OpenPlotter Documentation

Release 3

Sailoog

Mar 08, 2024

DESCRIPTION

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.

1.1 Releases

OpenPlotter 3

> **Code Name**
> > STOPmaremortum
>
> **Docs**
> > https://openplotter.readthedocs.io/en/3.x.x/

OpenPlotter 2

> **Code Name**
> > OpenArms
>
> **Docs**
> > https://openplotter.readthedocs.io/en/2.x.x/

1.2 Licenses

The code of all OpenPlotter apps is licensed under the GNU General Public License v3.0

This documentation is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License

Icons used in OpenPlotter apps and documentation are from flaticon.com

1.3 About us

You will see everywhere the nickname *Sailoog*. That is me, a guy from Barcelona, Catalonia, who started OpenPlotter as an exercise to learn English and Python in 2014. After a few years, OpenPlotter has become a very popular project and has helped me make many friends and a few enemies, so as long as this is still this fun, this project will last a long time ;)

My motivations are scientific and political, but OpenPlotter is free and that means that your motivations to use or collaborate with this project can be whatever you want, even the opposite of mine. OpenPlotter is not my main activity, I currently work as employee at o-charts that is part of the OpenCPN development team.

You will also see that I always use the plural when I speak about OpenPlotter. Although it seems that I am always that boring guy behind everything, many people collaborate on this project that always needs more hands and brains. I cannot mention all the people who have gone through this project because I would forget many of them, but I would like to make a special mention of e-sailing who contributed a lot of code.

OpenPlotter is made up of various pieces of great open source software developed by great teams like OpenCPN, Signal K, Pypilot ... We just sew it up with a bit of Python to turn it into a time saving tool for advanced users and a perfect introduction for novice Linux users. Enjoy it!

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. You can also run AvNav as server to have a chart plotter in any device remotely connected to it using a web browser.

Weather

Download and display GRIB files using XyGrib.

NMEA 0183

Connect your NMEA 0183 devices to receive and send data.

NMEA 2000

Connect your NMEA 2000 network to receive and send data.

Seatalk1

Connect your old Seatalk1 network to receive data.

Signal K

The free and open source universal marine data exchange format.

Access point

Share data with laptops, tablets, phones. . .

Headless

You can connect OpenPlotter to any HDMI monitor and/or access to OpenPlotter desktop from the cockpit through your mobile devices.

Dashboards

Customize your instrument panels to visualize data or create charts to see its evolution.

AIS

Build open source AIS receivers/transmitters.

Compass

Get magnetic heading, heel and trim using cheap Inertial Measurement Units (IMU).

Autopilot

Full pypilot integration.

Sensors

Easily connect all kinds of sensors (temperature, pressure, humidity, voltage, current, luminance, tank level, RPM, doors. . .)
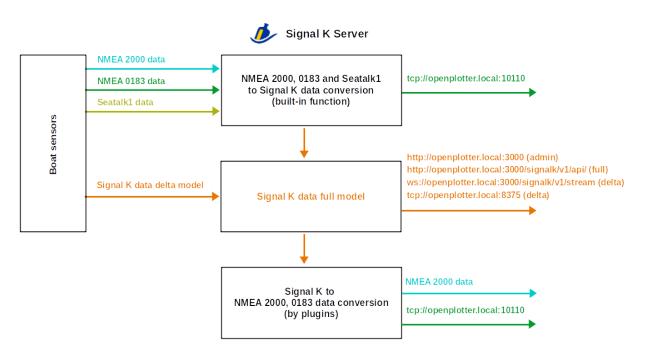
Notifications

Set thresholds for any parameter to trigger visual and sound notifications or trigger multiple custom actions.

IoB

Internet of Boats. Monitor what happens on your boat when you are not there or activate devices remotely.

HOW DOES IT WORK?

Signal K Server

| NMEA 2000 data |
| NMEA 0183 data |
| Seatalk1 data |

NMEA 2000, 0183 and Seatalk1 to Signal K data conversion (built-in function)

tcp://openplotter.local:10110

Boat sensors

Signal K data delta model

Signal K data full model

http://openplotter.local:3000 (admin)
http://openplotter.local:3000/signalk/v1/api/ (full)
ws://openplotter.local:3000/signalk/v1/stream (delta)
tcp://openplotter.local:8375 (delta)

Signal K to NMEA 2000, 0183 data conversion (by plugins)

NMEA 2000 data

tcp://openplotter.local:10110

OpenPlotter v3 Data Routing

The center of OpenPlotter is the **Signal K Server**. All the data collected by the boat's sensors in any of the supported formats are converted to Signal K format and stored. Once the server has the data in Signal K format, these can be sent by different ways to any program that supports this open source universal data exchange format or they can be converted again to any of the supported formats.

Note: The main function of the different apps that make up OpenPlotter is to facilitate the connection of the sensors with the Signal K server and in some cases to directly generate data in Signal K format from the raw sensor data. There is also another group of apps dedicated to processing and displaying the data from the Signal K server. You will learn how to use these apps in successive chapters.

Depending on the initial format of the data, it can follow different routes or be available in different ways:

3.1 Signal K

We encourage companies and developers to use the Signal K format for new sensors and devices. OpenPlotter can obtain Signal K data from sensors using USB, GPIO, UDP, TCP and Websockets connections.

There are two Signal K data models, *delta* and *full*. The delta format is used to exchange data between devices and/or servers and the full format is used to store data on servers. Read the Signal K documentation for details on both models.

When the Signal K server receives delta messages from sensors, it forwards them immediately through a Websocket and a TCP connection at:

```
ws://openplotter.local:3000/signalk/v1/stream
tcp://openplotter.local:8375
```

It also stores data using the full model. This data can be queried using the HTTP REST API at:

```
http://openplotter.local:3000/signalk/v1/api/
```

You can manage the Signal K server through a web application at:

```
http://openplotter.local:3000
```


Note: You can learn more about Signal K here: https://signalk.org/specification

3.2 NMEA 0183

You can get NMEA 0183 data from USB, GPIO, TCP and UDP connections. The Signal K server will automatically convert the data to Signal K format and store it. NMEA 0183 data will also optionally be forwarded to:

```
tcp://openplotter.local:10110
```

> **Caution:** If the same application gets NMEA 0183 data over the TCP port 10110 and also from any of the Signal K data outputs (HTTP, TCP, or WS) at the same time, it will probably get the same duplicate data in different formats.

If you have data in Signal K format that has not been converted from NMEA 0183, you can convert it to NMEA 0183 and send it through the TCP port 10110 using the **signalk-to-nmea0183** plugin.

> **Note:** To better understand the management of NMEA 0183 data in the Signal K server, please refer to the chapter *NMEA 0183 multiplexing*.

3.3 NMEA 2000

You can get NMEA 2000 data from USB, GPIO, TCP and UDP connections. The Signal K server will automatically convert the data to Signal K format and store it.

If you have data in Signal K format that has not been converted from NMEA 2000, you can convert it to NMEA 2000 and send it through the same CAN bus adapter using the **signalk-to-nmea2000** plugin.

3.4 Seatalk1

You can get data in the old Seatalk1 format from a GPIO. The Signal K server will automatically convert the data to Signal K format and store it. There is currently no way to convert Signal K data to Seatalk1 or send data in Seatalk1 format from a Signal K server.

EXAMPLES

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

4.1 Sigma 33 Build
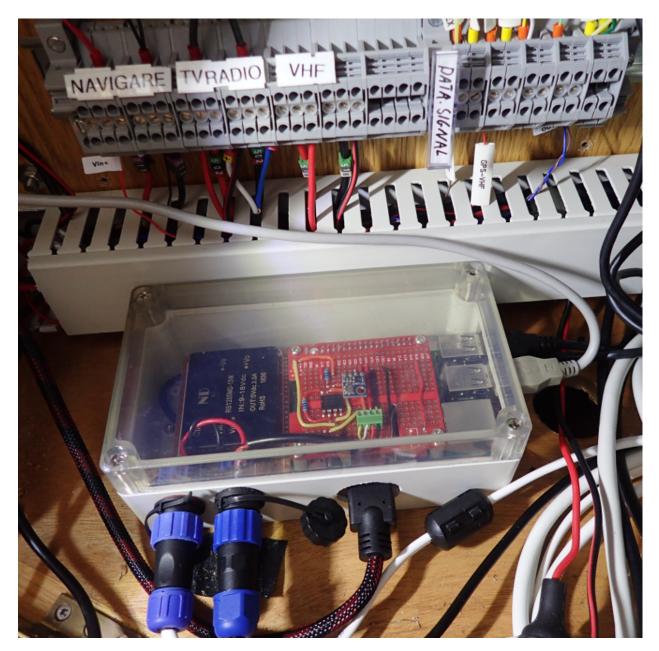
Description

> 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

4.2 Yacht server on board
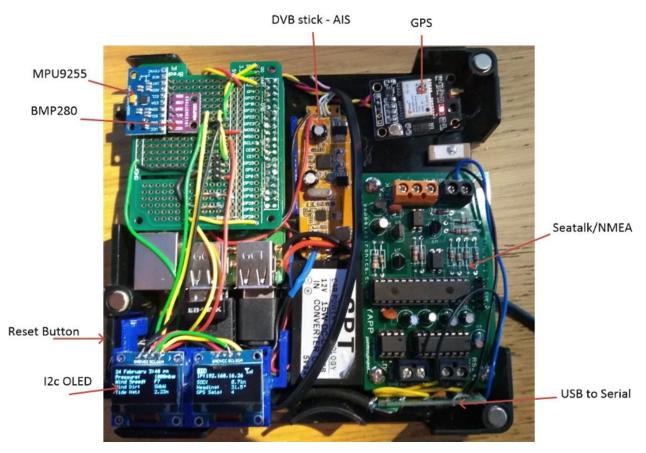
Description

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

4.3 UK OpenPlotter Build
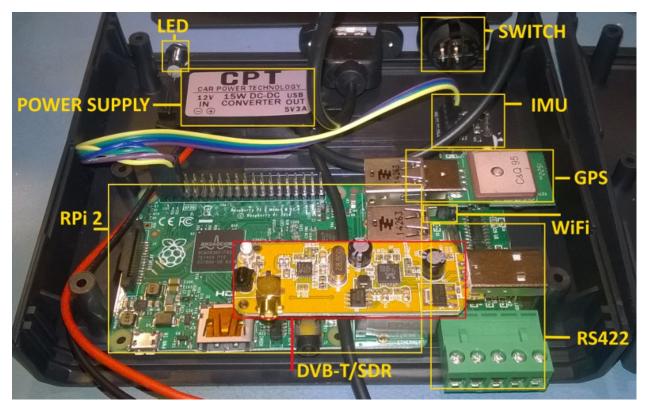
Description

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

4.4 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/

4.5 *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

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.

5.1 Translations

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

5.2 Documentation

We will no longer maintain translations, just the source in English, we tried but failed. This project is too dynamic and even maintaining this documentation is a hard job. Translators coordination is not an option either.

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 need 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. 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 a lot of grammar mistakes, please help us to solve it too.

WHAT DO YOU NEED?

Raspberry Pi or desktop/laptop computer?

OpenPlotter is optimized to be used on Raspberry Pi computers, but you can also install OpenPlotter on any desktop or laptop computer running Linux Debian or any derivative like Ubuntu, Mint … Some OpenPlotter apps that are used to manage some sensors connected via GPIO will not be available when installed on desktop and laptop computers. See a list *here*.

In the *downloading* and *installing* chapters you will learn how to get the software, let's see here what hardware we need.

6.1 Basic hardware

If you want to take full advantage of all the capabilities of OpenPlotter, your choice should be to install it on a Raspberry Pi.

Although a Raspberry Pi model 3 can run OpenPlotter, we only recommend the Raspberry Pi model 4 in any of its RAM configurations or a Raspberry Pi 400 unit. Raspberry Pi Zero, 1 or 2 models are not suitable to run OpenPlotter.

You will also need a keyboard, a mouse, a power supply, a microSD card and a monitor. Read this helpful guide for details on each item: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

6.2 Recommended hardware

This is the hardware that we have verified to work perfectly with OpenPlotter. We have been directly involved in the design or manufacture of some of these pieces of hardware.

6.2.1 Monitors
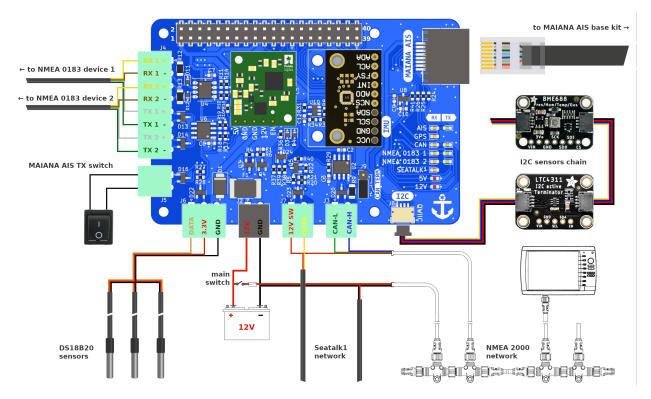
OpenPlotter is designed to be used on monitors with a minimum resolution of 800x480 and 7" in size.

Any HDMI-connected monitor that can handle this resolution, or any higher, will work just fine, but brightness on monitors that are connected via the DSI display port can also be controlled via software.

If the monitor is touchscreen, we recommend the data input via I2C to simplify installations. We do not recommend touchscreens that require drivers.

The monitor that meets all these qualities is the official monitor for Raspberry Pi and its clones. We particularly recommend this clone. We have a special edition of OpenPlotter for these touchscreens. These monitors typically have

a brightness of 250/300 cd/ and that means they need to be installed indoors. For outdoor installations, you need a sun-readable monitor with at least 1000 cd/ and waterproof.

6.2.2 MacArthur HAT

After a few years of developing software for OpenPlotter, we have identified exactly what we need in terms of hardware to achieve our goals and the result is the **MacArthur HAT** (Hardware Attached on Top), an add-on board for Raspberry Pi 4 running OpenPlotter v3. With this HAT we want to get the fully open-source boat to free ourselves from dependence on big companies and make our boats more respectful with the environment.

Its main function is to be able to communicate with any old or new marine electronic device using the proprietary and closed protocols **Seatalk1**, **NMEA 0183** or **NMEA 2000** and the free and open protocol **Signal K**.

> **More info**
> MacArthur HAT

6.2.3 AIS receivers/transponders

You can connect the AIS receiver/transponder you already have on the boat or you can choose from some open source solutions. We recommend the **dAISy HAT** (receiver) and the **MAIANA AIS transponder** (receiver/transponder).

> **More info**
>> *Connecting the dAISy HAT*

> **More info**
>> *MAIANA AIS transponder*

6.2.4 Autopilot

Pypilot is an open source autopilot for your boat and is fully integrated into OpenPlotter.

> **More info**
>> *Pypilot*

6.3 Alternative hardware

6.3.1 USB GPS receiver

This is the cheapest way to get position, speed and course data. Most devices on the market will work in OpenPlotter without the need for drivers.

> **Example**
>> *Connecting a USB GPS receiver*

6.3.2 RS422 converters

NMEA 0183 communication protocol was designed to run over the RS422 serial interface, which can support a single talker and up to 10 listeners and data rates as high as 10 mbit/sec. RS422 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.

