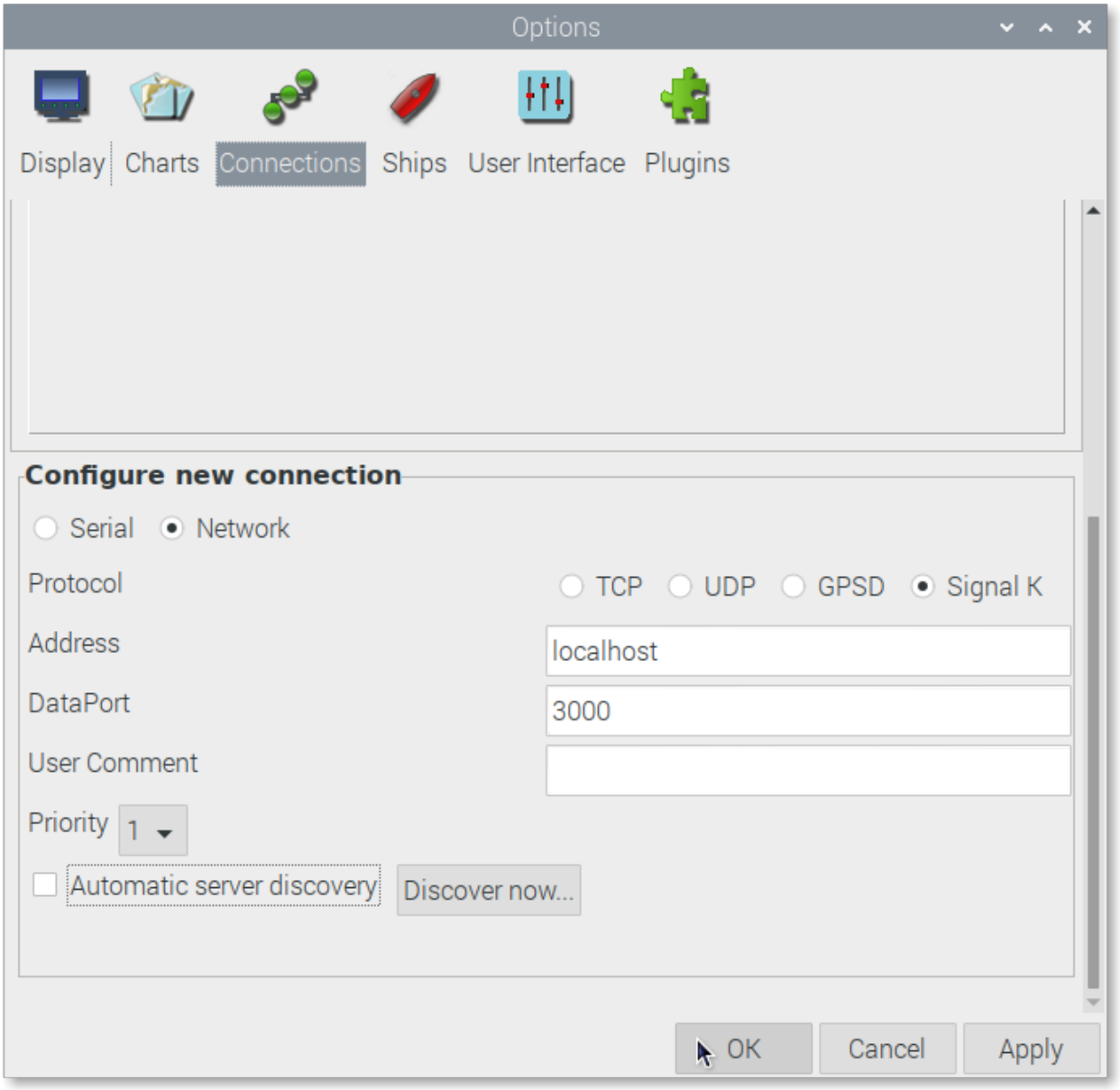
Open the `CAN Bus` app to confirm that the device has been added to the `CAN-USB` tab:



And go to Signal K server to confirm that the connection has been made:



Check OpenCPN to make sure there is a connection to the Signal K server and you are getting data from your NMEA 2000 network:

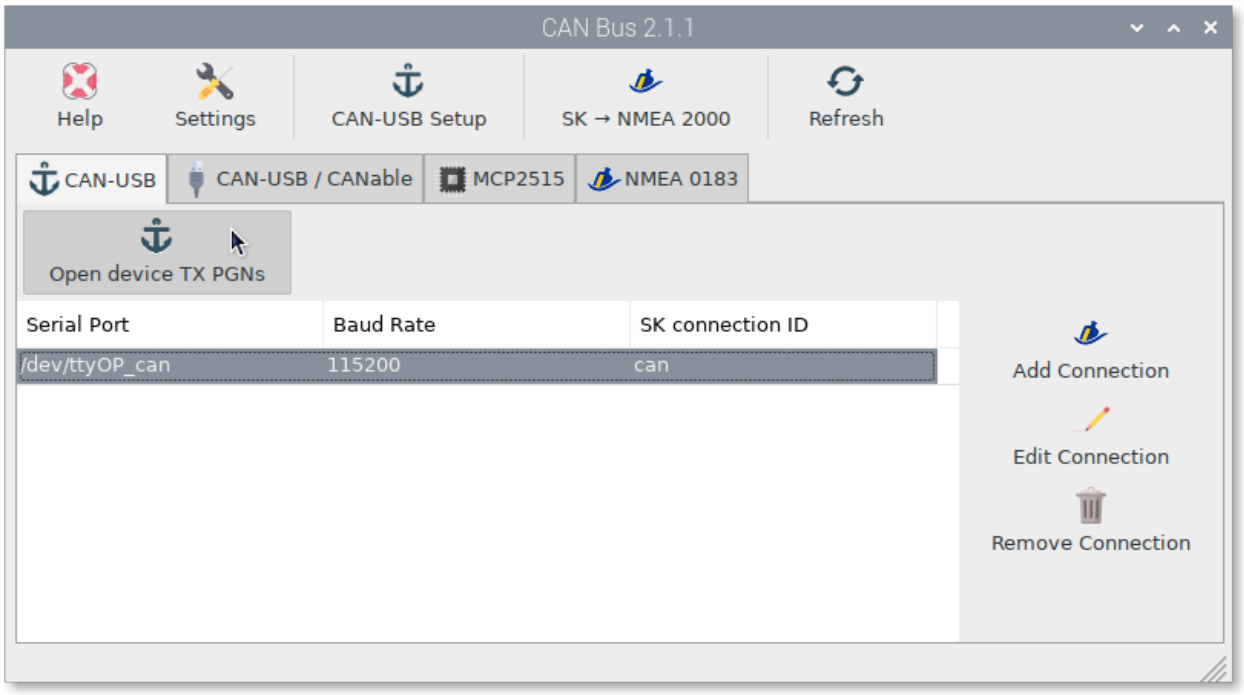


28.2.2 Input + output data

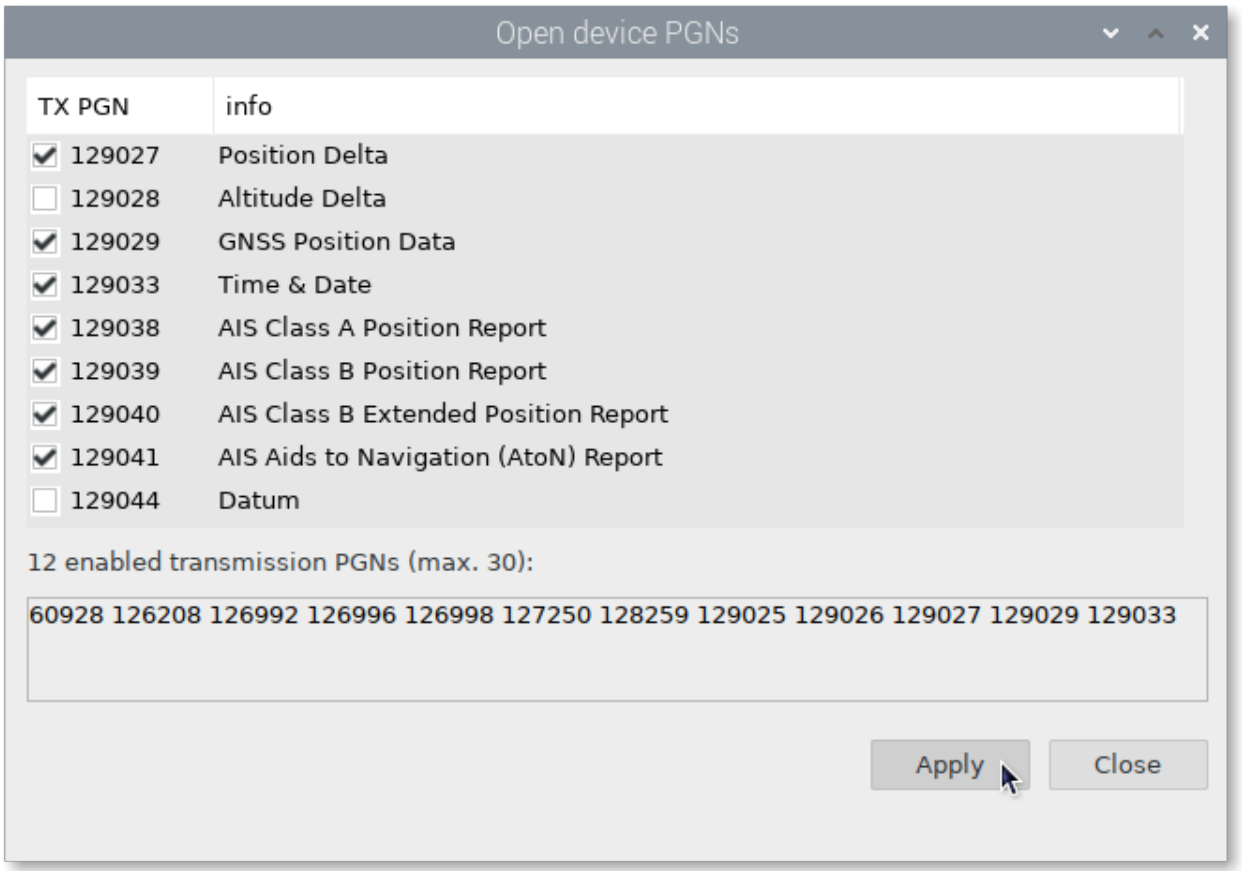
If you have any sensor in OpenPlotter sending data to the Signal K server, you can use the same USB-CAN converter to send this data to your NMEA 2000 network.

To protect your network, the *Actisense NGT-1* and the *OpenMarine CAN-USB Stick* (discontinued) devices have most PGNs blocked for transmission. On *CANable* devices, PGNs transmission is not blocked.

To unblock the PGNs you want to send to your NMEA 2000 network, go to `CAN Bus` app, select the device and click on `Open device TX PGNs`:

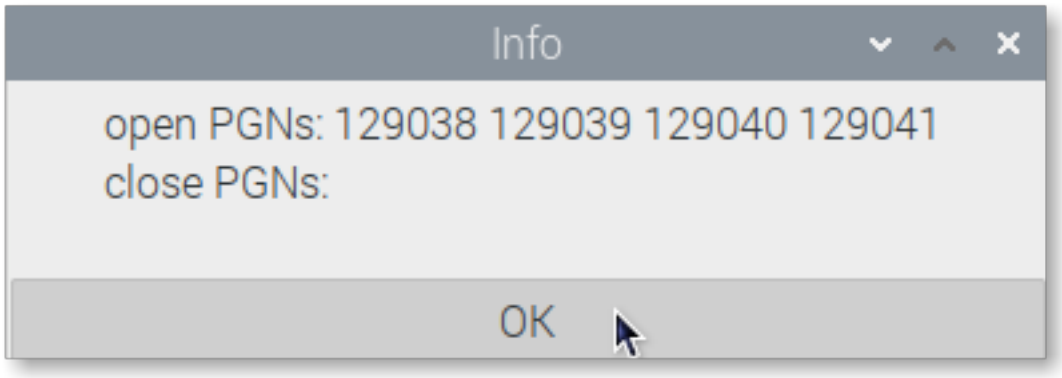


Enable the PGNs you want to unblock and click `Apply`:

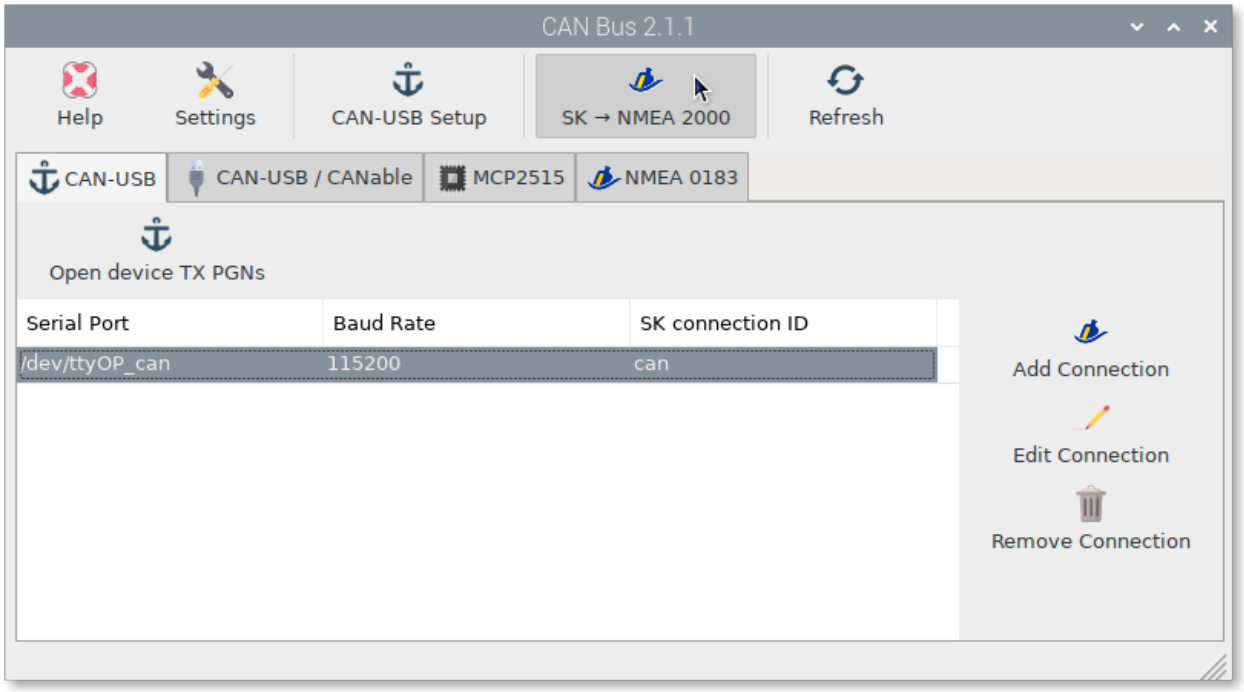


Note: If you see this message: *The list of enabled PGNs is empty, you may need to try a different baudrate or reset your device to 115200 bauds*, click on `CAN-USB Setup` to fix your device baud rate.

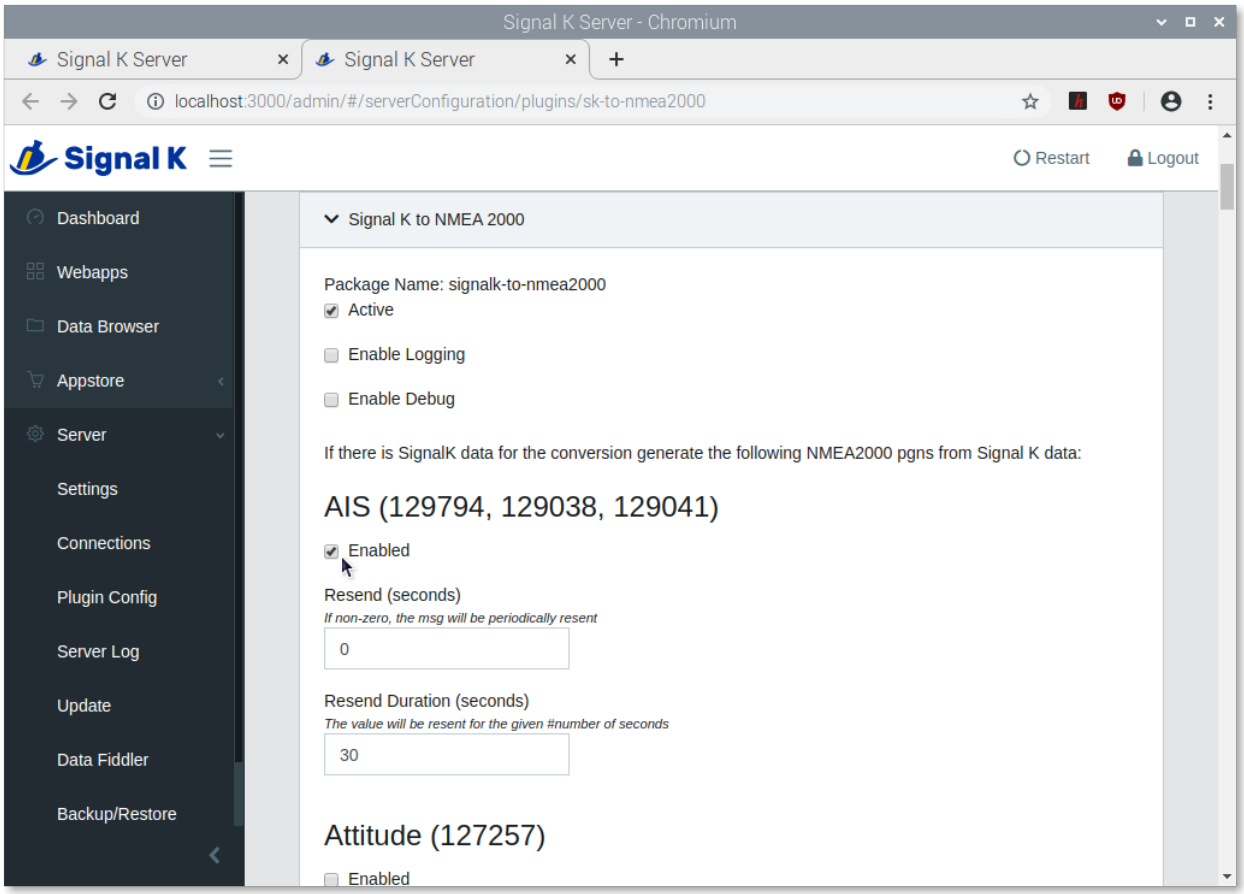
Click `OK` to write changes to the device:



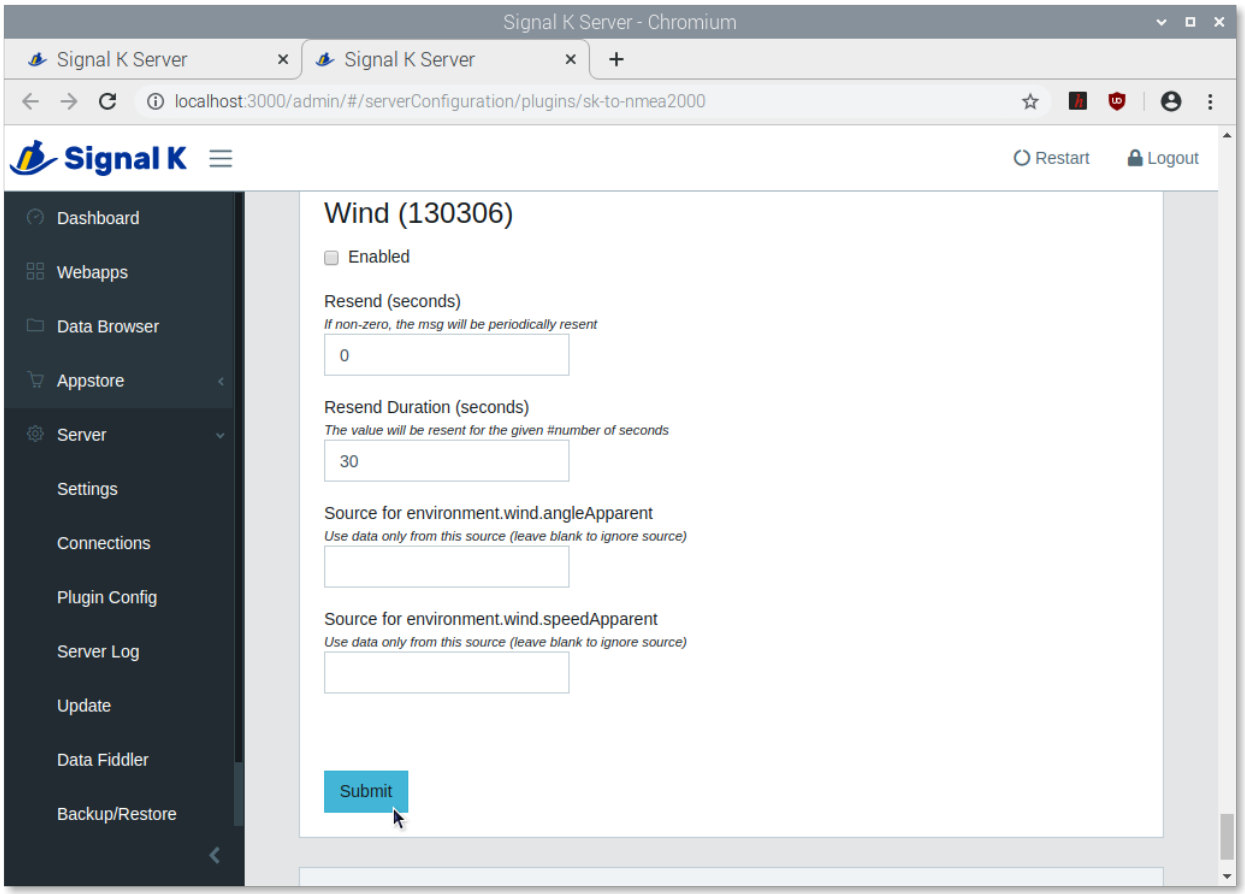
Finally, you have to tell the Signal K server what PGNs you need to convert from Signal K format to NMEA 2000 format (for any device model). To do this we use the plugin `Signal K to NMEA 2000`. Click on `SK → NMEA 2000` and you will be directed to the configuration page of this plugin:



Enable `Active` and the desired PGNs:



Click on `Submit` at the bottom of the page and you are done:


Signal K Server - Chromium
Signal K Server
Signal K Server
localhost:3000/admin/#/serverConfiguration/plugins/sk-to-nmea2000
Signal K
Restart
Logout
Dashboard
Webapps
Data Browser
Appstore
Server
Settings
Connections
Plugin Config
Server Log
Update
Data Fiddler
Backup/Restore
Wind (130306)
Enabled
Resend (seconds)
If non-zero, the msg will be periodically resent
0
Resend Duration (seconds)
The value will be resent for the given #number of seconds
30
Source for environment.wind.angleApparent
Use data only from this source (leave blank to ignore source)
Source for environment.wind.speedApparent
Use data only from this source (leave blank to ignore source)
Submit

28.3 Connecting the dAISy HAT

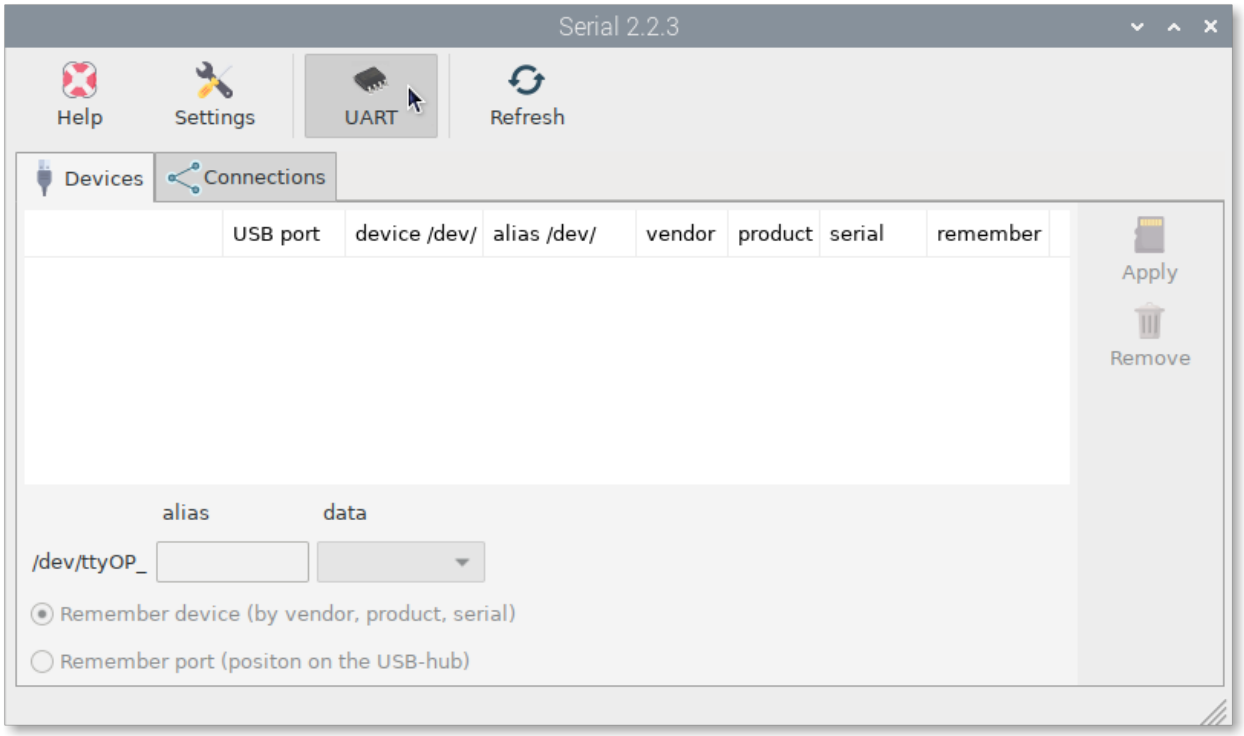
Specification

- True two channel receiver, continuously receiving on AIS channels A (161.975 MHz) and B (162.025 MHz)
- Superior sensitivity compared to other low-cost AIS receivers
- Low power, less than 200mW in receive mode (<40mA at 5V)
- 38400 baud serial output in industry standard NMEA format (AIVDM)
- Communicates with Raspberry Pi via UART0 (serial0)
- Works with Raspberry Pi 1 (A+/B+ only), Pi 2, Pi 3 and 4 (see note below), and Pi Zero
- Shape and size compliant with Raspberry Pi HAT standard
- Breakout pads for 2 independent TTL serial outputs, 3.3 and 5 volt rails, and Raspberry Pi I2C port
- SMA antenna connector
- SMA-to-BNC adapter and hex standoffs included

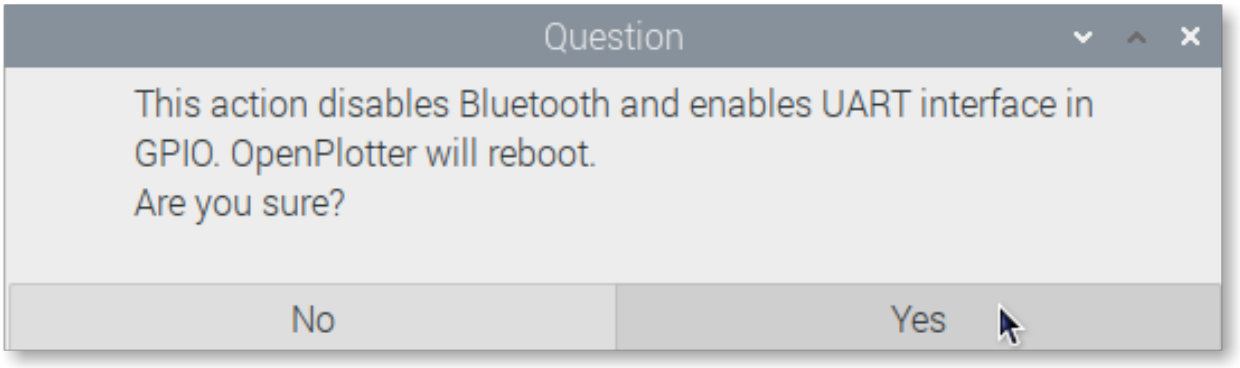
Note: This product is available in the OpenMarine Shop. Buying at OpenMarine Shop helps us keep the project alive. On the original product page you will find the full specification and a better choice for US buyers.

Configuration

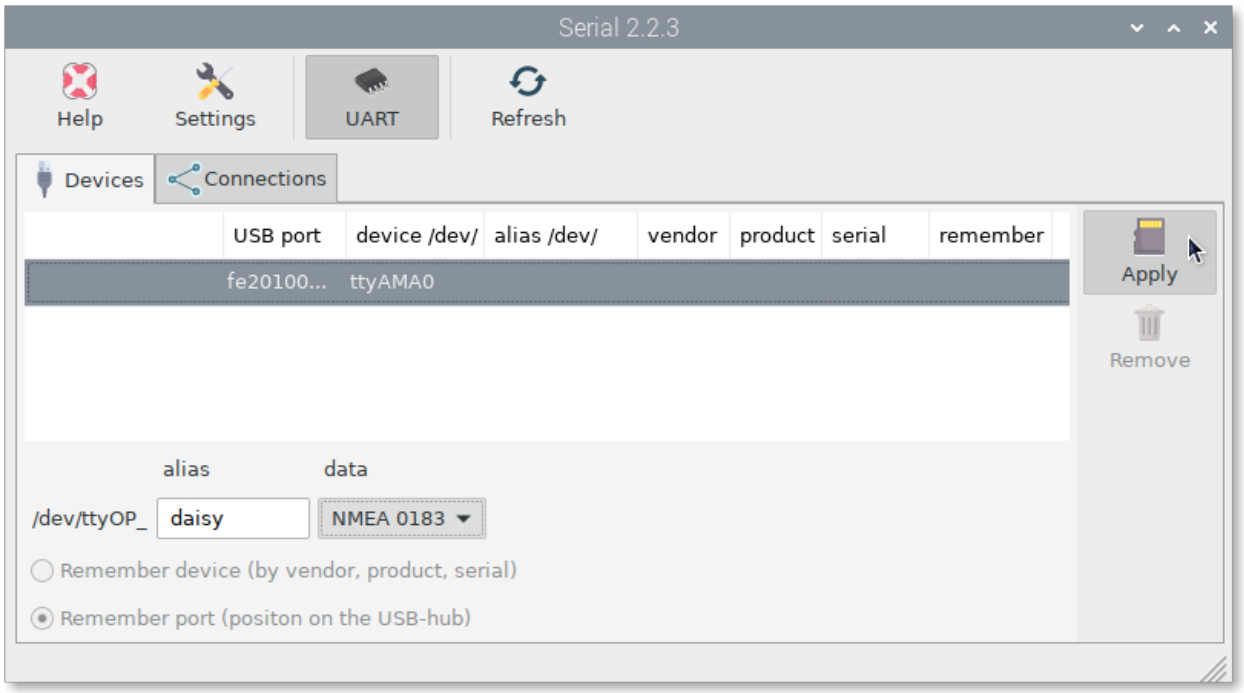
Mount the dAISy HAT in your Raspberry Pi and enable the serial port on the GPIO header of the Raspberry Pi by clicking the `UART` icon:

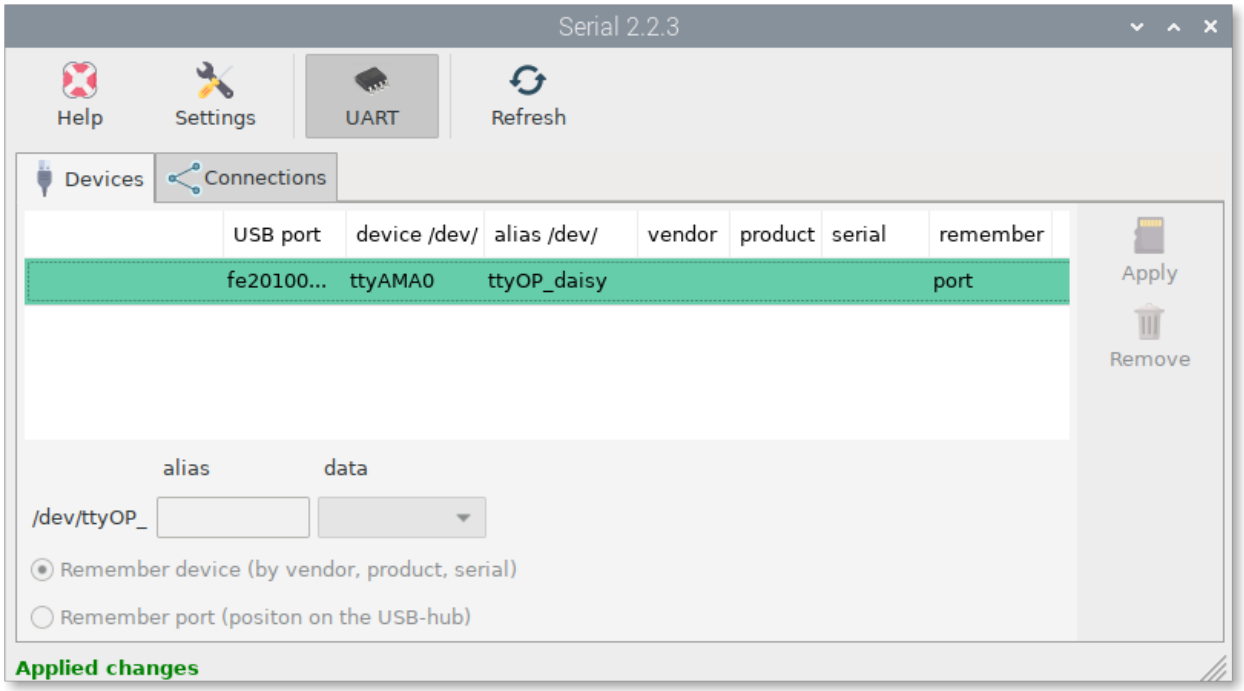


Acknowledge the warning, and reboot the Raspberry Pi:

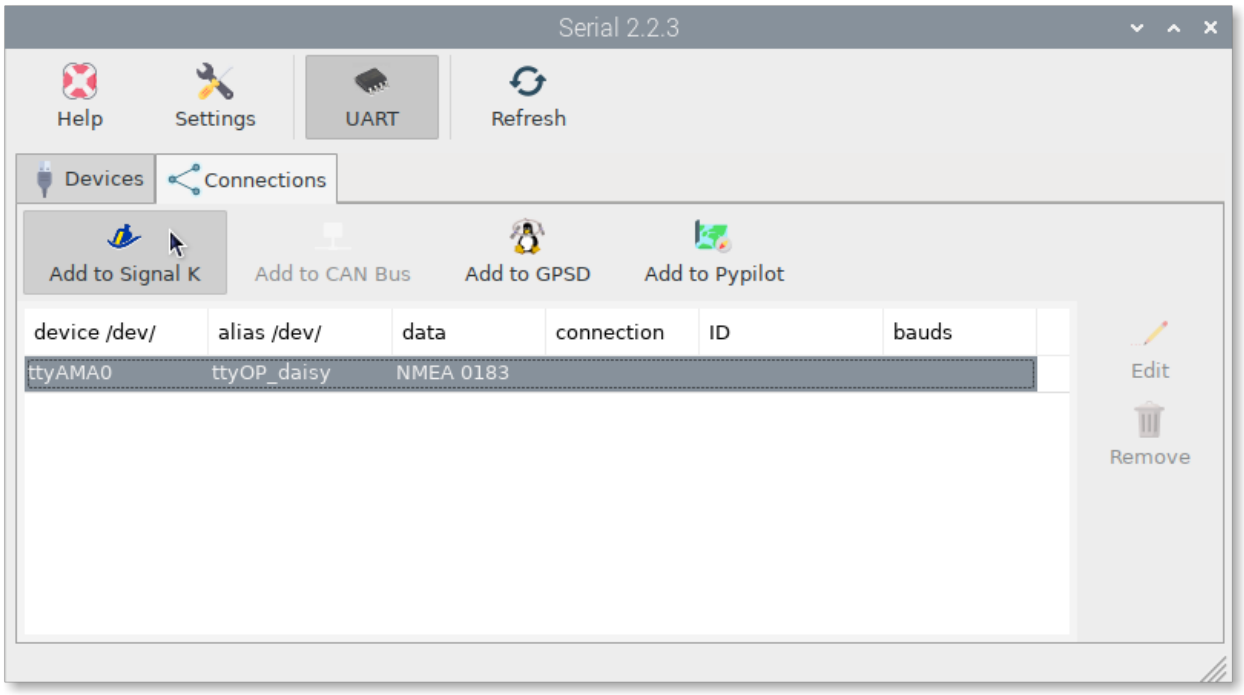


After the reboot, launch the *OpenPlotter Serial app* again. On the *Devices* tab, you should now see an entry *ttyAMA0*. Select the line with *ttyAMA0* and give it an alias (for example daisy) and select *NMEA 0183* from the data dropdown, then press `Apply`:

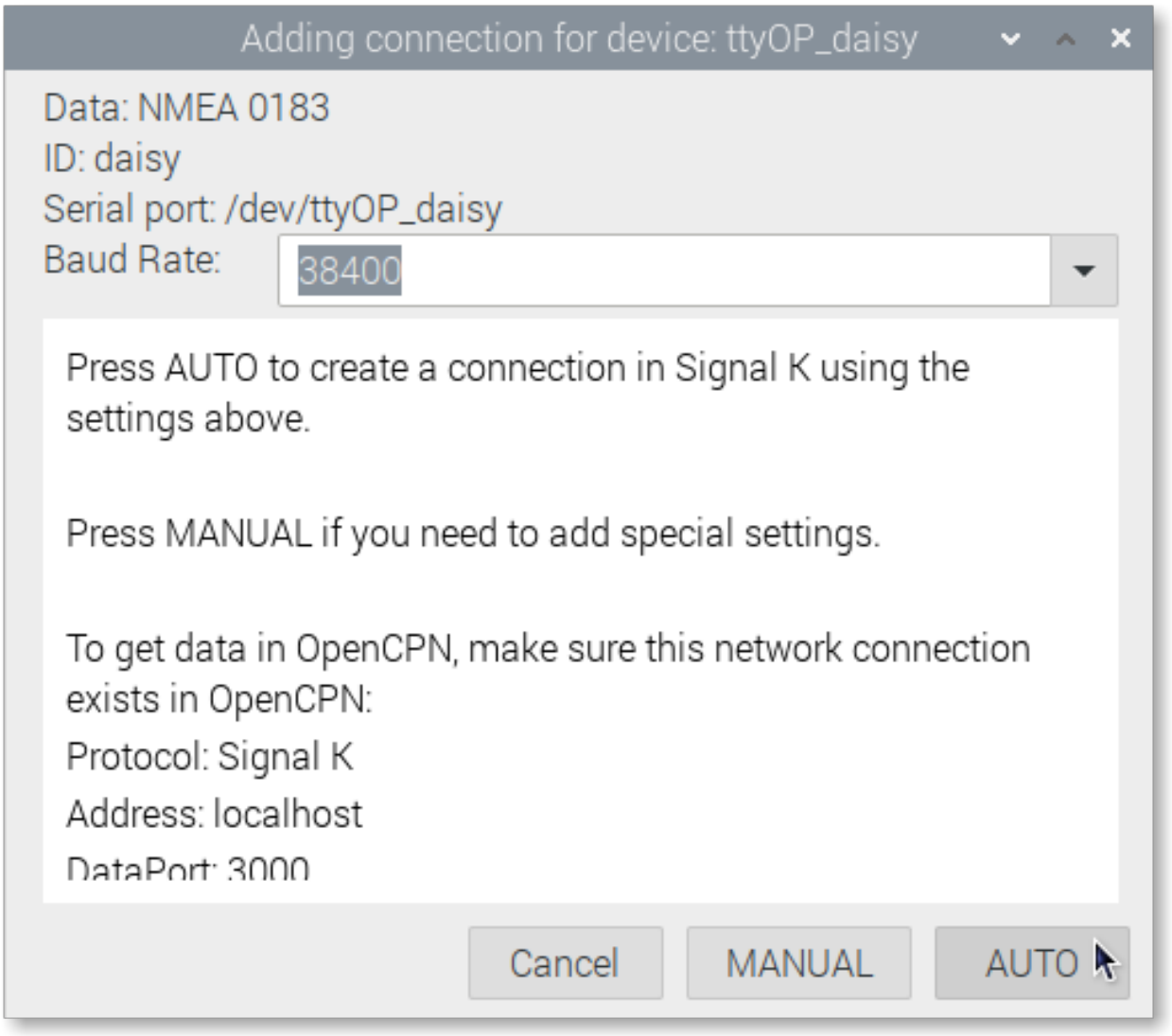




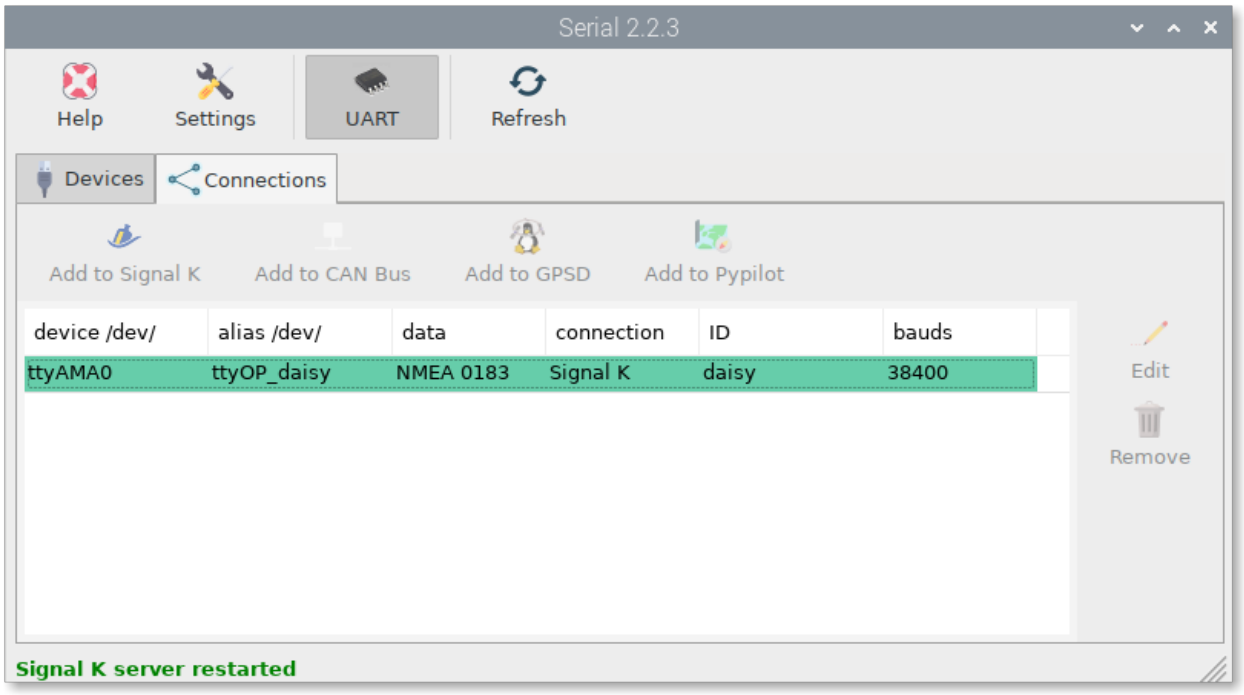
We now need to connect the *ttyOP_daisy* device with the Signal K server, the central data processing hub of OpenPlotter. Switch to the *Connections* tab, select the *ttyOP_daisy* device and click `Add to Signal K`:



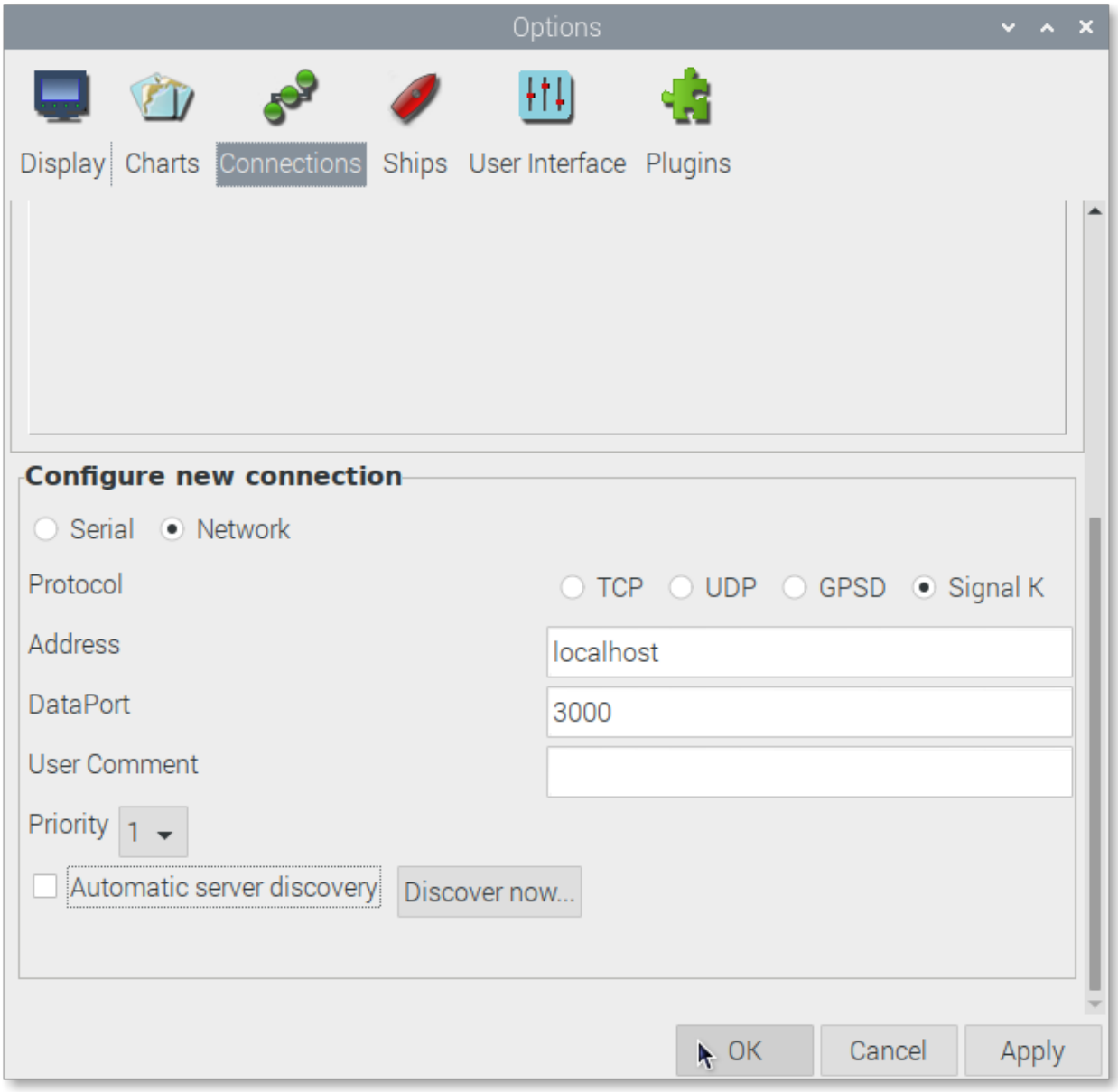
From the *Baud Rate* dropdown menu select *38400*, then press `AUTO`:

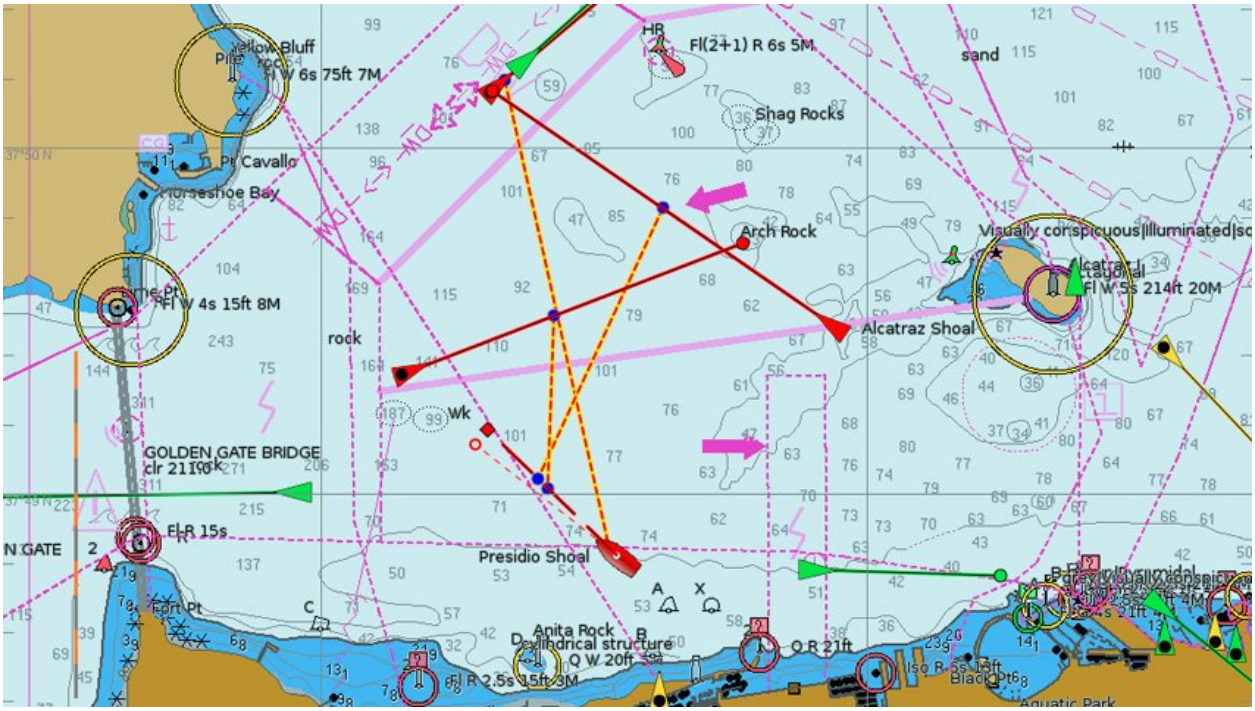


The Signal K server and applications connected to it, like OpenCPN, should now receive AIS data:



Check OpenCPN to make sure there is a connection to the Signal K server and you are getting data from your dAISy HAT:

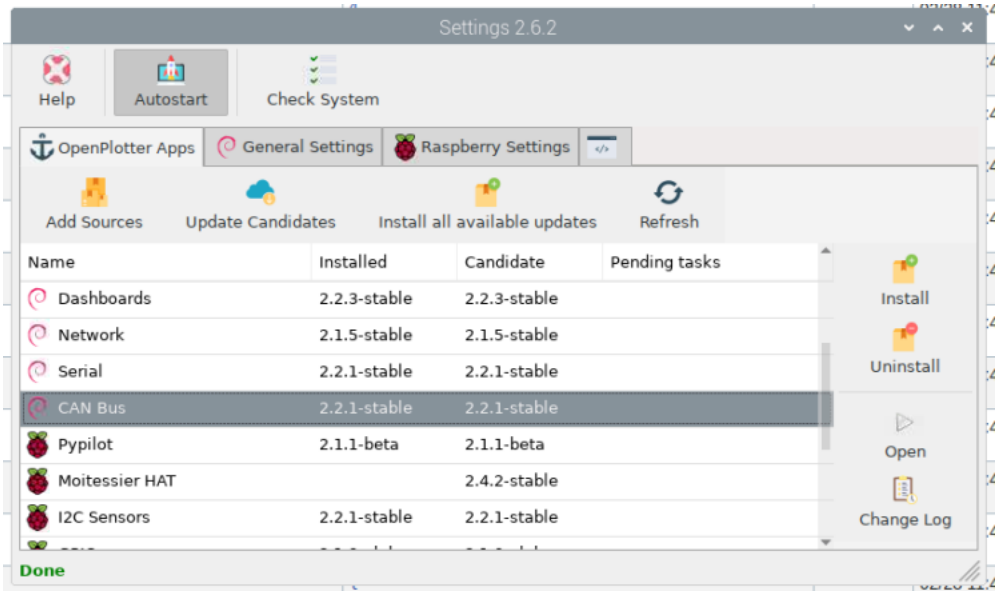

Options
Display
Charts
Connections
Ships
User Interface
Plugins
Configure new connection
Serial
Network
Protocol
TCP
UDP
GPSD
Signal K
Address
localhost
DataPort
3000
User Comment
Priority
1
Automatic server discovery
Discover now...
OK
Cancel
Apply


Yellow Bluff
Fl W 6s 75ft 7M
Pt Cavallo
Horseshoe Bay
Fl W 4s 15ft 8M
rock
GOLDEN GATE BRIDGE
clr 211ft
Fl R 15s
N GATE
2
Fort Pt
C
Presidio Shoal
Anita Rock
Cylindrical structure
Q W 20ft
Fl R 2.5s 15ft 3M
Q R 21ft
Iso R 5s 14ft
Black Pt
Aquatic Park
HR
Fl(2+1) R 6s 5M
Shag Rocks
sand
Arch Rock
Alcatraz Shoal
Visually conspicuous illuminated sc
Alcatraz I
Fl W 5s 214ft 20M
Wk

CHAPTER 29

CAN Bus

Go into settings (–> openplotter –>) and ensure that the CAN Bus App is installed and the latest



29.1 CAN_USB

To be added

29.2 Slcand

To be added

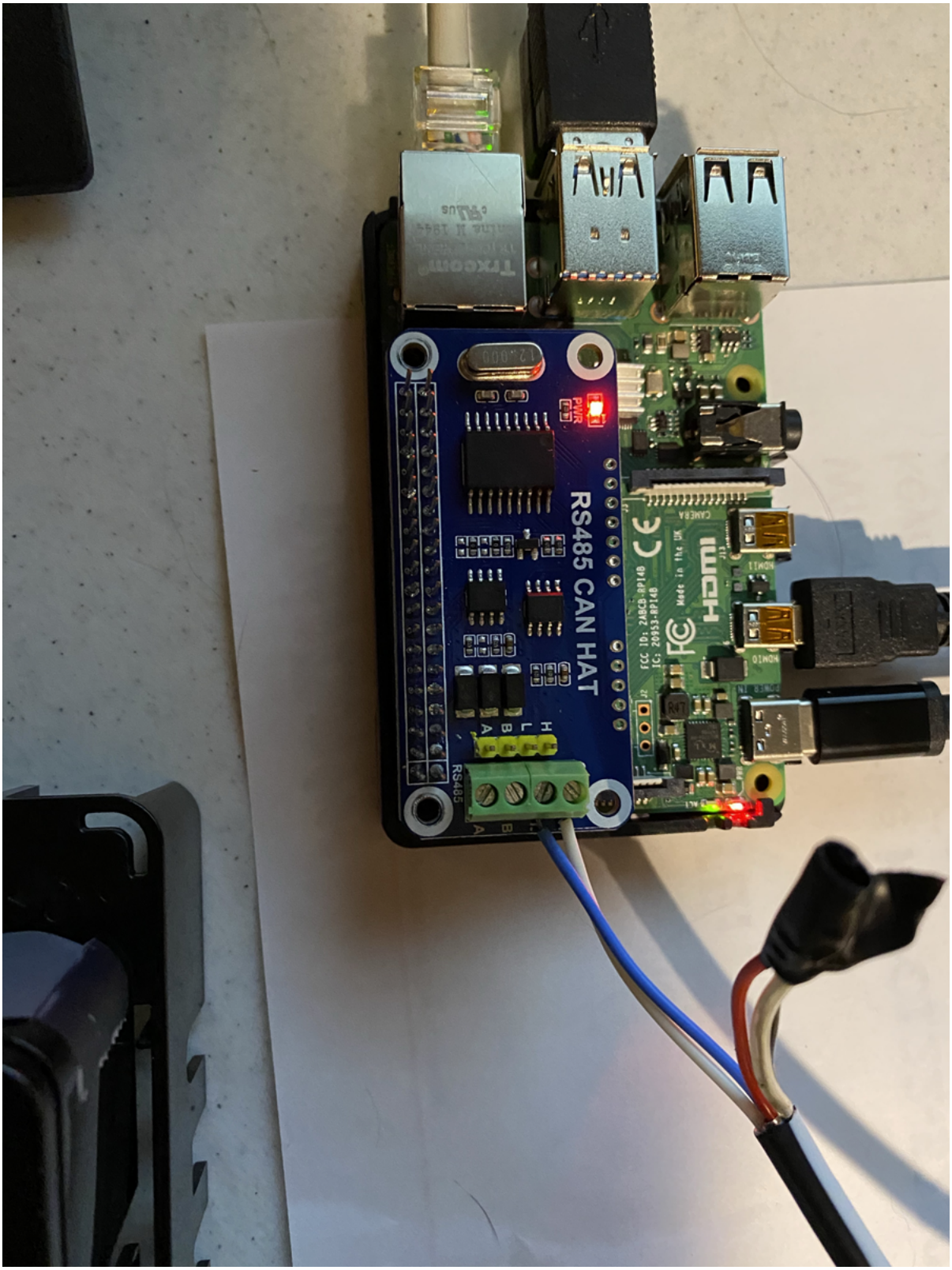
29.3 MCP2515

There are some boards available with the MCP2515 chip on board that can be added into openplotter below is a list of the ones that work:

- Waveshare RS485 CAN HAT - https://www.amazon.com/gp/product/B07VMB1ZKH

29.4 Physical install

Shutdown the Pi and install the Can Board (if it is a CAN Hat). Connect the Seatalk NG Can H and Can L or the NMEA2000 (N2K) Can H and Can L to the H and L

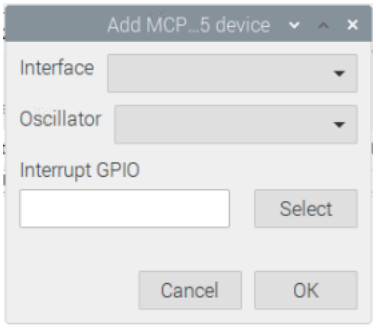


Note: The Seatalk NG bus needs to be powered - this would normally already be so if you have other devices on the bus but some Multi-function Displays need the bus powering, as well as the device - the power does not connect to the

HAT in any way.

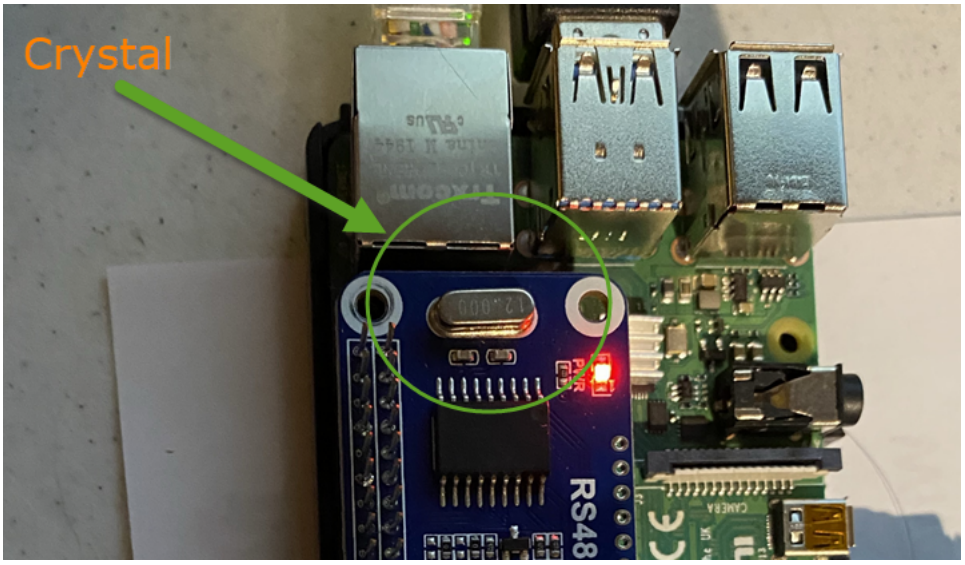
Note: Make sure the CAN network is appropriately terminated. CAN network termination is required to reduce reflection and consists of 60 Ohm resistors at the end points of the CAN networks - some devices have these built in and the Raymarine network has these already if you are just plugging in to an existing network

Reboot the Pi and open the CAN Bus app. In the app go to the MCP2515 tab and press the 'Add MCP2515 device' button.

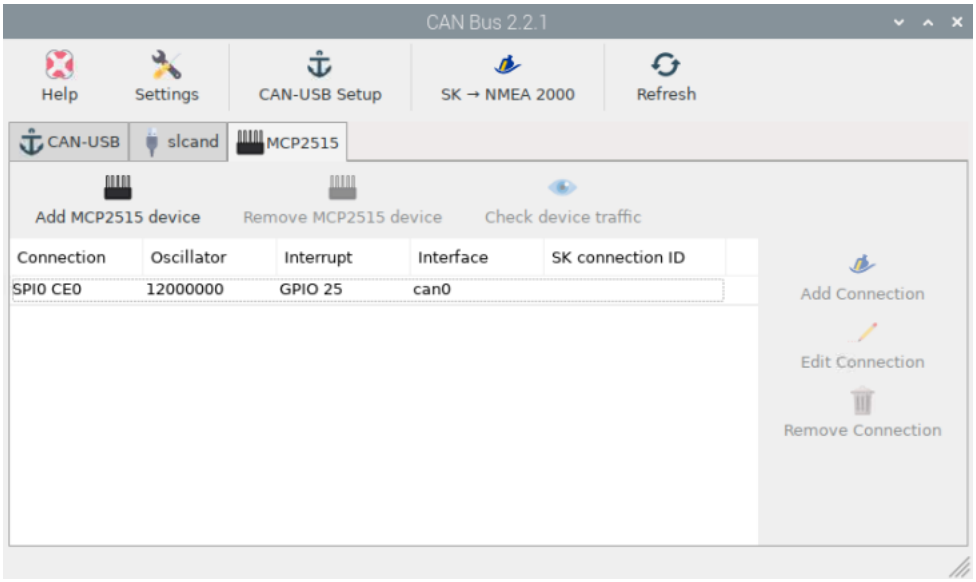


Pick the correct interface, Oscillator and select the correct GPIO pin.

On the Waveshare RS485 CAN HAT, this would be SPIO CE0 for the interface, the oscillator can be found by looking at the crystal chip on the HAT as per the pic below. The Interrupt GPIO is GPIO 25, pin 22



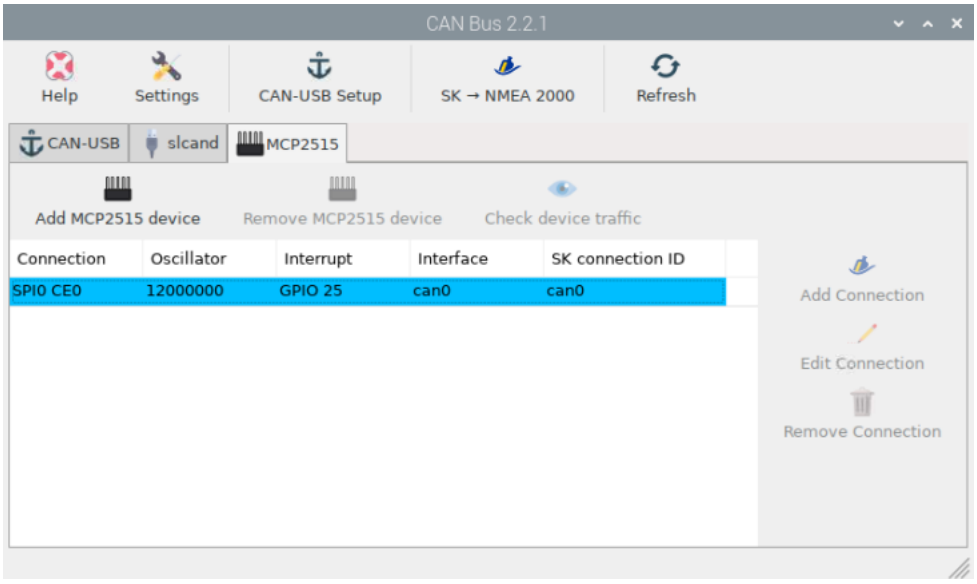
Once accepted the CAN bus app should like the pic below



At this point the device is set up but not connected to Signal K. By selecting the device and then pressing the 'Check device traffic' button, you will see a terminal and the data running through the device.

29.5 Connect to Signal K

To connect to Signal K, Select the device and press the 'Add Connection' button. You will see Signal K restart and the connection ID will be added



If you go into SignalK and there is data on the bus you will see it in the signalK Data Browser. See an example below

29.6 Signal K Output

We need to select which data is sent out by SignalK on the CAN bus. This data will be sent out to the MFD and other devices on the CAN bus. Press the 'SK –> NMEA 2000' button and select the items you want to send out. Press submit and make sure the 'SignalK to NMEA 2000' plugin is enabled.

CHAPTER 30

GPIO

To be added

30.1 Connecting to Signal K server

To be added

30.2 Connecting sensors

There are some considerations to take into account when connecting this type of sensor to our board. Most of them need an element called a pull up or pull down resistor.

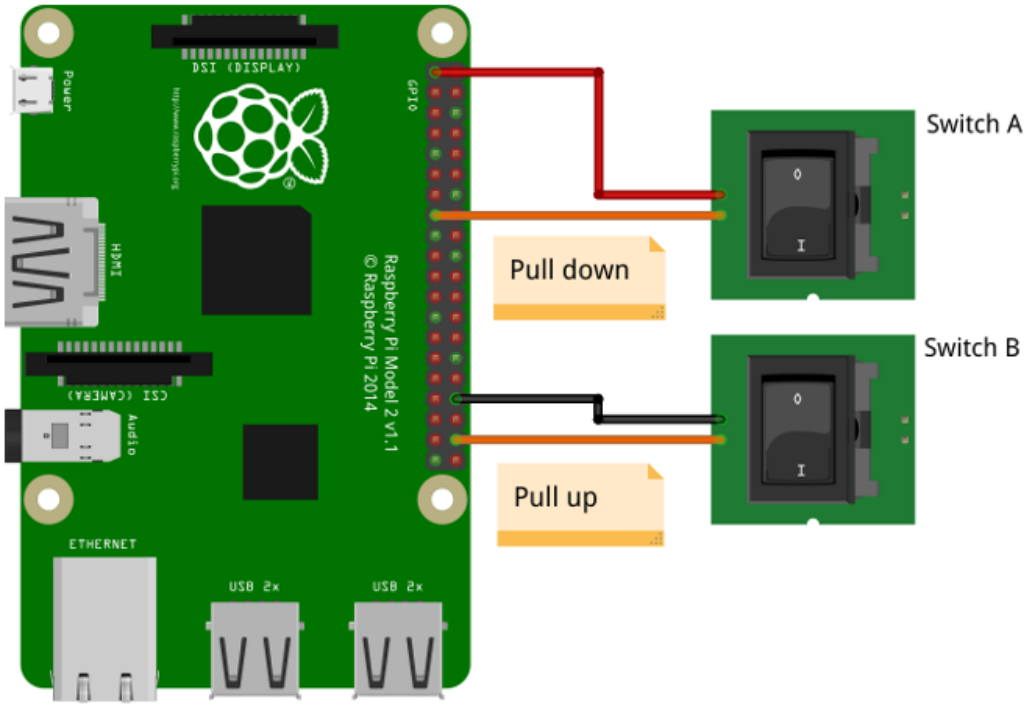
In electronic logic circuits, a pull up resistor or pull down resistor is a resistor used to ensure a known state for a signal. It is typically used in combination with components such as switches and transistors, which physically interrupt the connection of subsequent components to ground or to VCC. When the switch is closed, it creates a direct connection to ground or VCC, but when the switch is open, the rest of the circuit would be left floating (i.e., it would have an indeterminate voltage). For a switch that connects to ground, a pull-up resistor ensures a well-defined voltage (i.e. VCC, or logical high) across the remainder of the circuit when the switch is open. Conversely, for a switch that connects to VCC, a pull-down resistor ensures a well-defined ground voltage (i.e. logical low) when the switch is open.

Fortunately, the Raspberry Pi incorporates internal resistors that can be defined by software and you can directly connect the sensors to the pins of the Raspberry Pi.

30.2.1 Internal pull resistors

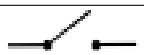
You can connect `digital` and `pulse` sensors using either external or internal pull resistors. To simplify your installation we recommend using internal pull resistors.

You have to connect one terminal of the sensor to any GPIO of your choice and you can choose between the GND pins or the 3.3V pins to connect the other sensor terminal. This is an example of 2 common switches connected in the two ways:



Danger: Never connect a digital or pulse sensor to the 5V pin.

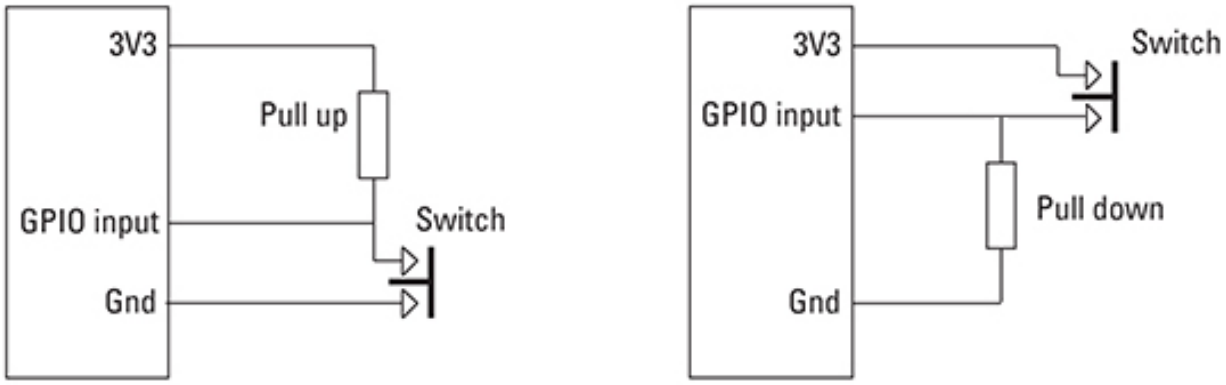
OpenPlotter will read the status of each sensor by sending 1 or 0 to the Signal K server according to how it has been connected and configured:

Settings	connected to GPIO + GND	connected to GPIO + 3.3V
none	external pull up resistor	external pull down resistor
pull up	 = 1 = 0	No data
pull down	No data	 = 0 = 1

See the chapter dedicated to each type of sensor for more information about the configuration.

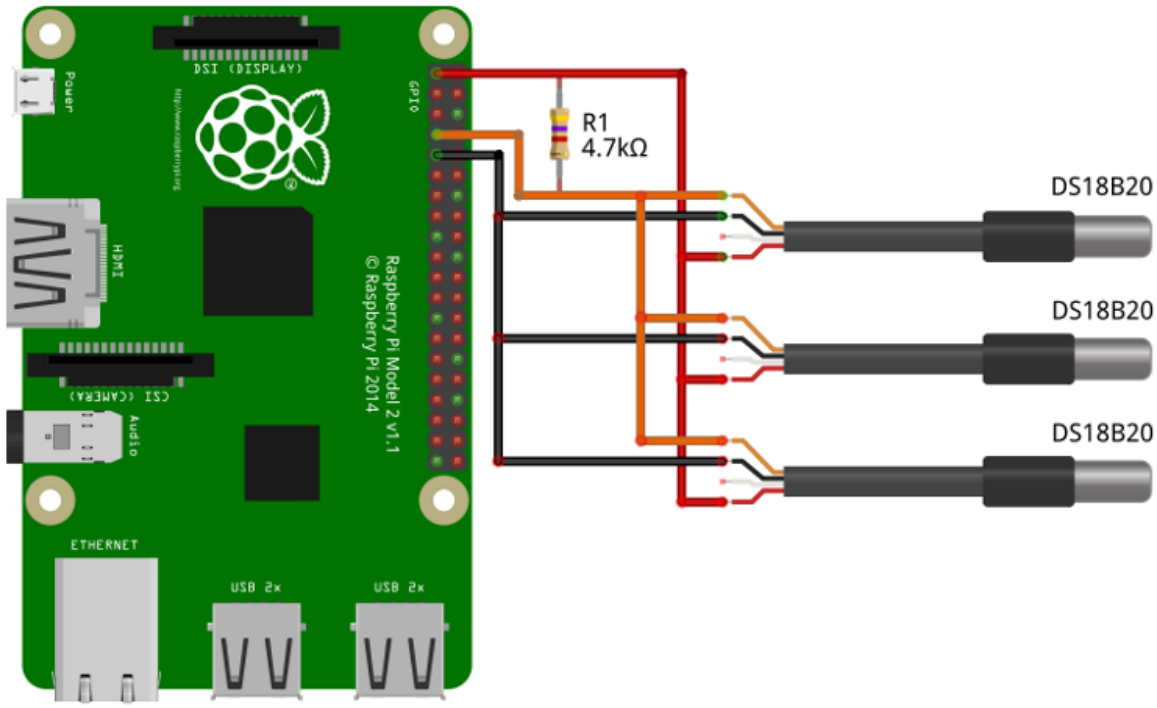
30.2.2 External pull resistors

There are some cases where it is necessary to use external pull resistors. The pull resistor value can be from 4.7KΩ to 10KΩ depending on the sensor specifications. This is an example of the connection of both methods:



This is the case for most `1W` sensors. The internal pull resistor of the Raspberry Pi is around 50KΩ and is too high for the proper functioning of these sensors. The specifications indicate that you have to use a 4.7KΩ pull up resistor and that you can connect multiple sensors using a single resistor.

Multiple 1W sensors (DS18B20) connection example using a single pull up resistor:



Danger: Never connect a 1W sensor to the 5V pin.

30.2.3 Seatalk-1

To be added

CHAPTER 31

Digital

CHAPTER 32

Pulse

CHAPTER 33

1W

CHAPTER 34

Seatalk-1

CHAPTER 35

Pypilot

CHAPTER 36

Compass calibration

Follow these steps in order:

36.1 1. Accelerometer bias

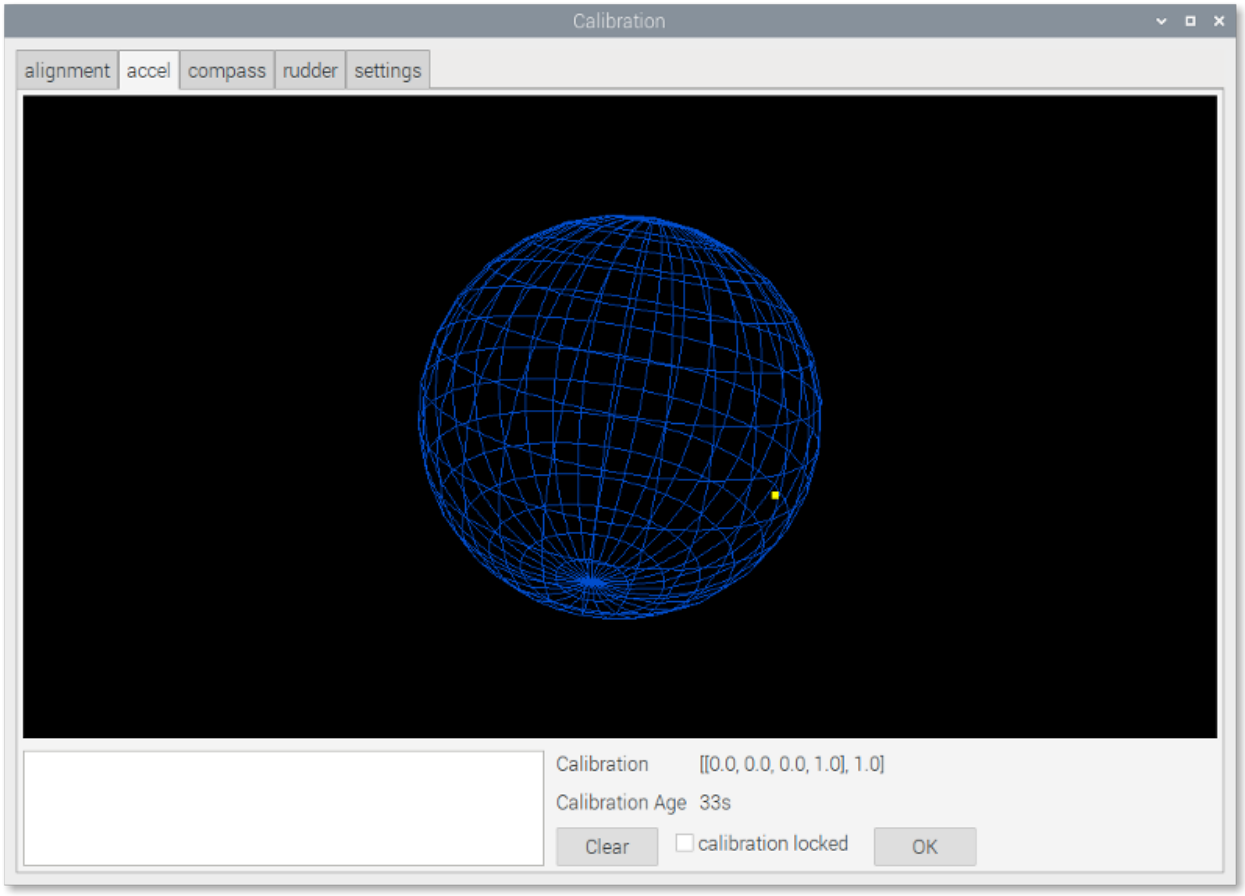
Most IMU require accelerometer bias calibration. Without it, there will be significant pitch and roll errors. The mpu9250 and mpu9255 usually are factory calibrated meaning you could skip this step. However, some of these do not have this calibration, these usually have an orange rather than yellow capacitor. The IMU on the Moitessier HAT should be ok. In any case, it is recommended to calibrate the accelerometer bias, even if factory calibrated as it will improve the factory calibration slightly.

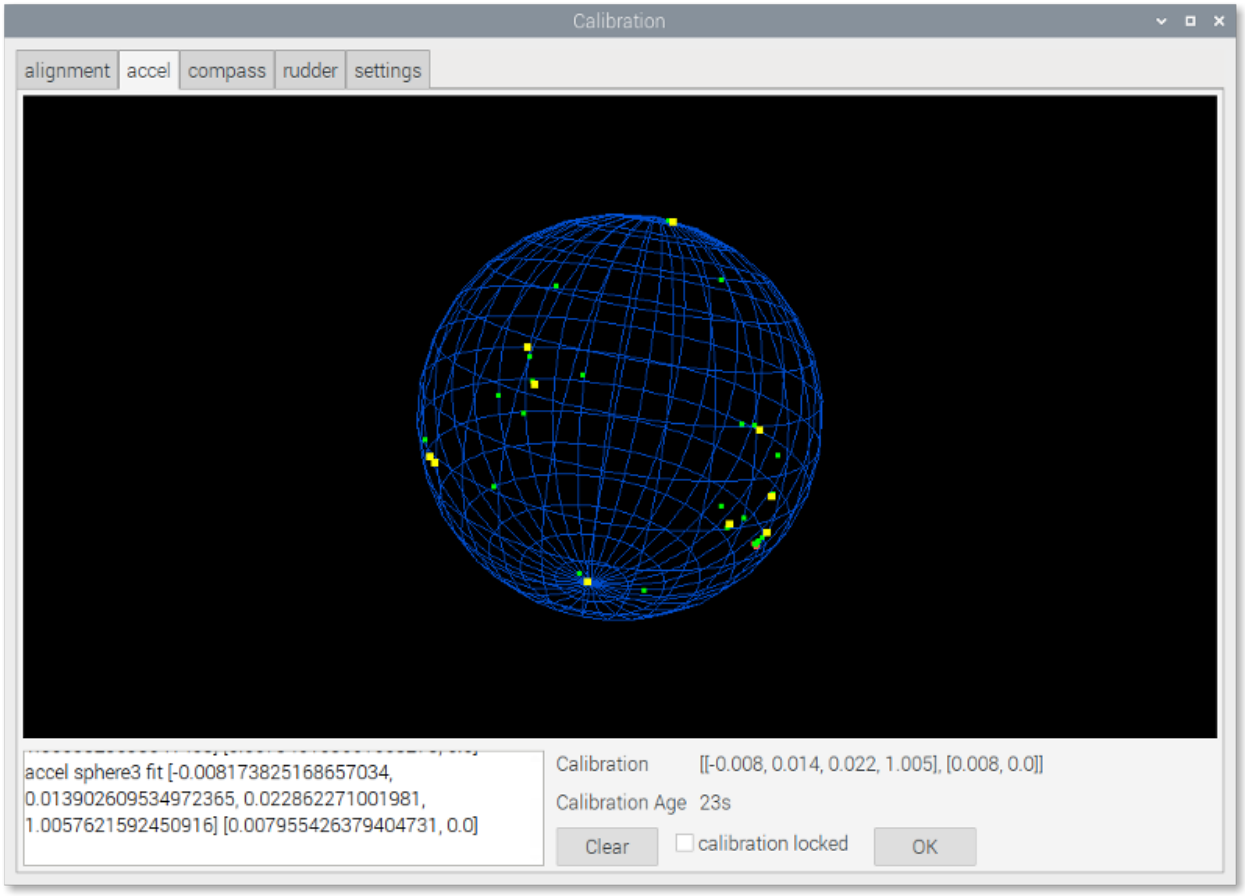
To calibrate the accelerometer bias, you must be on a "mostly" stable platform. It may be impossible to do at anchor if the boat is moving too much, so either in flat water, or land for this step.