> **Example**
>> *Connecting a USB RS422 converter*

6.3.3 CAN converters

NMEA 2000 communication protocol was designed to run over a Controller Area Network (CAN bus). This is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer. You can find USB converters or some Raspberry Pi HATs or modules to connect to the GPIO header.
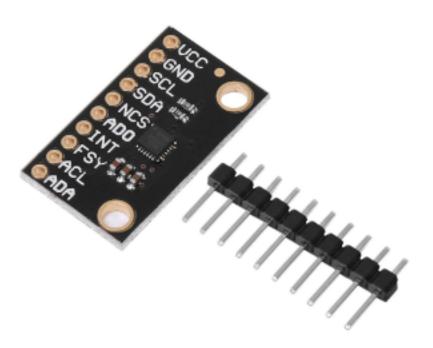
> **Example**
>> *Connecting a USB CAN converter*

> **Example**
>> *Connecting a GPIO CAN converter*

6.3.4 IMU

Adding an Inertial Measurement Unit (IMU) you can get heading, heel and trim data. OpenPlotter currently supports the following I2C-connected IMU models:

- ICM-20948 (recommended)

- MPU-9250

- MPU-9255.

> **Example**
>> *Configure an IMU with OpenPlotter Pypilot app*

> **Example**
>> *Calibrating the compass*

DOWNLOADING

Raspberry Pi or desktop/laptop computer?

OpenPlotter is optimized to be used on Raspberry Pi computers, but you can also install OpenPlotter on any desktop or laptop computer running Linux Debian or any derivative like Ubuntu, Mint . . . Some OpenPlotter apps that are used to manage some sensors connected via GPIO will not be available when installed on desktop and laptop computers. See a list *here*.

7.1 Raspberry Pi images

This is the easiest and fastest way to get started with OpenPlotter. We publish different editions according to the most demanded uses that contain all the required apps installed and preconfigured. We try to provide solutions for everyone from beginners to experts. We even have a fully customizable option that will save you a lot of time. Just plug and sail!

Our OpenPlotter editions are based on Raspberry Pi OS. You do not need any prior knowledge of Linux to install and use them. Follow the *Installing* chapter to learn how.

Important: With OpenPlotter 3 you can now choose between 32bit or 64bit.

7.1.1 OpenPlotter Starting

All required apps to fulfill most OpenPlotter marine features.

> **Download**
>   OpenPlotter Starting
>
> **Image name**
>   OpenPlotter Starting
>
> **Hostname**
>   openplotter
>
> **User**
>   pi
>
> **Password**
>   raspberry
>
> **Language**
>   en_GB.UTF-8

Keymap
   gb

Layout
   English (UK)

TimeZone
   Europe/London

Wifi client
   SSID: none, Password: none, Country: none

Wifi AP
   SSID: none, Password: none, IP: none

SSH
   Disabled

Remote desktop
   Disabled

Touchscreen
   Disabled

Backlight
   Disabled

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

7.1.2 OpenPlotter Headless

Same as *OpenPlotter Starting* but ready to be used remotely without monitor.

Download
   OpenPlotter Headless

Image name
   OpenPlotter Headless

Hostname
   openplotter

User
   pi

Password
   raspberry

Language
   en_GB.UTF-8

Keymap
   gb

Layout
   English (UK)

TimeZone
   Europe/London

Wifi client
    SSID: none, Password: none, Country: none

Wifi AP
    SSID: openplotter, Password: 12345678, IP: 10.10.10.1

SSH
    Enabled

Remote desktop
    Enabled

Touchscreen
    Disabled

Backlight
    Disabled

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

7.1.3 OpenPlotter Touchscreen

Same as *OpenPlotter Starting* but ready to be used on DSI touchscreens as the official monitor for Raspberry Pi and its clones.

Download
    `OpenPlotter Touchscreen`

Image name
    OpenPlotter Touchscreen

Hostname
    openplotter

User
    pi

Password
    raspberry

Language
    en_GB.UTF-8

Keymap
    gb

Layout
    English (UK)

TimeZone
    Europe/London

Wifi client
    SSID: none, Password: none, Country: none

Wifi AP
    SSID: none, Password: none, IP: none

SSH
    Disabled

> **Remote desktop**
>> Disabled
>
> **Touchscreen**
>> Enabled
>
> **Backlight**
>> Enabled
>
> **Installed apps**
>> Settings - Docs - Signal K installer - OpenCPN installer - Xygrib - Dashboards - Serial - CAN - Network

7.1.4 OpenPlotter À la Carte

Fill in a form with all the available customization options and in a few minutes you will receive an image built by a robot from scratch and to your liking that will save you a lot of time. Another advantage over the other editions is that all packages that make up the OS, including Openplotter apps, will be updated to the latest versions.

> **Download**
>> Under construction
>
> **Image name**
>> Customizable
>
> **Hostname**
>> Customizable
>
> **User**
>> Customizable
>
> **Password**
>> Customizable
>
> **Language**
>> Customizable
>
> **Keymap**
>> Customizable
>
> **Layout**
>> Customizable
>
> **TimeZone**
>> Customizable
>
> **Wifi client**
>> SSID: Customizable, Password: Customizable, Country: Customizable
>
> **Wifi AP**
>> SSID: Customizable, Password: Customizable, IP: Customizable
>
> **SSH**
>> Customizable
>
> **Remote desktop**
>> Customizable
>
> **Touchscreen**
>> Customizable

Backlight
>   Customizable

Installed apps
>   Customizable

7.2 Desktop and laptop

> **Danger:** Each new version of OpenPlotter should only be installed on the indicated system. **OpenPlotter v3 will work only on Debian 11 Bullseye, Ubuntu 20.04 Focal, Ubuntu 22.04 Jammy or any of their derivatives**. If you try to force an installation of OpenPlotter v3 over OpenPlotter v2 (based on Debian 10 Buster), your system will become unstable.

You can also install OpenPlotter in any desktop or laptop computer running your favourite Debian derivative distribution. Hovever, if your computer is not a Raspberry Pi, you will not be able to install some OpenPlotter apps:

Common
>   Settings - Docs - Signal K installer - OpenCPN installer - AvNav installer - Xygrib - Serial - CAN - Notifications - Dashboards - IoB - MAIANA AIS Transponder - SDR VHF

Only Raspberry
>   Network - I2C - Pypilot - GPIO

You just need basic knowledge of Linux to install OpenPlotter for desktop and laptop. Download this *OpenPlotter Settings* package: and follow the *Desktop and laptop* chapter to install OpenPlotter from scratch.

7.3 OpenPlotter 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 *openplotter32* and *openplotter64* branchs 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.

INSTALLING

8.1 Raspberry Pi images

You only need a micro SD card (minimum 8GB, recommended 16GB) and a computer with an SD card reader.

- Download your preferred OpenPlotter edition from the *Raspberry Pi images* section in *Downloading* chapter.

- Download and install the `Raspberry Pi Imager` program for your OS.

- Put the SD card you will use with your Raspberry Pi into the SD card reader and run *Raspberry Pi Imager*.

- Click on `CHOOSE OS` and then on `Use custom`:

- Select the file of your OpenPlotter edition (you do not need to unzip it).

- Click on `CHOOSE STORAGE` and select your SD card.

- Click on `WRITE` and take a coffe.
- Remove the SD card from the reader, insert it into the Raspberry Pi and you are done.

Unless you are using the **OpenPlotter À la Carte** edition, after the first boot you should customize and localize your system changing some important settings like the password or the system language. You can do this in *Main → Preferences → Raspberry Pi configuration*.

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

8.1.1 Headless

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 using the *OpenPlotter Network* app. Otherwise, anyone will be able to access your system easily.

> **Note:** Using **OpenPlotter À la Carte** edition you will be able to configure all these parameters and more by filling in a form before downloading your custom OpenPlotter.

8.1.2 Backups

Once you are satisfied with your final setup, it is highly recommended to make a backup to replace the SD card in case it gets damaged. This is especially recommended if we have installed a paid nautical chart because if we have to reinstall the system, we will surely lose the license.

Raspberry Pi OS incorporates an excellent application to make a backup of the entire SD card called *SD Card copier* that you can find in *Main → Accessories*

You will need to connect an external USB card reader with a new SD card the same size or larger than the internal one. In *Copy From Device* field you should select the internal card (something like */dev/mmcblk0*) and in *Copy To Device* field you should select the external card. The name of the external reader may vary, connect and disconnect the external reader several times to be sure which name appears and disappears.

Important: DO NOT check the option *New Partition UUIDs*. Some programs require the original and the copy to be exactly the same in order to function properly.

8.2 Desktop and laptop

Danger: Each new version of OpenPlotter should only be installed on the indicated system. **OpenPlotter v3 will work only on Debian 11 Bullseye, Ubuntu 20.04 Focal, Ubuntu 22.04 Jammy or any of their derivatives**. If you try to force an installation of OpenPlotter v3 over OpenPlotter v2 (based on Debian 10 Buster), your system will become unstable.

First of all you have to install some dependencies. Open a terminal and type:

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

Now you have to install the *OpenPlotter Settings* app from the .deb file you will found in *Desktop and laptop* section in *Downloading* chapter

After downloading the .deb file, you can install it by double click or typing this in a terminal replacing *x.x.x-stable* by your version:

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

And that's all. Open *OpenPlotter Settings* app typing this in a terminal:

```
openplotter-settings
```

You have to install the rest of OpenPlotter apps from this app. Read the *Settings* chapter to learn how.

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:
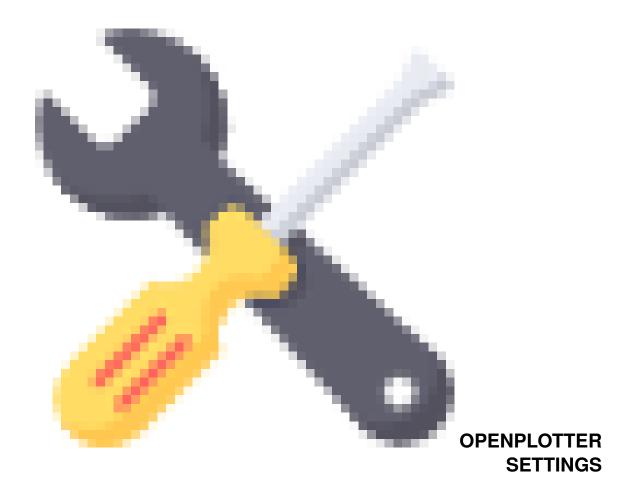
```
sudo visudo
```

Add this line to the end of the document replacing *myuser* by your user name and save:

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

NINE

OPENPLOTTER
SETTINGS

Note: To run this app type this in a terminal:
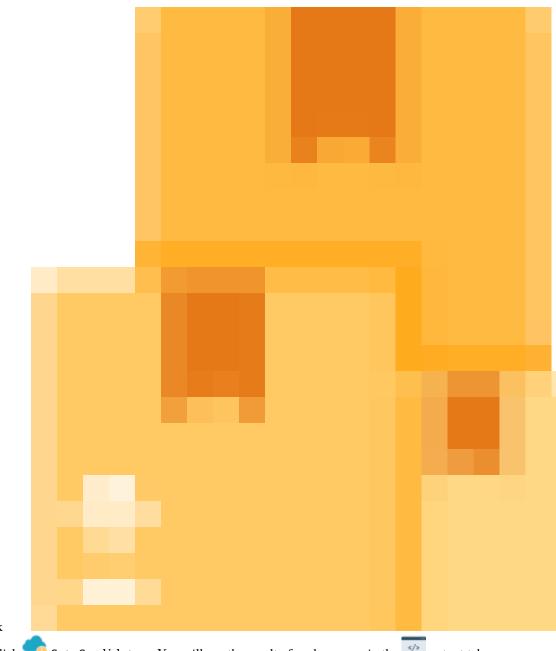
```
openplotter-settings
```

This is the main OpenPlotter app. You need it to install the rest of the apps. If you are using any of the OpenPlotter images for Raspberry Pi, the sources for the repositories where the OpenPlotter apps live will be installed and working, but if you are installing OpenPlotter from scratch you will see something like this:
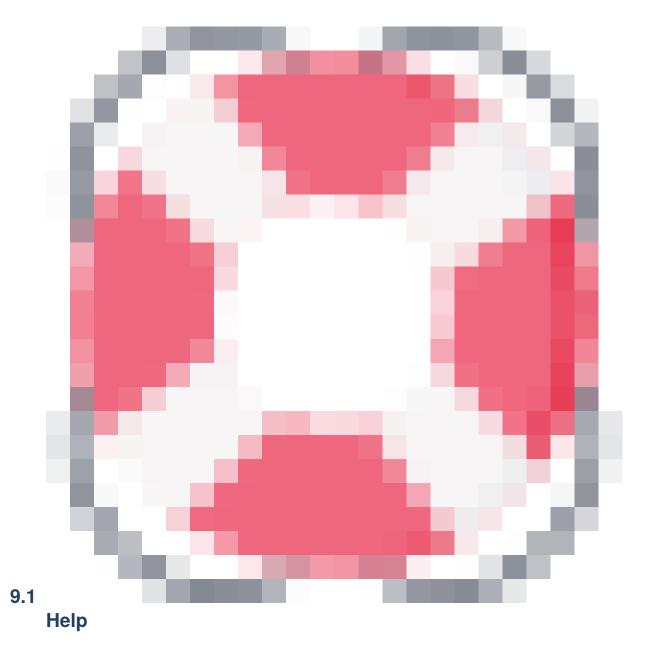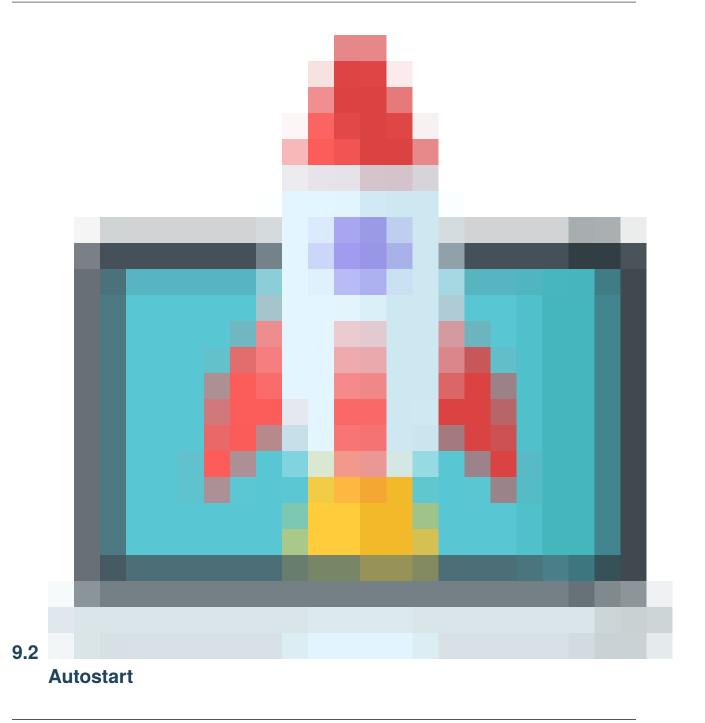
To install the sources, click

`Add sources` and then click `Get Candidates`. You will see the result of each process in the output tab. After adding the sources you are ready to install the rest of the apps:

9.1
Help

This button will be disabled until you install the *Documentation* app. After installation, the button will be enabled and clicking on it will open an offline copy of this documentation in a browser.

9.2

Autostart
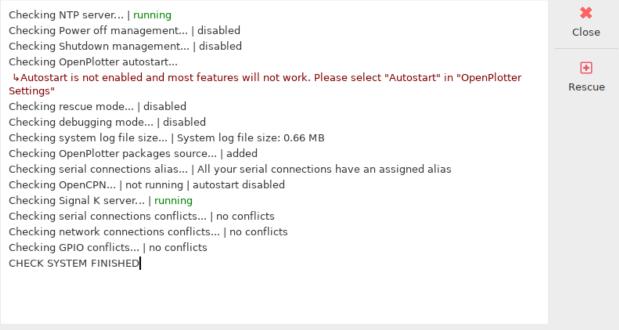
Important: This button should always be checked.

Allows self-diagnosis at startup of all OpenPlotter apps and some important settings. It will also trigger some programs and tools configured to run at startup.
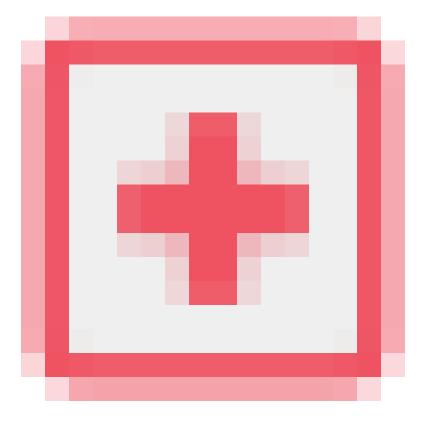
9.3 ⚏ Check System

By clicking this button, you can run the self-diagnosis at any time. Each OpenPlotter app installed will add new processes to diagnose its operation. If something does not work as expected, a red message will show the problem and the solution:

```
Checking NTP server... | running
Checking Power off management... | disabled
Checking Shutdown management... | disabled
Checking OpenPlotter autostart...
 ↳Autostart is not enabled and most features will not work. Please select "Autostart" in "OpenPlotter
Settings"
Checking rescue mode... | disabled
Checking debugging mode... | disabled
Checking system log file size... | System log file size: 0.66 MB
Checking OpenPlotter packages source... | added
Checking serial connections alias... | All your serial connections have an assigned alias
Checking OpenCPN... | not running | autostart disabled
Checking Signal K server... | running
Checking serial connections conflicts... | no conflicts
Checking network connections conflicts... | no conflicts
Checking GPIO conflicts... | no conflicts
CHECK SYSTEM FINISHED
```

✖ Close

⊞ Rescue

There are some warnings. Check your system. Closing in 58 seconds

This process runs automatically at startup and it is also the time when some important internal OpenPlotter processes are started. OpenPlotter is highly configurable and some parameters can produce unexpected effects such as data loops, unstable or even unusable systems. By clicking

`Rescue`
you can prevent these processes from starting in order to modify the settings and recover the system.

⚓ **OPENPLOTTER APPS**

OpenPlotter apps versions consist of 3 digits separated by periods (a.b.c), a code name and a development 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.
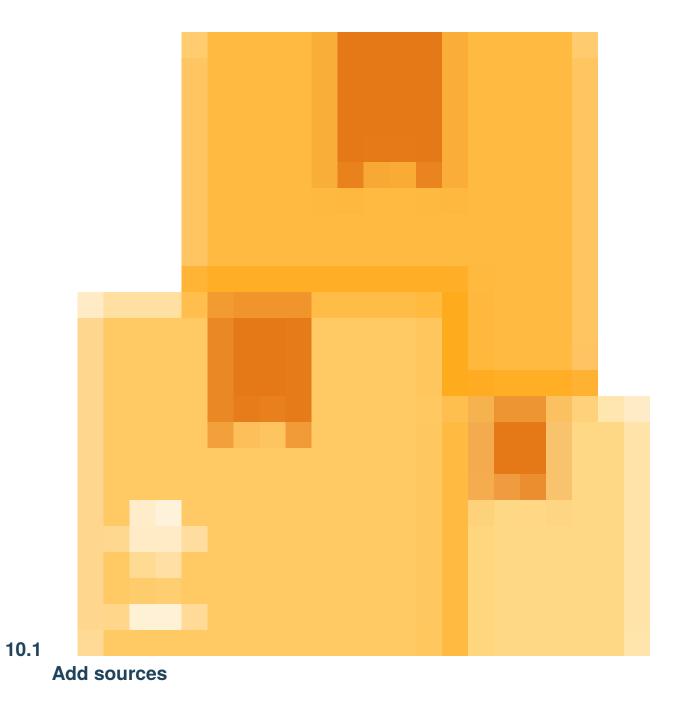
codeName
Name to identify the OpenPlotter version (a).

state
Alpha: new features need to be added; **Beta**: all features have been added, but need to be tested; **Stable**: all functions should work properly.

10.1

Add sources

The sources of the OpenPlotter apps repositories should be added once at the beginning and then only if you see the *missing source* message in any of the OpenPlotter apps.
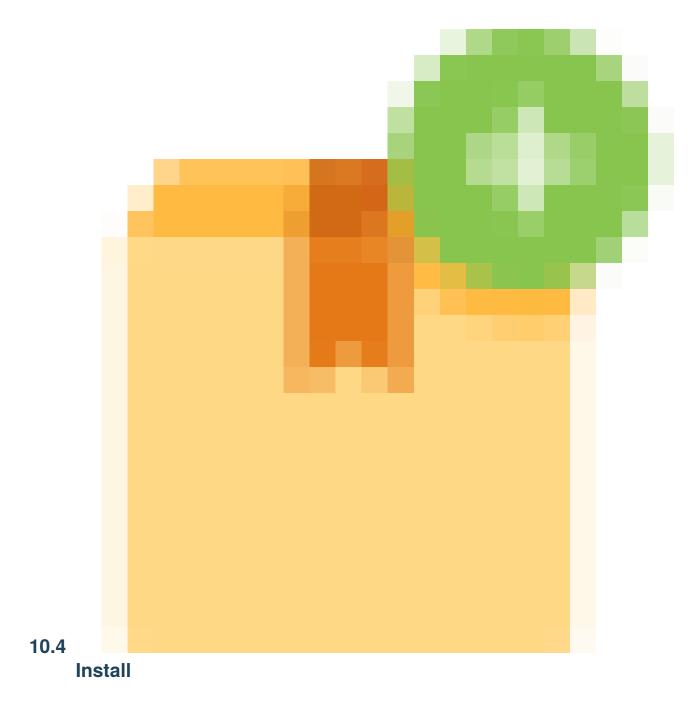
10.2 ☁ Get Candidates

Occasionally, you should check if there are new versions of OpenPlotter apps in the remote repositories to enjoy new features and correct errors. If there is a new version of an installed app, it will be shown in green:

10.3 ↻ Refresh

Run this option when you want to know the status of your apps. This option only compares versions locally and checks for pending tasks. To compare versions in the remote repositories click ☁ `Get Candidates`.

10.4 Install

To install a new version of any app select the item in the list and click this button. If the updated app was running while the installation, you will have to close and open it again to see changes.
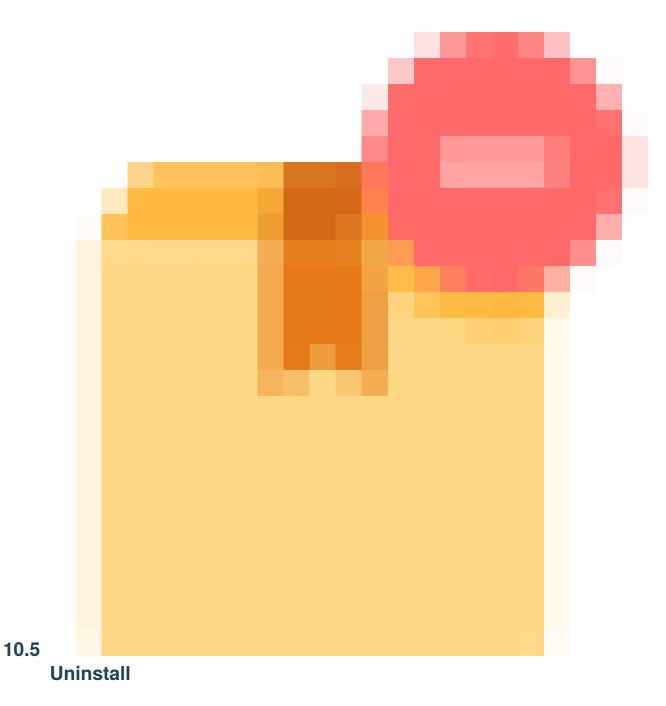
Note: It is highly recommended to install and update the OpenPlotter apps from *OpenPlotter Settings* because often the installation process involves extra changes to the system that will be carried out automatically. If you install the apps manually from a terminal or they are updated automatically due to a general system update, they will be marked in pink to indicate that there are pending actions:

These extra actions will be executed the first time you open the app:
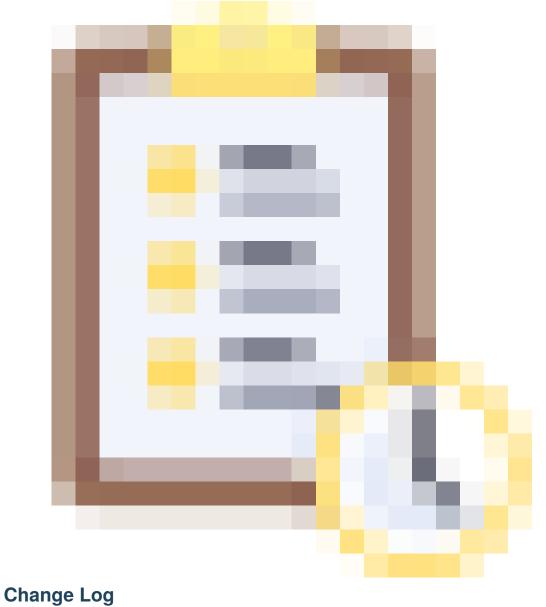
Post-installation actions

This application has been updated recently and it needs to configure your system to work properly. Please be patient, it could take some time.

Press Start.

Start

Cancel

Close

10.5 Uninstall

To uninstall any app select the item in the list and click this button. It is also important to uninstall OpenPlotter apps from *OpenPlotter Settings* to undo the changes in the system.

10.6 ▷ Open

If you have trouble locating and opening the OpenPlotter apps installed on your system, you can open them from here.

10.7
Change Log

Select any app and click this button to see all changes across versions.
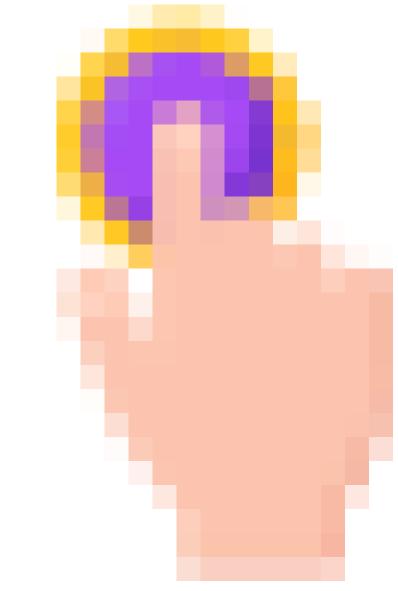
$\copyright$ **GENERAL SETTINGS**

11.1 Language

You can select any of the available languages and all OpenPlotter apps will be translated the next time you open them.

You can help us with translations by clicking Translate.
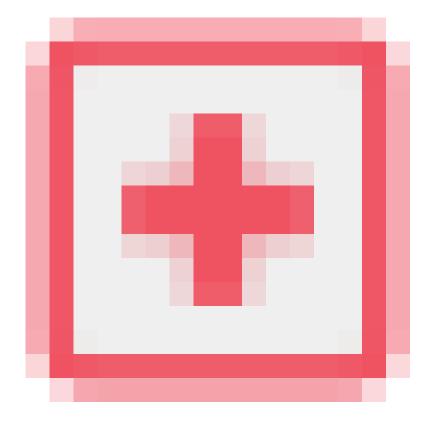
11.2

Touchscreen

After enabling this setting, most programs will display monster scrolls better for sailors' fingers and some programs like OpenCPN will be optimized for touchscreens.

11.3 ⬛ Maximize

If you enable this option, all OpenPlotter applications will be maximized the next time you open them.

11.4
Rescue
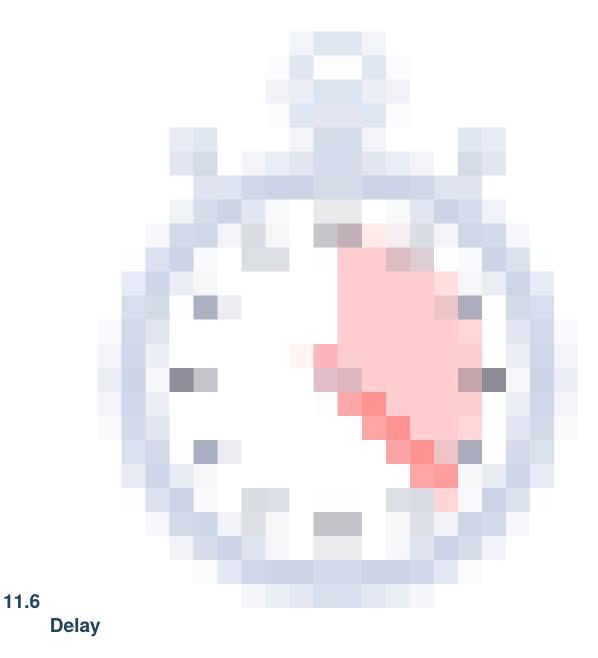
Here you can enable/disable the *Rescue* mode to help recover unstable systems due to some misconfigurations. For example if you set an action to reboot the system using the *OpenPlotter Notifications* app in response to some value of Signal K, it can cause a reboot loop. Using the *Rescue* mode disables all actions temporarily and you will be able to remove that killer setting.

11.5 ⌨ Virtual keyboard
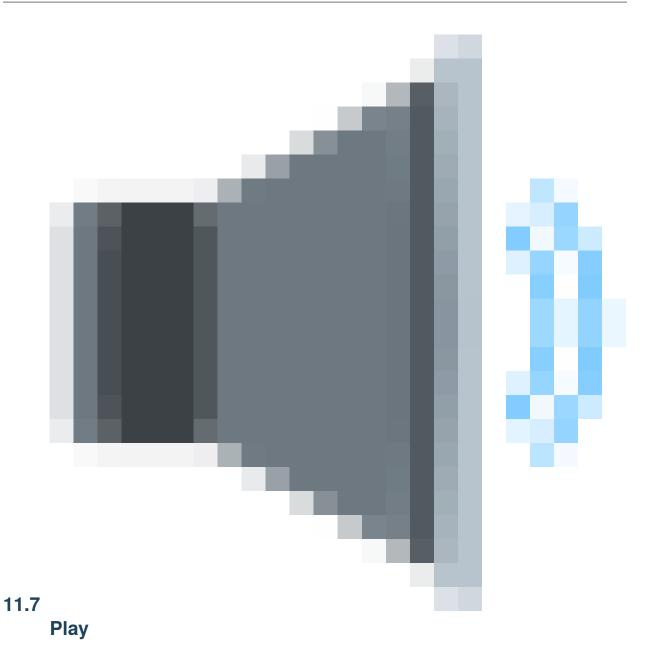
OpenPlotter includes a virtual keyboard for touch systems or systems that do not have a physical keyboard. You can customize your virtual keyboard by creating a layout that suits your requirements and your language.
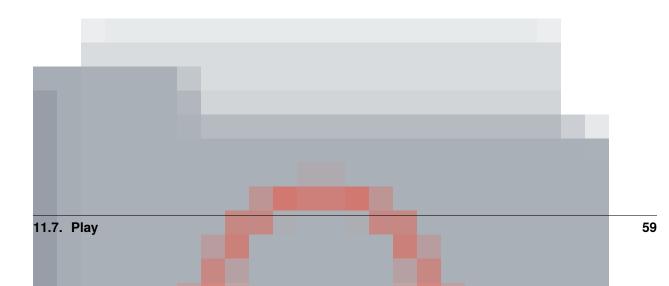
Note: Visit this link. to know how to create and share a keyboard layout.