Go to `Pypilot app` and click on `Calibration`. In Calibration window click on `accel` tab. Make sure `calibration locked` is not enabled.

Carefully place the sensor on each of the 6 sides of a box (+- 10 degrees will do) the actual orientation is not critical, so long as enough measurements can be taken to fit a sphere. Leave the sensors in each position for a few seconds.

Once a calibration is applied the accelerometer `Calibration Age` should reset and fit points become yellow. If it does not, repeat the process putting the sensors in different orientations until a calibration fix is found.


Calibration
alignment
accel
compass
rudder
settings
Calibration [[0.0, 0.0, 0.0, 1.0], 1.0]
Calibration Age 33s
Clear
calibration locked
OK

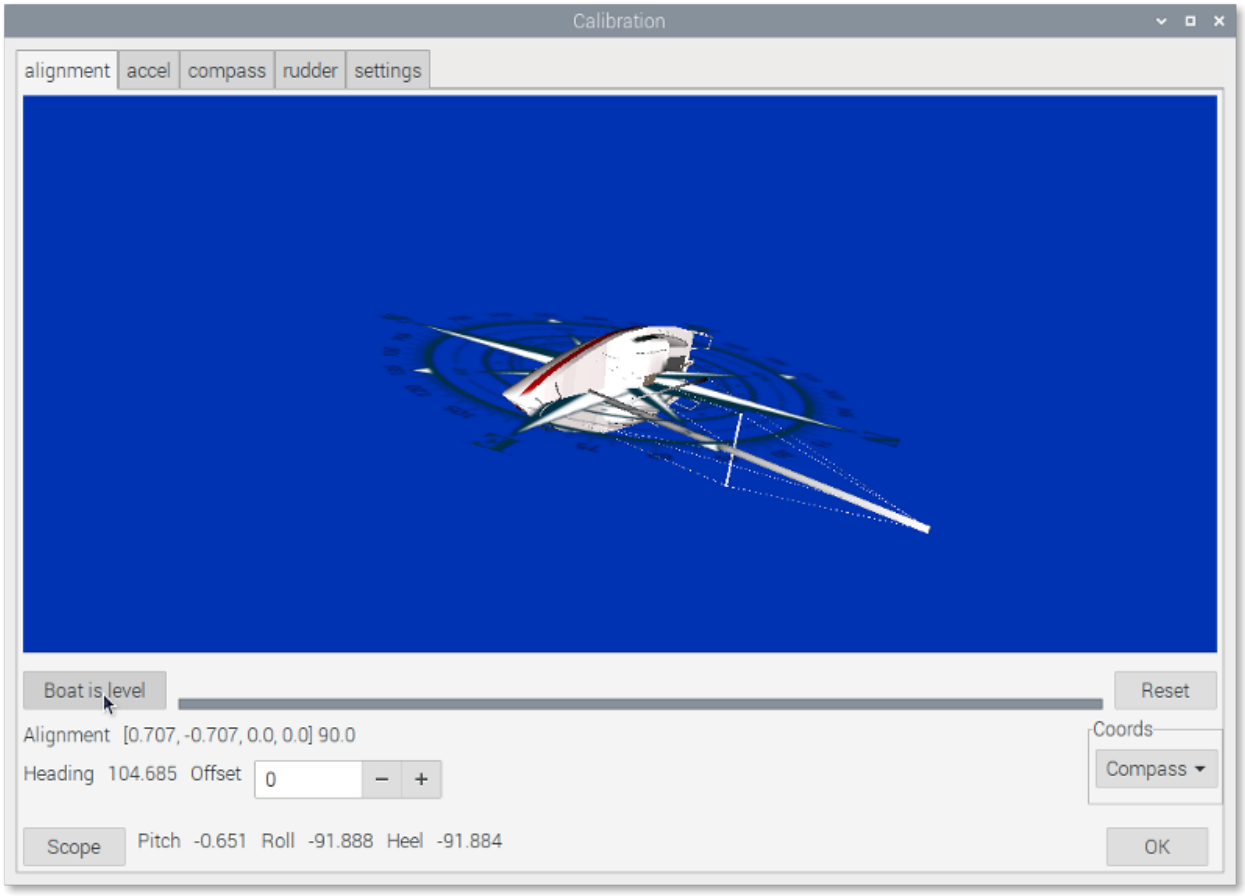


If you use the cheapest sensors, sometimes they have bad accelerometers. Either one axis will always read zero, or they will saturate because the bias is greater than 1g. This is easy to determine from the accelerometer calibration plot in calibration window.

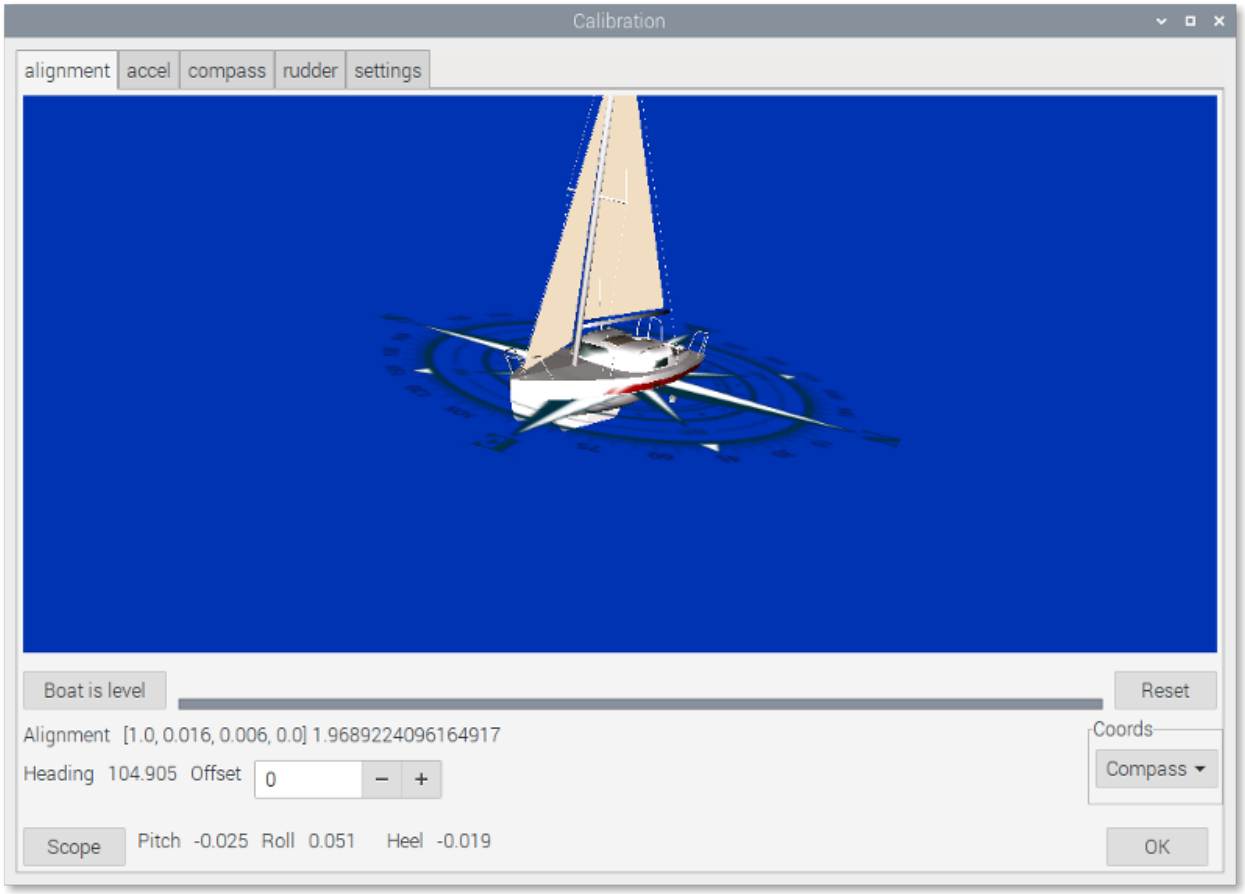
36.2 2. Alignment

Once the accelerometers are calibrated, the sensor should be fixed securely to the boat. Alignment and compass calibration are required for correct operation. If sensors are moved or remounted, this must be performed again (but not accelerometer calibration).

To perform alignment, ensure the boat is level (not heeling or pitching) and in relatively calm water (small waves motion of a few degrees is ok). Go to `alignment` tab and click `Boat is level` button.



Correct alignment must be performed before the compass calibration can begin.



36.3 3. Compass

Be sure to locate the sensors away from:

- magnets - speakers and especially moving magnets like floating compasses
- current carrying wires - very simple rule is 2 cm (1 inch) for every amp
- iron and steel - less critical. If you are in a steel boat, just do not fix the sensors to a steel wall and try to locate them several inches at least offset from it.

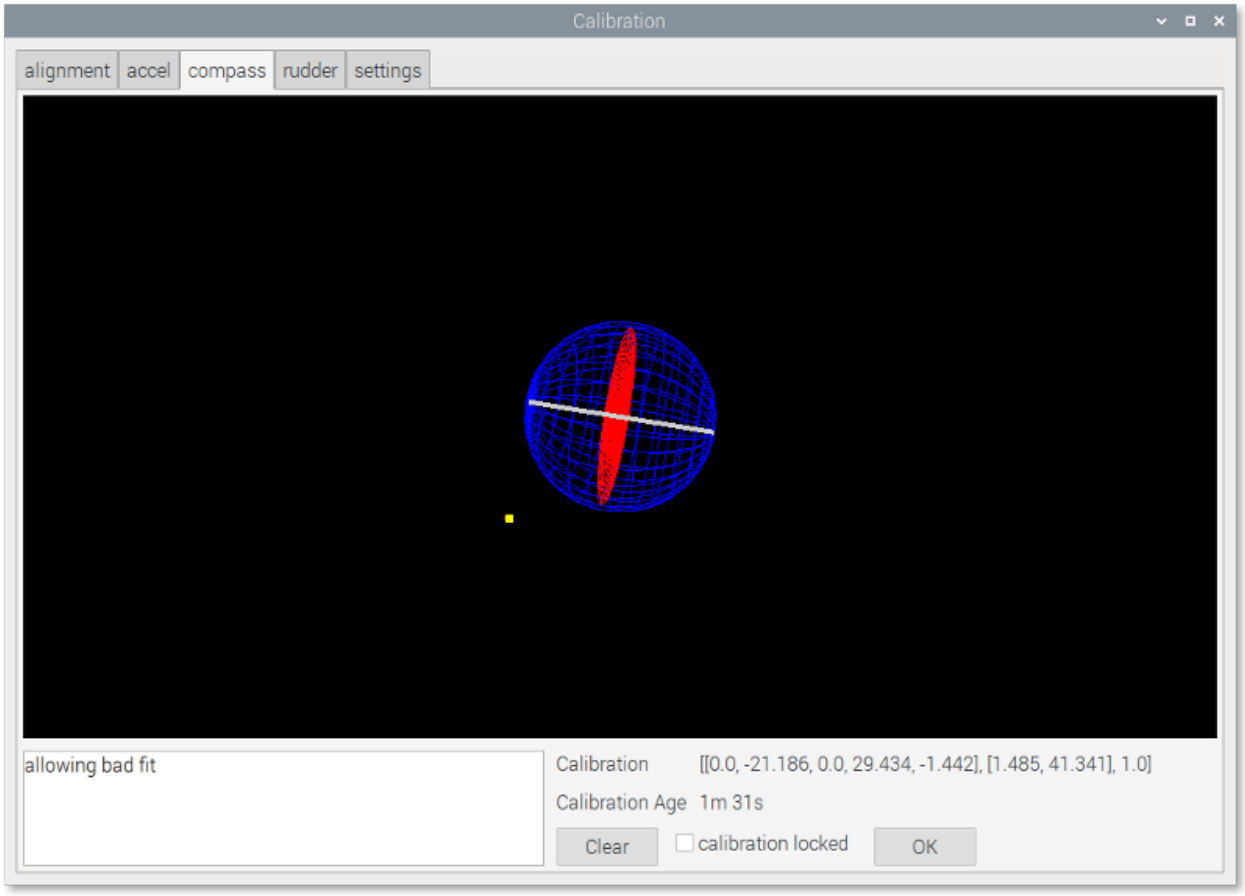
The compass calibration is mostly automatic. If the accelerometer and alignment are calibrated, you just need to sail turning more than 180 degrees to calibrate the compass.

Go to `compass` tab and make sure `calibration locked` is not enabled or updates will not occur.

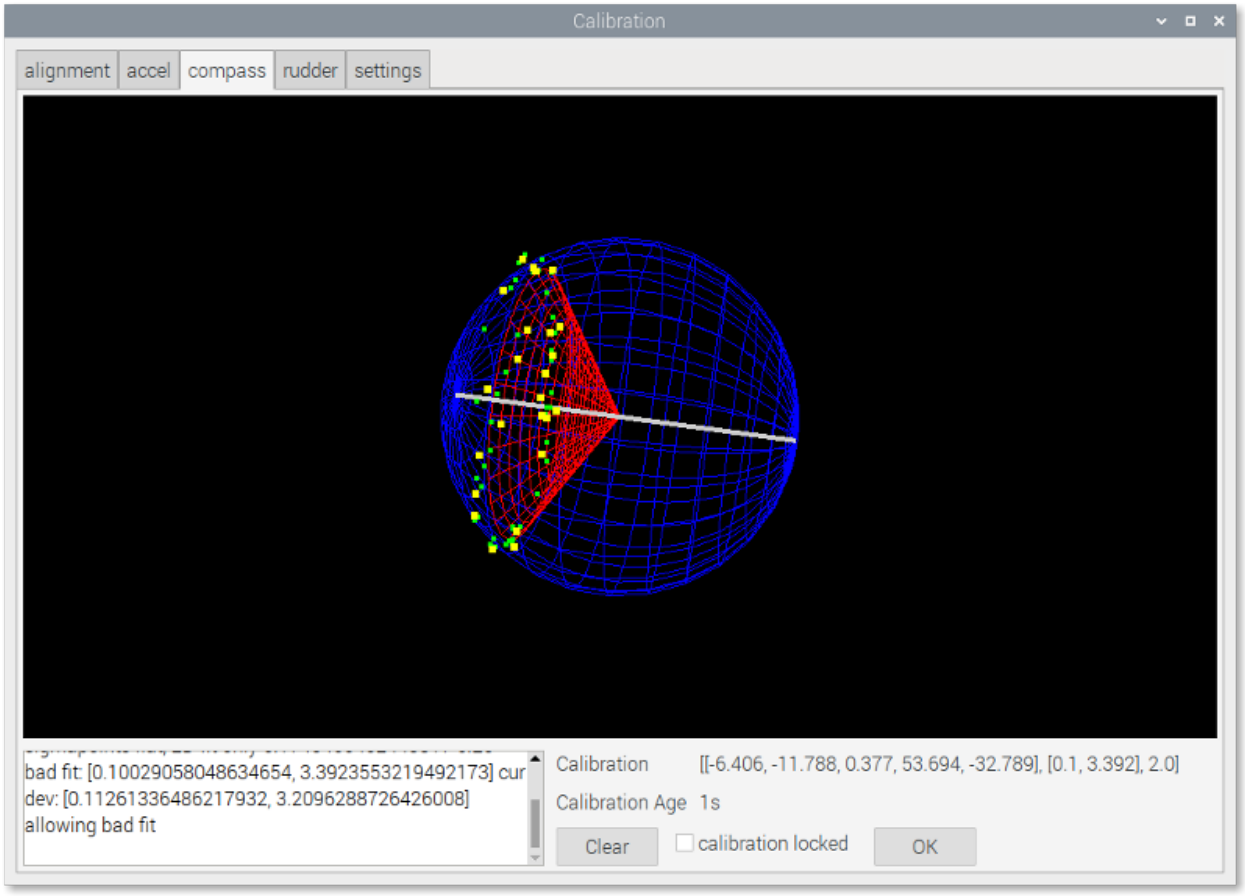
There are both 2D and 3D compass calibration fixes. A 2D fix will occur from turning without pitching or heeling. When heeling there may be some error without a 3D fix. To obtain a 3D fix, you should make a circle with sufficient heeling, such as tacking against the wind, or rolling in waves.

Subsequent 2D fixes will use the previous undetermined value for 3D fix, combining the new 2D fix with the past information from a 3D fix. Performing accelerometer calibration will give a rough 3D fix in most cases making a subsequent 2D fix sufficient for most use.

Compass calibration is continuous and always updates unless locked. You may wish to lock it to prevent future calibration updates.



Once a new calibration is applied, the accelerometer `Calibration Age` should reset and fit points become yellow.

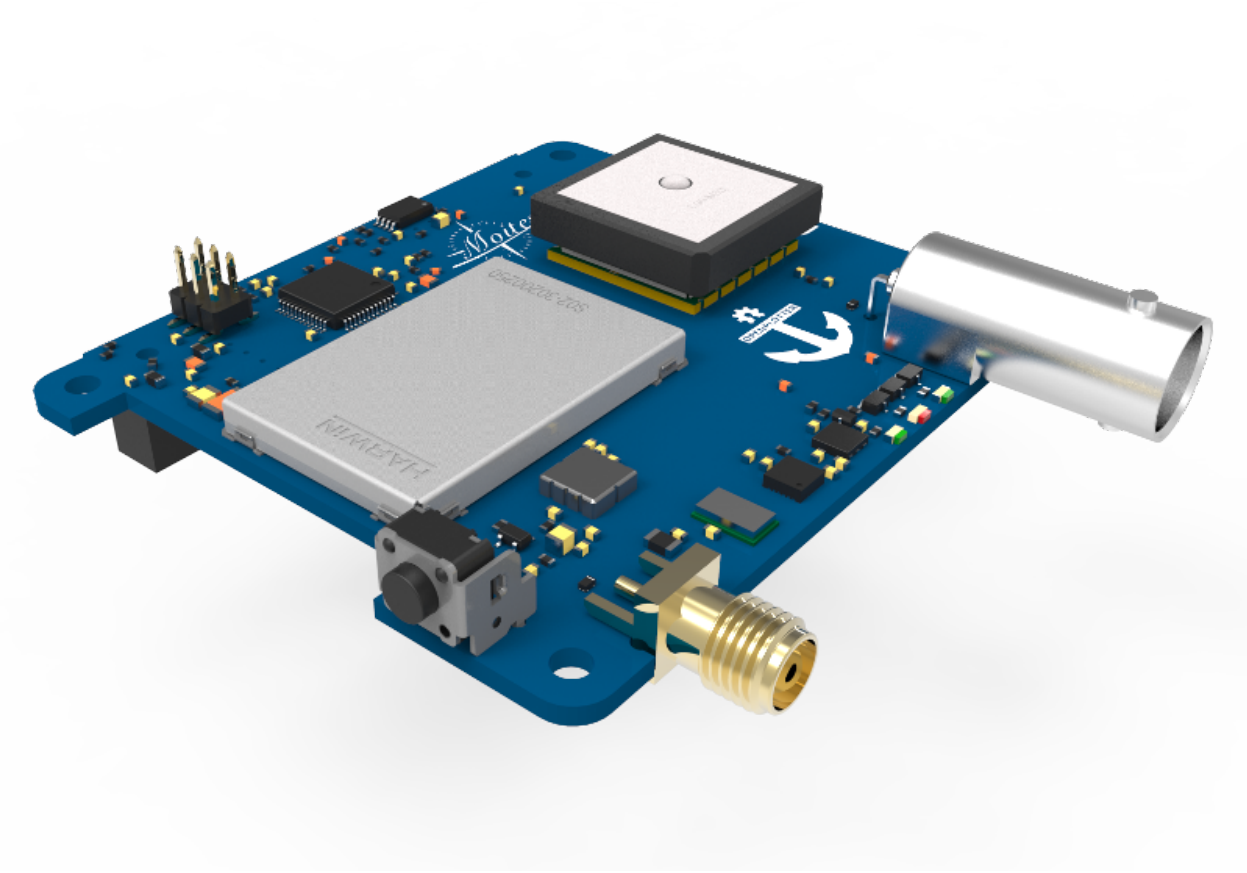


If the sensors are remounted, they must be re-aligned and the compass recalibrated.

If metal objects are moved around the sensors, the compass must recalibrate.

CHAPTER 37

Moitessier HAT 2



37.1 Features

- High-sensitivity (better than -112 dBm) dual channel AIS receiver with SMA antenna connector.
- High-performance GNSS receiver with integrated patch antenna and external antenna support via BNC connector.
- Compass, heel and trim from gyroscope, accelerometer and magnetometer sensors (IMU).
- Barometric pressure.
- Standalone usage or in combination with Raspberry Pi (). Sensors are directly accessible via Raspberry Pi. Standalone usage requires 3.3V power supply and sensors are controlled by the HAT's microcontroller.
- Fully compatible with Raspberry Pi models supporting 40-pin IO header.
- Data communication via SPI (AIS, GNSS and meta data) and via I2C (sensor data).Data accessible via device driver and device file.
- Supports ID EEPROM and automatic device tree loading.
- 3 status LEDs (AIS status, GNSS status, error).
- Shutdown button
- Firmware upgradeable via Raspberry Pi
- Full OpenPlotter compatible. Plug and play.

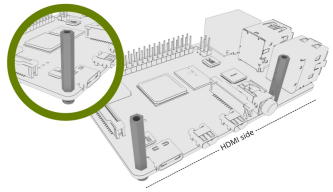
Shutdown button

You may now safely shutdown your OpenPlotter / Raspbian OS via the Moitessier HAT 2 shutdown switch. This will prevent your SD card image from crashing when turning off your Pi with power-off only.

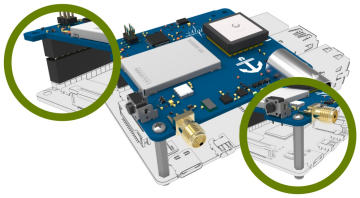
37.2 Mounting the HAT

Installing

Step 1: Fix the two spacers with the screws on the Raspberry Pi
(HDMI side only)



Step 2: Attach the pin header and HAT
Screw Raspberry Pi and HAT together

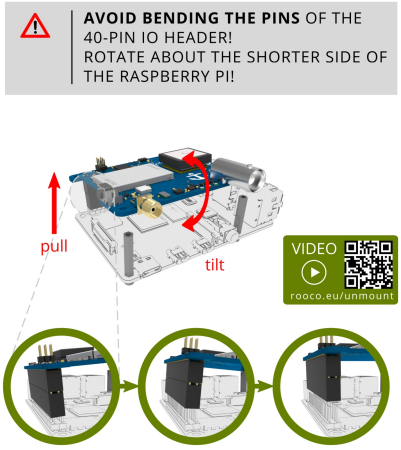


37.3 Removing the HAT

Removing

Step 1: Remove the screws

Step 2: Tilt and pull the HAT gently until you can remove it completely



Danger: You can damage your Raspberry or your HAT if you do not remove it carefully, please watch this video before removing:

37.4 Mounting the HAT into the case

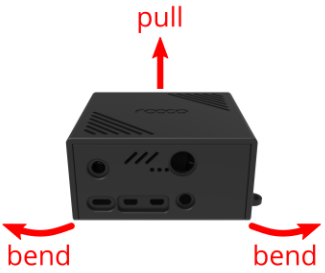
Installation

Raspberry Pi Case Moitessier HAT 2 Edition
RC101E02

Moitessier

Step 1: Open the case

Bend the side walls of the top part outwards and pull the top simultaneously.

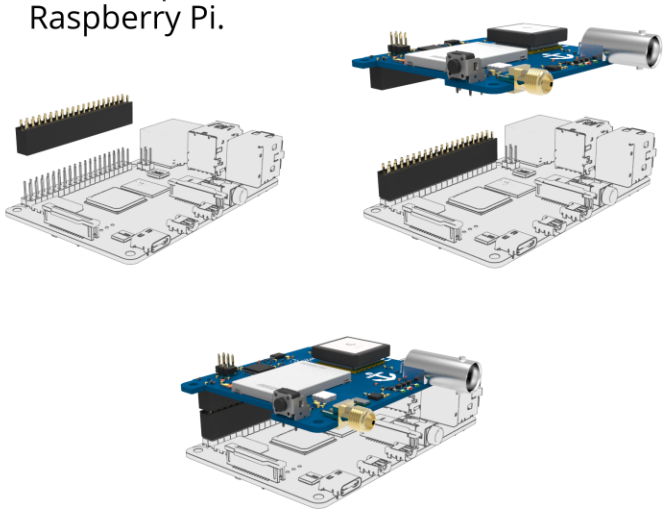


rooco

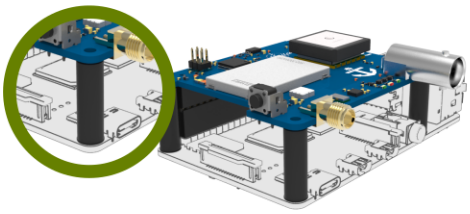
designed for OPENPLOTTER

Step 2: Assemble HAT and Pi

2.1: Connect pin header and HAT with Raspberry Pi.

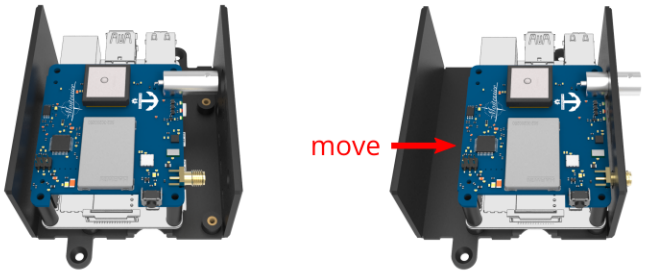


2.2: Place the 4 spacers at their position.
The HAT is not yet screwed together with the Pi.

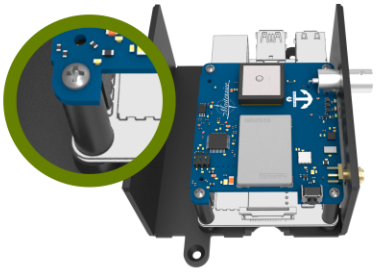


Step 3: Assemble Pi, HAT and case

3.1: Place the Pi with HAT inside the bottom part of the open case. **Be careful as the spacers are still loosely assembled.** Then move Pi and HAT until they are properly aligned on top of the screw threads.

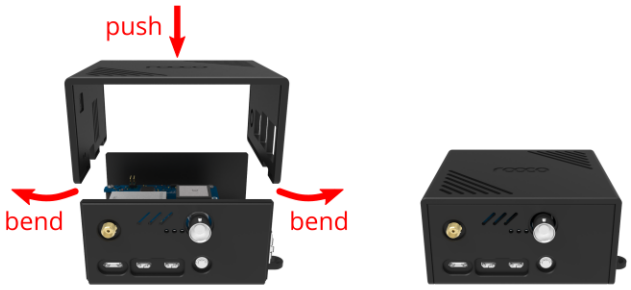


3.2: Screw together Pi, HAT and case with the 4 screws.



Step 4: Close the case

Slightly bend the side walls of the top part outwards and push it on the bottom part until it's locked.



Scope of delivery

Case (two-part), 4 spacers (19mm),
4 screws (M2.5x25mm)

Support

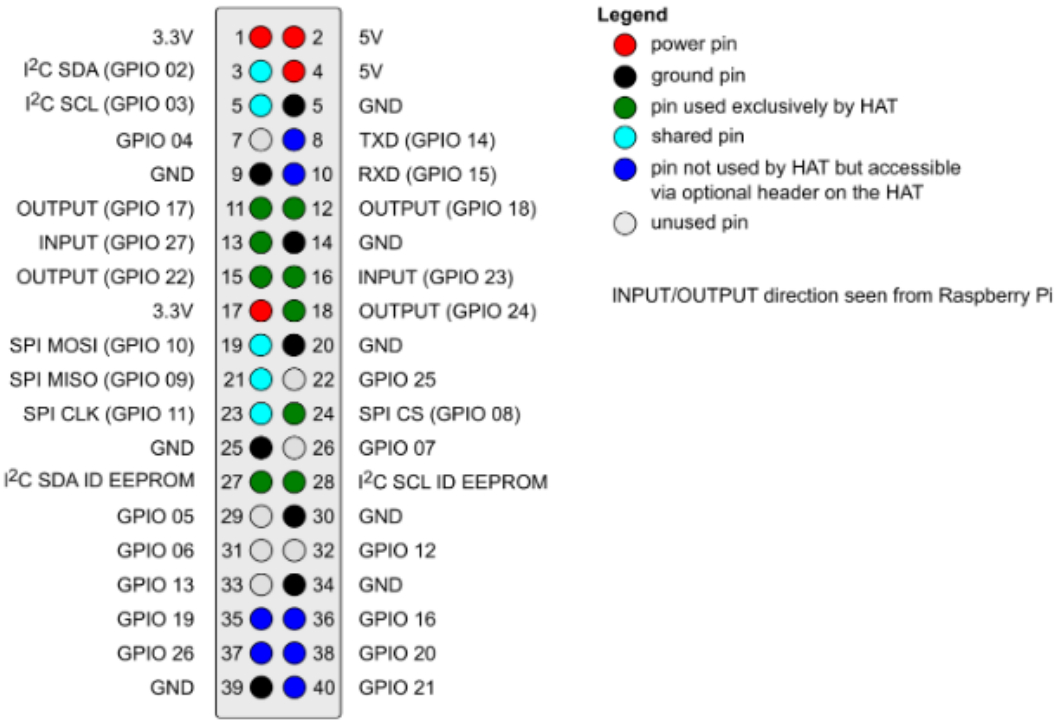
Thank you for buying a product of Rooco.
Find more information on Raspberry Pi case and more products of Rooco on
https://www.rooco.eu

Contact:
support@rooco.eu

Rooco
Wiener Straße 2
8665 Langenwang
Austria

37.5 Pinout

The HAT is controlled by the Raspberry Pi using several GPIOs. Green marked signals are not shareable with other hardware. Pins marked blue are not used by the HAT itself, but are accessible for extension purpose on optional headers on the HAT. I2C and SPI bus can be shared with other hardware. Keep in mind that this is not applicable for the chip select used with the SPI bus, which is exclusively used by the HAT.

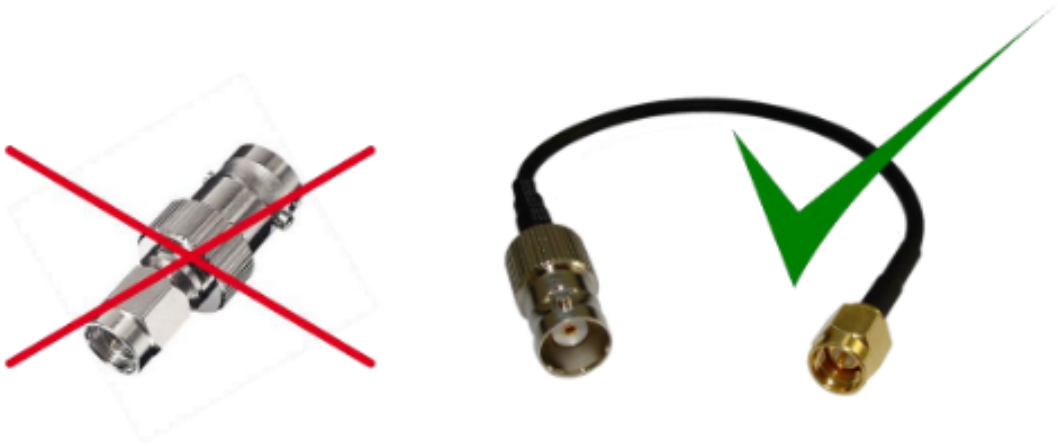

3.3V 1 2 5V
I2C SDA (GPIO 02) 3 4 5V
I2C SCL (GPIO 03) 5 5 GND
GPIO 04 7 8 TXD (GPIO 14)
GND 9 10 RXD (GPIO 15)
OUTPUT (GPIO 17) 11 12 OUTPUT (GPIO 18)
INPUT (GPIO 27) 13 14 GND
OUTPUT (GPIO 22) 15 16 INPUT (GPIO 23)
3.3V 17 18 OUTPUT (GPIO 24)
SPI MOSI (GPIO 10) 19 20 GND
SPI MISO (GPIO 09) 21 22 GPIO 25
SPI CLK (GPIO 11) 23 24 SPI CS (GPIO 08)
GND 25 26 GPIO 07
I2C SDA ID EEPROM 27 28 I2C SCL ID EEPROM
GPIO 05 29 30 GND
GPIO 06 31 32 GPIO 12
GPIO 13 33 34 GND
GPIO 19 35 36 GPIO 16
GPIO 26 37 38 GPIO 20
GND 39 40 GPIO 21
Legend
power pin
ground pin
pin used exclusively by HAT
shared pin
pin not used by HAT but accessible via optional header on the HAT
unused pin
INPUT/OUTPUT direction seen from Raspberry Pi

CHAPTER 38

Antennas

Tip: For the best receiving performance, ensure that the cable lengths of the antennas are as short as possible.

Caution: It is recommended to use pigtail adapters to reduce mechanical force on the antenna connectors.



38.1 AIS Antenna

The Moitessier HAT supports all popular VHF/AIS antennas. Please note the following features when selecting an antenna:

- 50 Ohm impedance
- SMA male connector for direct connection, or any other connector using a proper pigtail adapter

- Frequency range at least 161.95 MHz to 162.05 MHz
- RG 174 coaxial cable or better

Caution: The coaxial cable attached to the SMA connector should have a maximum outside diameter of 3.7 mm. Larger diameters might cause mechanical force to the antenna connector.

A suitable splitter also enables the Moitessier HAT to share the VHF antenna of other radio equipmenton a ship.

Danger: Use splitters only that physically decouple the Moitessier HAT from any transmitter while transmission is in progress.

38.2 GNSS Antenna

Your device has an internal patch antenna. If it is not possible to fit the HAT with an unobstructed view of the sky (such as below deck), an external GNSS antenna is required. Use a standard, active GNSS antenna that is fitted with a BNC connector.

CHAPTER 39

Configuration

OpenPlotter Moitessier HAT edition

Download the img or NOOBS file and follow the *basic manual* to install it on your SD card.

The easiest way to make Moitessier HAT work on Raspberry Pi is to download and install a special OpenPlotter edition. Everything is preinstalled and preconfigured in *OpenPlotter Moitessier HAT* edition and it will work out of the box, just plug and sail!

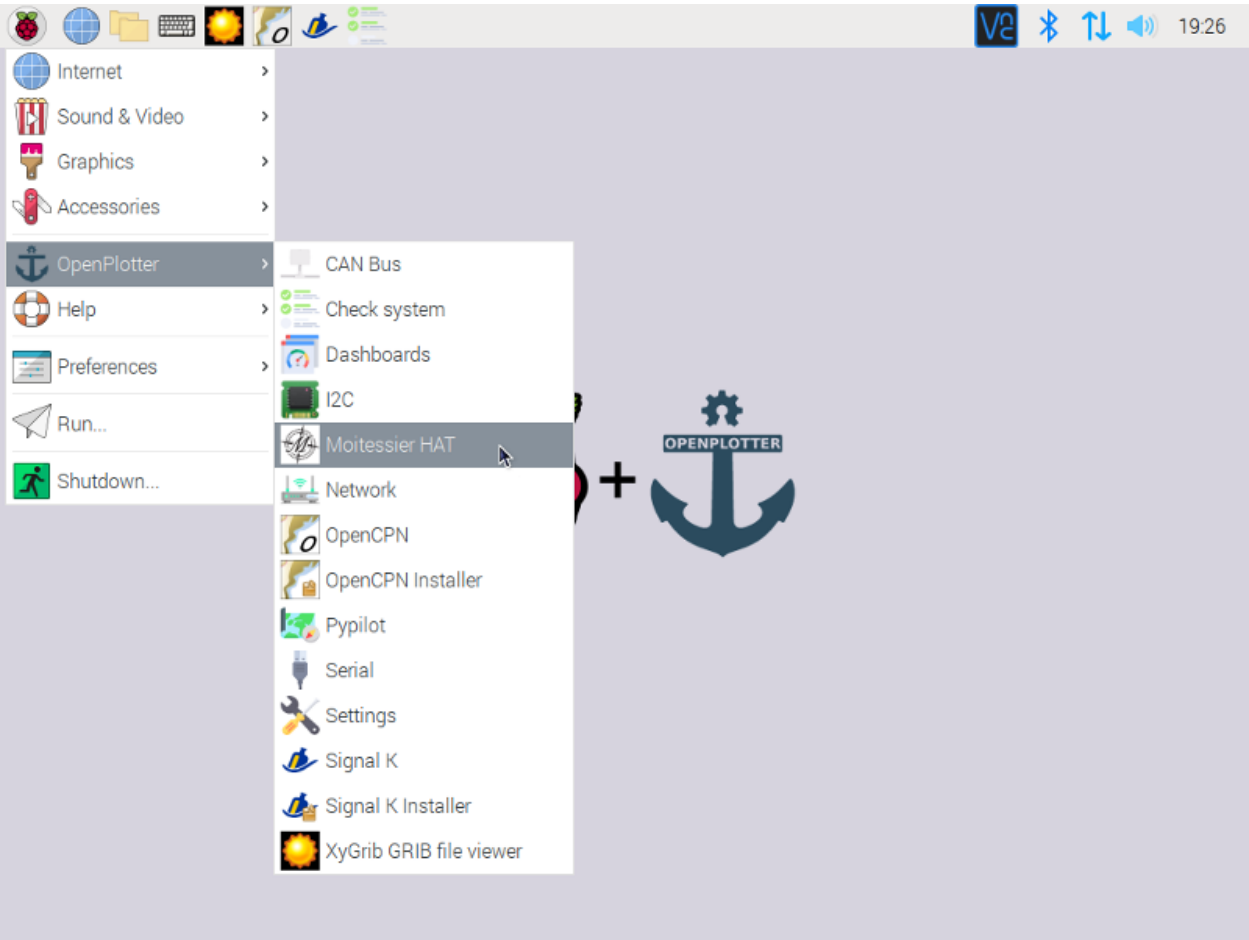
It is recommended to read the rest of the chapter to learn how to configure the HAT on your own and be able to play with its settings.

Important: If you are using the *OpenPlotter Moitessier HAT* edition, the only thing you should do is calibrate the compass following the steps of the *Pypilot compass calibration* chapter.

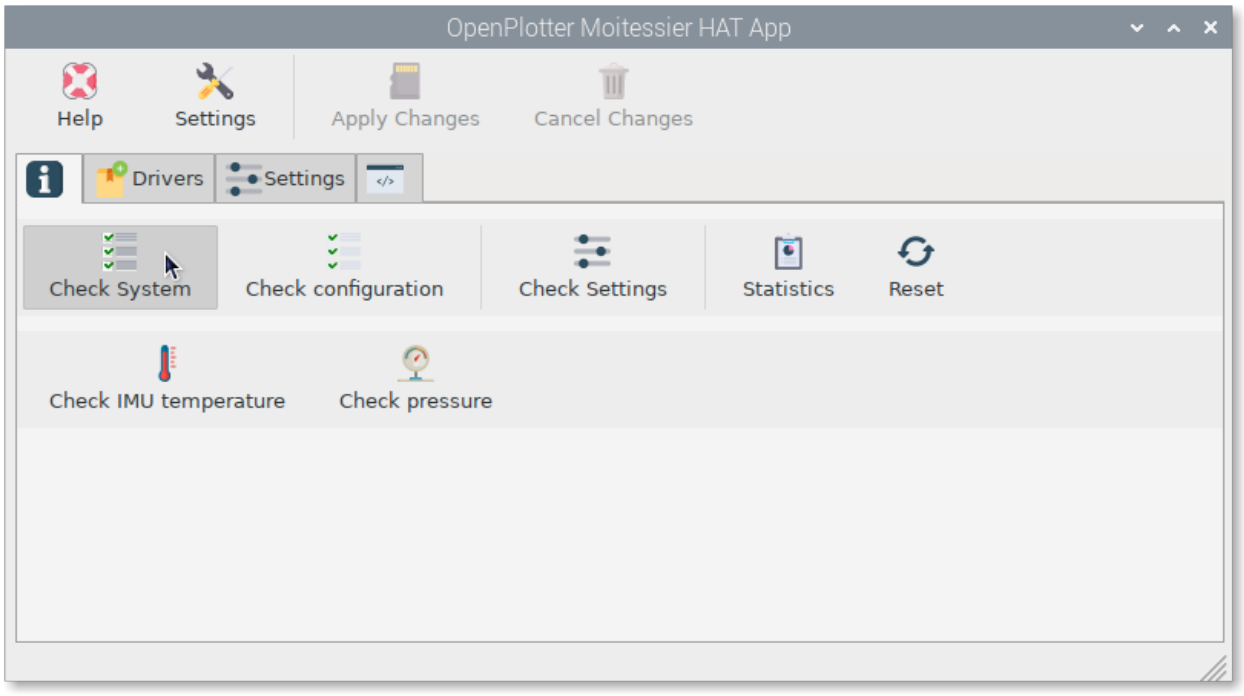
You should reinstall the driver at least once even if everything works fine to make sure that your HAT firmware is up to date, especially Moitessier HAT 1 users.