11.6

Delay
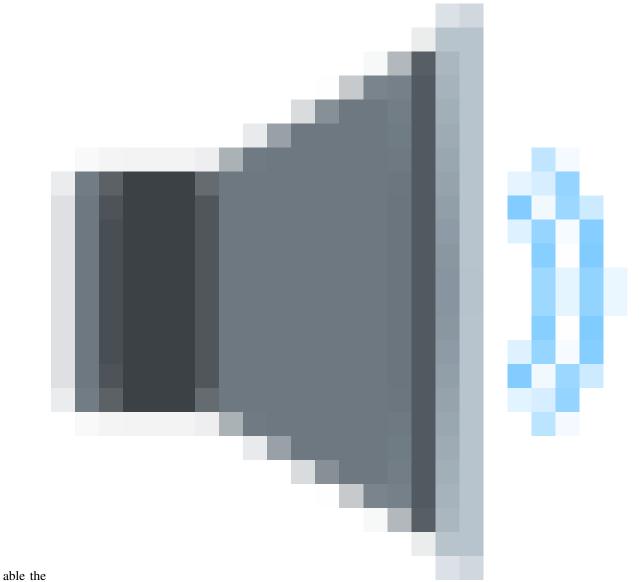
Sometimes you may need some devices or programs to be ready before they work normally. Adding seconds to this field and enabling this button will add a delay to the OpenPlotter startup process to allow time for these lazy devices or programs.

11.7 Play

You can play a sound to notify you when the OpenPlotter startup process is complete. Select a sound by clicking
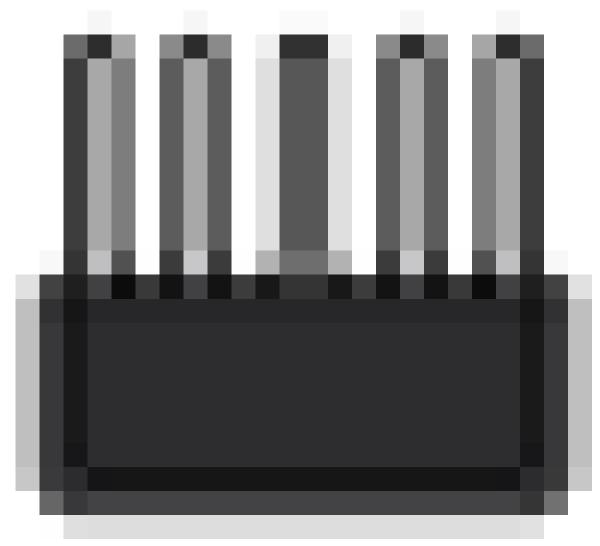
able the
`Play` button. This is especially useful for headless systems.

🍓 RASPBERRY SETTINGS
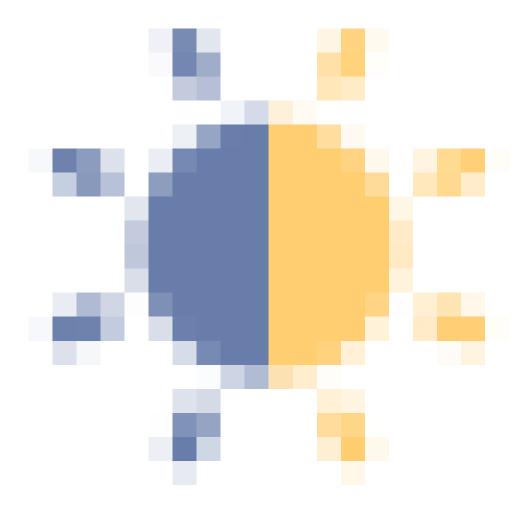
12.1
GPIO Map
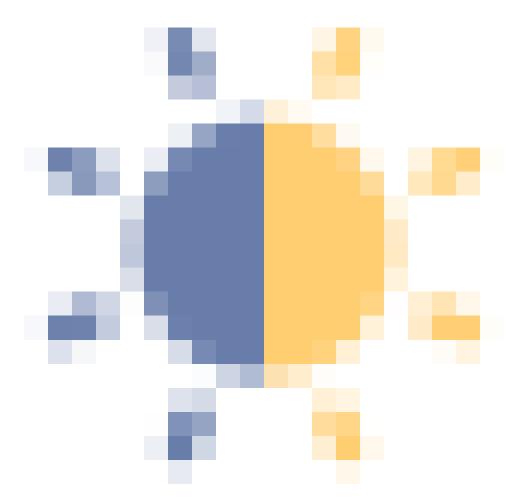
Some apps will report which GPIO they are using and you can check it here. Checking a GPIO will return useful information about its usage.

GPIO Map   ∨   ∧   ✕

```
      (Power) 3v3 - 1  ☐  ☐ 2 - 5v (Power)
      (I2C) GPIO 2 - 3  ☐  ☐ 4 - 5v (Power)
      (I2C) GPIO 3 - 5  ☐  ☐ 6 - Ground (Power)
           GPIO 4 - 7  ☐  ☑ 8 - GPIO 14 (UART)
    (Power) Ground - 9  ☐  ☐ 10 - GPIO 15 (UART)
         GPIO 17 - 11  ☐  ☐ 12 - GPIO 18
         GPIO 27 - 13  ☐  ☐ 14 - Ground (Power)
         GPIO 22 - 15  ☐  ☐ 16 - GPIO 23
      (Power) 3v3 - 17  ☐  ☐ 18 - GPIO 24
   (SPI) GPIO 10 - 19  ☐  ☐ 20 - Ground (Power)
    (SPI) GPIO 9 - 21  ☐  ☐ 22 - GPIO 25
   (SPI) GPIO 11 - 23  ☐  ☐ 24 - GPIO 8 (SPI)
   (Power) Ground - 25  ☐  ☐ 26 - GPIO 7 (SPI)
 (EEPROM) GPIO 0 - 27  ☐  ☐ 28 - GPIO 1 (EEPROM)
           GPIO 5 - 29  ☐  ☐ 30 - Ground (Power)
           GPIO 6 - 31  ☐  ☐ 32 - GPIO 12
          GPIO 13 - 33  ☐  ☐ 34 - Ground (Power)
          GPIO 19 - 35  ☐  ☐ 36 - GPIO 16
          GPIO 26 - 37  ☐  ☐ 38 - GPIO 20
   (Power) Ground - 39  ☐  ☐ 40 - GPIO 21
```

Host: localhost
Physical pin: 8
BCM name: GPIO 14
Interface: UART
Shared: No

Refresh
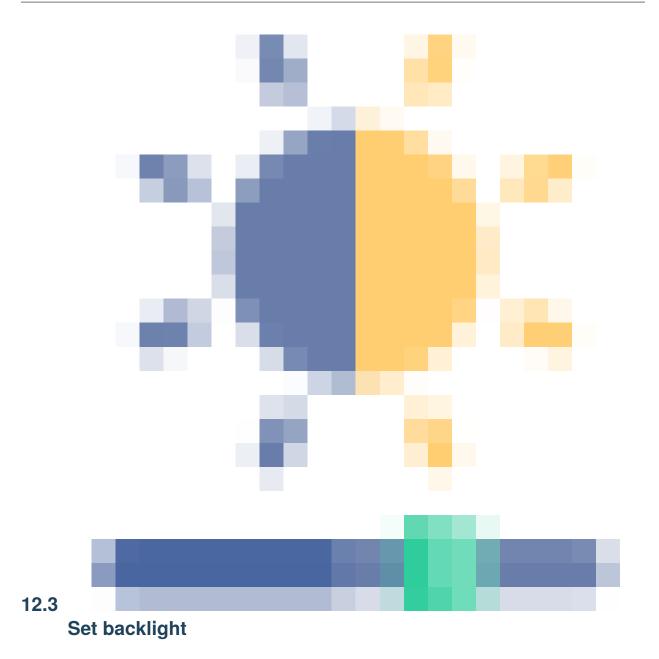
12.2

Install backlight
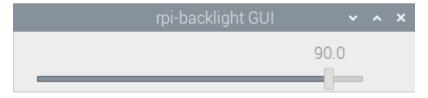
Brightness on monitors that are connected via the DSI display port can be controlled by software. If you have a DSI touchscreens as the official monitor for Raspberry Pi or any of its clones, you can install the necessary software to control brightness by clicking

Install `backlight` button.

12.3 Set backlight

After installing the required software, you will have access to a graphical interface to set the brightness using a slider.
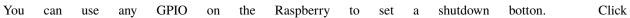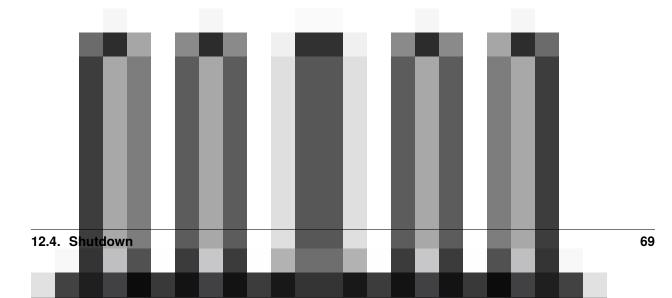
If you have the ⟨image⟩ *OpenPlotter Notifications* app installed, you will see a new actions added to the list to automatically set the backlight value upon receiving a specific notification:
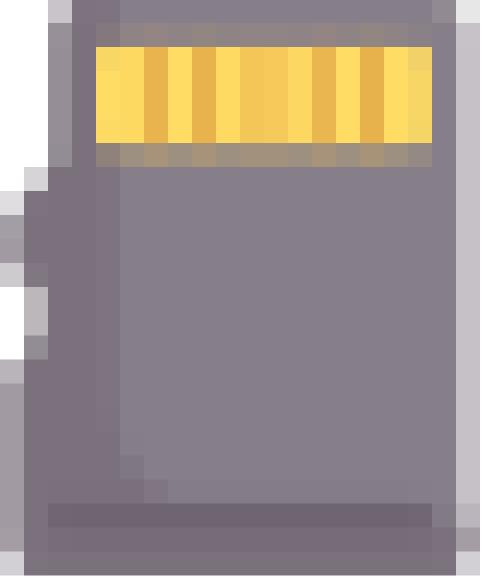
12.4 Shutdown

You   can   use   any   GPIO   on   the   Raspberry   to   set   a   shutdown   botton.   Click

to choose a GPIO, usually GPIO 21 at pin 40. Select a GPIO `Transition` to trigger the shutdown, *high->low* or *low->high*. Select an internal pull resistor, *pull-up* and *pull-down*, or *off* if you use an external pull resistor. Click `Apply` to save settings and changes will be applied after the next reboot.
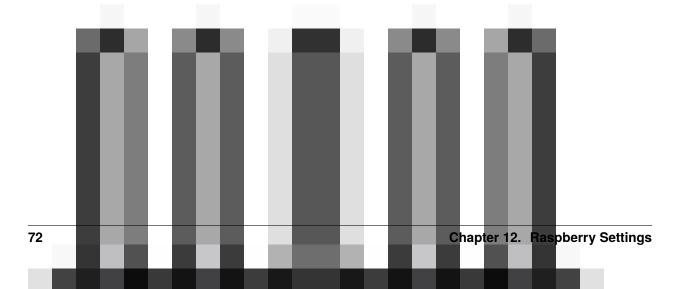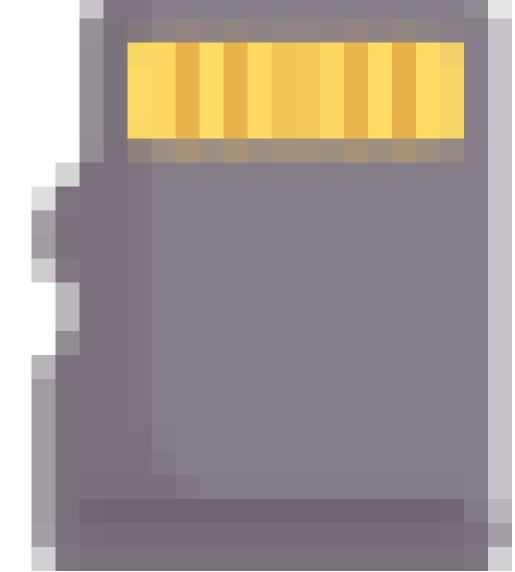
12.5

Power off
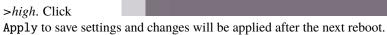
You can use any GPIO on the Raspberry to notify an external circuit that it can safely cut power.   Click
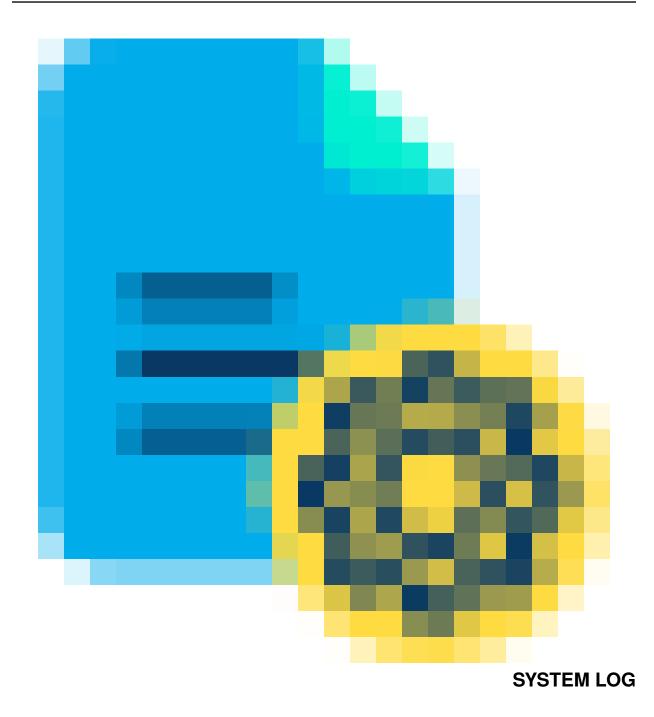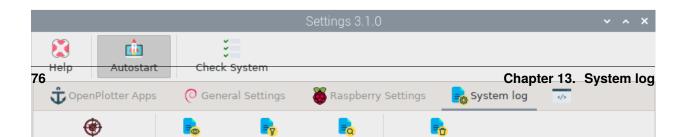
choose a GPIO, usually GPIO 26 at pin 37. Select a GPIO `Transition` to trigger the power off, *high->low* or *low-*

>high. Click
`Apply` to save settings and changes will be applied after the next reboot.

CHAPTER

THIRTEEN

SYSTEM LOG
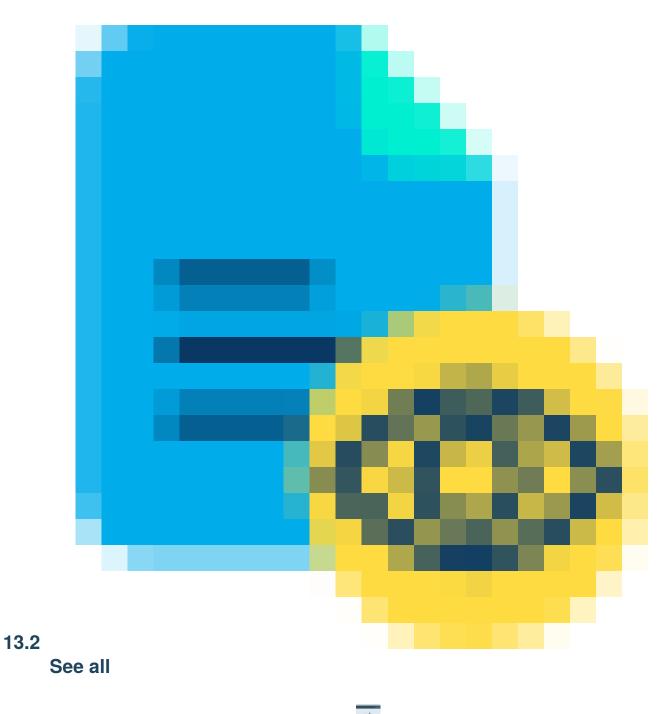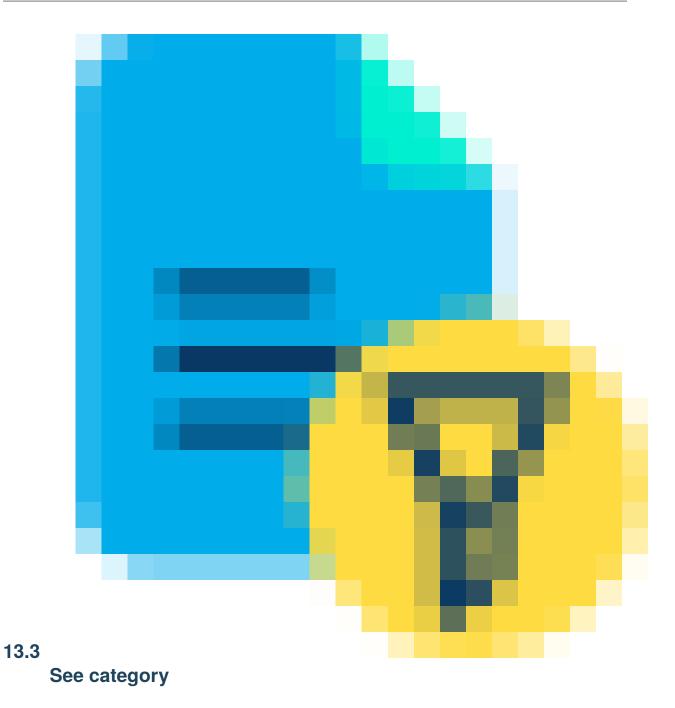
Here are some tools for debugging and troubleshooting your system. When an application malfunctions, it can generate excess messages and eventually create huge log files that consume all free storage space. By setting a maximum log file size, you can detect and resolve potential problems.

13.1 Debugging
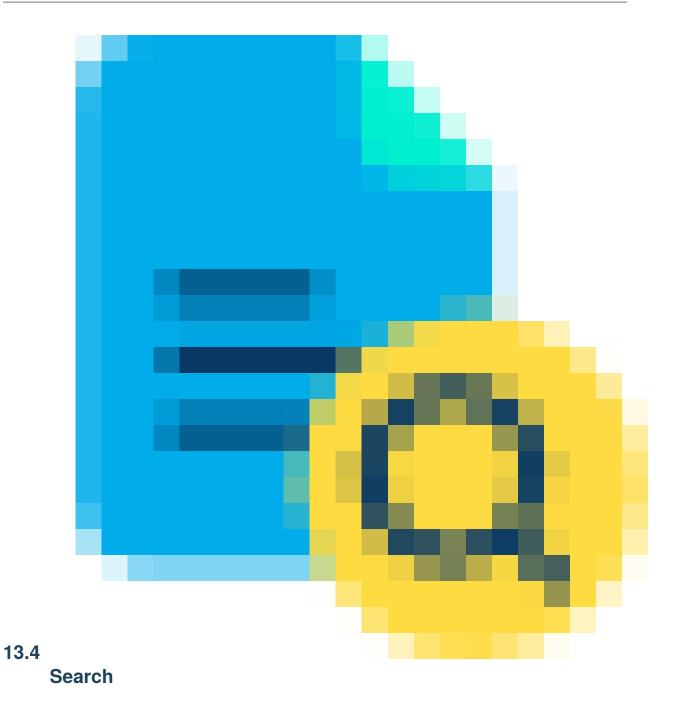
When this button is enabled, OpenPlotter applications will generate error messages that will be saved to the log file or printed on the screen when the application is run from a terminal. Use this feature only to check for errors, do not forget to disable it later because it might slow down performance or lead to oversized log files.

13.2
See all
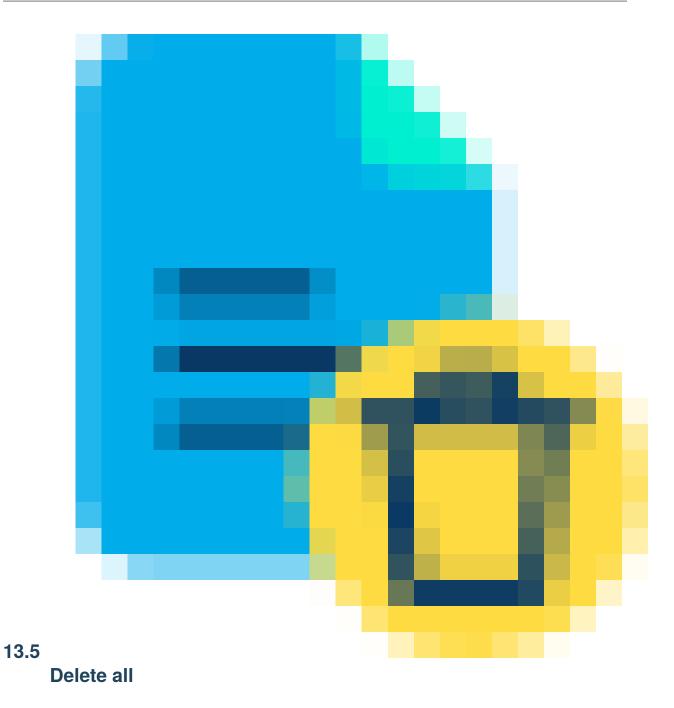
This button will print the full contents of the system log file in the output tab.

13.3

See category

Here you can filter the contents of the system log file by some keywords.

13.4

Search

Here you can filter the contents of the system log file by any term.

13.5 Delete all

If after debugging the system the log file is too large here you can delete it to start from scratch on the next reboot.
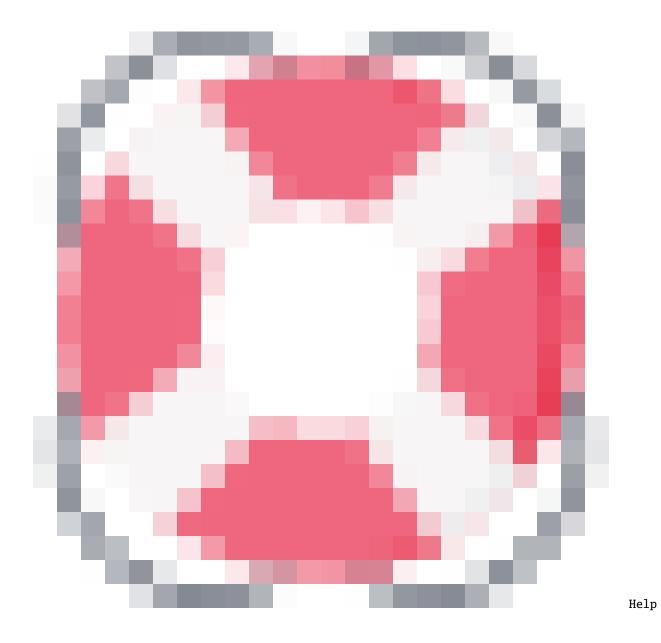
FOURTEEN

SIGNAL K INSTALLER

Note: To run this app type this in a terminal:

```
openplotter-signalk-installer
```

Help

opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

14.1 Reinstall Signal K

After installing *OpenPlotter Signal K Installer* app, the Signal K server should be also installed and you do not have
to do anything else to start using it. We add this option in case you need to reinstall the server from scratch if it ever
becomes unstable.

Caution: Reinstalling the signal K server will remove the current plugins, login credentials and settings.

14.2 ≣ Settings

OpenPlotter Signal K Installer app installs the server using port 3000 by default. To access the web administration panel of the Signal K server, you can use this URL from the browser included in OpenPlotter:
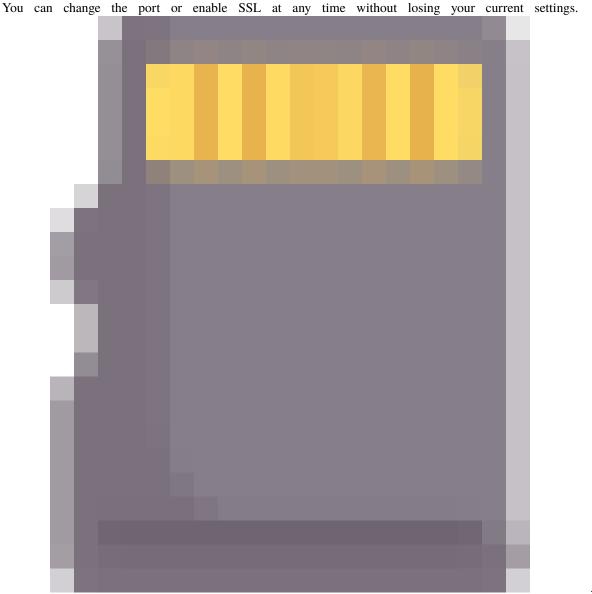
```
http://localhost:3000
```

Or this one from a browser running on any computer connected to the same network:

```
http://openplotter.local:3000
```
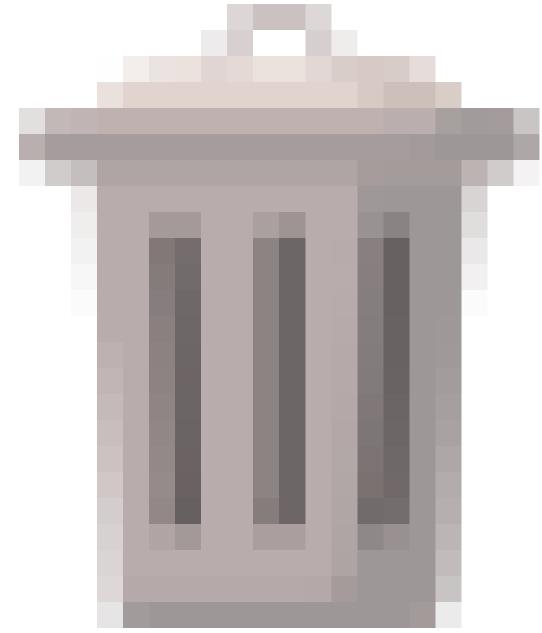
If you set the port 80 instead the default port 3000, the URLs would be:

```
http://localhost
http://openplotter.local
```

You can change the port or enable SSL at any time without losing your current settings. Use

Apply
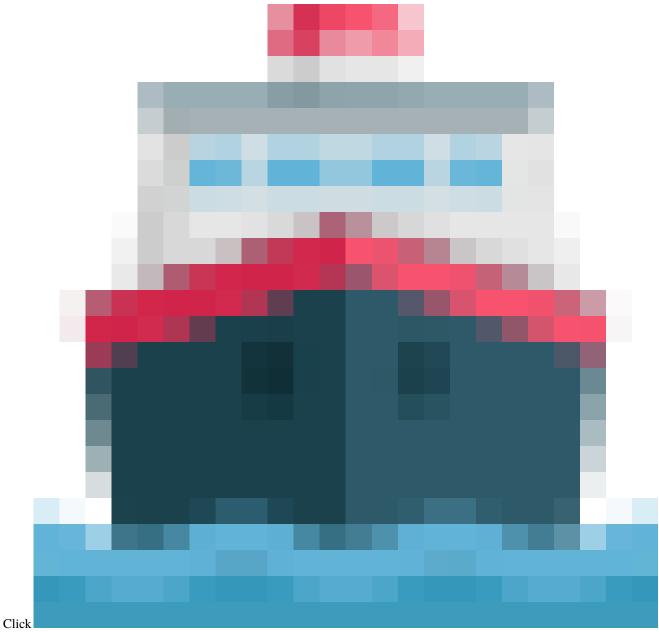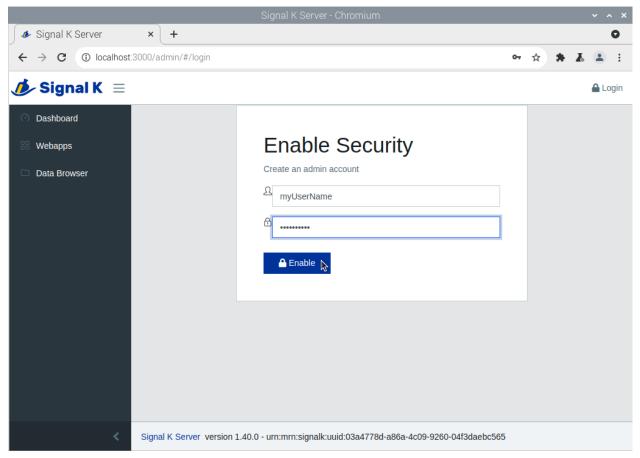
to save changes or
Cancel to reload current settings.
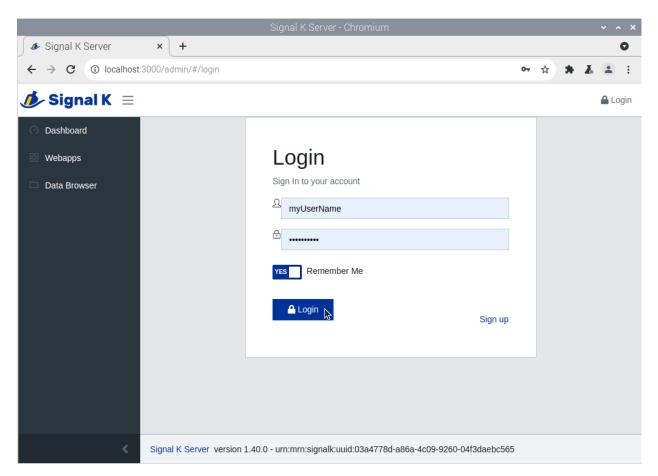
Click
`Vessel Data` to set some important data of your boat like name, MMSI, call sign, draft . . . You need to login to access this section of the web administration panel.

14.3 Logging in

When you first enter the Signal K web administration panel, it will ask you for a name and password to create an administrator account:

Once you do that you will be offered the login page:

The last menu item in the Signal K administrator is *Security*. You can add/delete users and change passwords there.

14.4 More info

To know how data is managed in OpenPlotter you should read the chapter *How does it work?* in the *Description* section.

On the official Signal K site you will find a lot of valuable information.

On the Signal K server github page you will also find some interesting information.