If you are not using the *OpenPlotter Moitessier HAT* edition, you have to be sure the list of apps below are installed. These apps have to be installed from `OpenPlotter Settings` interface.

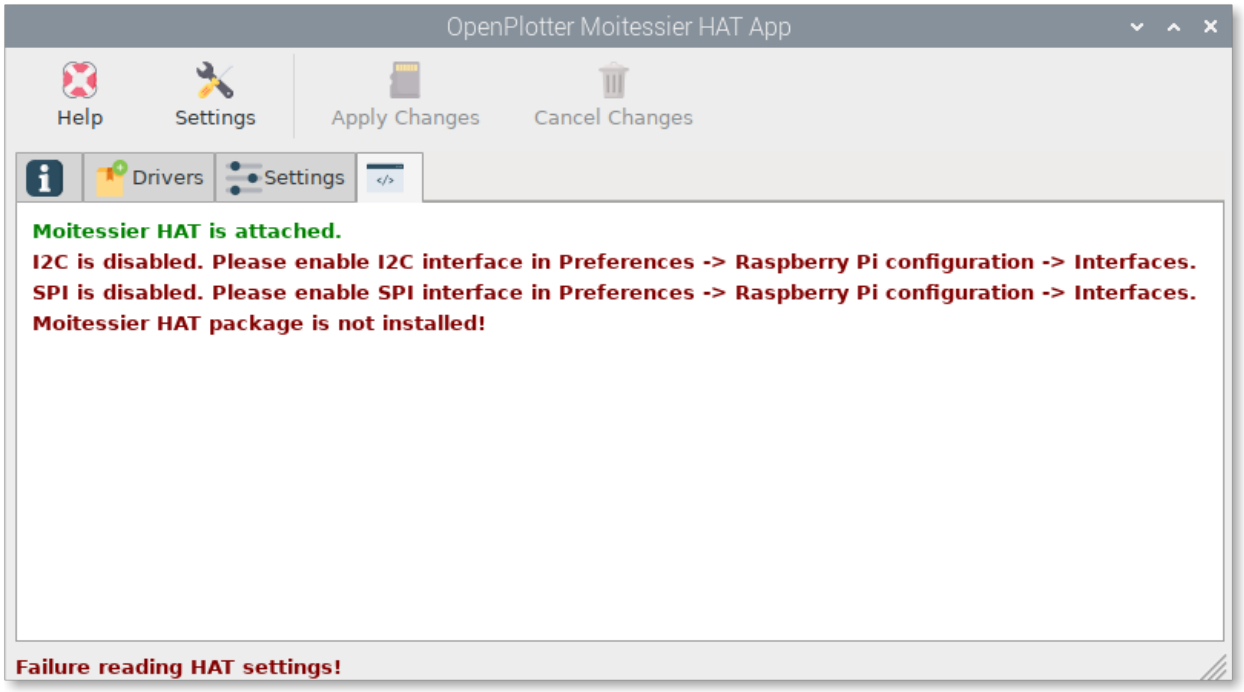
- `OpenCPN Installer`
- `Signal K Installer`
- `Serial`
- `Pypilot`
- `Moitessier HAT`
- `I2C`



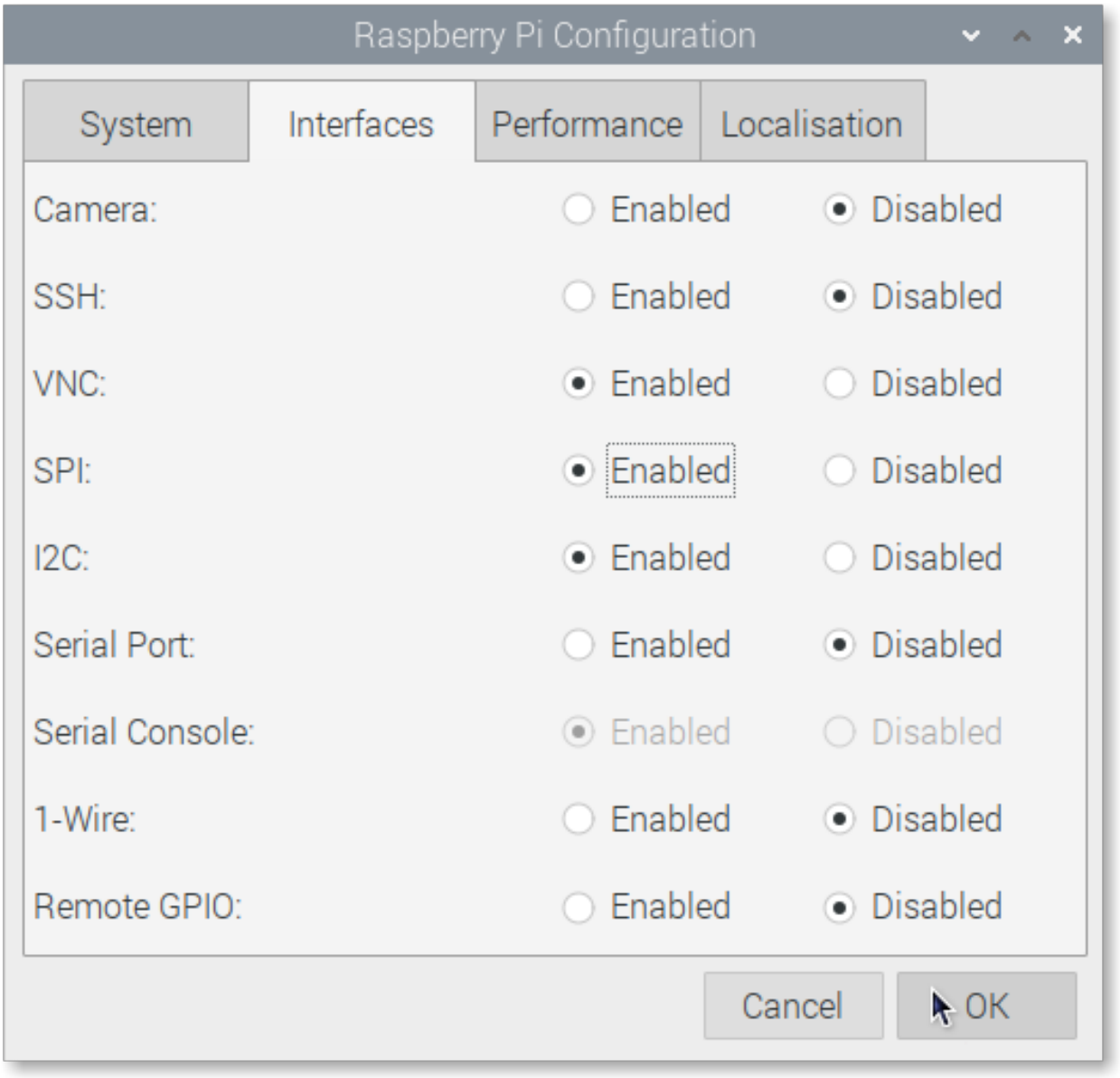
After installing these apps go to `Moitessier HAT` app and click on `Check System`:



You could see some error messages because SPI and I2C interfaces are disabled or the Moitessier HAT driver is not installed yet:

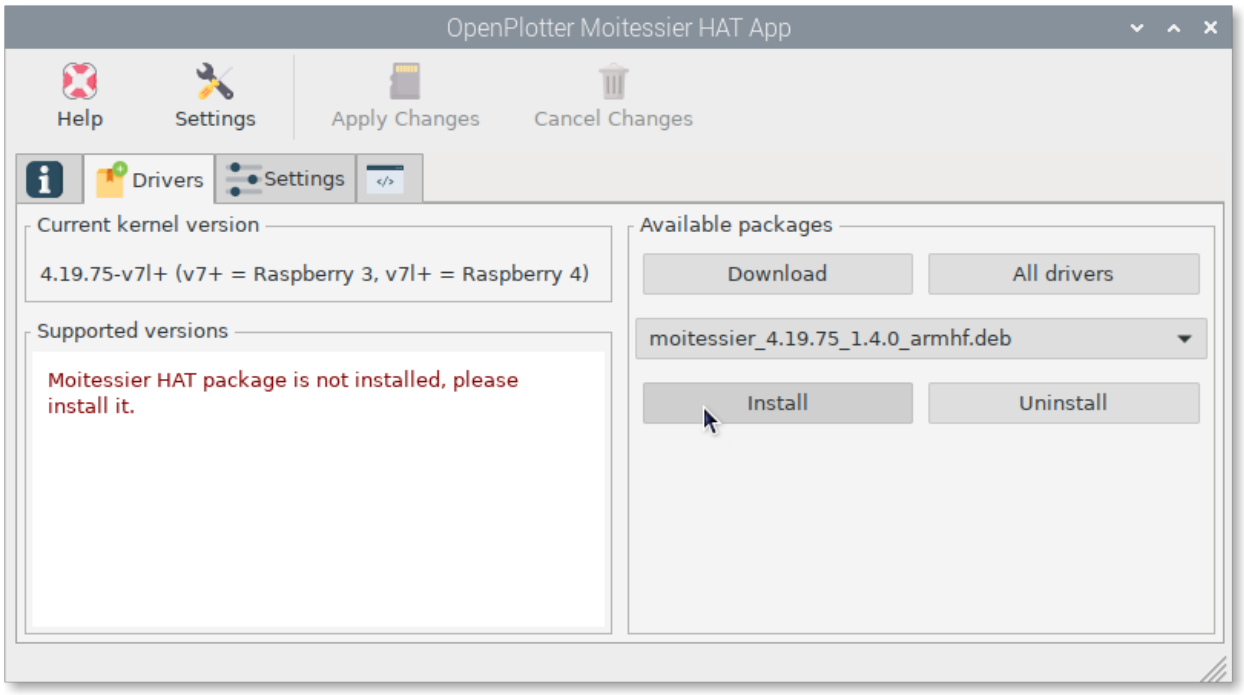


SPI and I2C must be enabled before installing the drivers. Go to applications menu and enable both of them in *Preferences → Raspberry Pi configuration → Interfaces*:

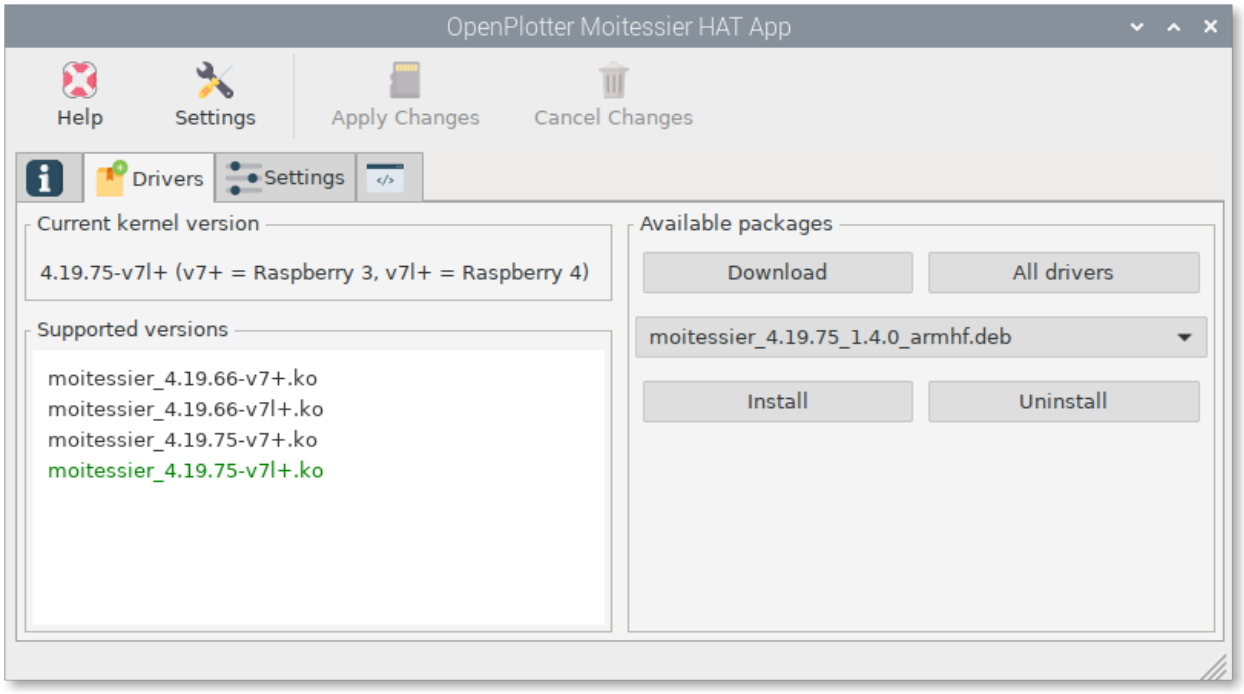


39.1 Installing drivers

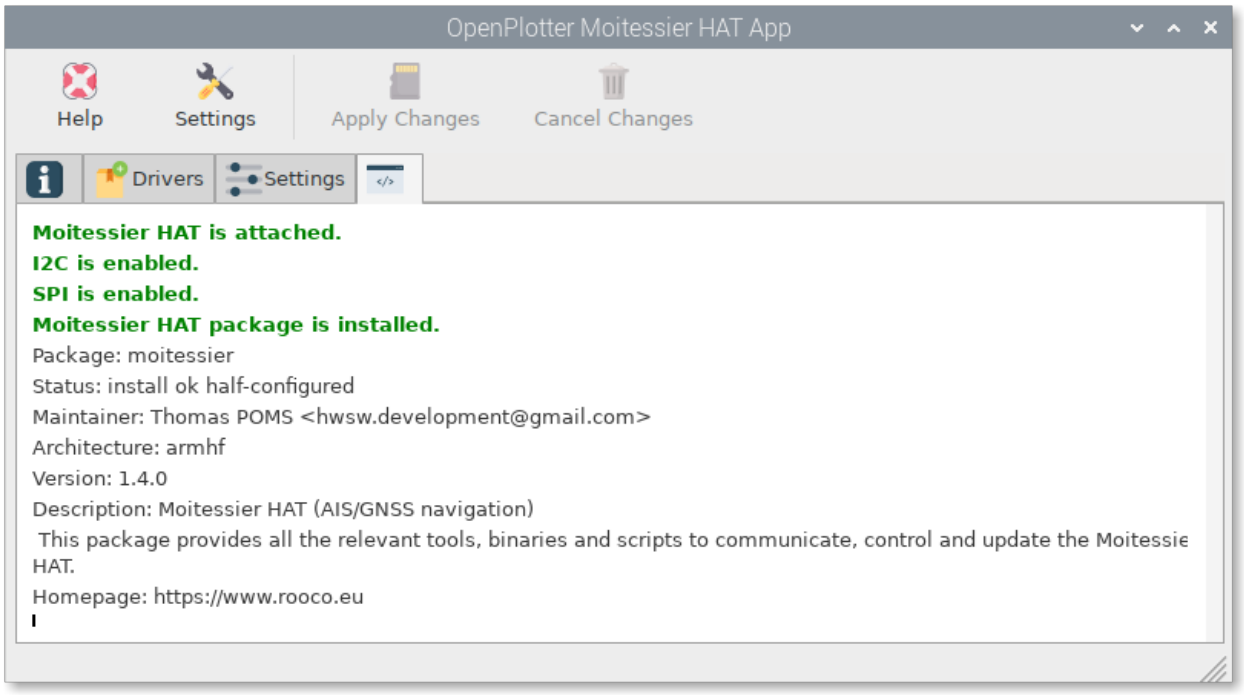
The driver must match your kernel version. After updating your system, the kernel could be also updated and your HAT will stop working. In `Drivers` tab you will find your current kernel version. Select the package matching your kernel and click on `Install`. The drivers will be installed and the system rebooted. The package may need to do this twice, you will be notified.



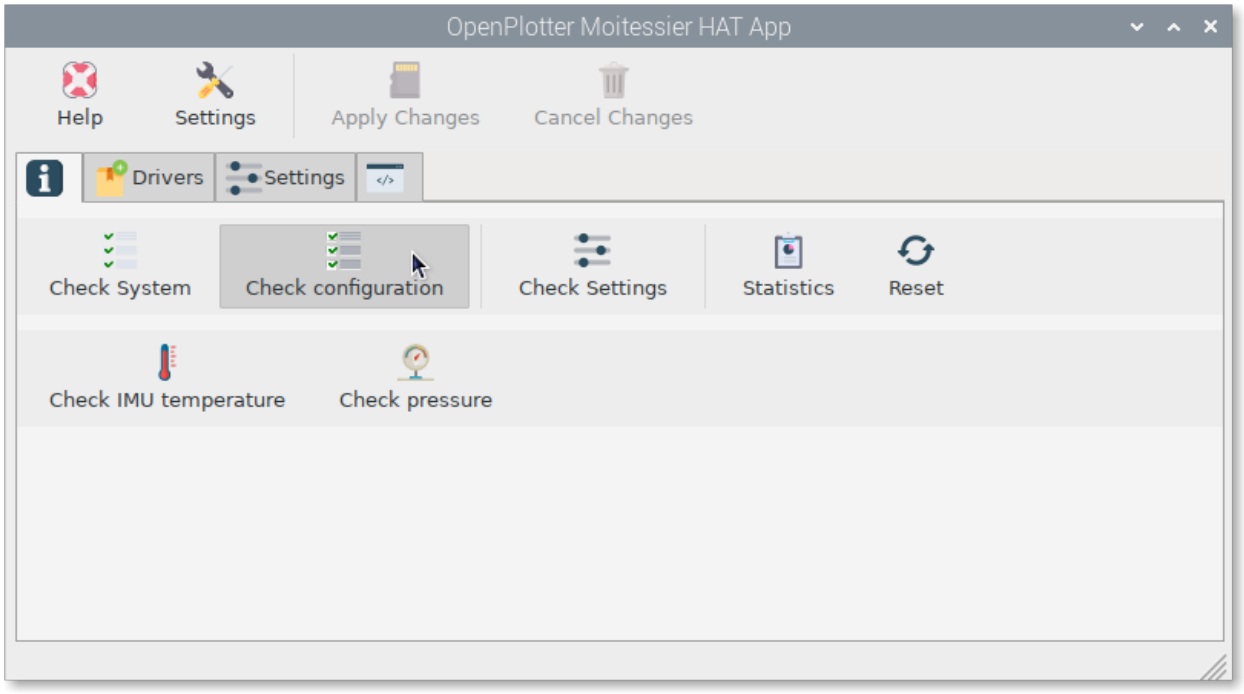
After rebooting you should see a list of kernels supported by the installed driver. If your current kernel is not supported by any of the available packages, click on `Download` and the system will try to find a suitable package from Internet. If this fails too, click on `All drivers` to go to *Rooco* site and ask them when the package will be available.

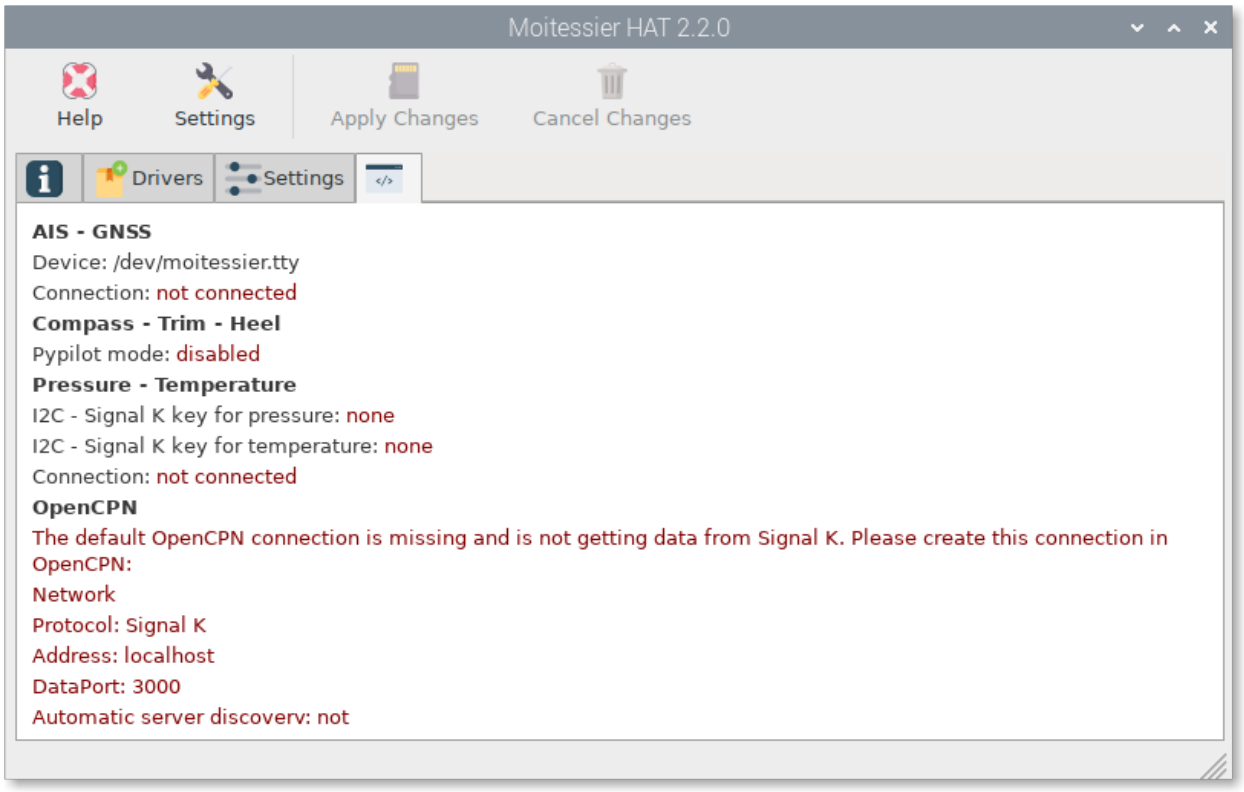


If everything went well, you should see something like this by clicking on `Check System` again:



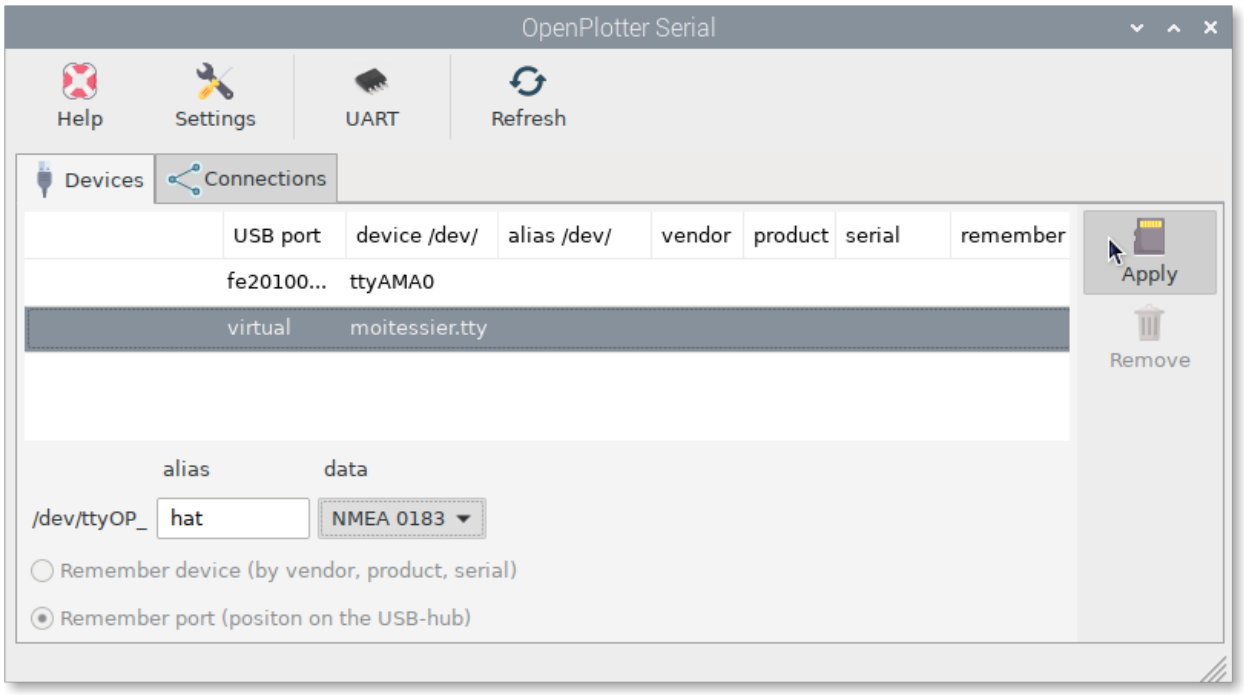
Now you are ready to configure your system. Go to `Info` tab and click on `Check configuration`. The goal is to send all data to the `Signal K` server to be shared with any program that need data from our HAT like OpenCPN. The rest of apps will help us to do this.



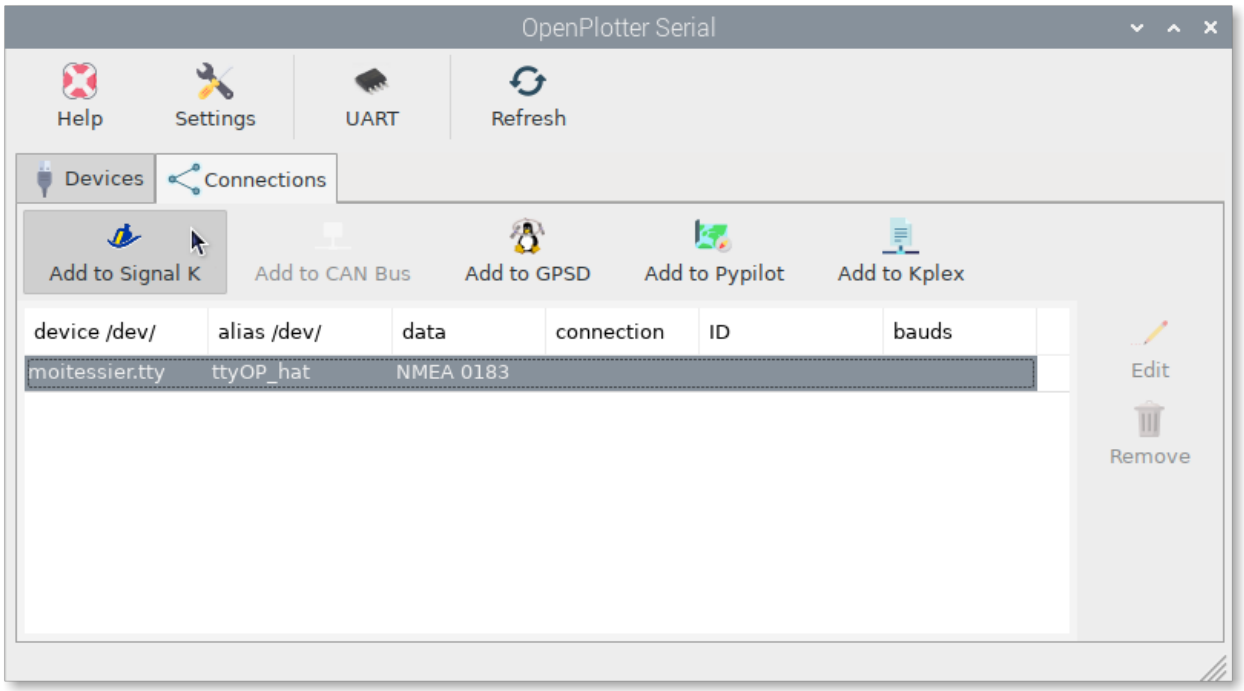


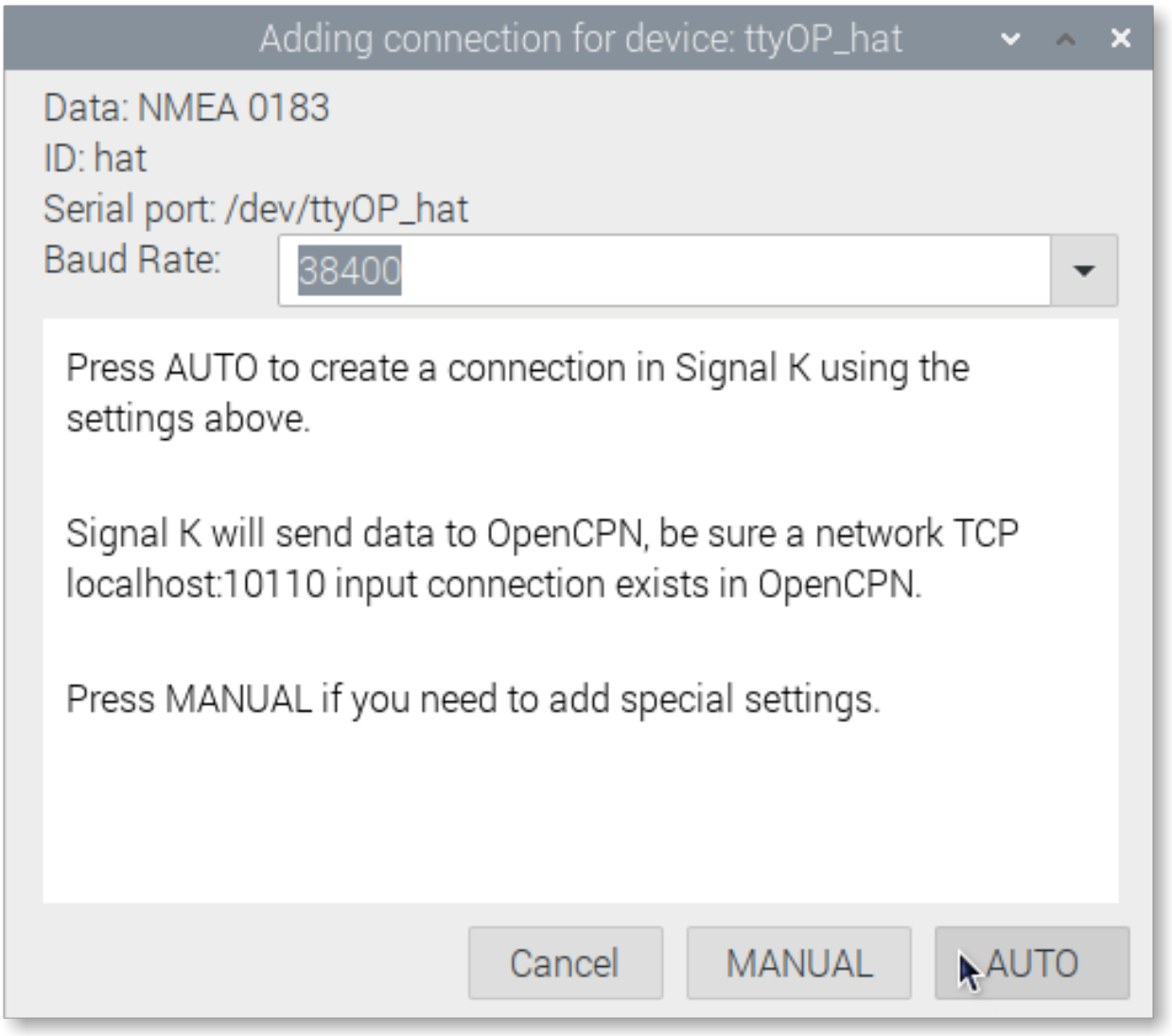
39.2 Configuring AIS and GNSS reception

Go to `Serial` app and select the HAT from the the list of detected devices in `Devices` tab. Provide a short `alias` and select NMEA 0183 as the format of the expected `data`. Finally click `Apply`:



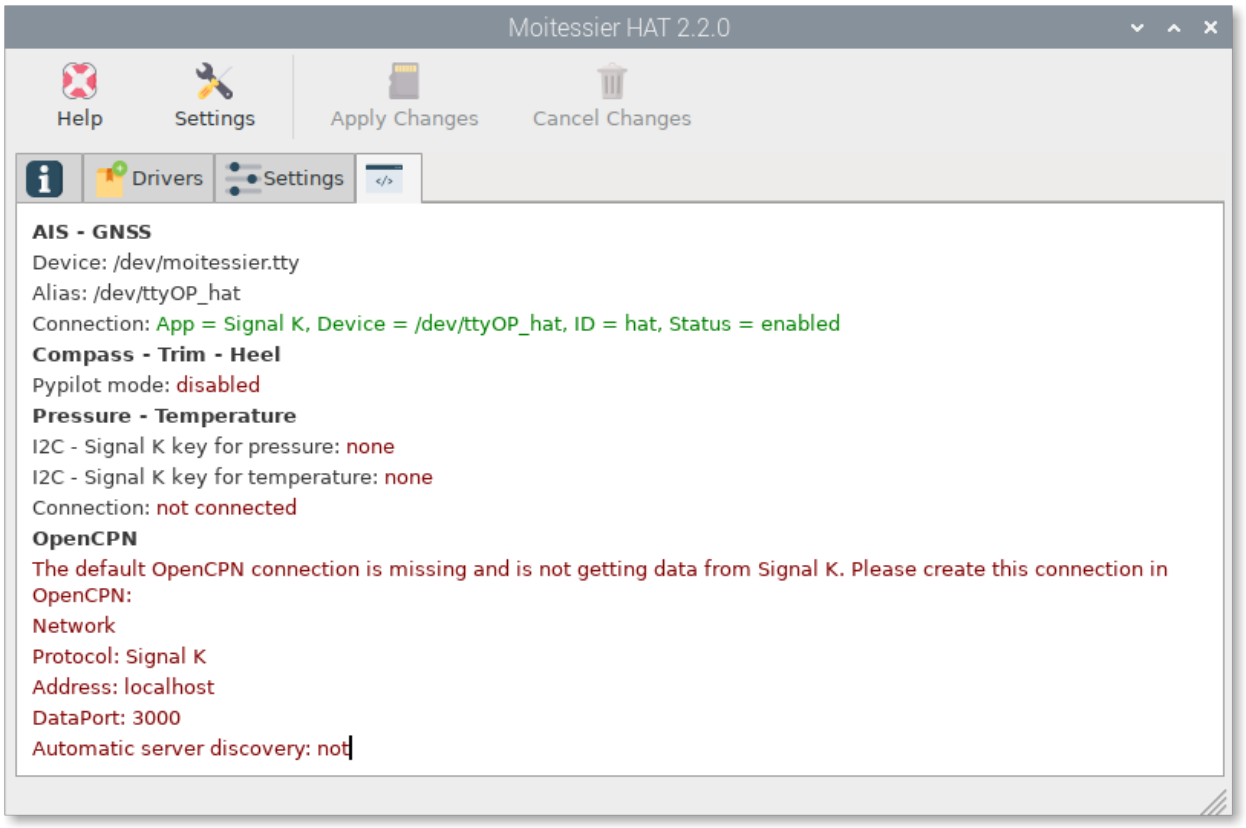
Go to `Connections` tab, select the HAT from the list, click on `Add to Signal K` and in the next window click `AUTO`. This way we are creating a serial connection in `Signal K` server using the settings provided by `Serial` app.





Note: If you are going to use an autopilot you should select `Add to Pypilot` and finally connect pypilot to `Signal K`. See *pypilot* chapter for details.

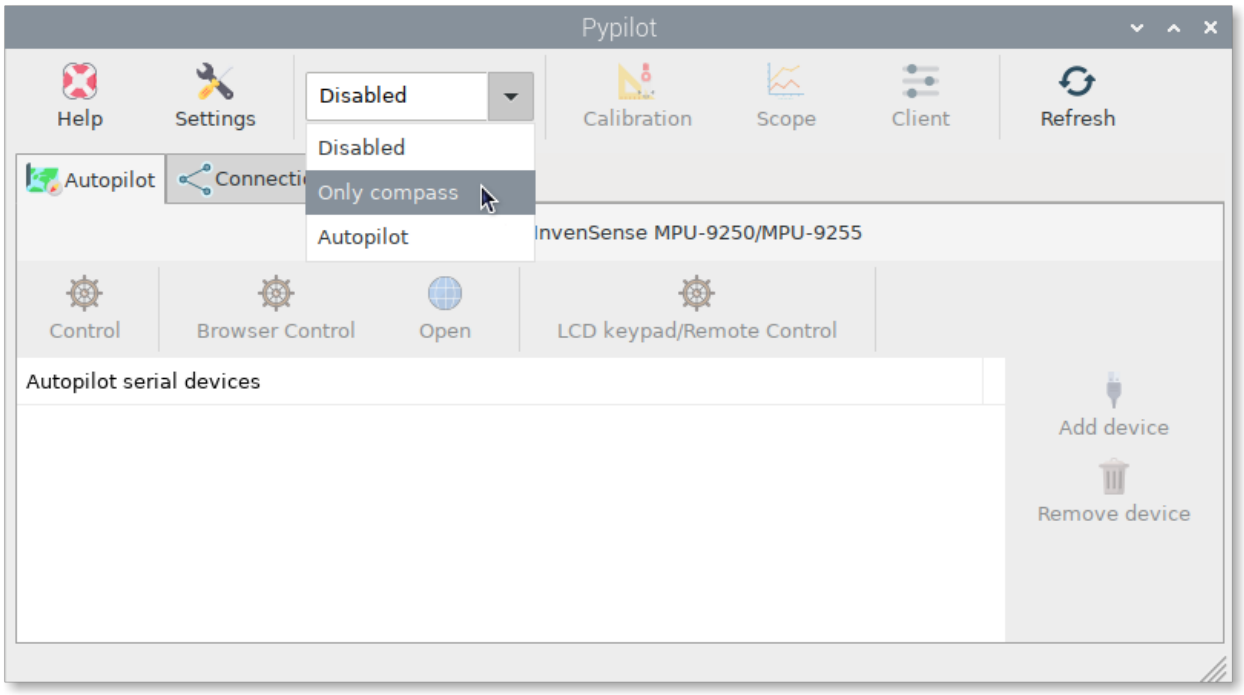
Go to `Info` tab and click on `Check configuration` again to see the changes:



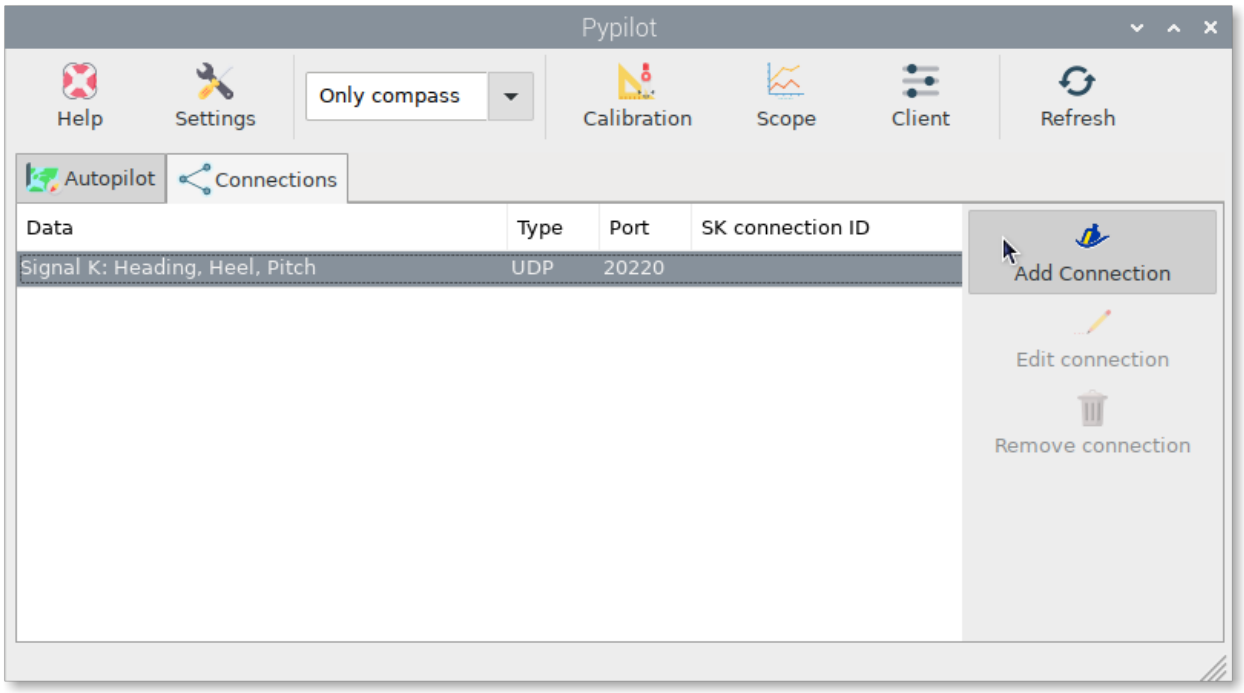
39.3 Configuring compass, heel and trim reception

Go to `Pypilot` app an select `Only compass`.