There are more than 150 plugins and more than 40 apps to extend the Signal K server features. Go to *Appstore →
Available* to install them:

FIFTEEN

NMEA 0183 MULTIPLEXING
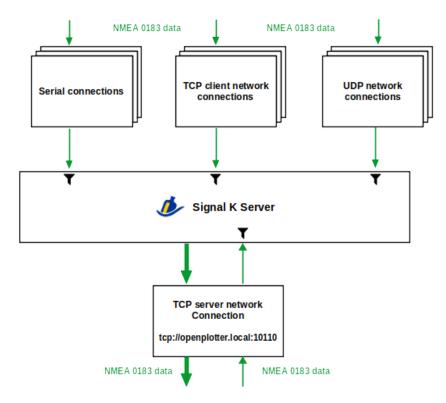
The Signal K server is a powerful NMEA 0183 multiplexer. It can merge all the NMEA 0183 devices on your boat into a single channel, add new data to this channel, redistribute data between devices, filter out unwanted data and even correct malformed data. As always, powerful tools can be a bit difficult to understand at first but here we will try to make it easier.

Using the Signal K server administrator you can set connections to get NMEA 0183 data through *Serial*, *TCP Client*, *TCP Server on port 10110* and *UDP*. There is another option called *GPSD* that works as a TCP client and that you will only use when you want to use that tool to control your GPS/AIS receiver. To add NMEA 0183 connections go to *Server → Data Connections*, click `Add`, select *NMEA 0183* in `Data Type` and choose an option in `NMEA 0183 source`:
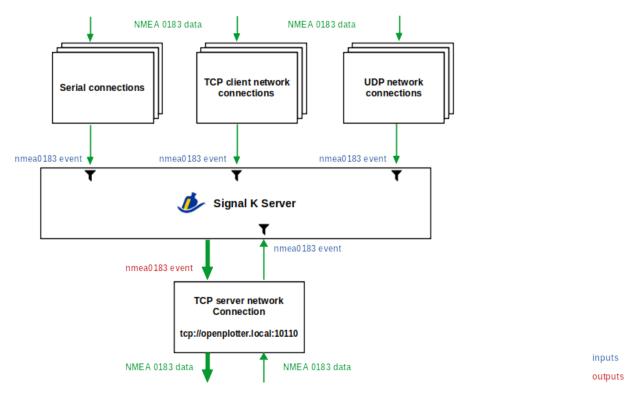
By default, the Signal K server always creates a TCP server on port 10110. **This TCP server on port 10110 will be the default output of all multiplexed input data**. If you create a new data connection by selecting *TCP Server on port 10110* in the `NMEA 0183 source` field, what you are actually doing is also enabling data input to this TCP server on port 10110.

You can also ▼ filter unwanted NMEA 0183 sentences in each input by entering them in the field `Ignored Sentences`:
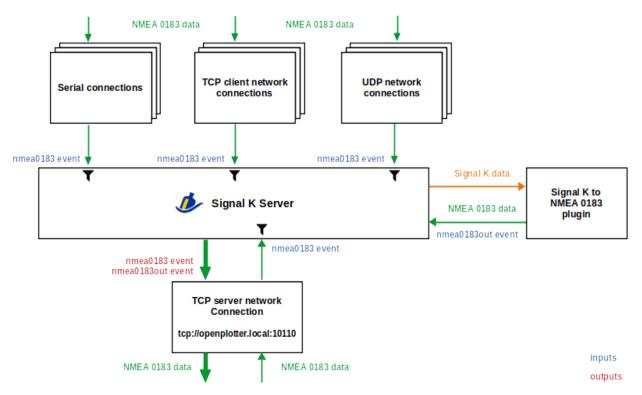
And now comes the most interesting. The Signal K server uses a mechanism called **Events** to internally tag data that comes from connections. When you create an NMEA 0183 connection, by default the data is assigned to the **nmea0183** event, this way the TCP server in port 10110 knows that any data tagged with *nmea0183* should be sent out to clients. If you want the data of any of the connections not to be sent to the output you just have to check `Suppress nmea0183 event` in the settings of that connection:

In addition to multiplexing data from different NMEA 0183 inputs, the Signal K server also converts the NMEA 0183 data to Signal K format and can also receive data in NMEA 2000 or Seatalk 1 format from other inputs which will also be converted to Signal K format. If you want to convert some data to NMEA 0183 because its original format was not NMEA 0183 and you want to have it in the output as well, you should use the *Signal K to NMEA 0183* plugin.

This plugin automatically adds the **nmea0183out** event to the converted data and since the TCP server on port 10110 also listens to this event, it will also send this data to the output. Using two event names here allows distinguishing between original and derived/generated data:
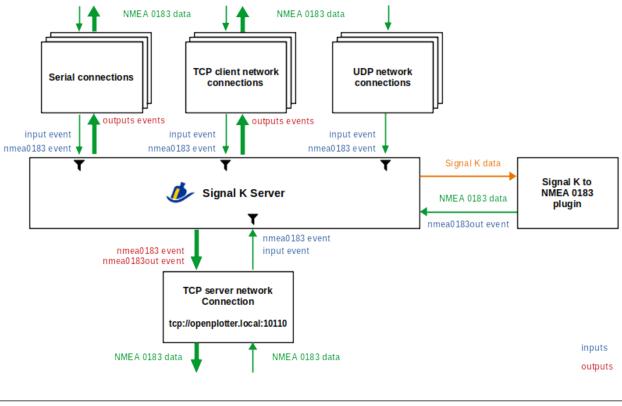
We have seen how the server uses some predefined system events to collect or discriminate NMEA 0183 data from several sources and send them merged by a single output. We will see now how using custom events we can create more than one output or share data between connections.

When creating NMEA 0183 connections, you will see an option called `Input Event`. That means you can configure an extra event to tag the input data in addition to the default *nmea0183* event.

When creating *Serial* or *TCP Client* connections, you will see an additional option called `Output Events`. That means that all data tagged with those events will be sent out the output of that device or network connection.
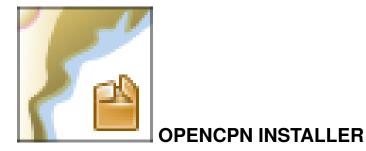
UDP connections are inbound only. The *Output Events* of the TCP server on port 10110 are *nmea0183* and *nmea0183out* and cannot be changed:

Note: More information about event handling can be found on the Signal K server github page.
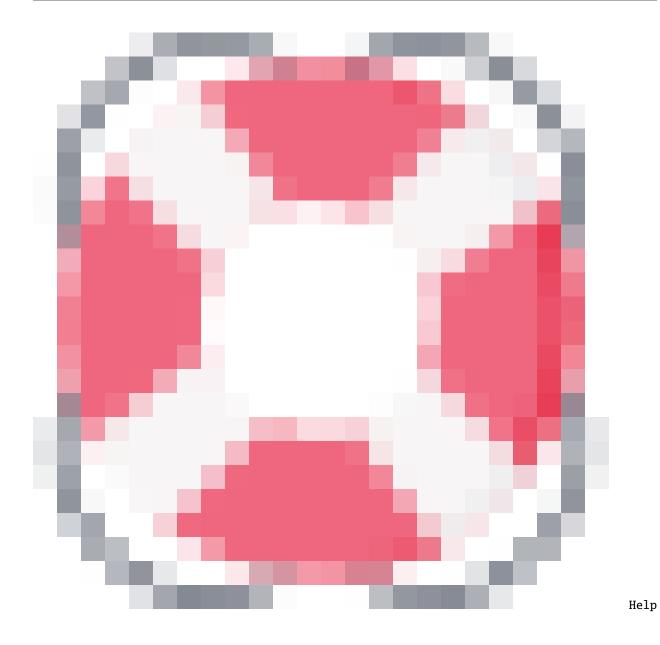
SIXTEEN

 OPENCPN INSTALLER

Note: To run this app type this in a terminal:

```
openplotter-opencpn-installer
```

Help

opens an offline copy of this documentation in a browser and `Settings` opens the main app *OpenPlotter Settings*.

OpenCPN can be installed on multiple Debian derivatives (Raspberry OS, Ubuntu, Mint...) and these OS can be installed on multiple architectures (i386, armhf, arm64, amd64...). We have added all the ways to install OpenCPN to this *OpenPlotter OpenCPN Installer* app so you can choose the one that best suits your system.

16.1 ⣿ Check Versions

When you open the app, all buttons are disabled. You have to check the current versions present in all the installed sources. This could take even a minute the first time:

A list will be displayed with the different versions of all the available sources and some recommendations:

You can install OpenCPN in two different ways, from the **Debian/Ubuntu** repositories or from **Flatpak**. You can use both ways and install two instances of OpenCPN that can be used simultaneously on the same machine without

problems.

In Flatpak there is only one source and therefore only one version but in Debian/Ubuntu there are several sources and several versions available:

- **Debian/Ubuntu**: This is the official Debian/Ubuntu repository. There is an OpenCPN package, but it will probably always be old.

- **Ubuntu PPA**: This is an special repository to be added in Ubuntu but it will also work in Debian and Raspberry OS. Packages in this repository are always up to date and are fully compatible with packages in the official Debian/Ubuntu repositories above.

- **Debian/Ubuntu Backports**: The official backports repositories are used to install packages that exist in higher versions of the system that have not yet been updated in the current system version.

16.2 Which source to choose?

After checking versions the buttons on the ⬚ `Install` tab will now be enabled:

- ⬚ **Debian/Ubuntu - Ubuntu PPA**: This option will install the highest version found in the official *Debian/Ubuntu* repository and the *Ubuntu PPA* repository.

- ⬚ **Debian/Ubuntu Backports**: This option will install the latest version in the official *Debian/Ubuntu Backports* repository.

- ⬚ **Flatpak**: This option will install the latest version in the *Flatpak* repository. This option runs OpenCPN in a kind of container independent of the host system and for this reason the time and size of the download will be larger.

Important: As a general rule, you should always choose the highest version, regardless of the source, unless you fall within one of the recommended uses or there is an OpenCPN plugin you need to use that is not included in any of the sources.

Note: At the time of writing this manual, OpenCPN Flatpak is the best option for touch screens because it allows right-clicking and two-finger zooming. However OpenCPN Flatpak may not work well in headless environments.

Note: The Debian/Ubuntu repositories only release OpenCPN packages for LTS (Long-Term Support) versions of your operating system. If you use a development version of Debian or Ubuntu you should use OpenCPN Flatpak.
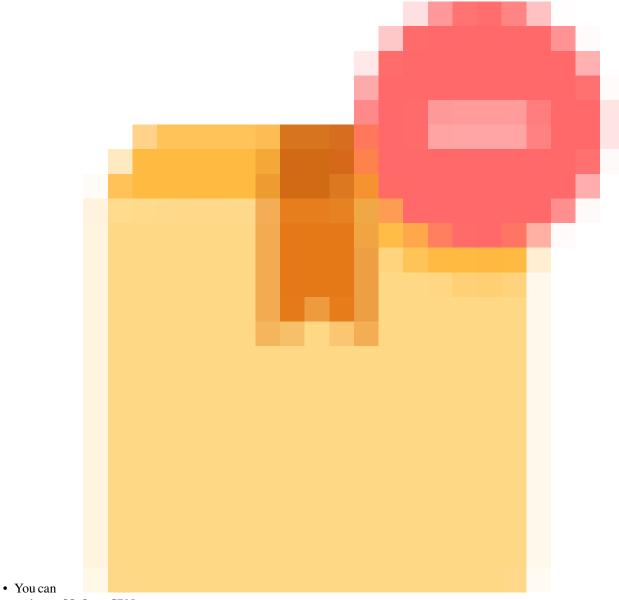
Note: Sometimes when we install a new version, some plugins may not be compatible and cause OpenCPN to crash or prevent it from opening. If this happens we can try to remove the old plugins by deleting the *~/.local/lib/opencpn* folder and reinstalling the plugins when OpenCPN opens normally again.

If you install OpenCPN twice, from Debian/Ubuntu and Flatpak, the system will differentiate between them by adding the FP suffix to the version installed from Flatpak:
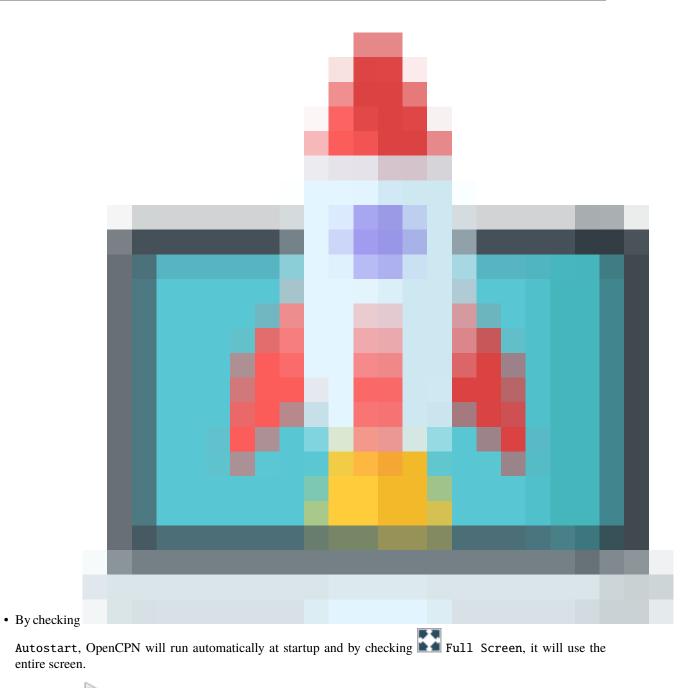
16.3 OpenCPN Installer actions

Once OpenCPN is installed, there are a few actions you can take in this app.

- You can
  Uninstall OpenCPN at any time.

- By checking

    `Autostart`, OpenCPN will run automatically at startup and by checking `Full Screen`, it will use the entire screen.

- By clicking `Open`, OpenCPN will run.