Note: If you are going to use an autopilot you should select `Autopilot`. See *pypilot* chapter for details.

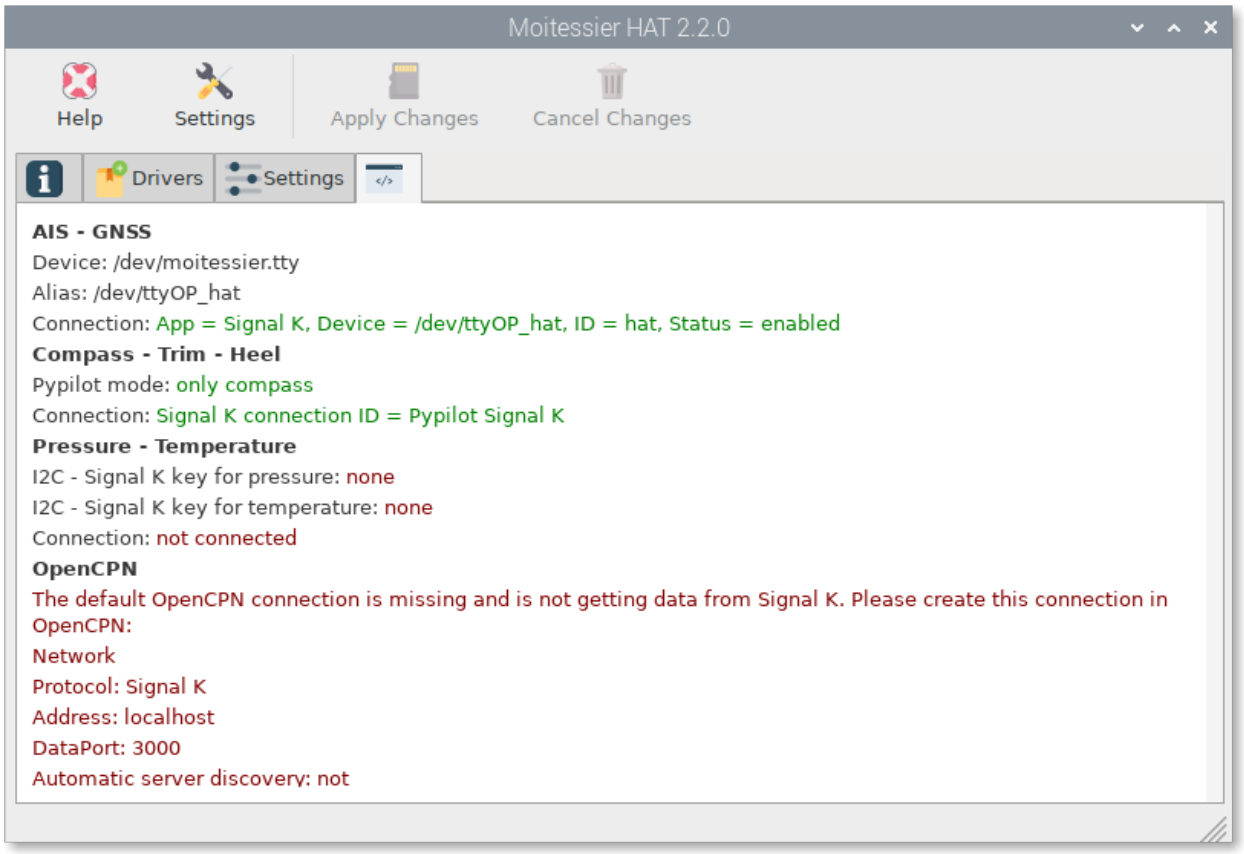


Then go to `connections`, select the available connection and click on `Add connection`. This way we are creating a network connection in `Signal K` to receive heading, pitch and heel data.



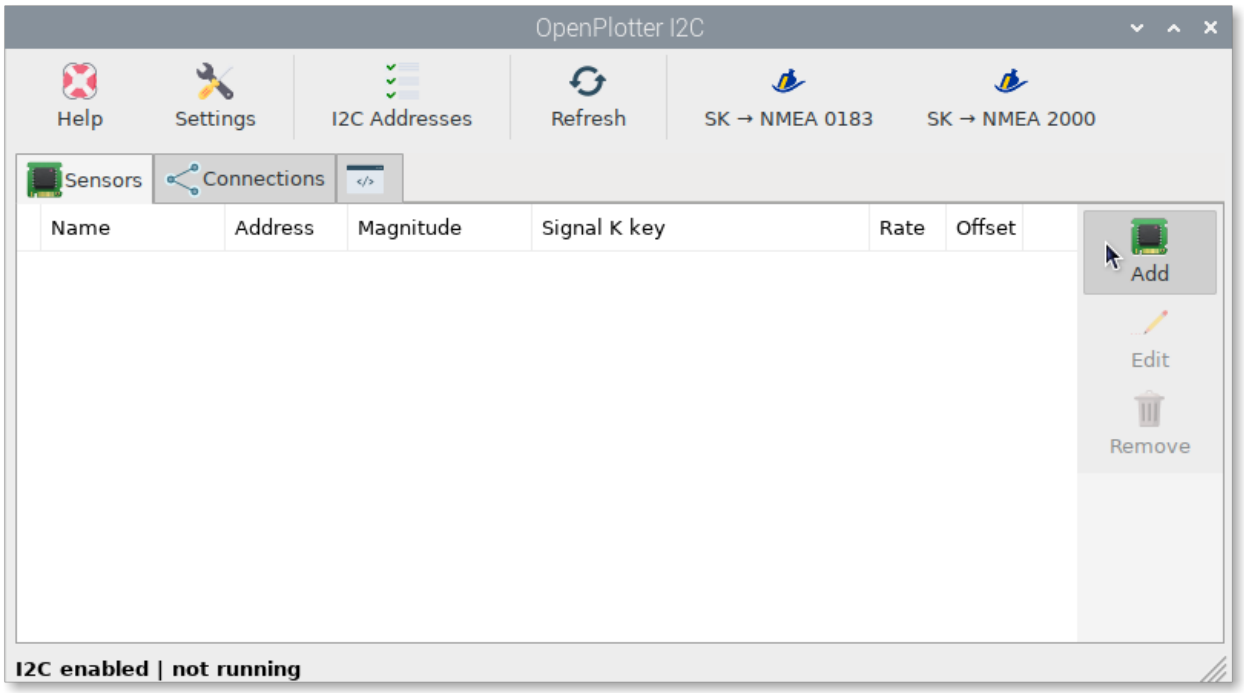
Important: To get reliable heading readings you have to calibrate the compass following the steps of the *Pypilot compass calibration* chapter.

Go to `Info` tab and click on `Check configuration` again to see the changes:

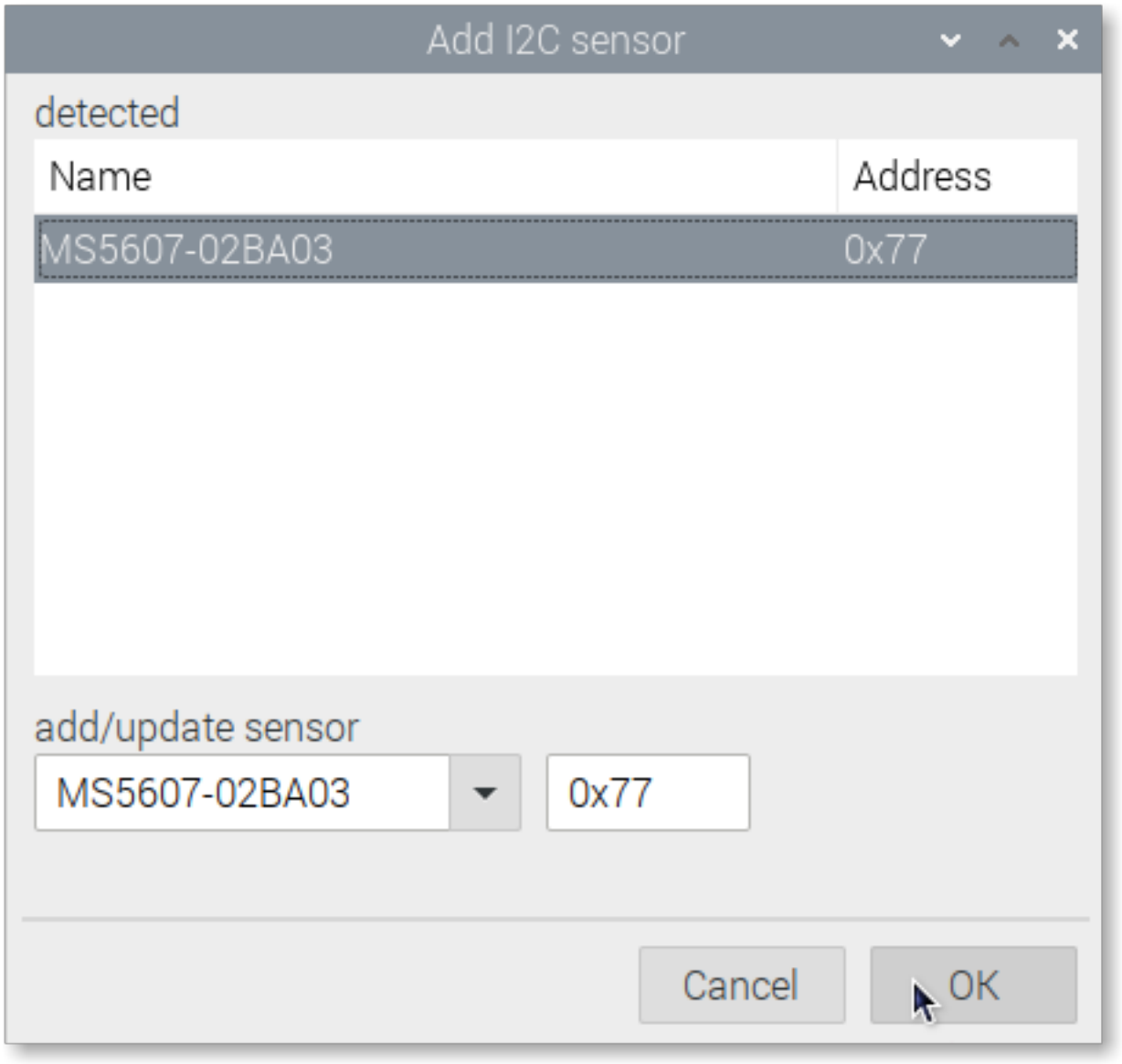


39.4 Configuring pressure reception

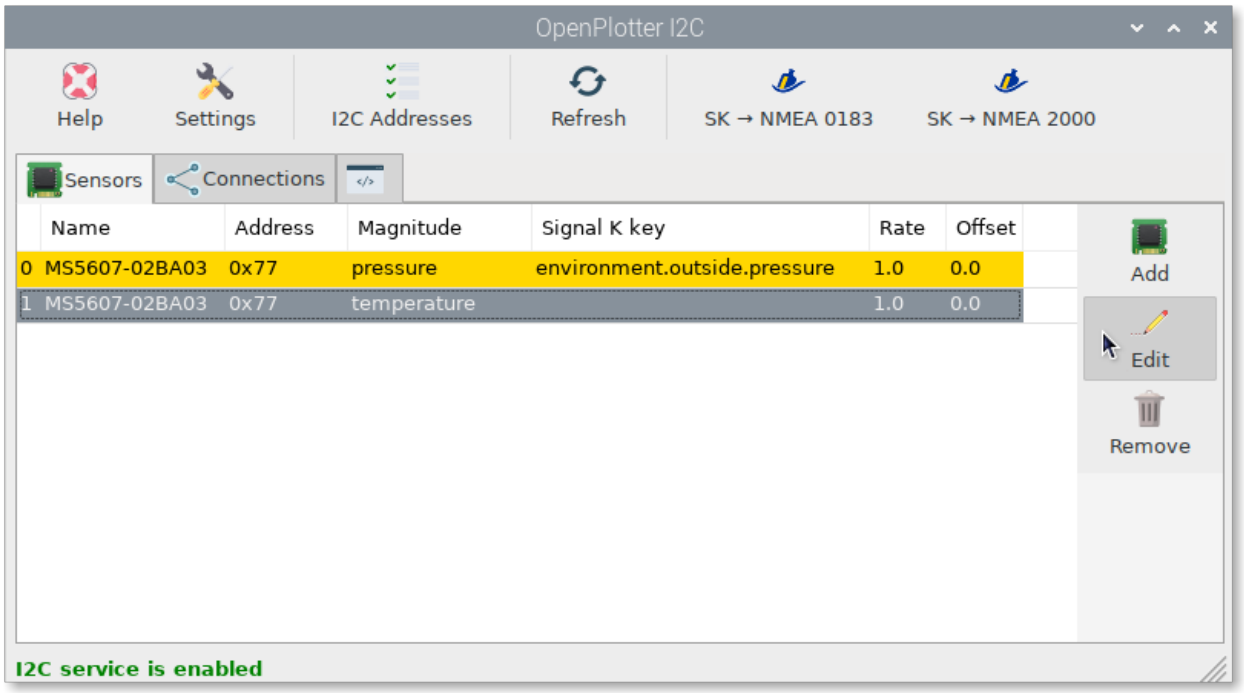
Go to `Sensors` tab in `I2C` app an click `Add`.



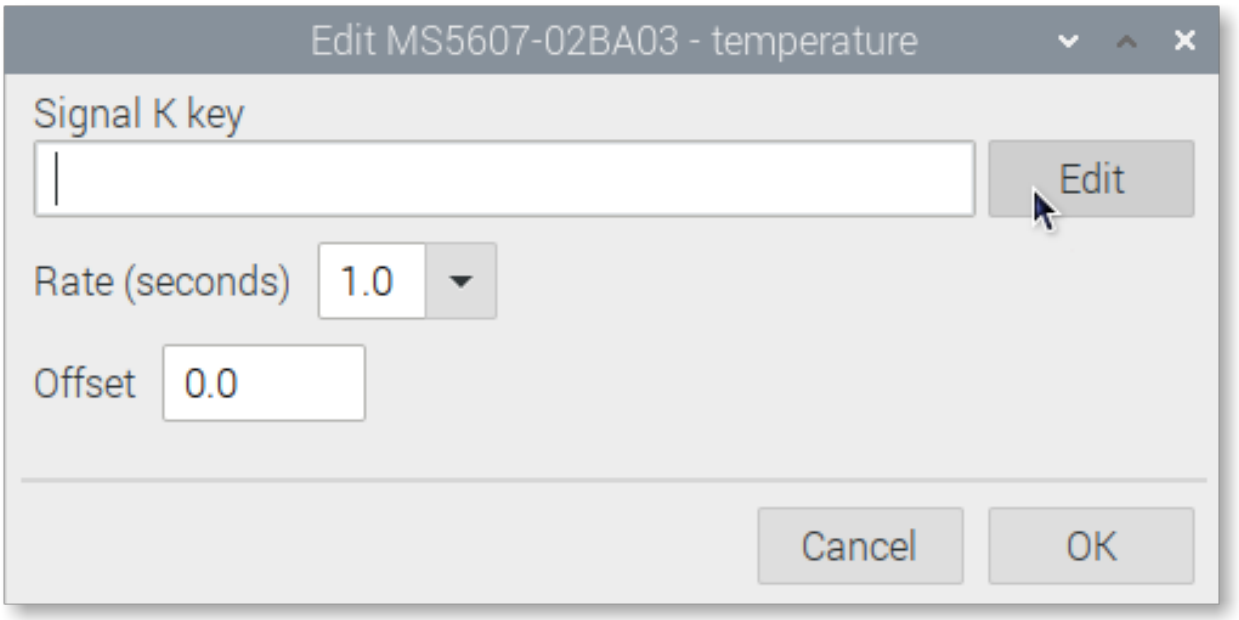
Select `MS5607-02BA03` in the list of detected sensors and click `OK`.



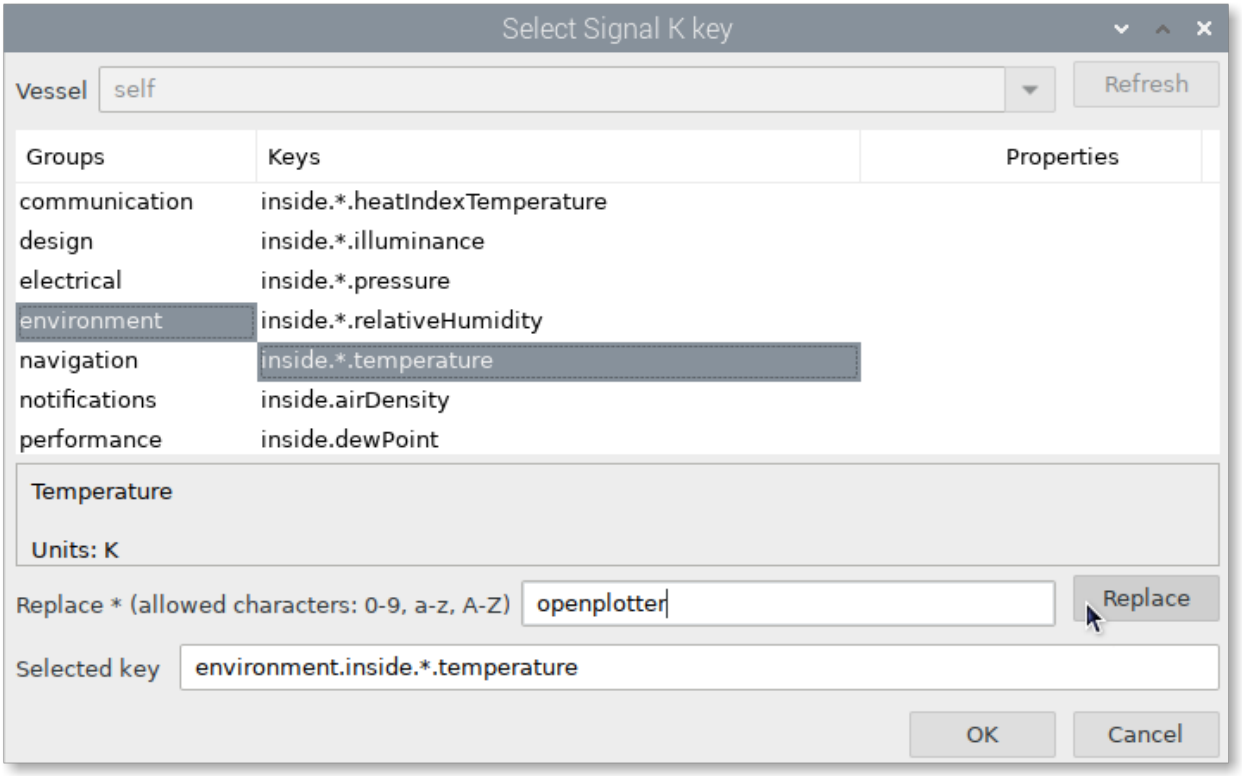
A Signal K key will be created for pressure by default. You can assign another one for temperature. The temperature sensor is affected by the heat produced by the Raspberry and the HAT itself, so we can not assign this value to environment.inside.temperature key, we should use something like environment.openplotter.temperature. Select `temperature` and click in `Edit`.



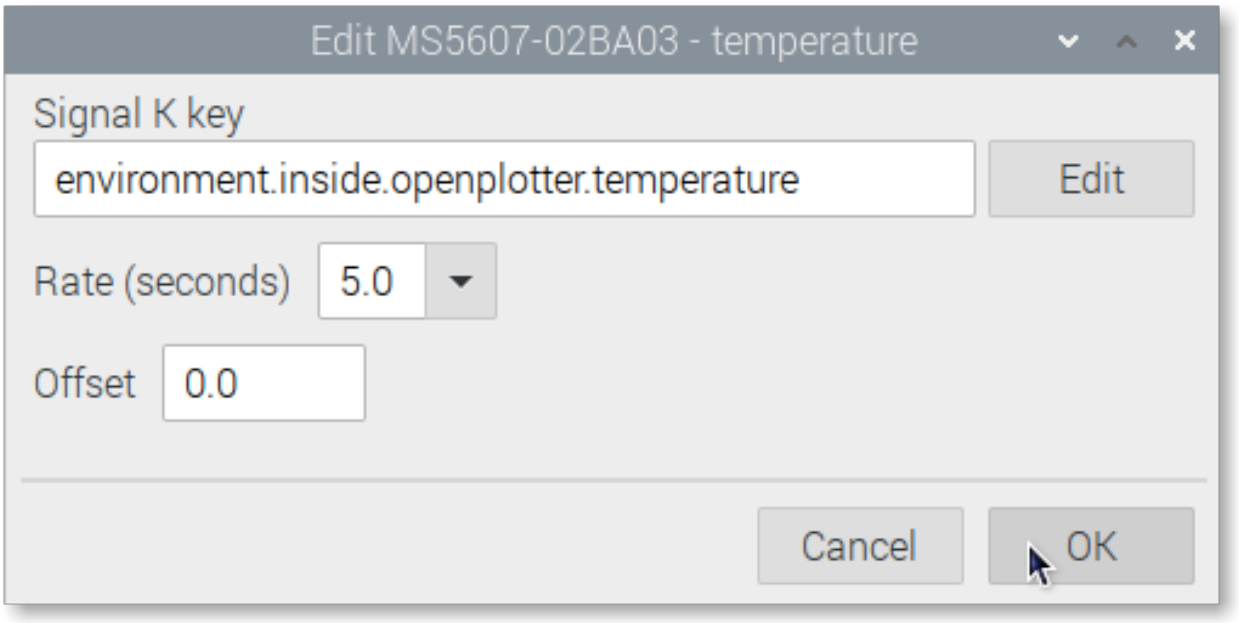
To choose a Signal K key click `Edit`.



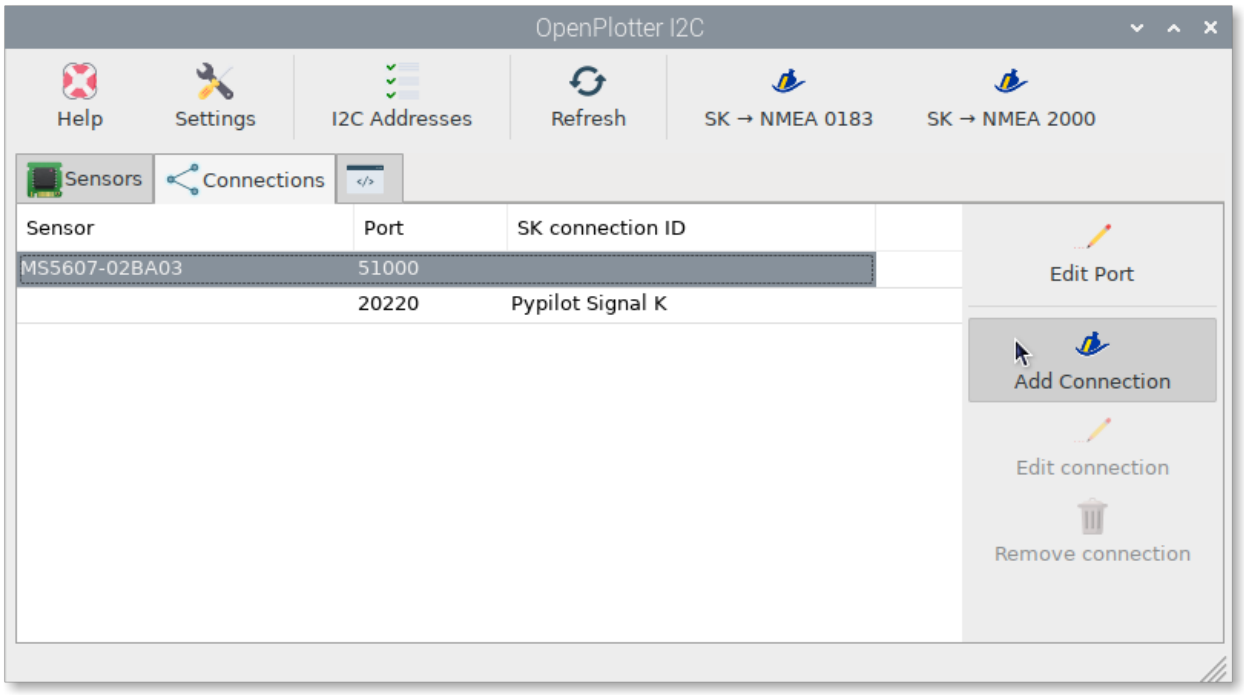
Select `environment` in the first column and `inside.*.temperature` in the second column. Write *openplotter* in the `Replace` field, press `Replace` button and the wildcard will be replaced by *openplotter*. Press `OK`.



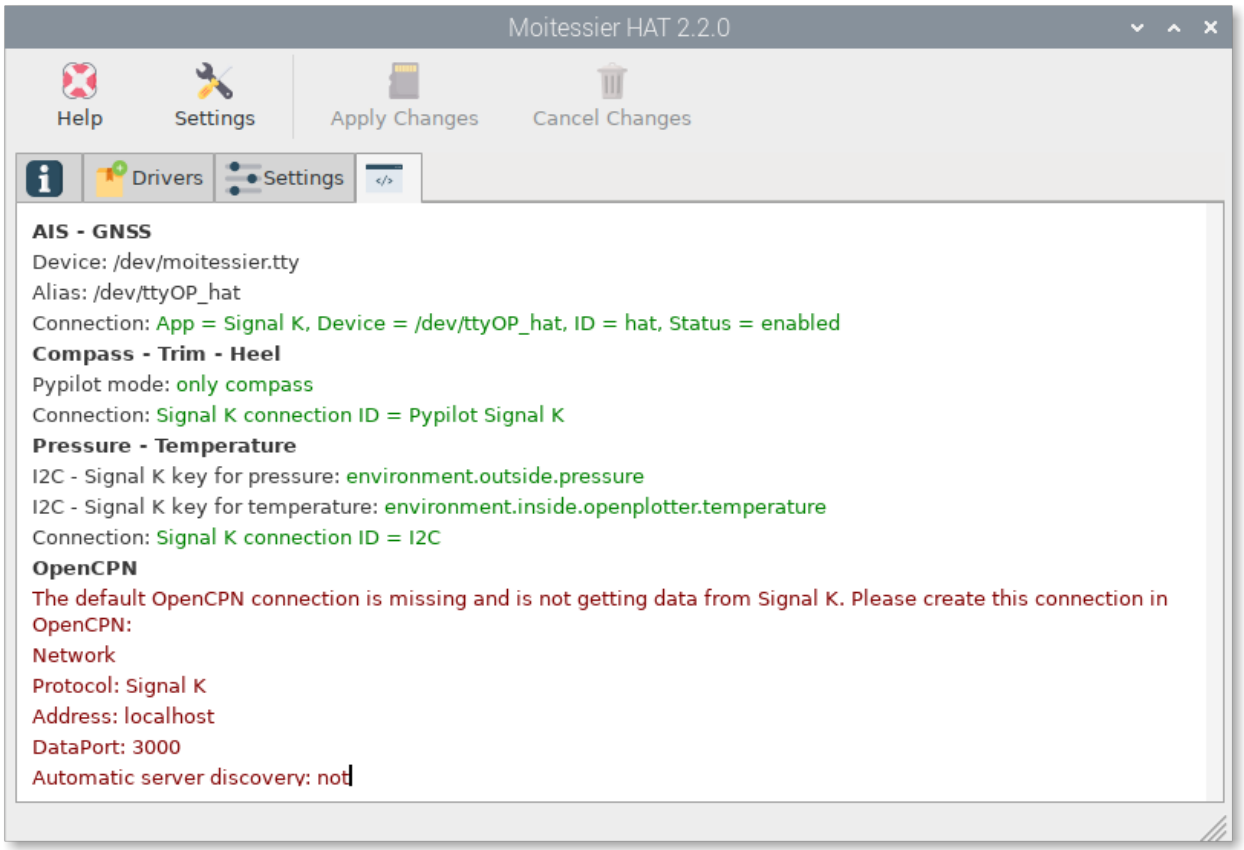
We do not need pressure or temperature data every second so we will select another `Rate`. Click `OK`. Edit the `pressure` value to select another `Rate` too.



Go to `Connections` tab, select `MS5607-02BA03` sensor and click in either `Add Connection` to create a new network connection in `Signal K` or `Edit port` if you want to send these data to any existing network connection in `Signal K`.

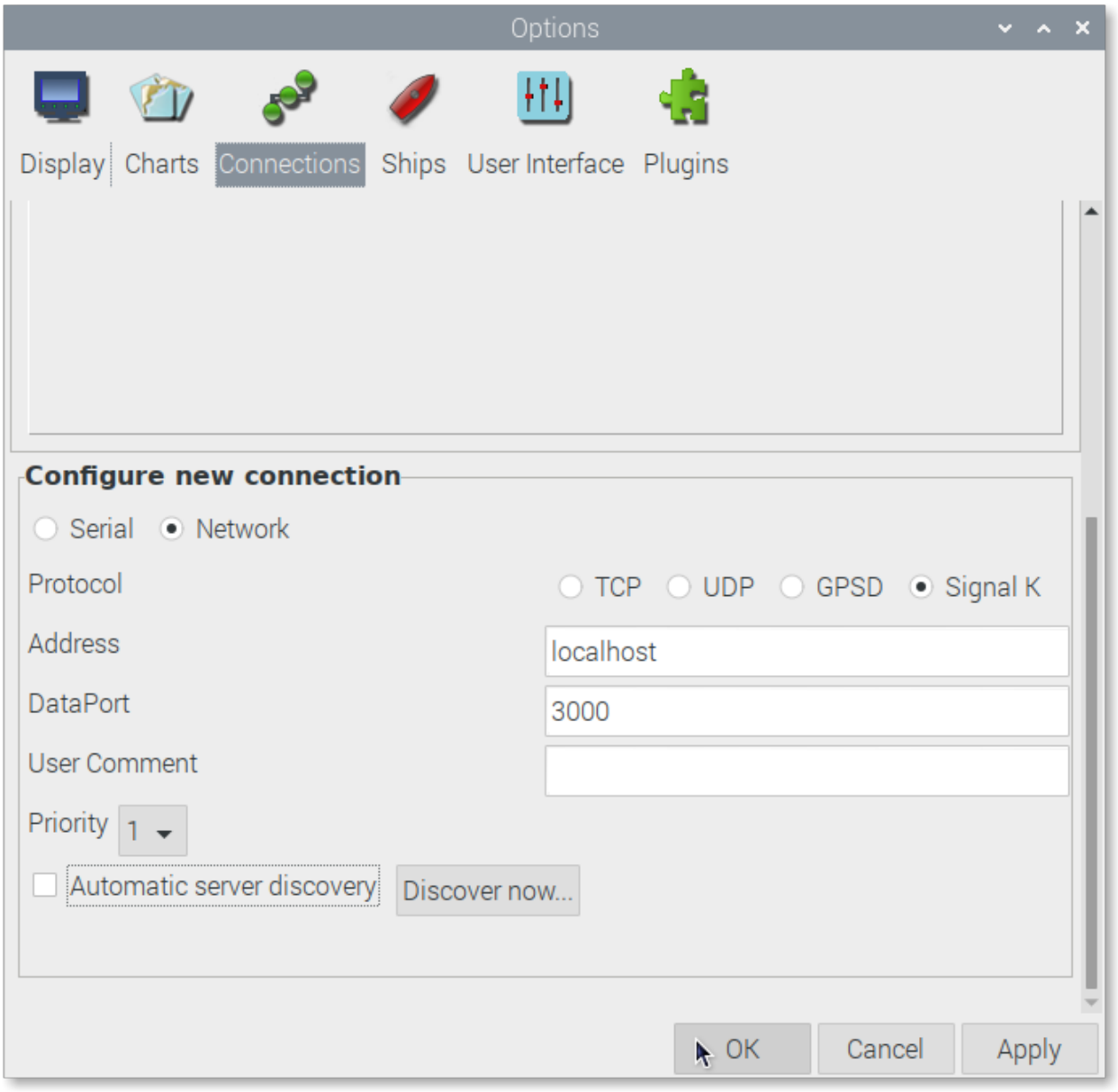


Go to `Info` tab and click on `Check configuration` again to see the changes:



39.5 Configuring OpenCPN

As of version 5.2, OpenCPN can manage Signal K data, so we no longer need to convert Signal K data to NMEA 0183. You just need to create this connection in OpenCPN:



The default port of the Signal K server is 3000, but you can change it. If you are not sure which port you have configured, go to the `Signal K Installer` app to check.

Go to `Info` tab and click on `Check configuration` again to see the changes:

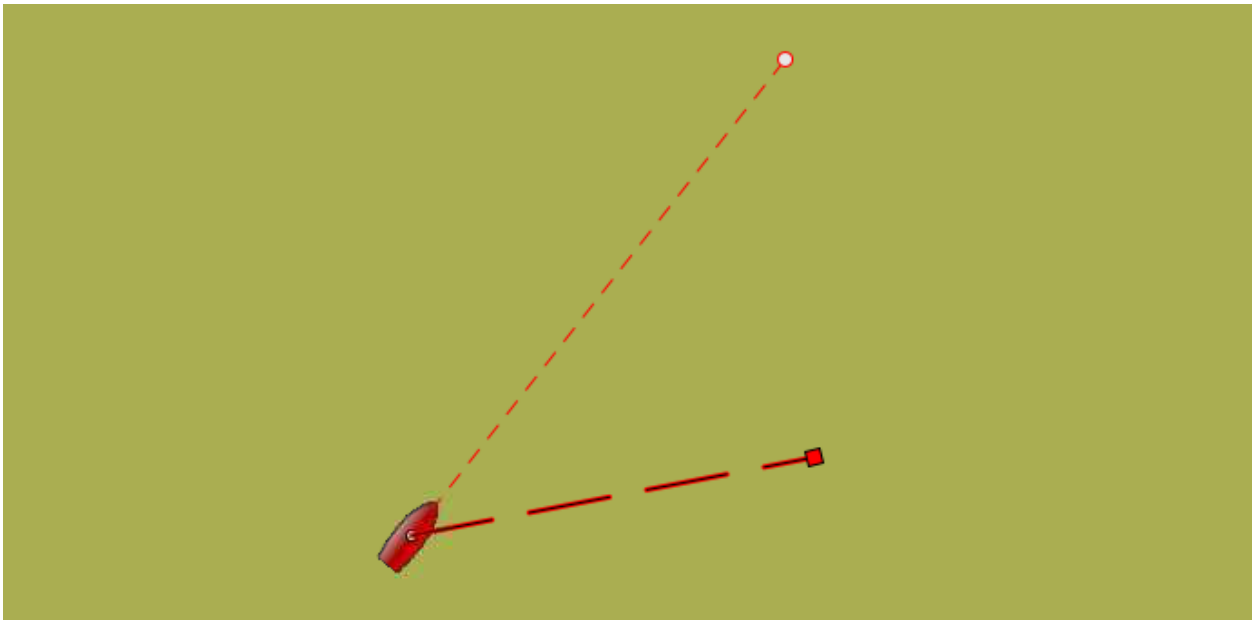
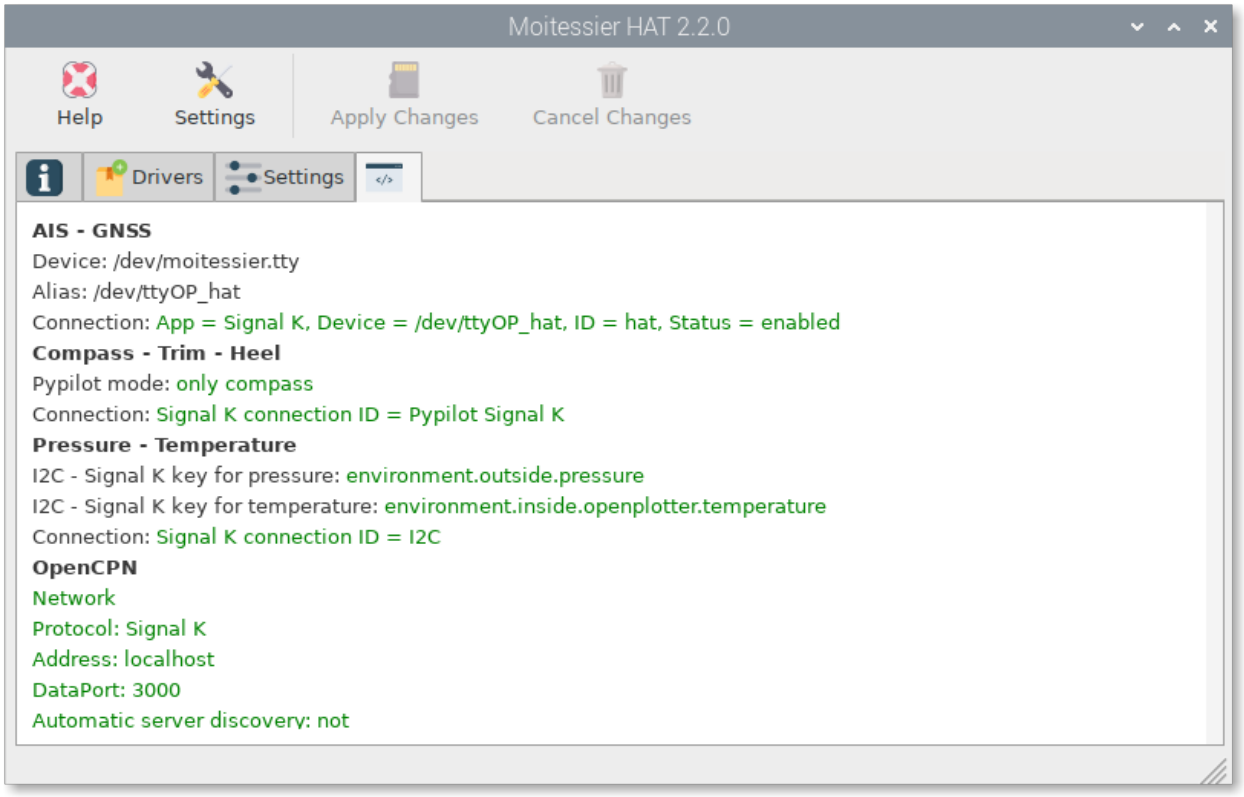


Fig. 1: Magnetic Heading (circle), Course Over Ground (square)