SIGNAL K CONNECTION

To receive data from all your devices and sensors on OpenCPN, the recommended way is to create all connections on the Signa K server and then create a single connection to the server on OpenCPN using these settings:

Type
> Network

Protocol
> signal K

Address
> localhost

DataPort
> 3000

Uncheck
> Automatic server dicovery
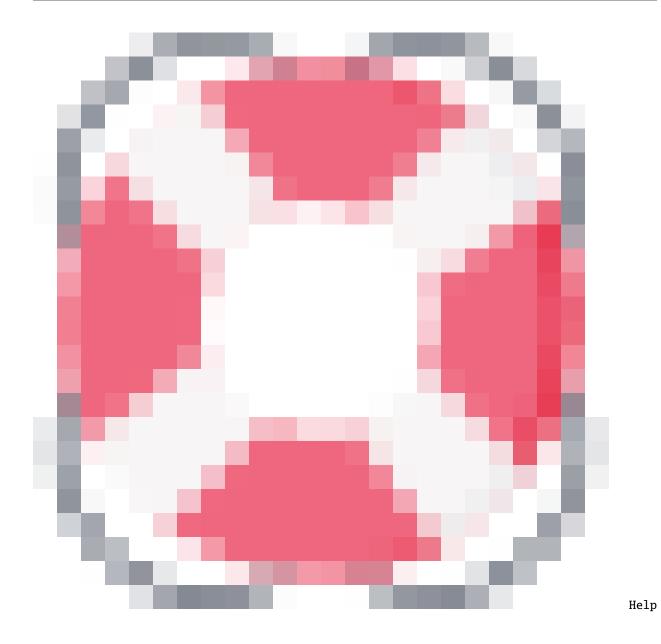
DASHBOARDS

Note: To run this app type this in a terminal:
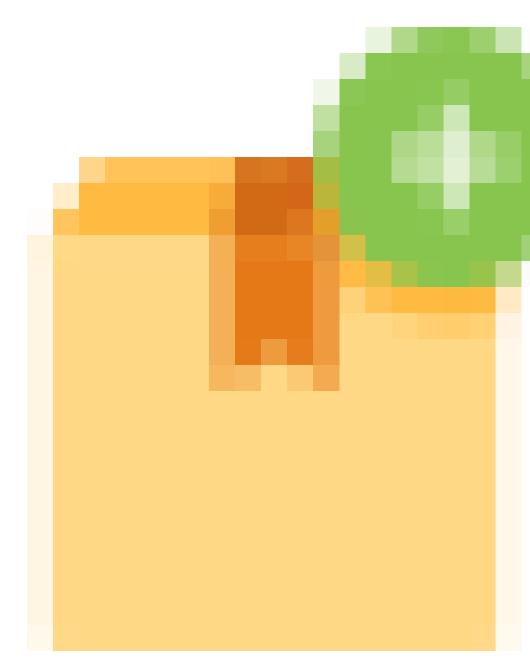
```
openplotter-dashboards
```
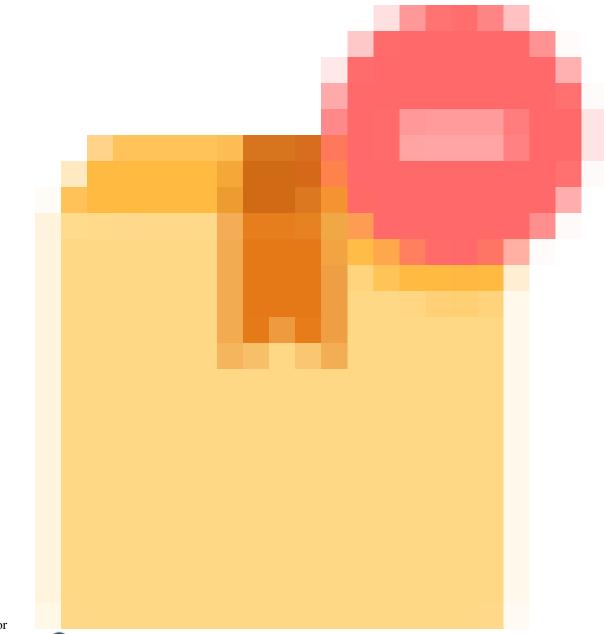
Help

opens an offline copy of this documentation in a browser and
Settings opens the main app *OpenPlotter Settings*.
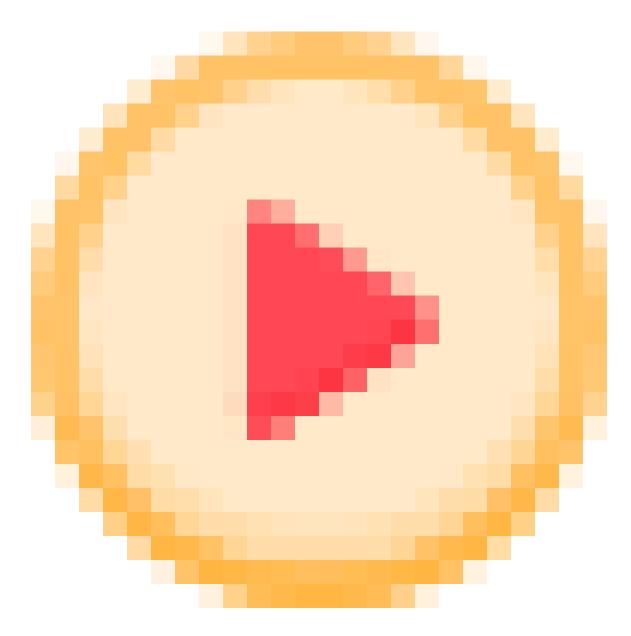
Select any app from the list and click
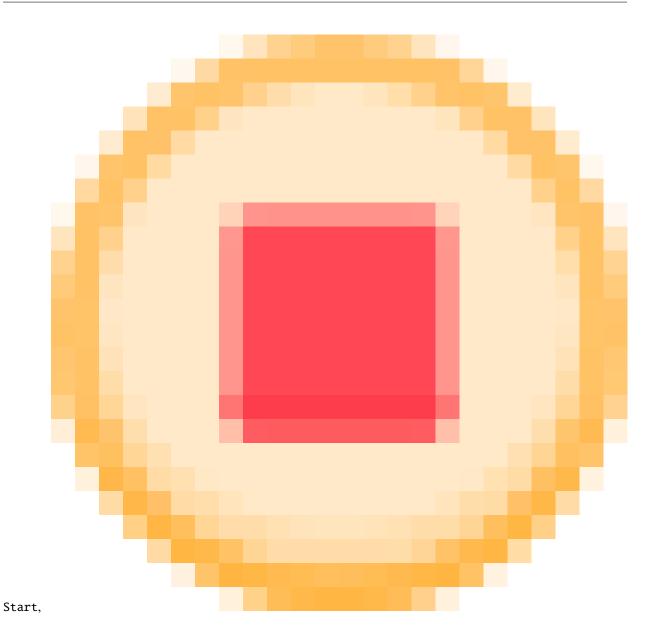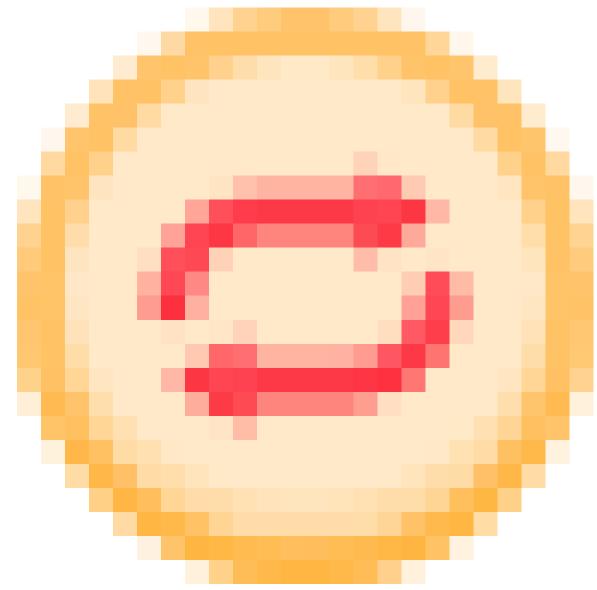
`Install` or

`Uninstall`. Click ↻ `Refresh` after any of these actions to refresh the list.
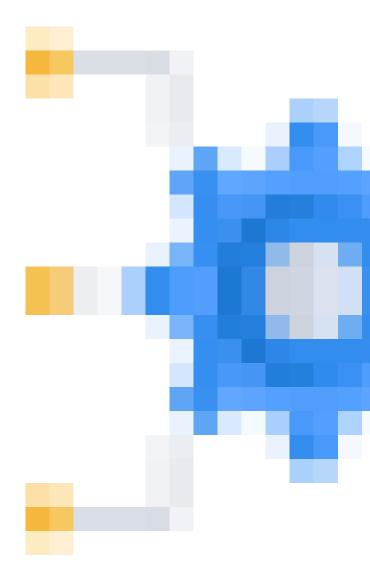
Most of these dashboards are Signal K server plugins, but some are standalone third-party applications such as Grafana or InfluxDB. Once these applications are installed you can

Start,

Stop,
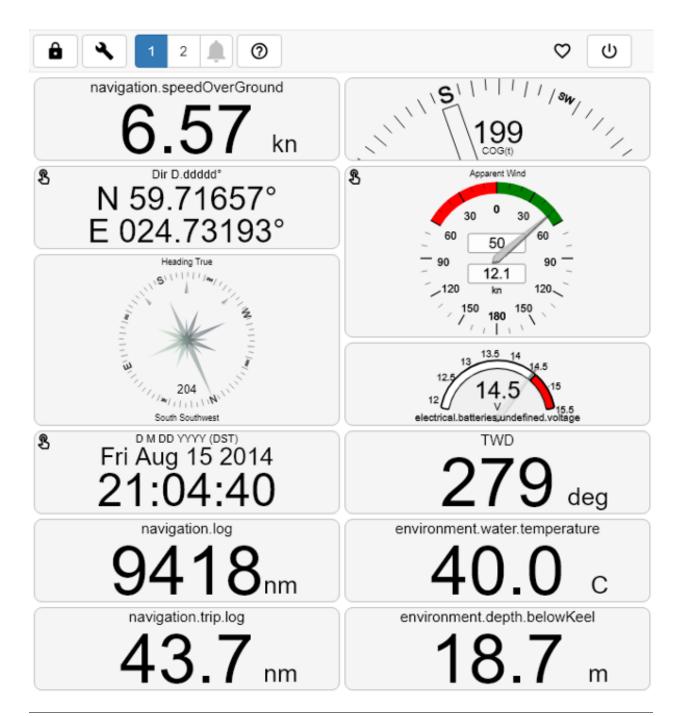
Restart and configure automatic startup by toggling Autostart in the
Processes tab:
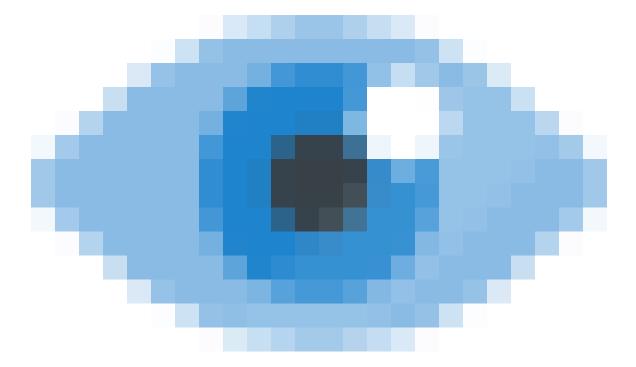
INSTRUMENT PANEL

This is the dashboard built into the Signal K server. It is highly configurable and you can even embed panels from other applications like Grafana.

You can access this dashboard by selecting *Instrument Panel* in the list and clicking

Open
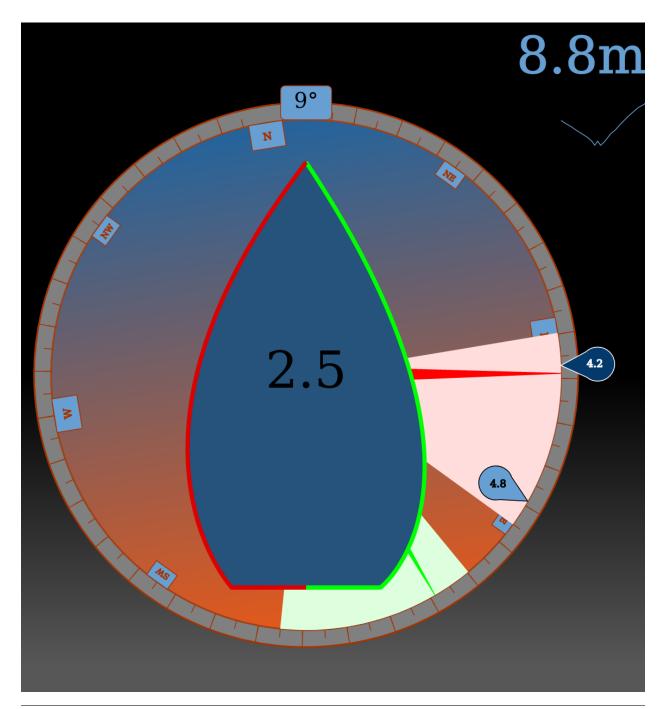
or by typing this adddress in your local browser: *http://localhost:3000/@signalk/instrumentpanel/*. You can also connect to this page from any device connected to the same network as OpenPlotter using the address *http://openplotter.local:3000/@signalk/instrumentpanel/*.

Click on the help icon to access full documtnation.
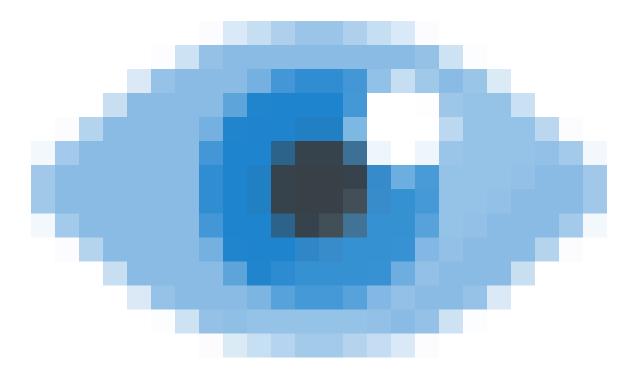
CHAPTER

TWENTY

SAILGAUGE
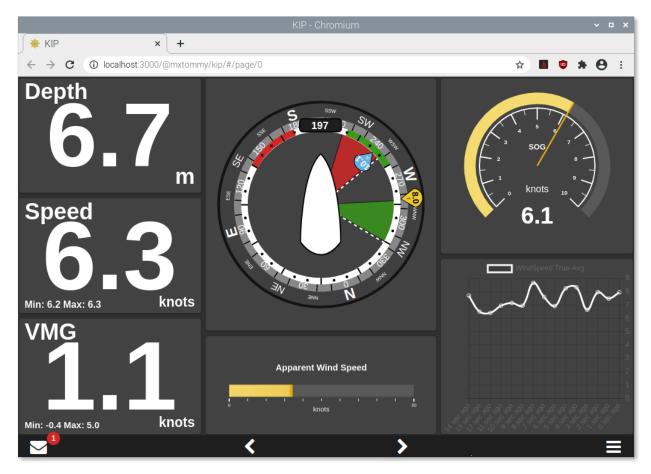
This signal K server plugin provides an all-in-one gauge for sailboats.

You   can   access   this   dashboard   by   selecting   *SailGauge*   in   the   list   and   clicking

Open or by typing this adddress in your local browser: *http://localhost:3000/@signalk/sailgauge/.*   You can
also connect to this page from any device connected to the same network as OpenPlotter using the address
http://openplotter.local:3000/@signalk/sailgauge/.
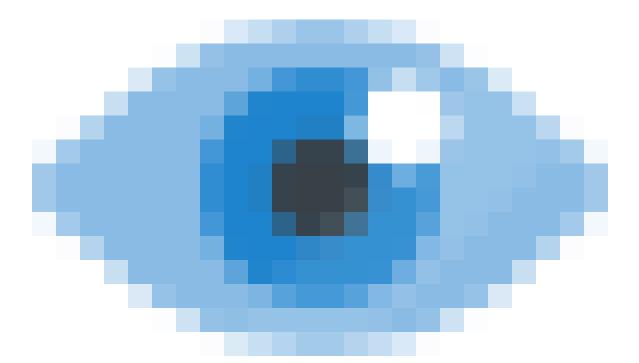
KIP

Using this wonderful Signal K plugin you will be able to easily build sexy dashboards containing graphs, gauges, numeric and level panels...