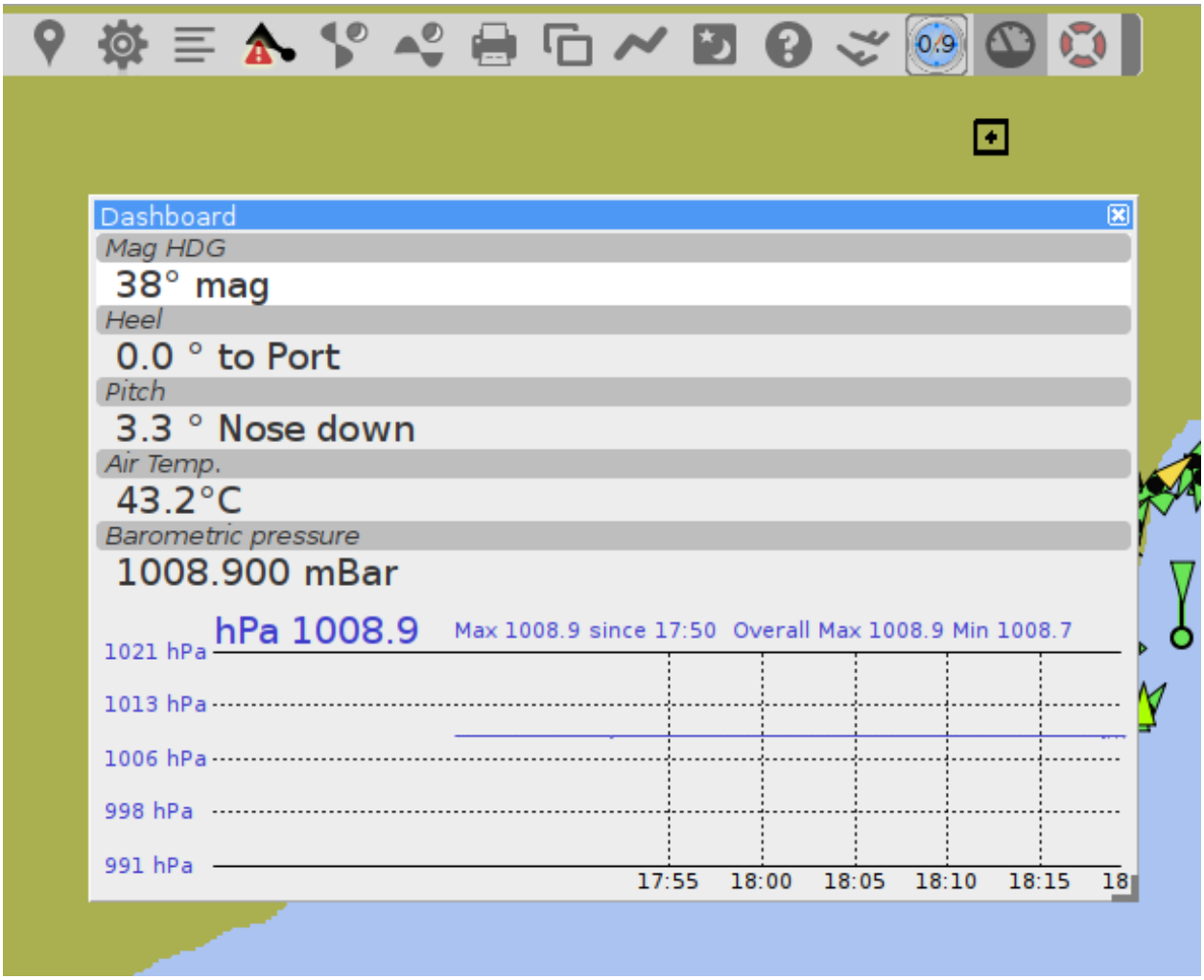


Fig. 2: Heel, Pitch and Pressure

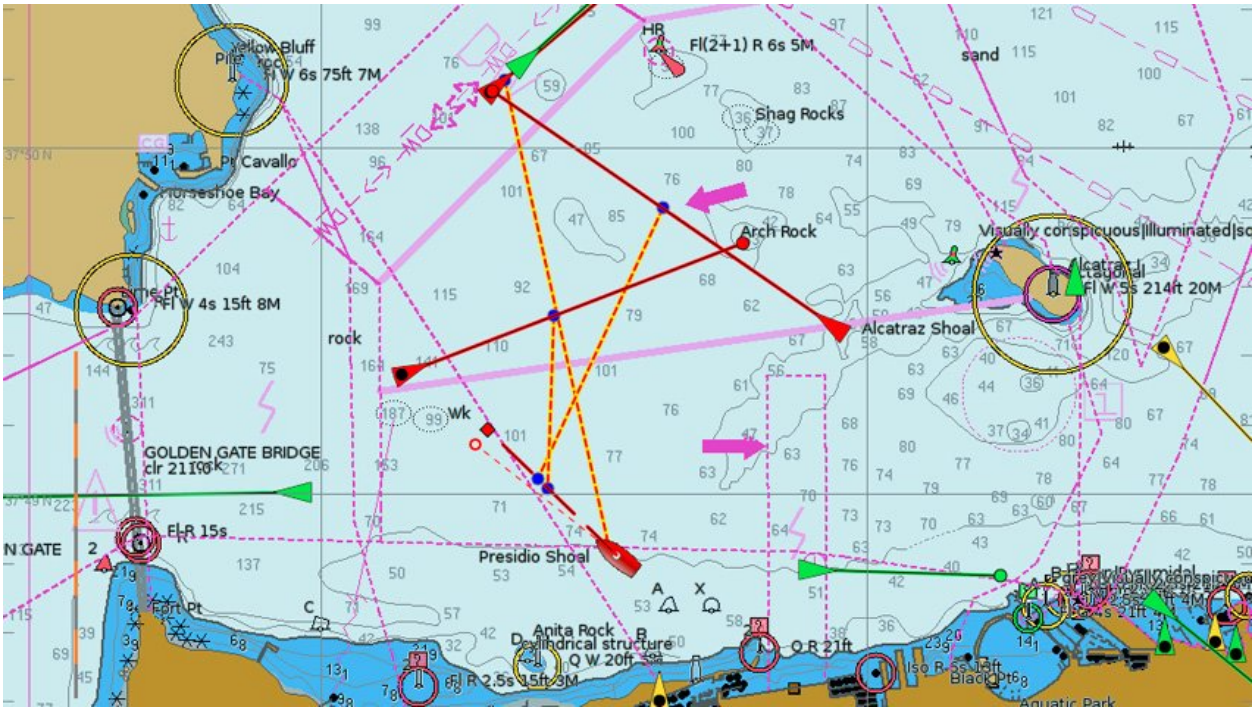
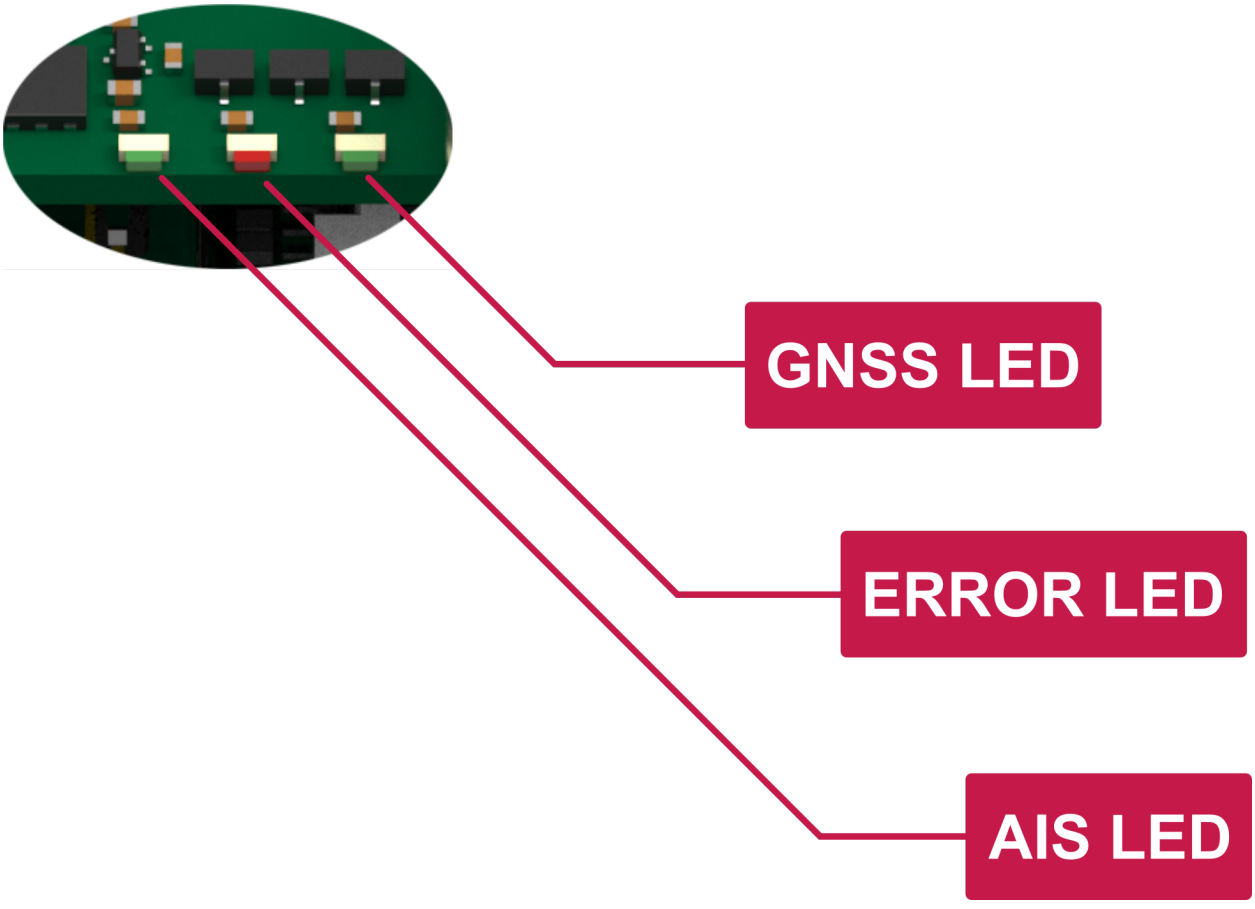

Fl(2+1) R 6s 5M
sand
Shag Rocks
Arch Rock
Alcatraz Shoal
Fl W 4s 15ft 8M
rock
Wk
GOLDEN GATE BRIDGE
Fl R 15s
Presidio Shoal
Anita Rock
Q W 20ft
Q R 21ft
Fl R 2.5s 15ft 3M
Fl W 5s 214ft 20M

Fig. 3: AIS

CHAPTER 40

Status LEDs



Meaning of LED Sequence

LED switched off
LED switched on
LED flashing at variable/inconsistent frequency
LED flashing at consistent frequency

LED Patterns

Status LED	Color	Pattern	Meaning
ERROR	red		No errors occurred
		Duration: as long as the error exists	Internal buffer overflow. The data is processed too slowly by the Raspberry Pi.
		Duration: until reset	Error occurred on device self-test
AIS	green		No AIS data available
			AIS data being received
GNSS	green		No GNSS fix
			GNSS fix

Error Output

The ERROR LED indicates the following error types:

- Minor/temporary error: Every NMEA source (AIS, GNSS) and every NMEA output (SPI interface to Raspberry Pi) is assigned a memory area in the microcontroller in the form of a buffer. The data of multiple inputs is written to the SPI buffer. If data is written to this buffer quicker than it is read by the Raspberry Pi, the buffer will be full after a certain time due to its memory restrictions. In such a case, no more data can be written and a buffer overflow occurs. As soon as the buffer is read, the device continuous to process data.
- Serious (system) errors: The device is running a self-test after each reset. If an error occurs during hardware initialization, the ERROR LED flashes in a constant pattern. The device can no longer be used in this case. If a restart does not resolve the problem, please contact the Rooco support team.

You can read the system errors clicking `Check settings` in `info` tab of `Moitessier HAT` app.

The error is coded as bit pattern:

- Bit 0: GNSS failed
- Bit 1: AIS receiver 1 failed
- Bit 2: AIS receiver 2 failed
- Bit 3: SPI host interface failed
- Bit 4: UART host interface failed

e.g. system errors = 0x00000011→GNSS and UART host interface failed

CHAPTER 41

I²C

41.1 What is I²C?

In a nutshell, I²C (Inter-Integrated Circuit) is a reliable, cheap, well-defined means of connecting sensors to OpenPlotter.

For details, refer to https://en.wikipedia.org/wiki/I%C2%B2C.

41.2 Setting up I²C on Raspberry

41.2.1 Install OpenPlotter I²C app

Go into >Openplotter>Settings and then hit refresh.

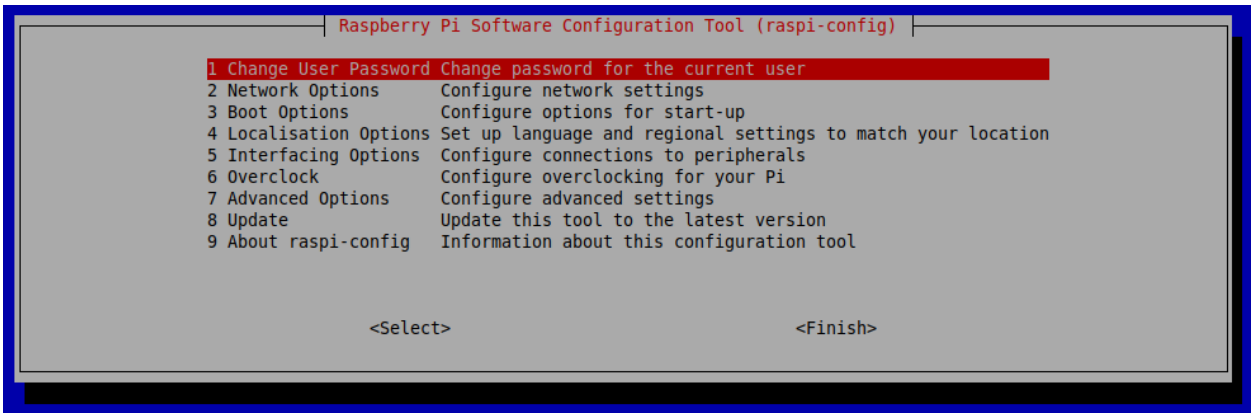
In the list of openplotter Apps, go down to I²C Sensors and select then install.

41.2.2 Switching on I²C on your Raspberry

In the menu of Raspbian OS, go to >Preferences>Rasp Pi Config to start the Raspberry Config tool.

Select [5 Interfacing Options], and then enable I2C.

Once enabled, hit <Finish> and reboot your Raspberry. After reboot, I2C bus is enabled.



Further information can be found on the official Raspberry website.[1]

41.2.3 Powerdown(!) the Pi and install the sensor(s)

I²C require four connections ("cables") between you sensor and your Raspberry.

Two pins, GND "ground" and 3.3V "power" provide exactly that, 3.3 V power to your sensor. It's obvious, your sensor has to be specified to run with 3.3 V. Sensors rated for 5.0 V will not work on your Raspberry. Some sensors on the market can handle both voltages.

Two more pins, SDA "Data" and SCL "Clock", are used to transmit data between your Raspberry and the sensor chips.

On a Raspberry pi, you need to connect pins 1 (3.3V), 3 (SDA), 5 (SCL), 9 (GND).[2]

41.2.4 Soldering or Plug'n'Play?

There are sensors on the market, that are connected with the four ports (3.3v power, GND, SDA, SCL) by simply soldering four cables and connect them the PI.

To make life easier for prototyping, plug-and-play connectors have been developed. Amongst them "Stemma QT"[3] and "QWIIC"[4] are quite common and offer a variety of sensors and other peripherls.

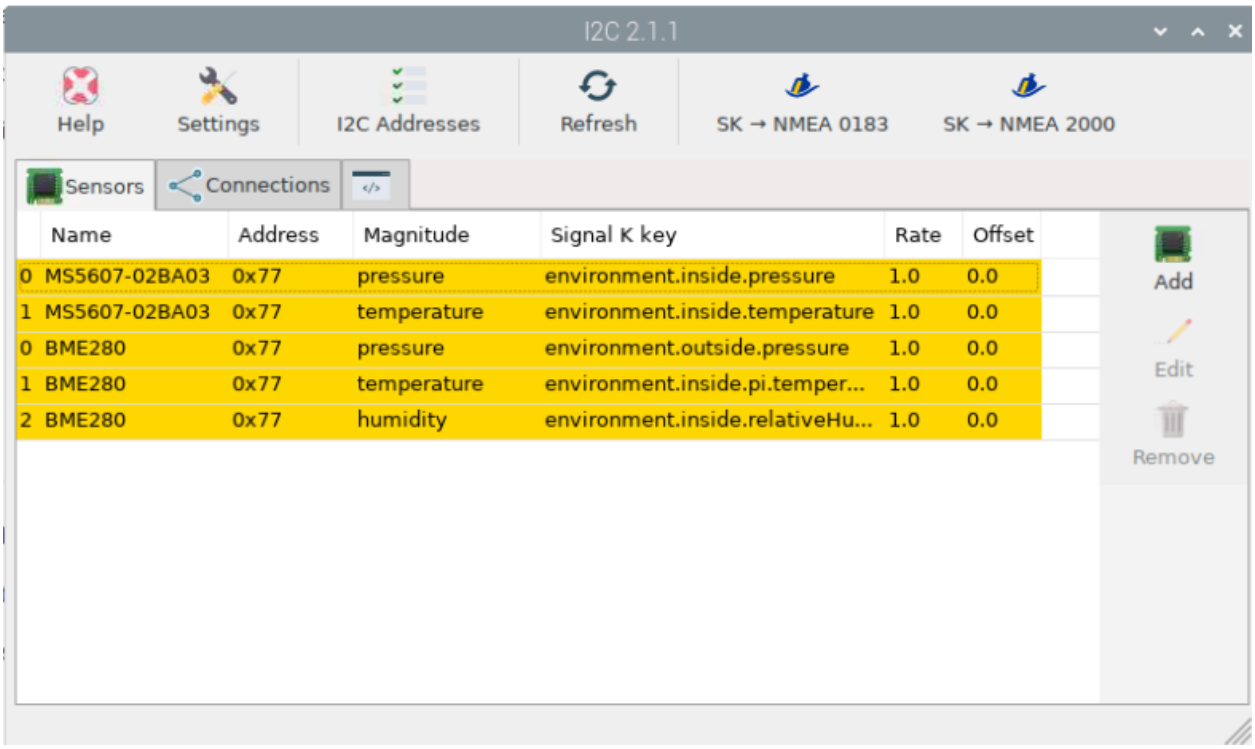
If you use those connectors, you require on the Raspberry side of things a 4-Pin female 2mm socket cable.

Once all connections are solid, power the Pi back up.

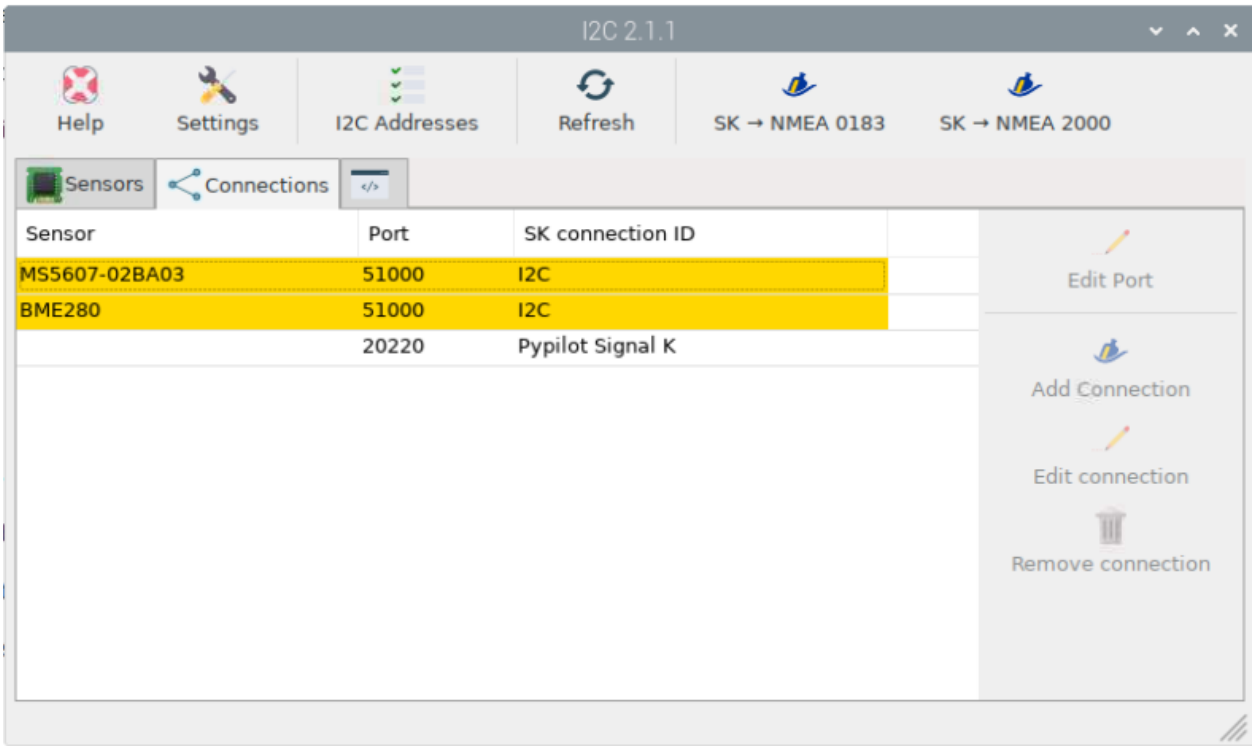
41.3 Configure the I²C OpenPlotter app

Got to >Openplotter>I2C and add all sensors providing a name for each that makes sense and fits with the Signal K specification, see http://signalk.org/specification/1.0.4/doc/signalk.pdf, see the picture below:

[1] https://www.raspberrypi.com/documentation/computers/configuration.html#raspi-config
[2] https://pinout.xyz/pinout/i2c
[3] https://learn.adafruit.com/introducing-adafruit-stemma-qt/what-is-stemma-qt
[4] https://www.sparkfun.com/qwiic



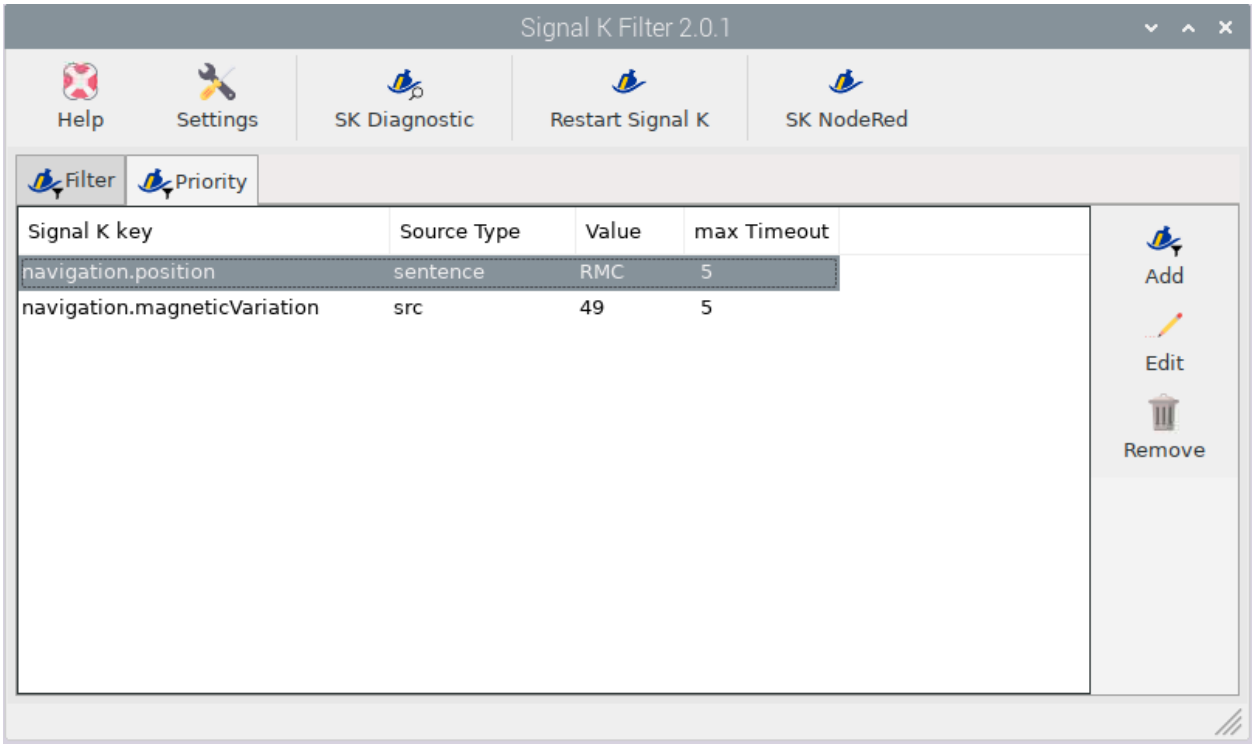
On the connection Tab, add connection to Signal K for each of the sensors:



Note: The Pypilot entry will not show up until you have done the Pypilot configs

CHAPTER 42

Signal K filter



42.1 What is Signal K Filter?

The Signal K server wants to get all data it gets.

Why should we filter Signal K?

- Some devices send data which are everytime wrong (You can't disable unwanted sentences on a nmea 2000 bus, or a defective water temp sensor in the log on nmea 0183, or . . .)
- You have a backup device (A second GPS or AIS)

To explain the problem we look at two GPS with a distance to each other.

If you zoom your nautical chart, your boat will constantly zigzag. Or one GPS has a bad reception.

The Signal K path will be identical but not the source.

The typical way would be to deside for every path which source you want to have. As long as you are able to make this decision everything is fine. (When you start converting nmea 2000 to nmea 0183 or the other way. Is there a chance to say only convert the Signal K path from a special source? No.

There are enough reasons to filter the data before it gets into the Signal K server.

There are two ways to filter.

In OpenPlotter SKfilter the page:

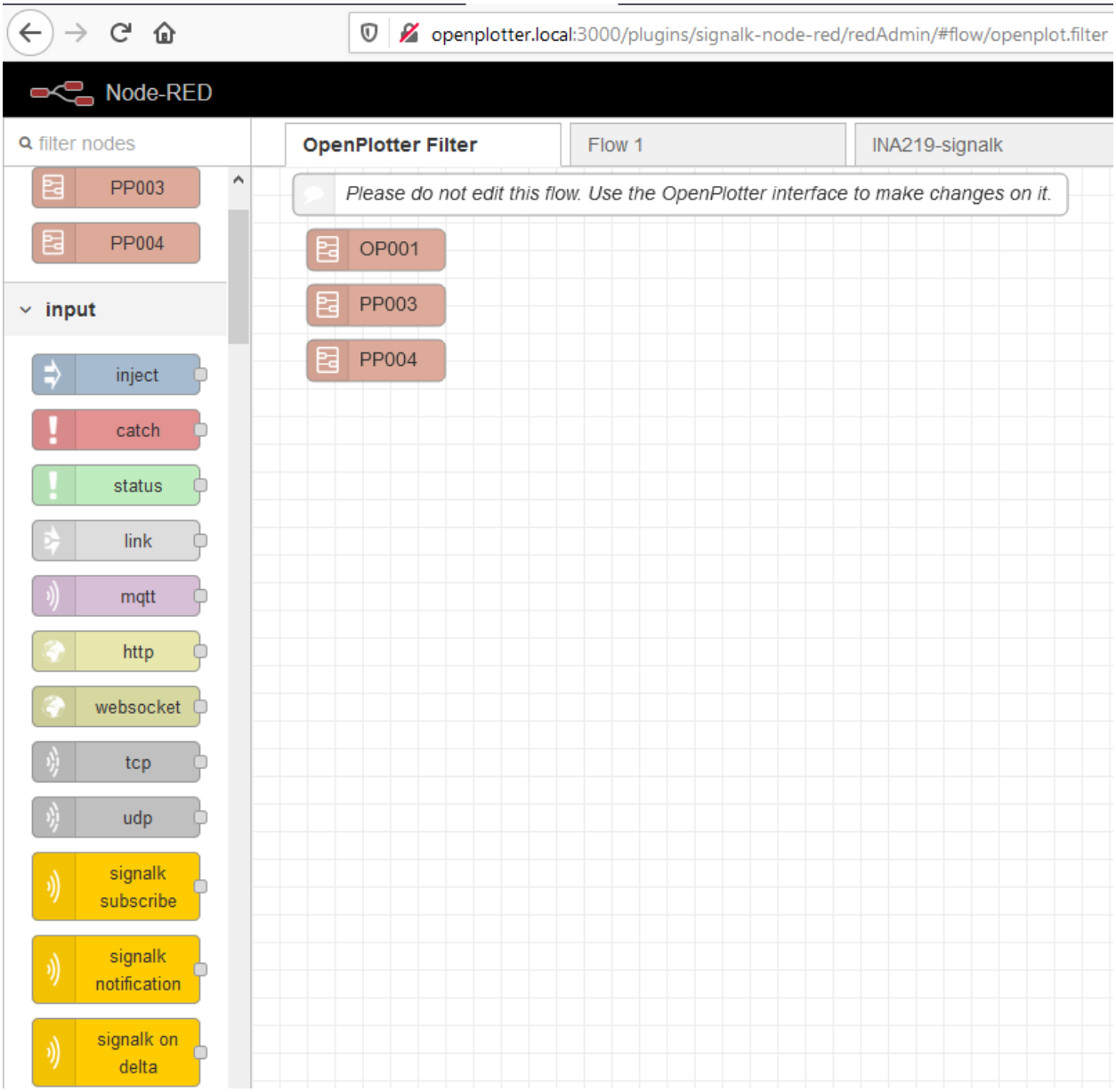
- Filter is the hard way. Only Signal K pathes witch match the criteria come to the server.
- Priority is the smooth way to filter. The criteria are the same, but if there is no communication within a time limit (timeout) that meets the criteria, any source for that Signal K path comes to the Signal K server. This is a kind of fallback functionality

42.2 How does it work?

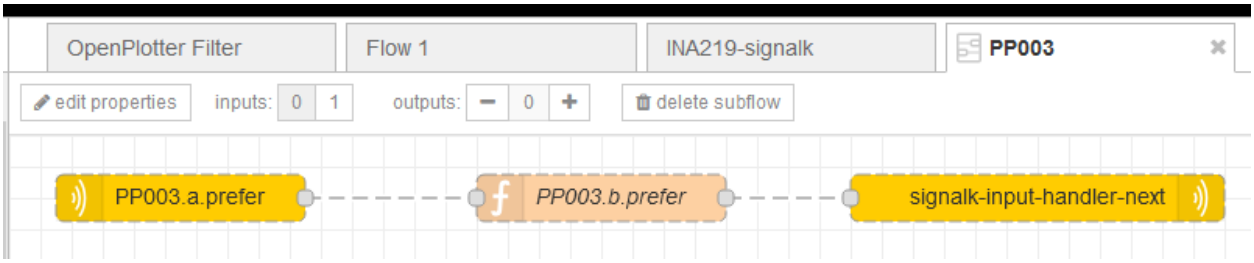
Signal K filter is based on the Signal K Node-RED library.

OpenPlotter does manage some Node-RED source code. (The flow page has the name OpenPlotter Filter. The code is put into subflows.)

Two pictures of Signal K filter in Node-RED created by this app.

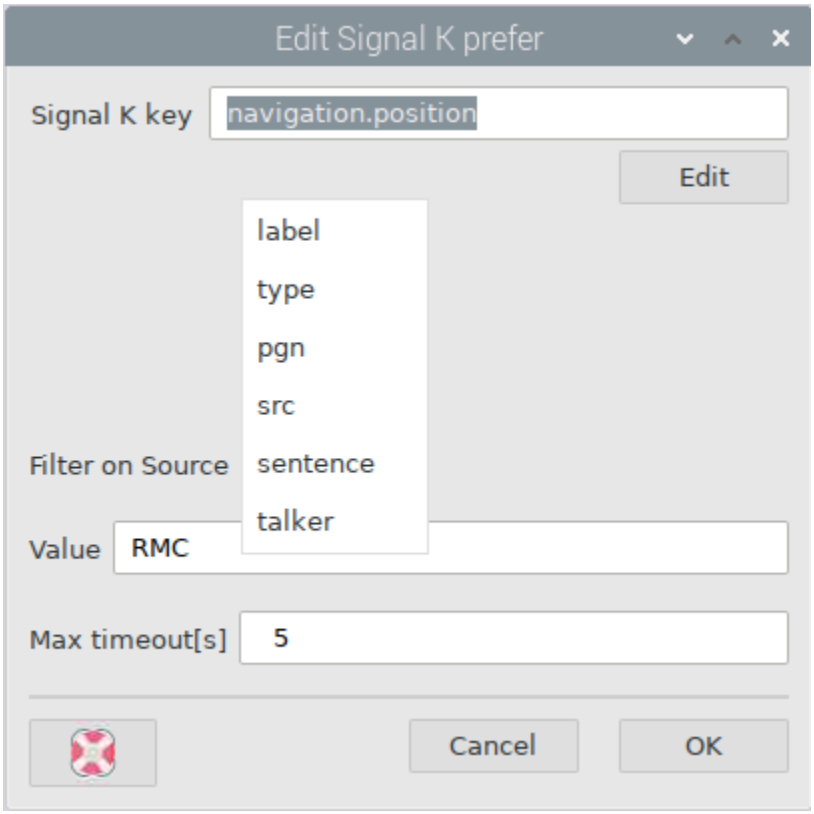


This picture show how it looks like in Node-RED

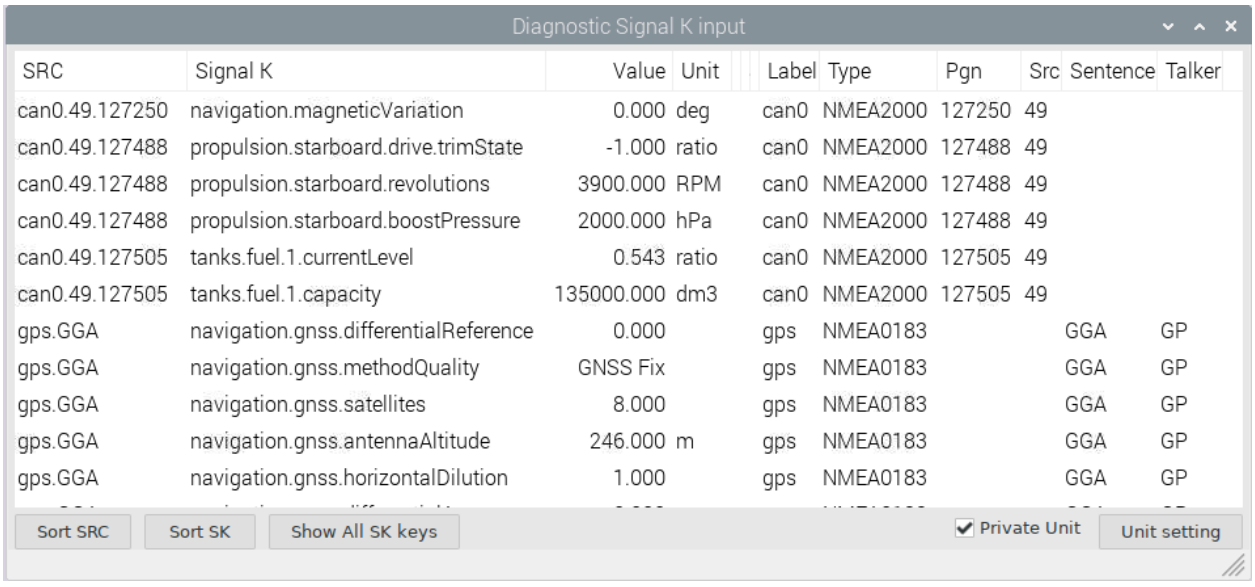


Here you see the edit view for an opened subflow created by OpenPlotter Signal K filter.

42.3 How can I get the criteria?



On the picture you can see "Filter on Source" (criteria) to choose. On the following picture you see that SK Diagnostic show you the available values.



Tip: To find double Signal K path click on "Sort SK"

Note: Signal K uses ONLY SI units!!! (rad, K, m, Hz, Pa, V, A, J, s)

To make the values readable they are converted in SK Diagnostic! Unselect "Private Unit" to see real Signal K values!!!
The "Unit setting" is only for SK Diagnostic "Private Unit" nothing else!!!

42.4 Known issues

The Signal K server starts before Node-RED. A few seconds you will get data unfiltered.

CHAPTER 43

Kplex

CHAPTER 44

SDR VHF

This application will allow you to easily use *Software Defined Radio* devices in OpenPlotter. These devices can be used as a wide band radio scanner. Applications include:

- Use as a police radio scanner.
- Listening to EMS/Ambulance/Fire communications.
- Listening to aircraft traffic control conversations.
- Tracking aircraft positions like a radar with ADSB decoding.
- Decoding aircraft ACARS short messages.
- Scanning trunking radio conversations.
- Decoding unencrypted digital voice transmissions such as P25/DMR/D-STAR.
- Tracking maritime boat positions like a radar with AIS decoding.
- Decoding POCSAG/FLEX pager traffic.
- Scanning for cordless phones and baby monitors.
- Tracking and receiving meteorological agency launched weather balloon data.
- Tracking your own self launched high altitude balloon for payload recovery.
- Receiving wireless temperature sensors and wireless power meter sensors.
- Listening to VHF amateur radio.
- Decoding ham radio APRS packets.
- Watching analogue broadcast TV.
- Sniffing GSM signals.
- Using rtl-sdr on your Android device as a portable radio scanner.
- Receiving GPS signals and decoding them.
- Using rtl-sdr as a spectrum analyzer.

- Receiving NOAA weather satellite images.
- Listening to satellites and the ISS (International Space Station).
- Radio astronomy.
- Monitoring meteor scatter.
- Listening to FM radio, and decoding RDS information.
- Listening to DAB broadcast radio.
- Listening to and decoding HD-Radio (NRSC5).
- Use rtl-sdr as a panadapter for your traditional hardware radio.
- Decoding taxi mobile data terminal signals.
- Use rtl-sdr as a high quality entropy source for random number generation.
- Use rtl-sdr as a noise figure indicator.
- Reverse engineering unknown protocols.
- Triangulating the source of a signal.
- Searching for RF noise sources.
- Characterizing RF filters and measuring antenna SWR.
- Decoding Inmarsat STD-C EGC geosynchronous satellites.

In OpenPlotter `SDR VHF` app we include some of these interesting tools for maritime use.

Note: You can buy this item in the store.

Before you start using any of these bundled tools, there are a few steps you should take.

44.1 Antennas

Each tool included in `SDR VHF` uses a specific frequency range and you will need a specific antenna for each of them. These are the recommended antennas:

AIS

AIS signals are broadcast at both 161.975 MHz and 162.025 MHz and have a maximum range of approximately 75 kilometers. So if you are more than 75 kilometers away from any boats, you will probably not be able to receive AIS signals. AIS is also considered a line of sight signal, meaning that if there are large buildings or mountains in the way of your antenna and the boats, AIS signals could be blocked. Because of this reason it is important to put your antenna as high up as possible. There are multiple commercial AIS or VHF antennas designed for marine use that will work. However, sometimes home made antennas work even better and they of course are cheaper.

ADS-B

GQRX

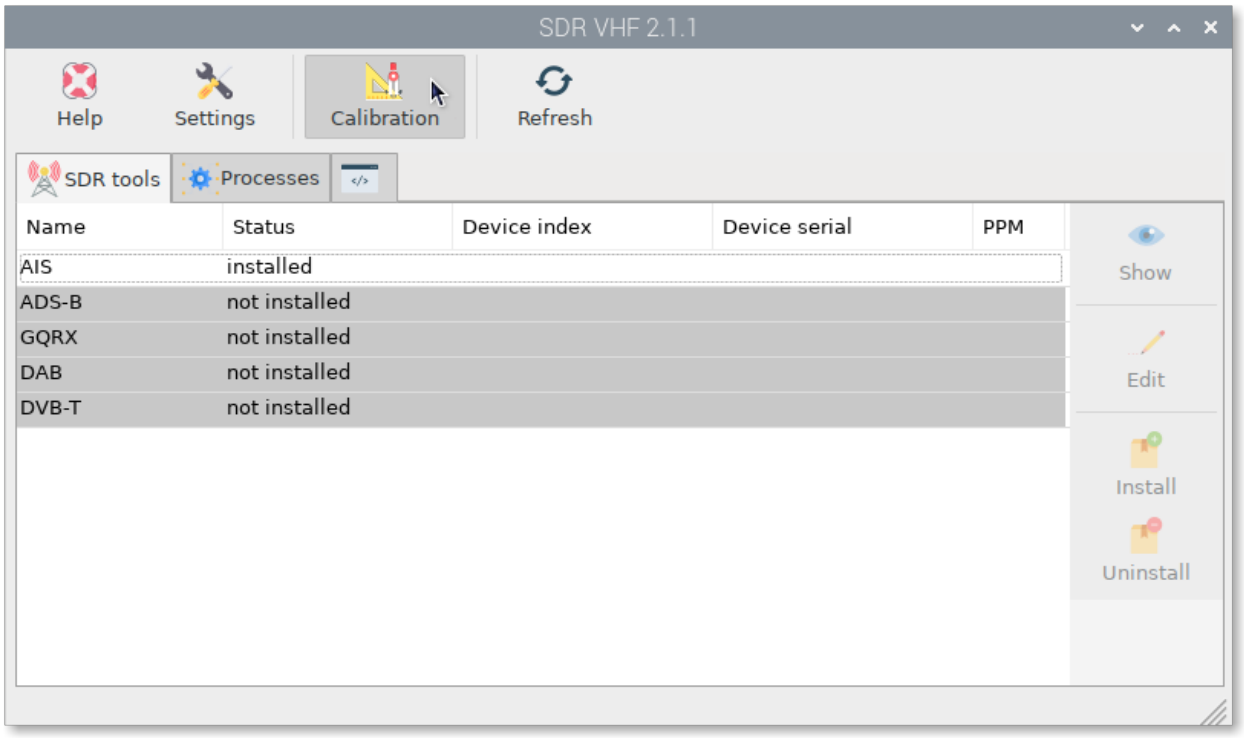
DAB

DVB-T

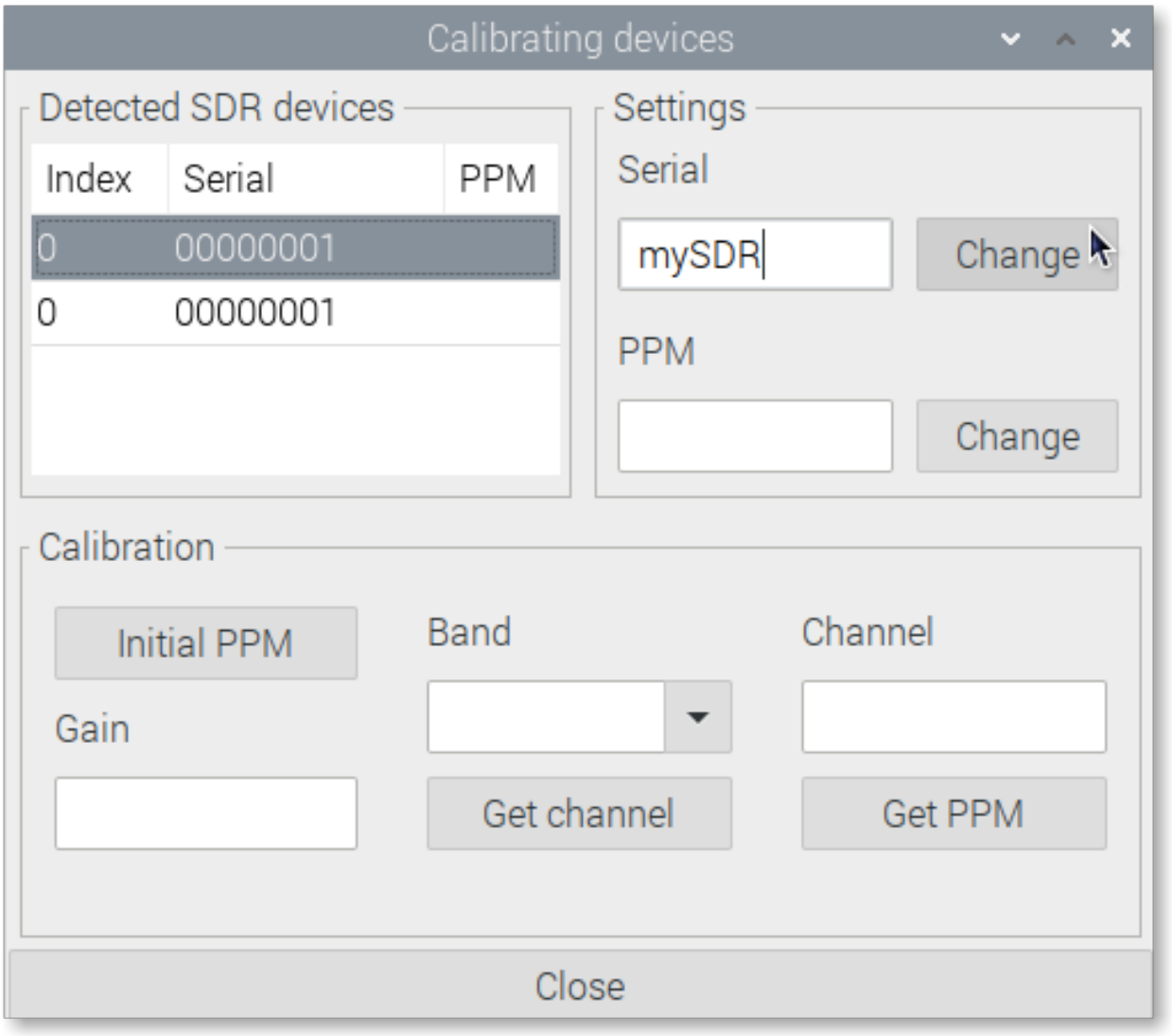
44.2 Edit device serial numbers

SDR devices can only be used by one program at a time. If you have more than one device you can select which one you want to use in some tools and some others will take the first available device. Most of the SDR devices available on the market have the same serial number (00000001) and this makes it difficult to identify them, so we have added a tool to change these serial numbers if necessary.

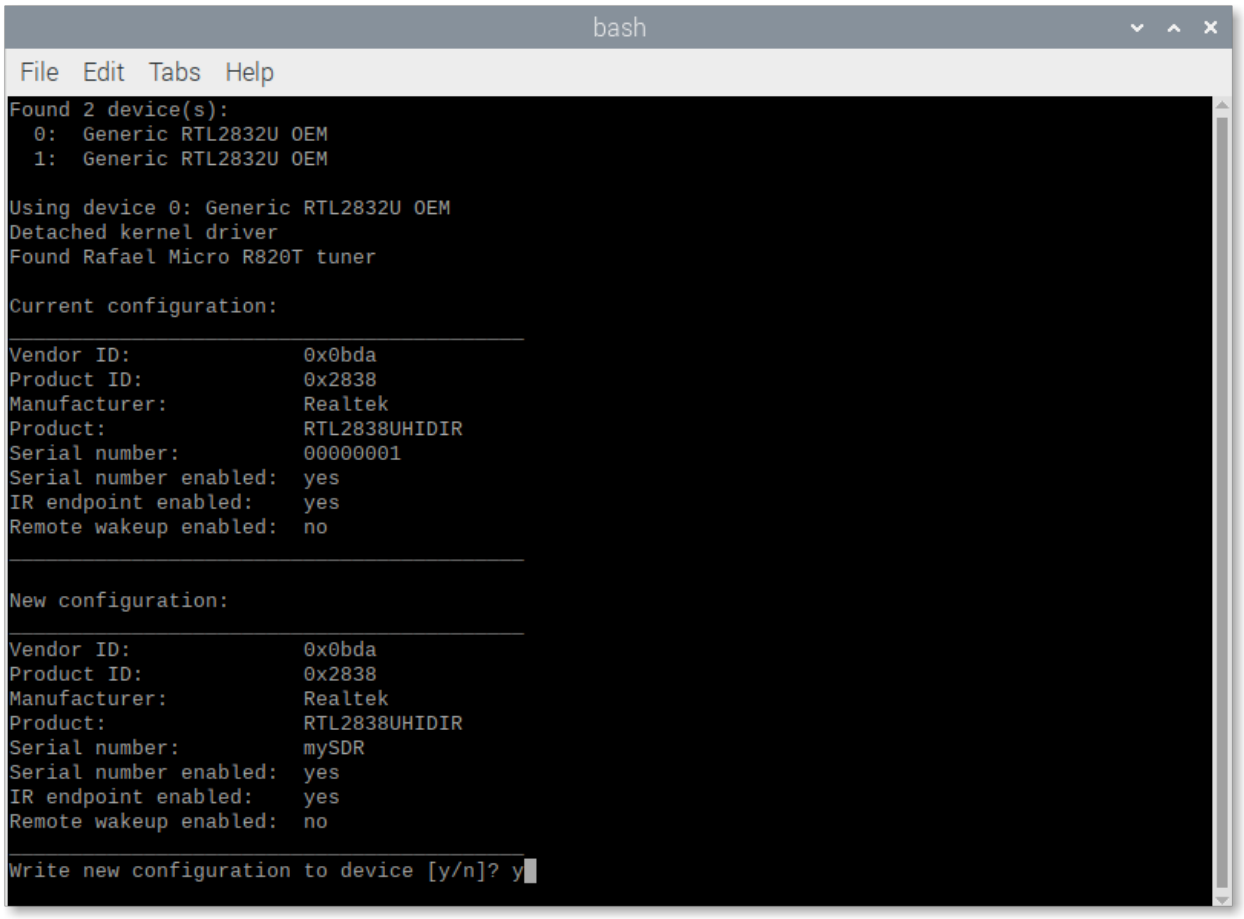
Click on `Calibration`



You will see the list of connected SDR devices. Select any of them, type a new name in `Serial` field and click on `Change`:



A new window will open asking for confirmation. Type `y` and press enter. Finally replug the device and open again `SDR VHF` to check the changes.



44.3 Calibrate devices

Every SDR device will have a small frequency error as it is cheaply mass produced and not tested for accuracy. This frequency error is linear across the spectrum, and can be adjusted in most SDR programs by entering a **PPM** (parts per million) offset value.

Important:

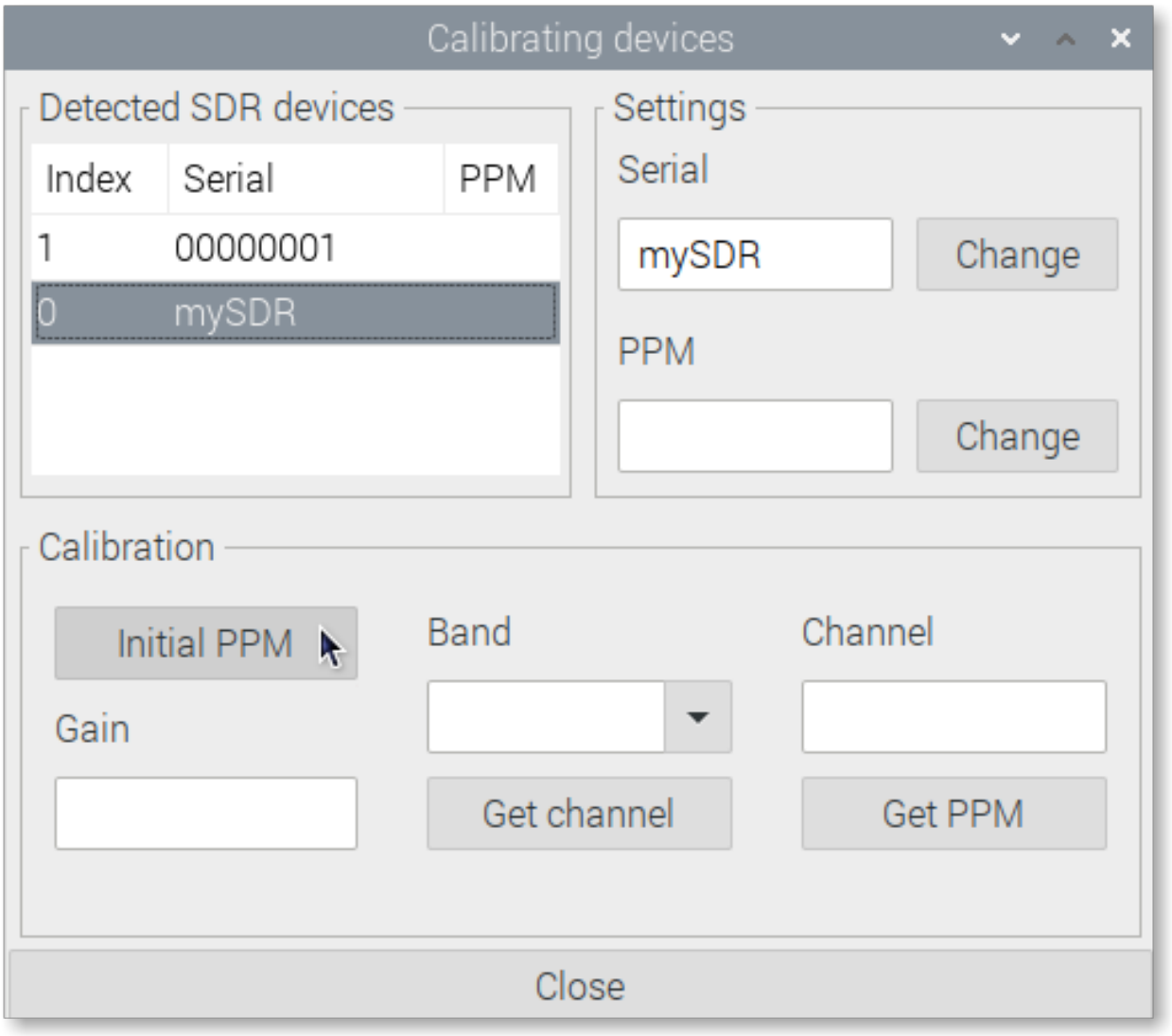
- PPM values have a tolerance of +/-7
- PPM values can be negative
- If you do not find the correct PPM you will not get AIS data
- Some devices have a built-in temperature compensated oscillator (TCXO) that provides a PPM close to 0. These devices do not require calibration

---

If your device does not have TCXO, you need to know what its PPM value is, that is why we have added a tool to find it using GSM frequencies.

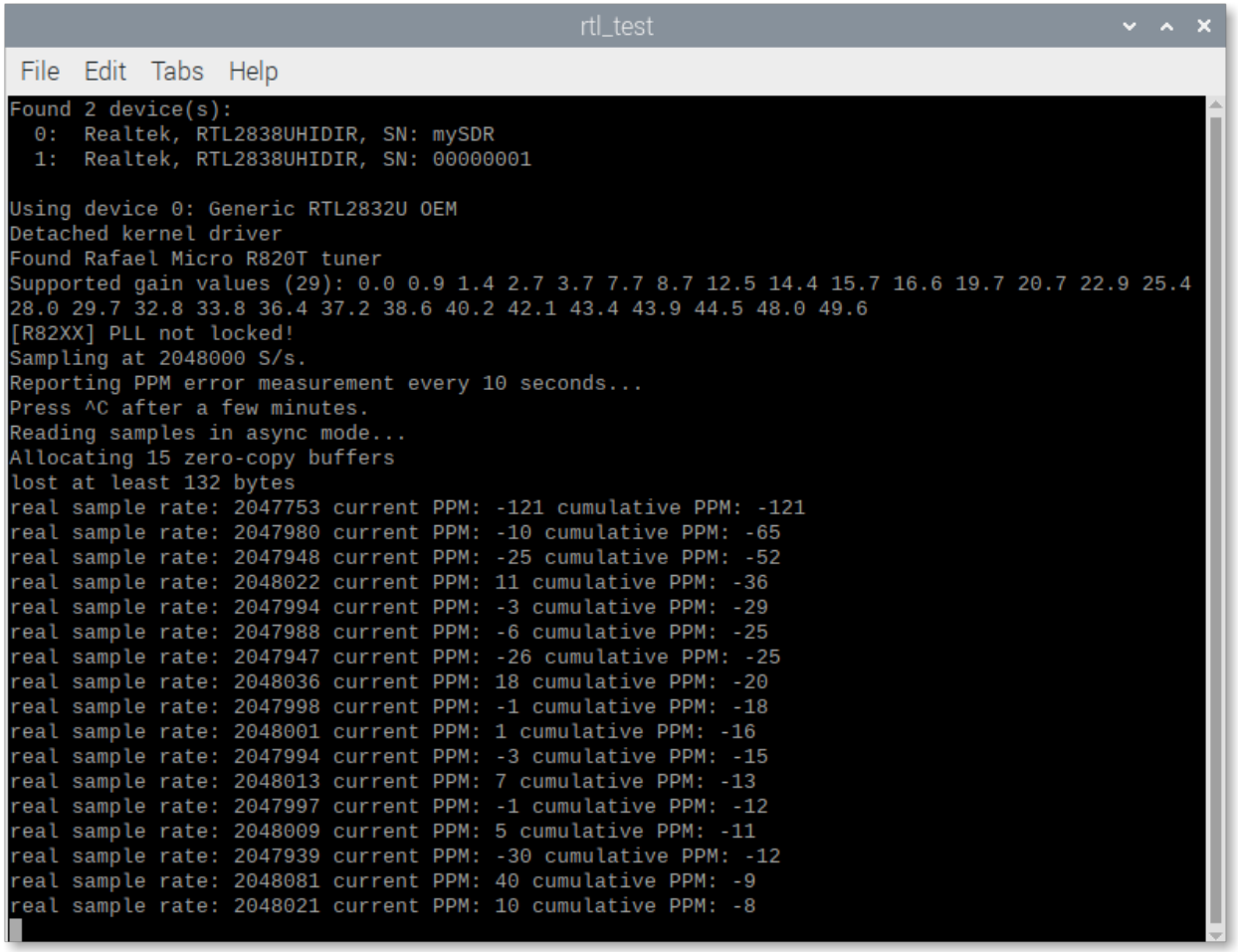
Before starting the calibration process, make sure there is an antenna connected to your device.

Go to `Calibration` again, select the device and click `Initial PPM` to get an approach to your PPM value:



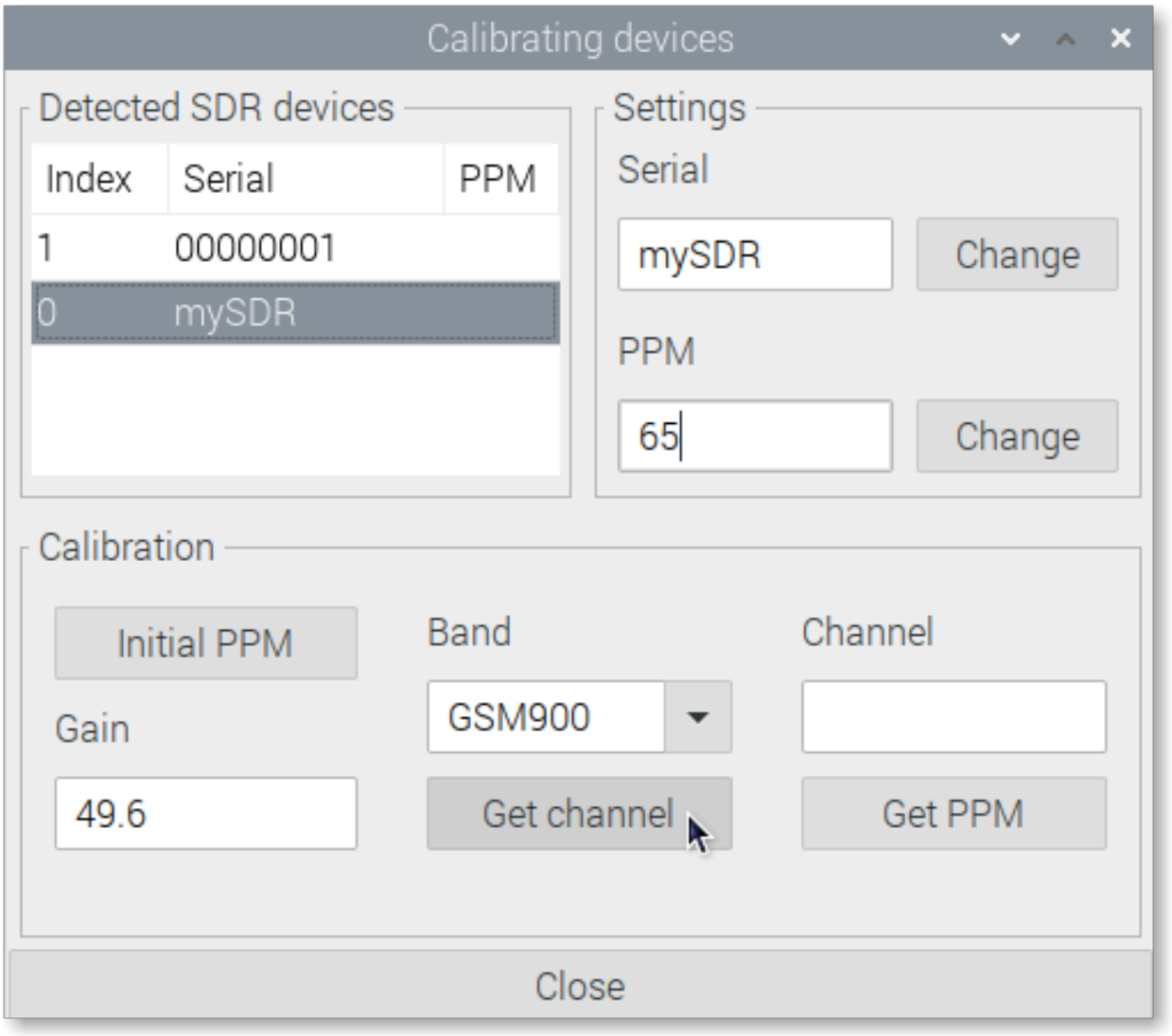
The PPM value will change with temperature, so let the device run for at least 30 minutes. The longer you let the program calculate the better result you will get. If you run the program for hours you will get almost the final PPM but if you do not have time just wait for the value to stabilize.

Write down the stabilized PPM value and the maximum supported gain value for your device (usually 49.6).

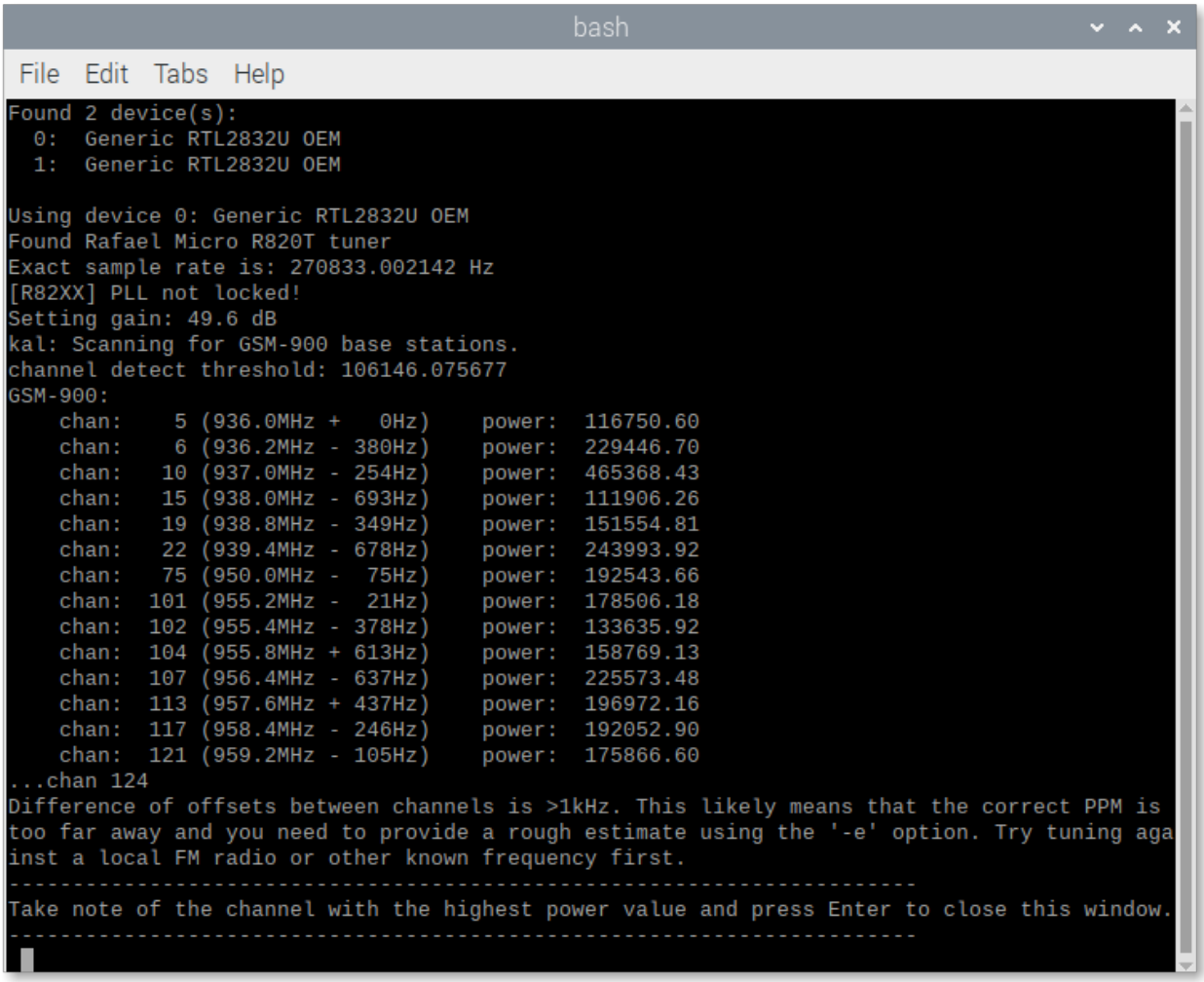


Close the program and put the PPM value in the `PPM` field and the maximum supported gain value in the `Gain` field.

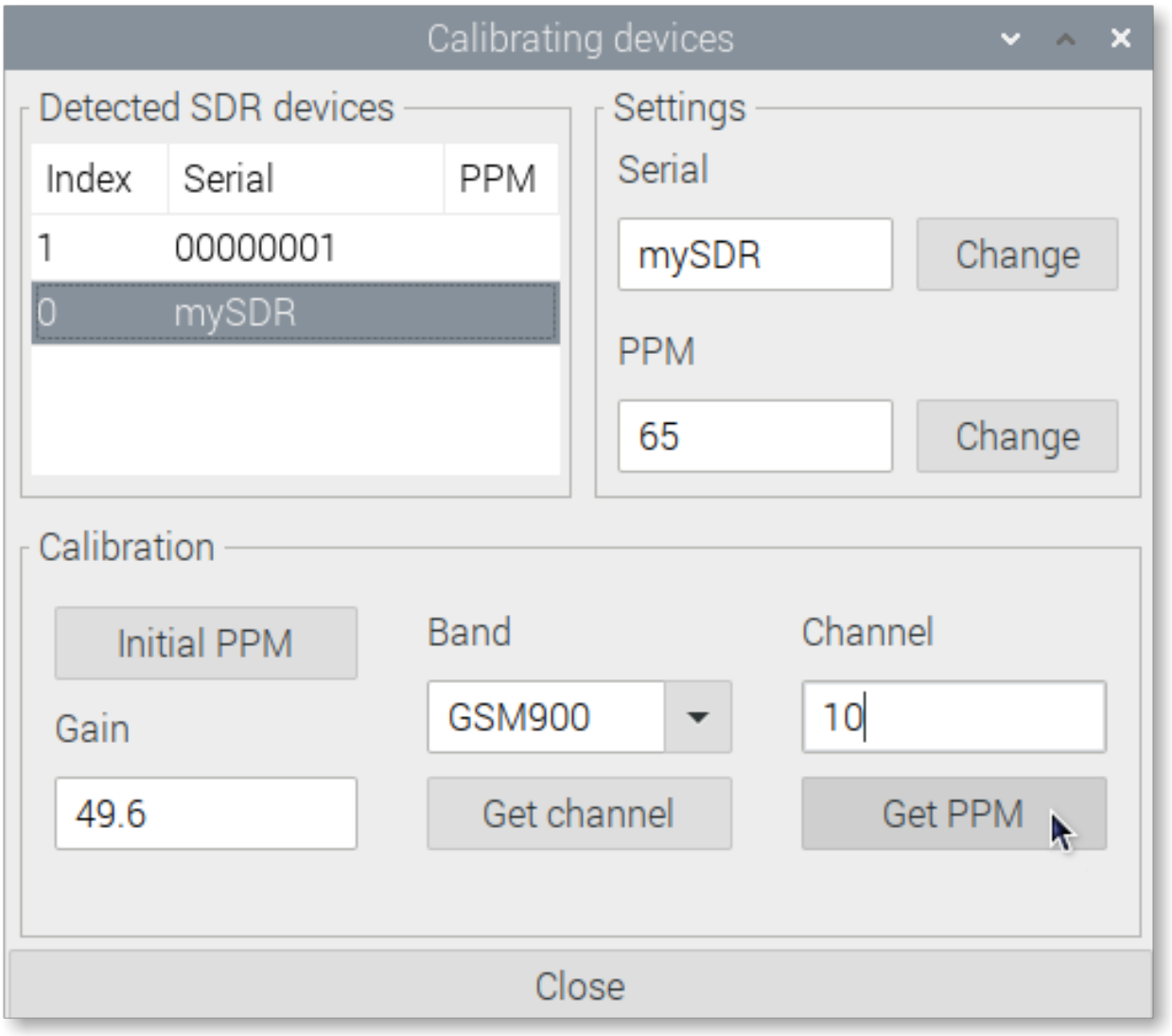
Select the `GSM band` for your zone and press `Get channel`:



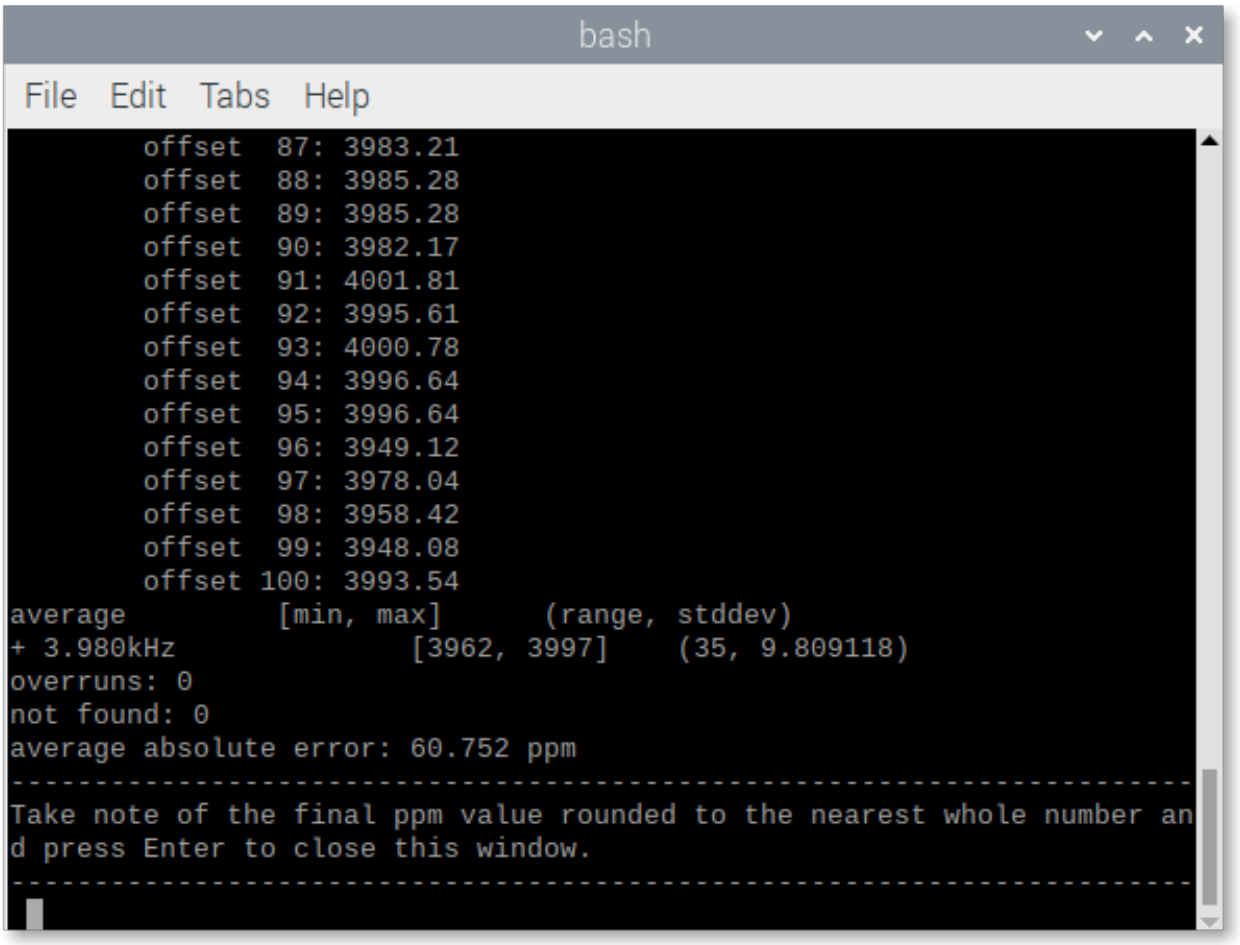
Write down the channel with the highest power value and close the window:



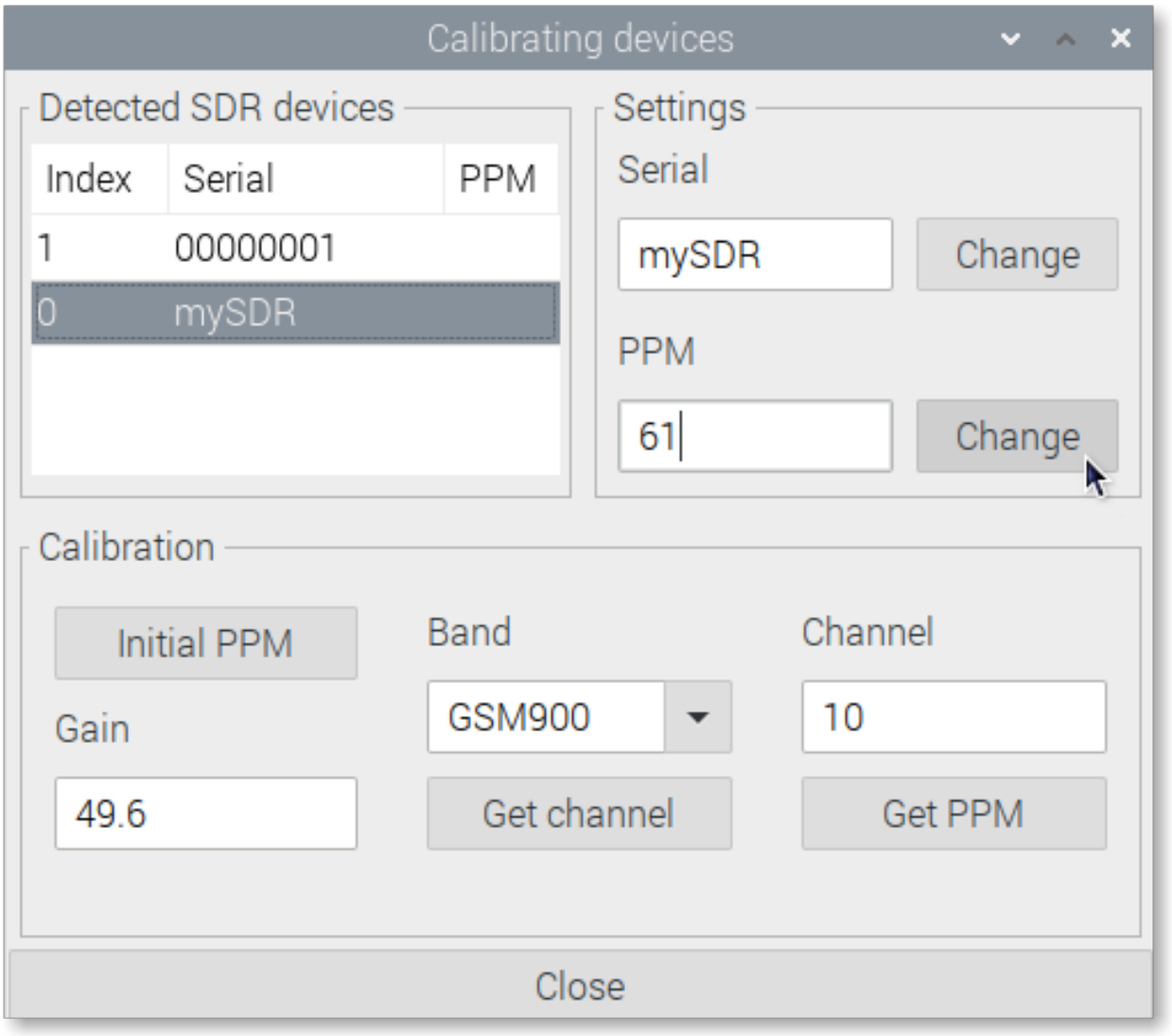
Put the strongest channel into `Channel` field and press `Get PPM`:



Write down the final PPM value and close the window:



Put the final PPM into the `PPM` field without decimals rounding the value to the nearest integer number, click on `Change` and you are done:

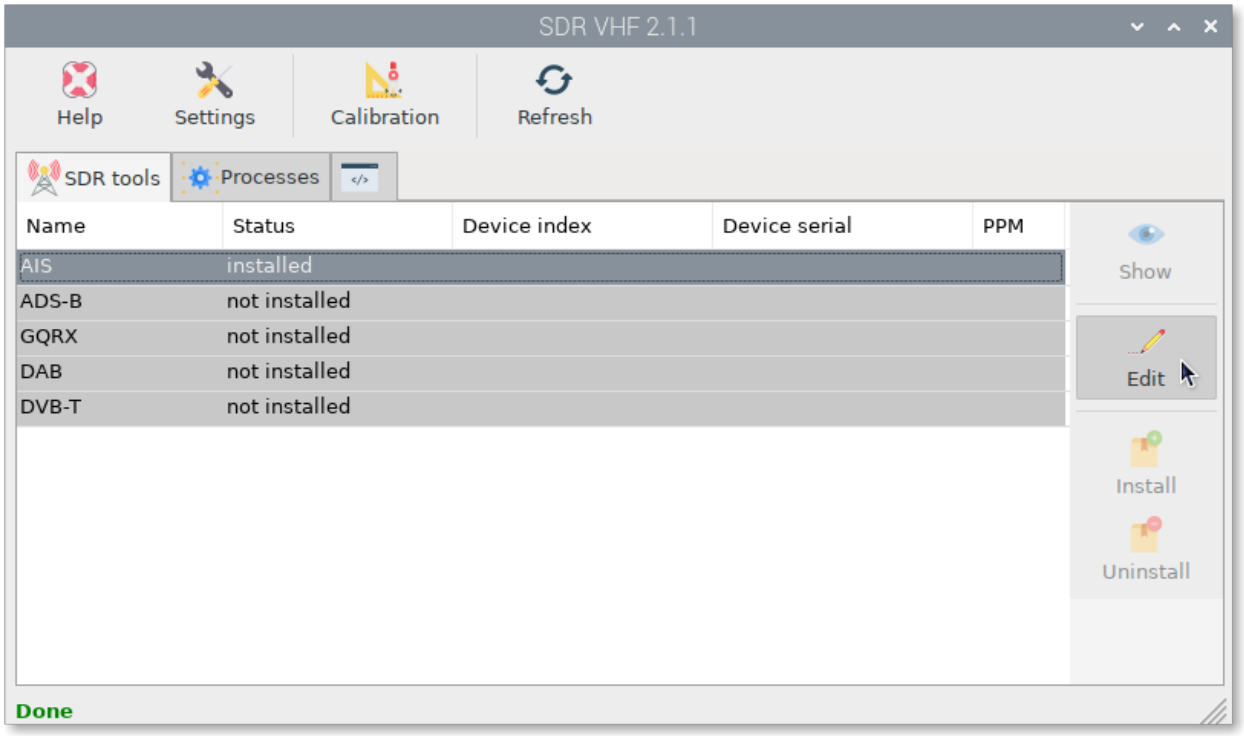

Calibrating devices
Detected SDR devices
Index
Serial
PPM
1
00000001
0
mySDR
Settings
Serial
mySDR
Change
PPM
61
Change
Calibration
Initial PPM
Band
Channel
Gain
GSM900
10
49.6
Get channel
Get PPM
Close

CHAPTER 45

AIS

AIS tool comes pre-installed in SDR VHF. To start receiving AIS data you just have to follow a few simple steps.

Select AIS app and click on Edit:

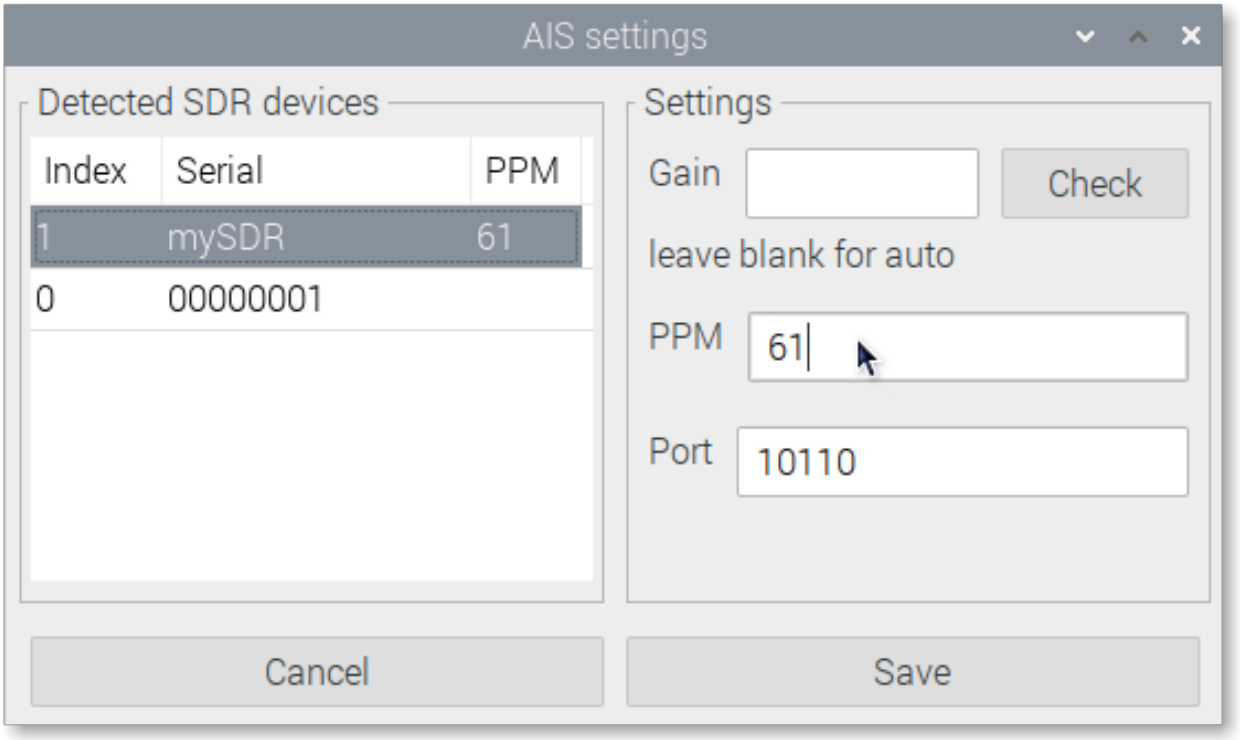


Select the device you want to use to get AIS data from the Detected SDR devices list.

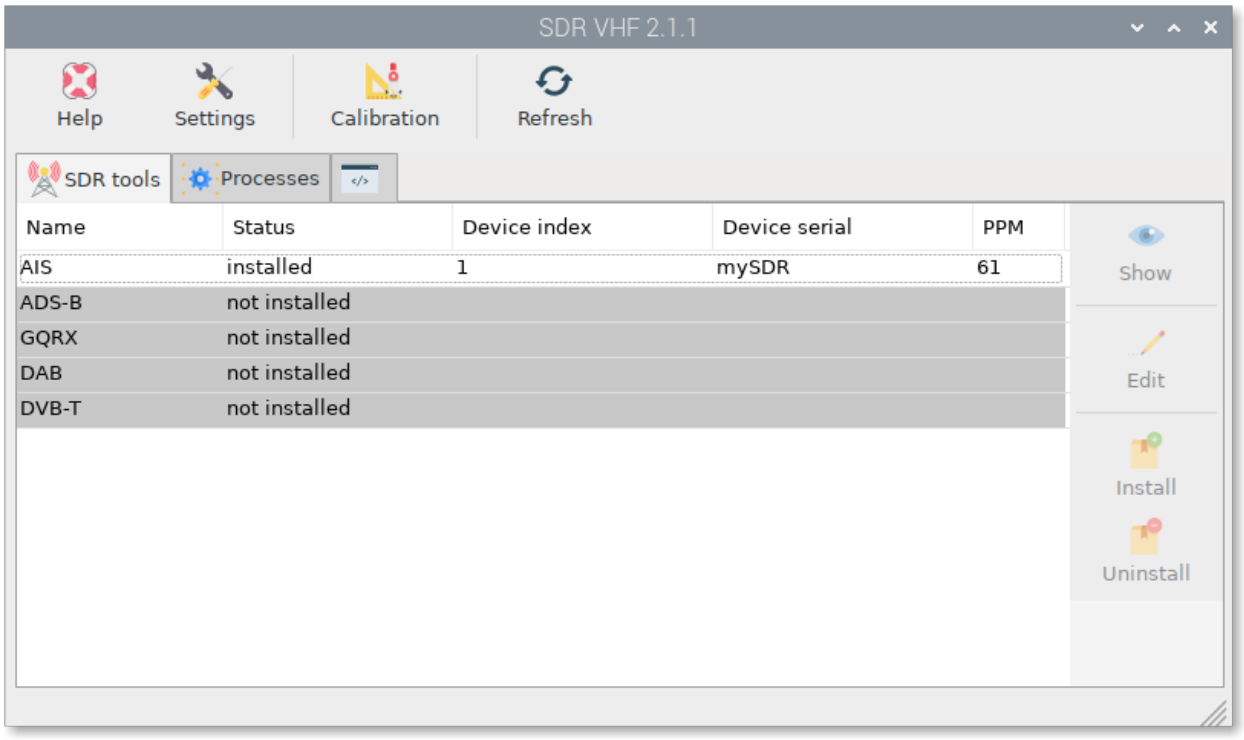
You can set the receive Gain. Not always the maximum gain will work better, we recommend leaving this field blank for auto.

Set the `PPM` value for your device. If you have calibrated your device, you should see this value in the device list. You can also set any value to test.

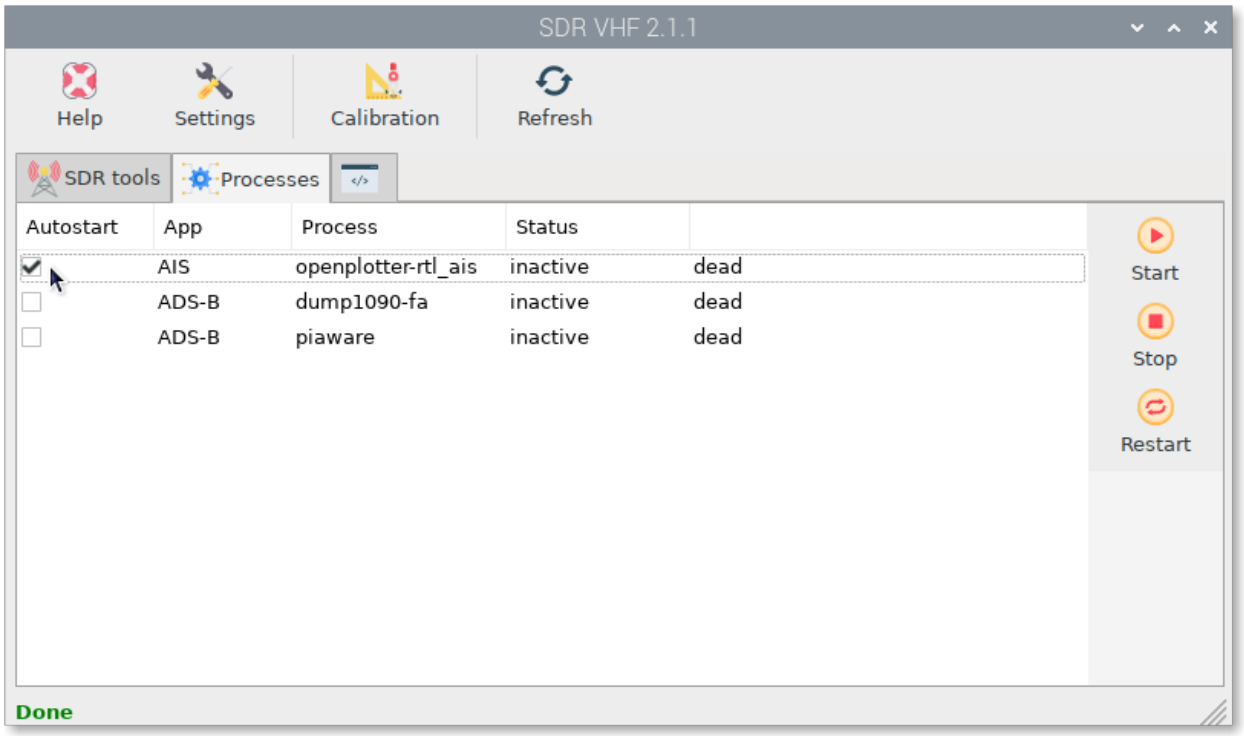
Provide a `Port` to send AIS data (default 10110). A UDP network connection will be created in Signal K automatically for that port.



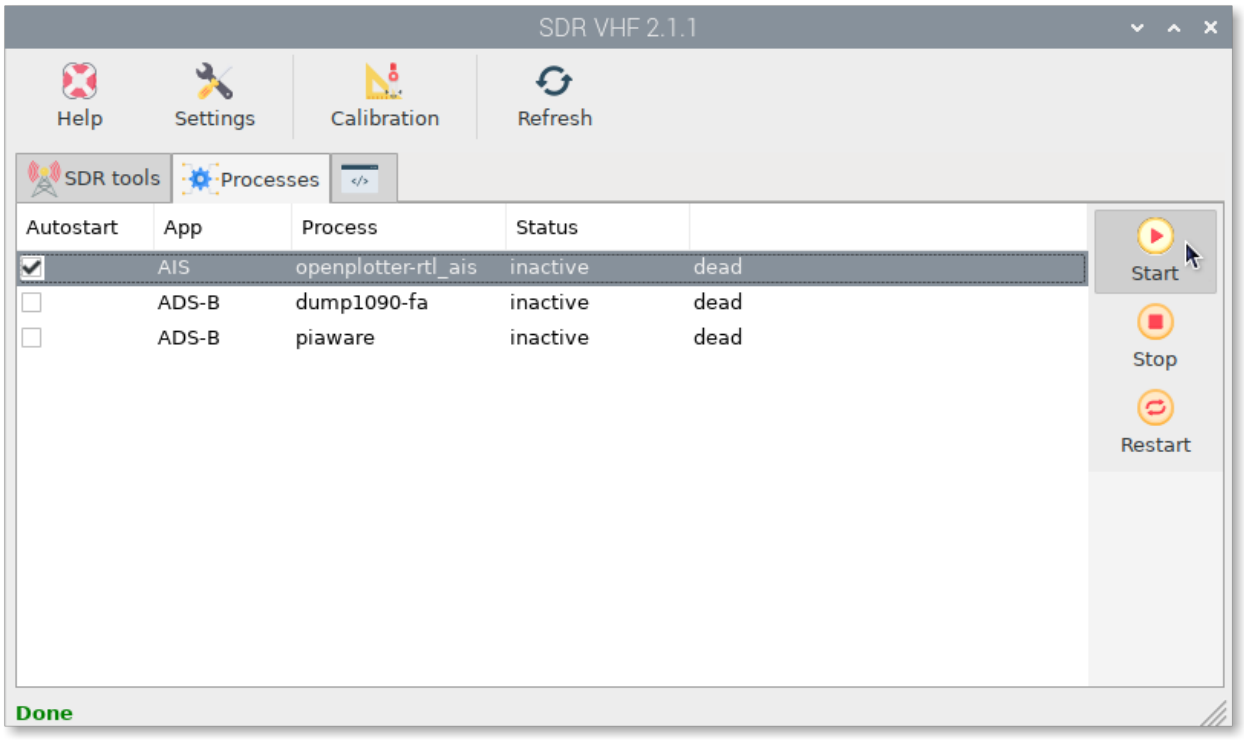
Click on `Save`.



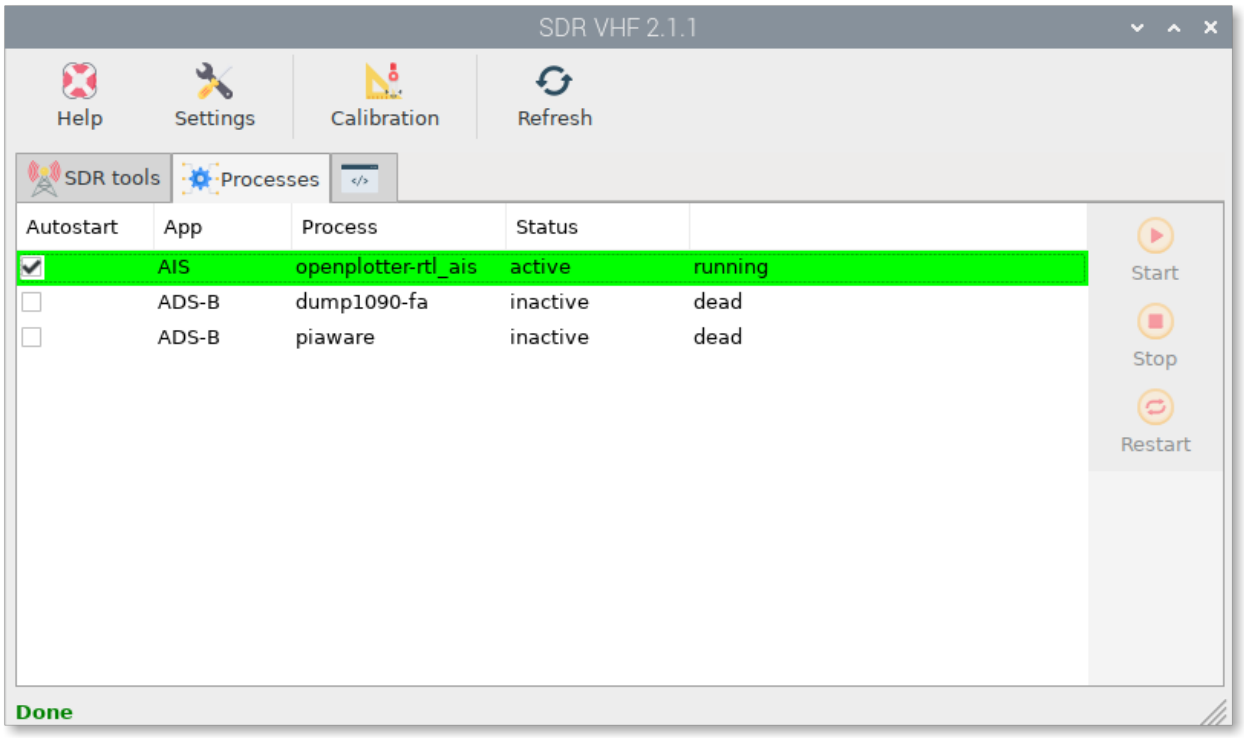
Go to the `Processes` tab and check `Autostart` on AIS process to start getting data at system startup:



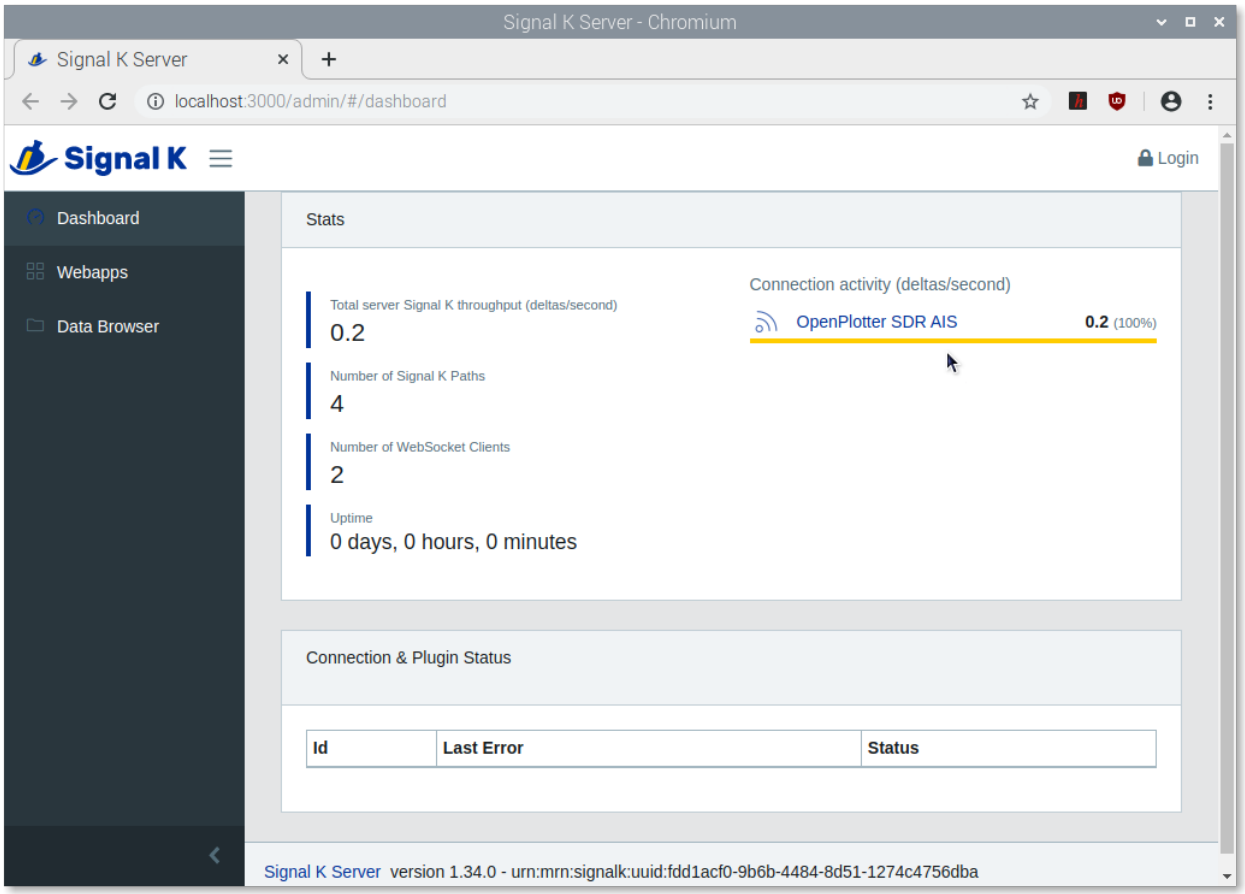
Finally, select the AIS process and click `Start` to start getting data:



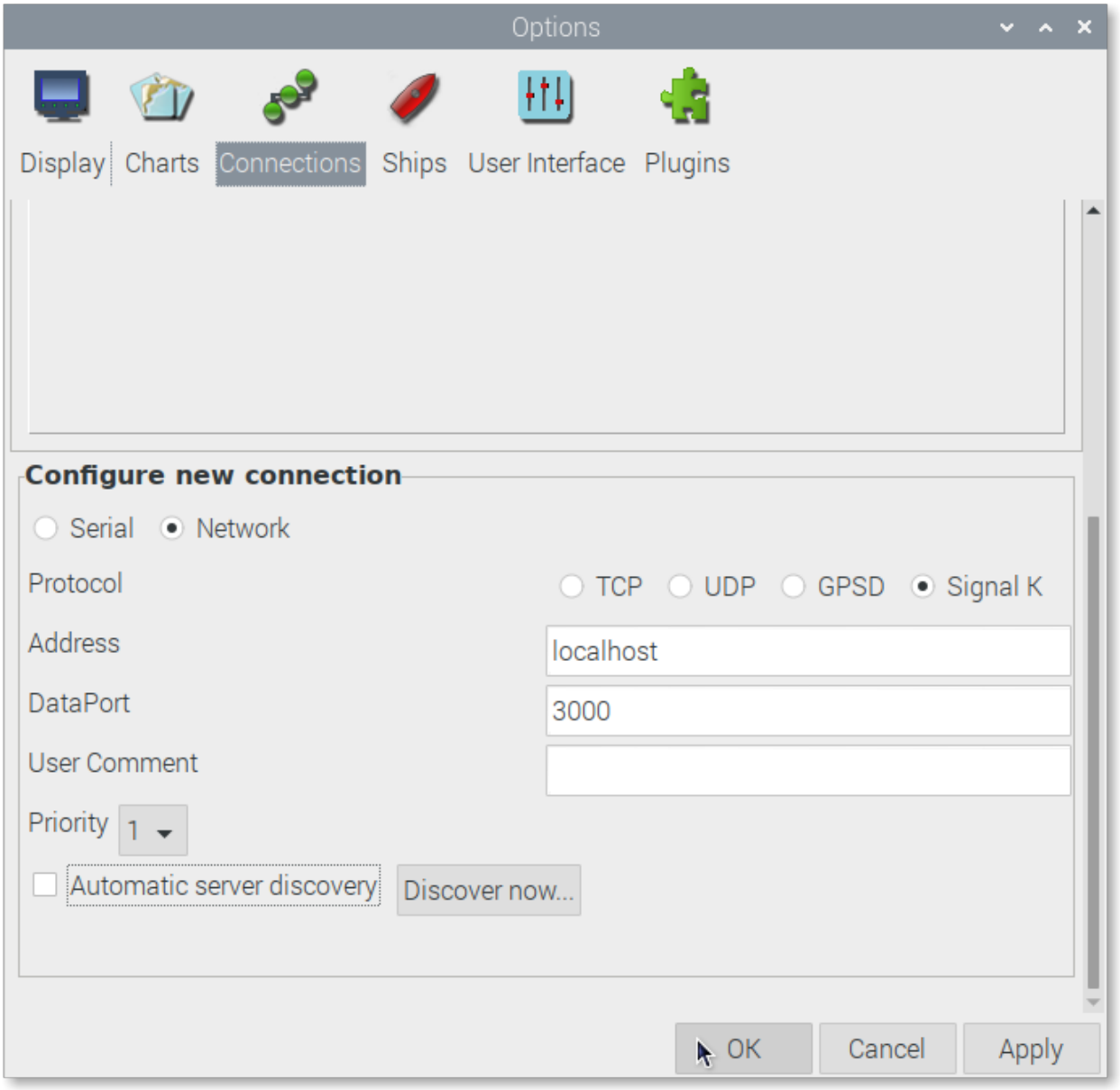
If you see the AIS process in green, you are done:

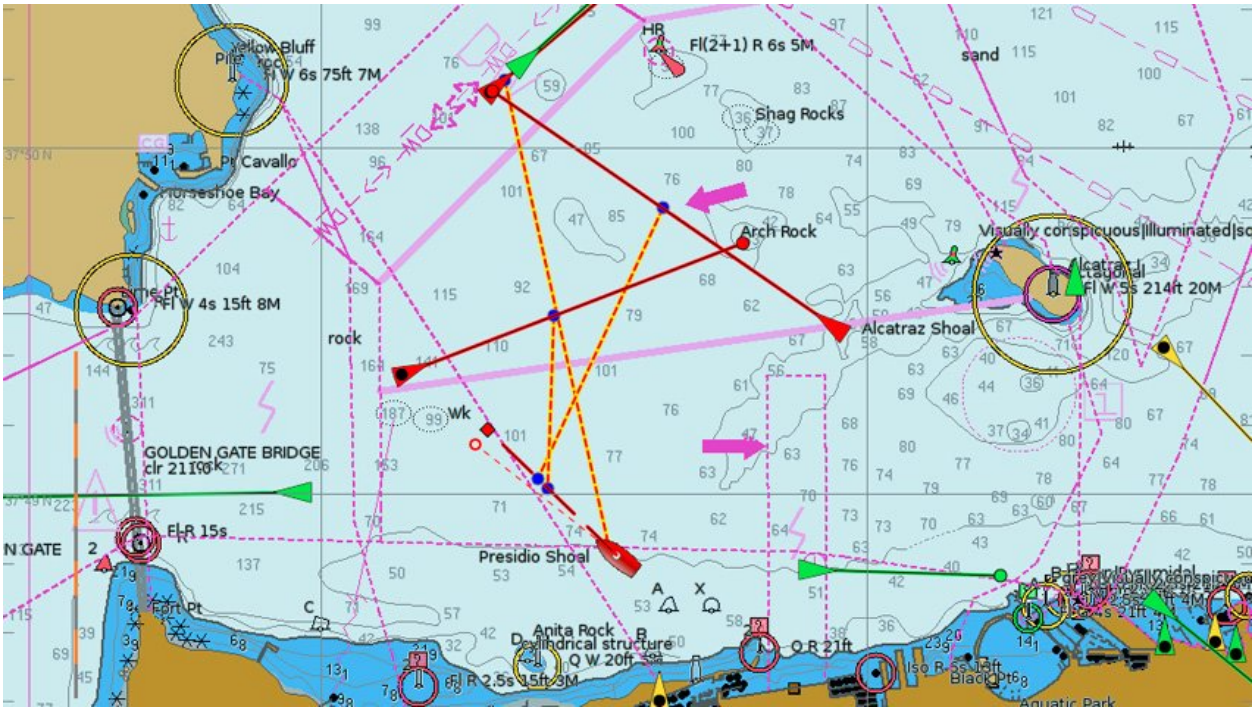


To confirm that everything is working fine, go to the Signal K server and check if an `OpenPlotter SDR AIS` connection has been created and is getting data:



Then, go to OpenCPN and confirm that a connection with the Signal K server exists and is getting AIS data:


Options
Display
Charts
Connections
Ships
User Interface
Plugins
Configure new connection
Serial
Network
Protocol
TCP
UDP
GPSD
Signal K
Address
localhost
DataPort
3000
User Comment
Priority
1
Automatic server discovery
Discover now...
OK
Cancel
Apply


Yellow Bluff
Fl W 6s 75ft 7M
Pt Cavallo
Horseshoe Bay
Fl W 4s 15ft 8M
rock
GOLDEN GATE BRIDGE
clr 211ft
Fl R 15s
Fort Pt
Presidio Shoal
Anita Rock
Cylindrical structure
Q W 20ft
Fl R 2.5s 15ft 3M
Q R 21ft
Iso R 5s 14ft
Wk
Arch Rock
Alcatraz Shoal
Fl(2+1) R 6s 5M
Shag Rocks
sand
Visually conspicuous illuminated sc
Alcatraz I
Fl W 5s 214ft 20M
Aquatic Park

CHAPTER 46

ADS-B

CHAPTER 47

GQRX

CHAPTER 48

DAB

CHAPTER 49

DVB-T

When a device is not being used the kernel recovers control of it and can be used to watch TV using DVB-T app. Remember to scan available channels.

CHAPTER 50

External Apps

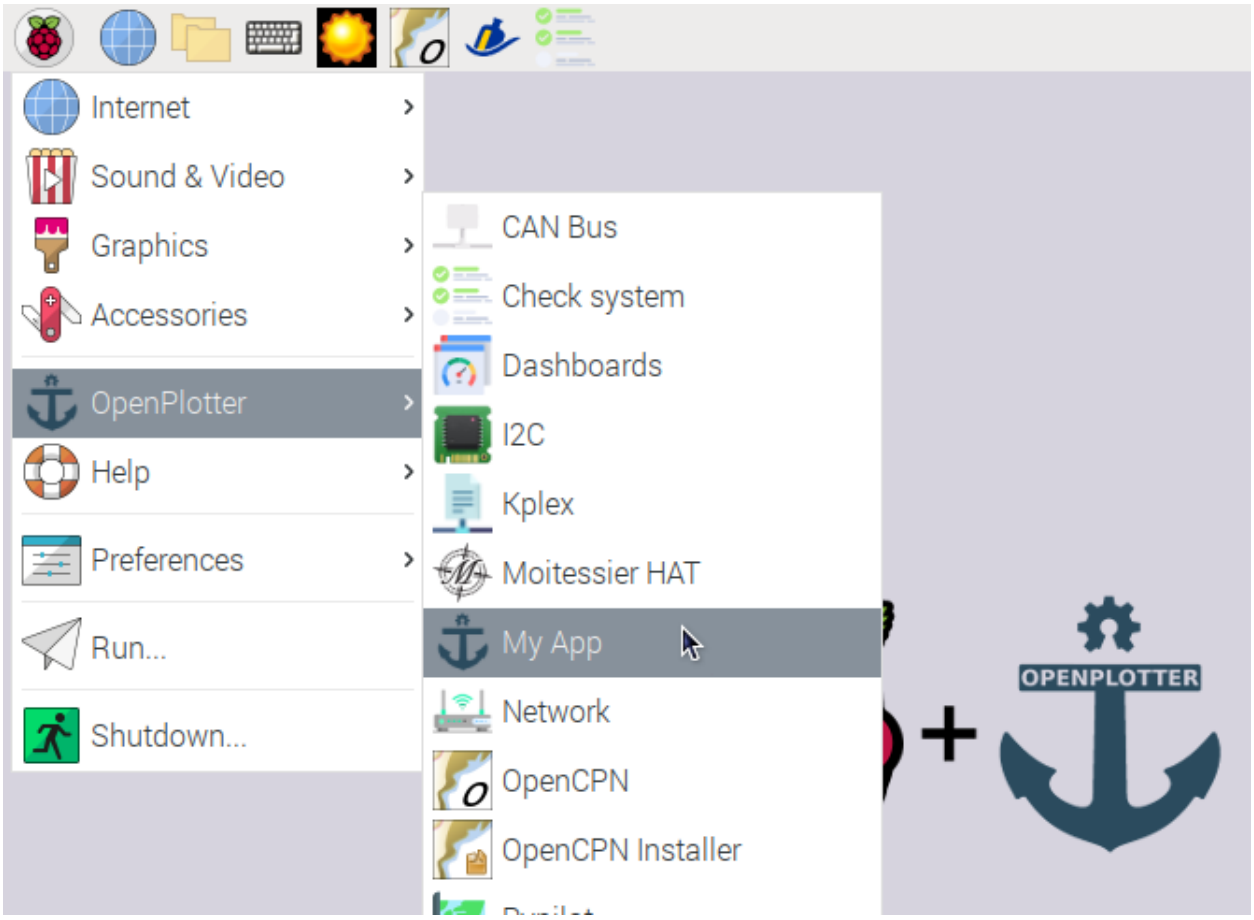
OpenPlotter contains a lot of useful apps but you can create your own apps too. External apps can share OpenPlotter resources but are autonomous and independent.

We have created an example of an external app called *My App* that can be used as a template for your app.

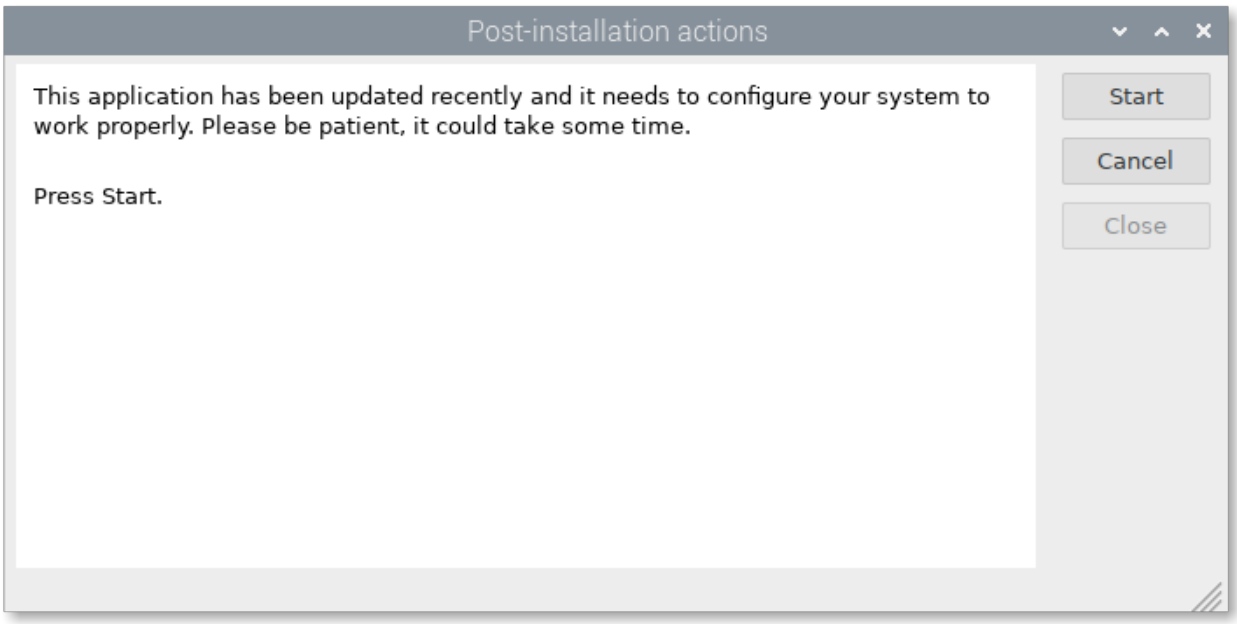
50.1 Installing *My App*

`Download` the latest version of *My App* and double-click the deb file.

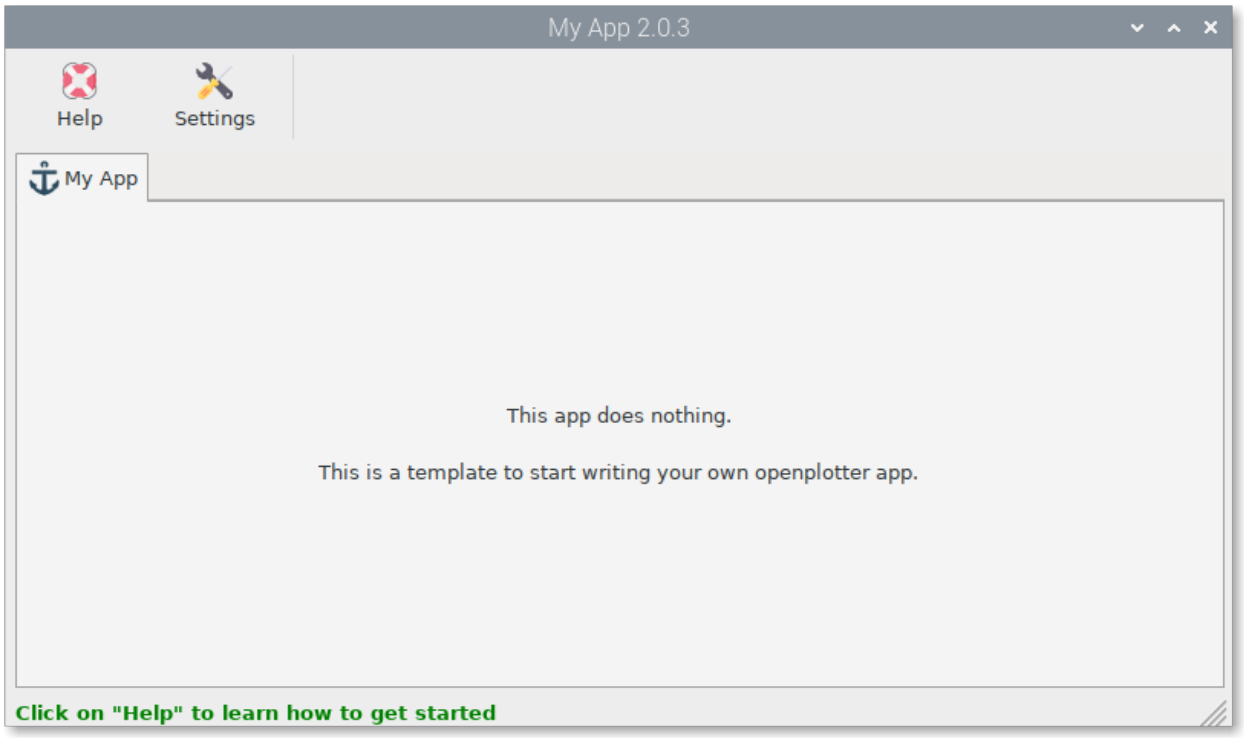
After installing go to *Menu* → *OpenPlotter* and click on `My App`.



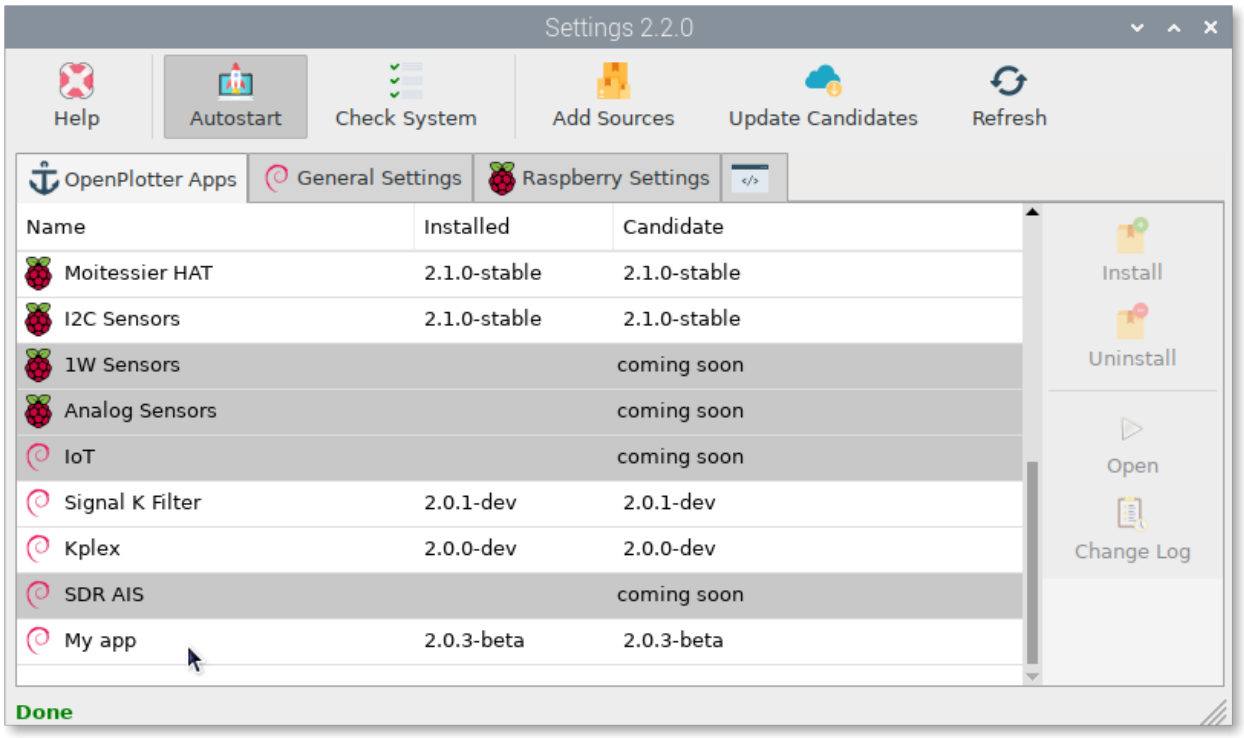
Most apps must perform some tasks for proper operation before being used. Click on `Start`.



After finishing the post installation tasks, try to open the app again.



My App is installed correctly and from this moment it will appear in the list of OpenPlotter apps that will be updated when a new version is published.



50.2 Creating your app from *My App* template

OpenPlotter apps should be written in python 3 and must be distributed as Debian packages.

First of all you need a repository to upload and distribute your app. You can use any Debian repository but we recommend https://cloudsmith.io because it is really easy for newbyes.

- Create an account in Cloudsmith.
- Create a repository in Cloudsmith and select "GNU Public General License v3.0" and "Open-Source". Your repository should look like this: https://cloudsmith.io/~openplotter/repos/openplotter-external/packages/
- To get the source of your new repository replace *myname* and *myrepository* by your data and type this in a browser:

```
https://dl.cloudsmith.io/public/myname/myrepository/cfg/setup/config.deb.txt?
↪distro=debian&codename=buster
```

- You will get a file that contains your source. Save it, you will need it later.
- Download the GPG key file you will find in the section *Signing Keys* of your repository. Save it, you will need it later.

Go to the *My App* GitHub page and click on `Use this template`. Provide a name for your new app and the project will be forked. It is recommended that you use the prefix *openplotter-* for your app name.

Browse the code of your fork of *My App*. There are comments on every piece of code you should change to customize your new app. Use the source and the GPG key files when required.

50.3 Packaging your app

After you have customized your app, you have to create a Debian package to test. To build deb packages you need to sign them with your GPG key. Follow this manual to get a GPG key using the same email you have in the file debian/changelog of your customized *My App*.

Once you have a GPG key installed in your computer you can create deb packages typing this in the root folder of your fork of *My App*:

```
dpkg-buildpackage -b
```

To increase the version of your app you have to edit the file *openplotterMyapp/version.py* and the file *debian/changelog*.

Attention: Versions in *openplotterMyapp/version.py* and *debian/changelog* must always match.

Tip: To help you edit the changelog file, type this in the root folder of your fork of *My App*:

```
dch
```

50.4 Uploading your app to your repository

Go to your Cloudsmith repository and select `Upload` for Debian. Then click on `Upload Debian Package`. In the next window click on `Upload File` to select the deb file from your computer and select the `Distribution`. OpenPlotter 2 works for *debian/buster - Debian - 10 (Buster)*.

50.5 Distributing your app

You are done. Send the URL of your repository to your users to download and install your deb file. Write a short manual to install your app just like the *first section of this page*.

Once installed your package can be updated from openplotter-settings app or typing:

```
sudo apt update
sudo apt install openplotter-myapp
```