You can access this dashboard by selecting *Kip* in the list and clicking
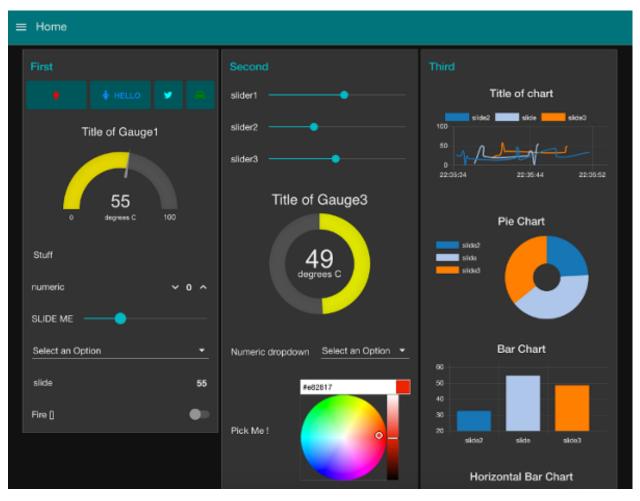
Open or by typing this adddress in your local browser: *http://localhost:3000/@mxtommy/kip/*. You can also connect to this page from any device connected to the same network as OpenPlotter using the address *http://openplotter.local:3000/@mxtommy/kip/*.

Go to the plugin homepage for the list of features and tips: https://github.com/mxtommy/Kip

NODE-RED DASHBOARD
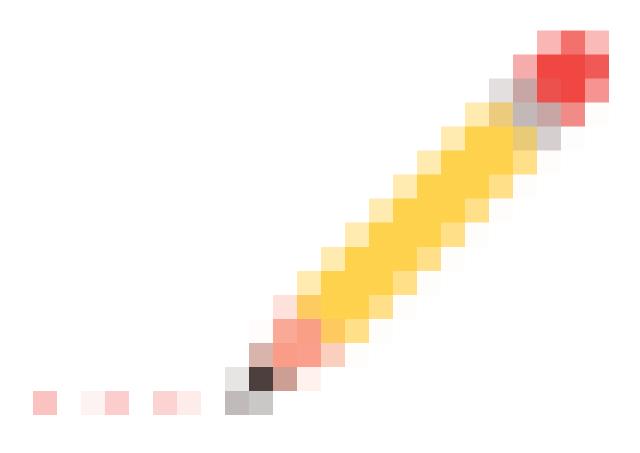
Node-RED is a tool that uses a browser-based visual editor to program the connections between hardware devices, and APIs such as Signal K. It allows you to gather, adjust, enhance and deliver the data without needing to understand complex programming languages such as C or Python.

In OpenPlotter, Node-RED is embedded in the Signal K server as a plugin, and contains some specific nodes to manage Signal K data. Go to the plugin homepage to see the list of nodes and some examples: https://github.com/SignalK/node-red-embedded

The installation also includes the node-red-dashboard plugin that adds more nodes for creating buttons, graphs, gauges, sliders, and so on.
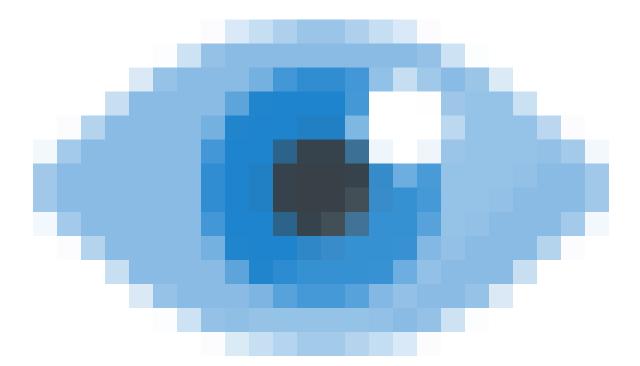
You can build Node-RED dashboards by selecting *Node-Red Dashboard* in the list and clicking

```
                                                                                                            Edit
```

or by typing this adddress in your local browser: *http://localhost:3000/plugins/signalk-node-red/redAdmin/*. You can also connect to this page from any device connected to the same network as OpenPlotter using the address *http://openplotter.local:3000/plugins/signalk-node-red/redAdmin/*.

You can see your Node-RED dashboards in action by selecting *Node-Red Dashboard* in the list and clicking

`Open` or
by typing this adddress in your local browser: *http://localhost:3000/plugins/signalk-node-red/redApi/ui/*. You can
also connect to this page from any device connected to the same network as OpenPlotter using the address
http://openplotter.local:3000/plugins/signalk-node-red/redApi/ui/.

Using and learning Node-RED takes some time and is not the goal of this document. If you are not a programmer
but would like to learn some basic principles, Node-RED is the perfect starting point. Everything will be possible in
OpenPlotter using Node-RED.
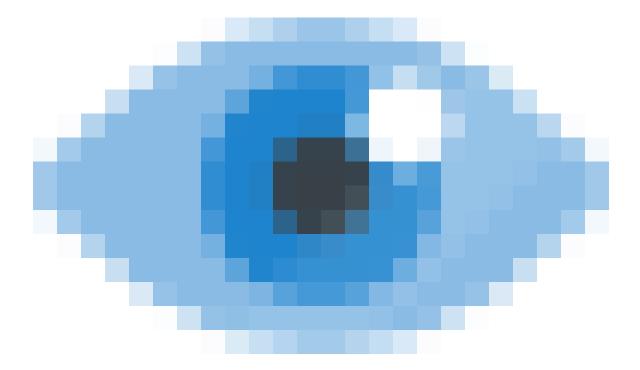
- Go to the Node-RED homepage to get started: https://nodered.org/docs

- Go to our dedicated forum to share or ask about Node-RED in OpenPlotter: https://forum.openmarine.net/
  forumdisplay.php?fid=15

GRAFANA

Grafana is one of the most powerful platforms to manage and visualize data. After installation, select *Grafana* in the dashboard list and click

Open:

The browser will open and show us the page *http://localhost:3001*. You can also connect to this page from any device connected to the same network as OpenPlotter using the address *http://openplotter.local:3001*.

The initial username and password are both set to *admin*. On the next page we will be asked to change and verify a new password.

There are many ways to manage and display data in OpenPlotter using Grafana, but in this short tutorial we will focus on how to display data from the Signal K server in real time. We want to catch the delta messages that the server sends over the websocket connection, so we will use the *WebSocket API* plugin that is pre-installed in OpenPlotter.

Go to *Configuration → Data sources* and click `Add data source`. Select *WebSocket API* at the end of the list and write *ws://localhost:3000/signalk/v1/stream?subscribe=all* in `Host:`

After saving, Grafana will be ready to receive all delta messages for all existing Signal K keys and values. Now we need to filter a Signal K key to be displayed on a dashboard, for this example we will use *environment.outside.temperature*.

Go to *Create → Dashboard* and click `Add a new panel`. In the `Query` tab, select *WebSocket API* as the `Data Source` and add 3 `Fields` by filling in the `Field`, `Type` and `Alias` fields using the data from the image below:

Finally go to `Transform` tab and add the `Filter data by values` transformation using the settings shown in the image below:

Signal K uses the Kelvin unit for temperature, so if we want to convert the data to Celsius, we can add another transformation `Add field from calculation` using the settings in the image above.

Once you are happy with your panel click `Save` and continue building your perfect dashboard by adding more panels.
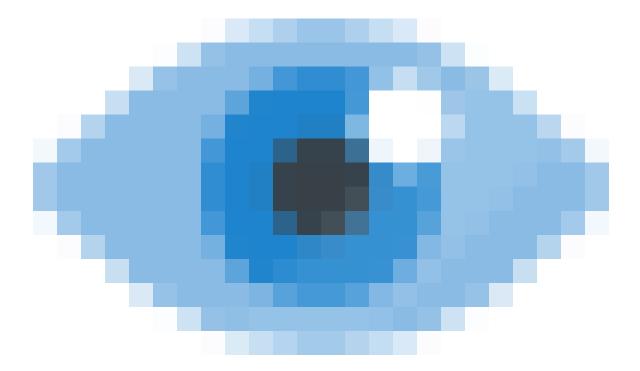
INFLUXDB OSS 2.X

Note: InfluxDB OSS 2.x is only available for 64 bit systems.

InfluxDB is a powerful time series database system that can be used to store data and track its' evolution over time. After installation, select *InfluxDB OSS 2.x* in the dashboard list and click

Open:

The browser will open and show us the page *http://localhost:8086*. You can also connect to this page from any device connected to the same network as OpenPlotter using the address *http://openplotter.local:8086*.

Fill in all the required fields using data that you can easily remember later. A *Bucket* is what we normally know as a database:

On the next page there are several options to learn the basics of InfluxDB but you can ignore them because here we are going to guide you step by step to store and display data for the Signal K key *environment.outside.temperature*. Click `Configure Later`.

The amount of space used by InfluxDB will depend significantly on the volume of data you wish to store, and the amount of data you wish to retain. It is important to control the size of your databases so you do not consume all of your valuable storage space. In InfluxDB this is done by configuring *data retention* on each bucket. The retention policy of the bucket we created during the initial setup is set to *Forever* by default, so we will create a new one. Go to *Data → Buckets* and click `Create Bucket`:

We will call it *myBucket24h* to signify its retention policy, and we will configure *Older Than 24 hours* in `Delete Data`:

Now we need to create a token so that we are allowed to write data to our buckets from external applications. Go to *Data → API Tokens* and select *Read/Write API Token* in `Generate API Token`:
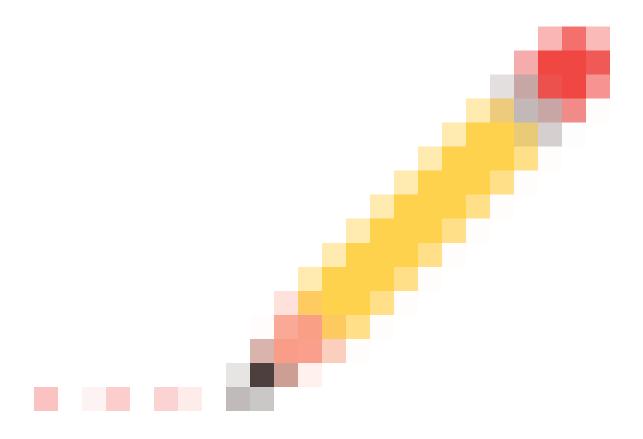
Write a description for your token and select *All Buckets* in both `Read` and `Write` options. This way we will use the same token for all current and future buckets:

After saving, open your new token and click `Copy to Clipboard`:

The most complex part of using InfluxDB is sending data to databases using an external application called *Telegraf* but the good news is that OpenPlotter will do the hard work for you. Go back to the *OpenPlotter Dashboards* app, select *InfluxDB OSS 2.x* and this time click

Edit:

Fill in `Organization`, `Bucket` and `Token` with the data that we have used in the previous steps. Be conservative with the `Ìnterval`, you do not need to store the air temperature every second because you are only interested in its evolution over time, so 10 minutes will be enough. Note that you can not only monitor your ship's data, but also the data of any ships you have data for, such as ships detected by AIS. For this example we will add

self.environment.outside.temperature in `Signal K key` field. Click to add a new item in the list:

When you have added all the items you want to monitor, click OK to save and your databases will begin storing data as it becomes available:

InfluxDB also contains tools for creating dashboards and these tools use *Flux*, an scripting language designed for

querying, analyzing, and acting on data. Do not worry, you do not have to learn Flux to create dashboards, just learn how to use these tools and how to make small changes in Flux scripts.

Go back to the InfluxDB web administrator and click `Explore`. Select *myBucket24h* in `FROM` column and filter by *_measurement: self.environment.outside.temperature* and *_field: value* adding 2 extra `Filter` columns. Click `Submit` to create a *Graph* cell using this query:

As you can see on the y-axis, Signal K uses Kelvin unit for temperature. If we want to convert the data to Celsius we have to make some changes in the Flux script that this *Query Builder* has created. Click `Script Editor`:

Replace these lines:

```
aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
yield(name: "mean")
```

By this one:

```
map(fn: (r) => ({ r with _value: r._value - 272.15 }))
```
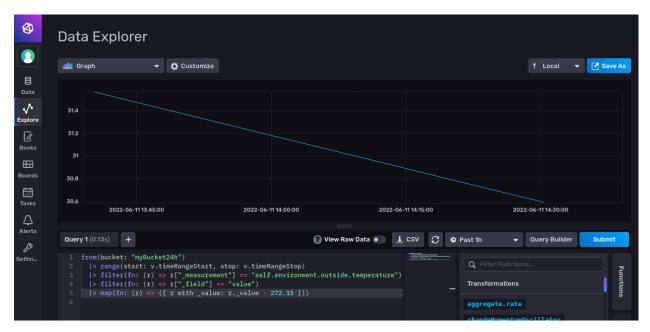
Note: To convert the data to other units you just have to apply the formula to this line. For example, to convert from Kelvin to Fahrenheit we have to apply the formula 1.8*(K-273)+32:

```
map(fn: (r) => ({ r with _value: 1.8*(r._value-273)+32 }))
```
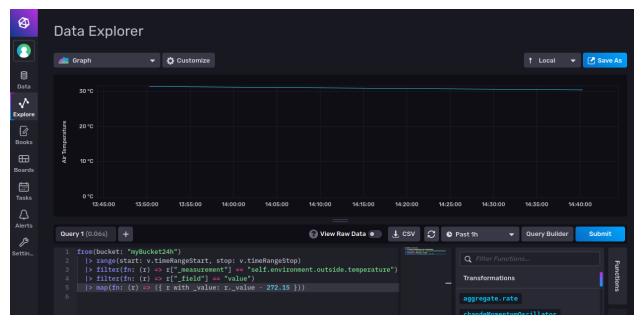
Your final script will look like this:

```
from(bucket: "myBucket24h")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "self.environment.outside.temperature")
  |> filter(fn: (r) => r["_field"] == "value")
  |> map(fn: (r) => ({ r with _value: r._value - 272.15 }))
```

After clicking Submit you will see that the y-axis now shows the data in Celsius:

Finally click `Customize` to add some aesthetic settings like labels, ranges or suffixes:

Once you are happy with your cell, click `Save As` and continue building your perfect dashboard by adding more cells.

24.1 InfluxDB in Grafana
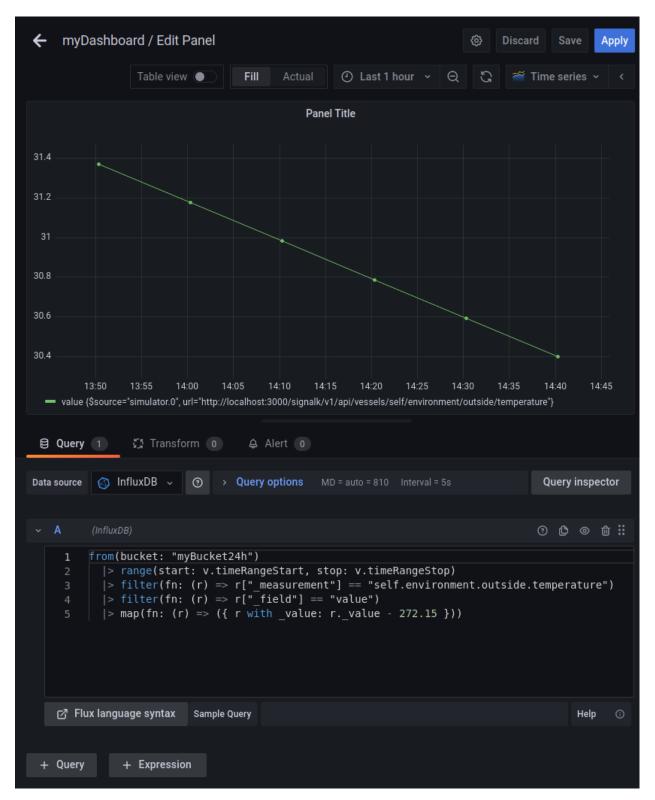
You may prefer to use Grafana to unify all your panels or cells with real-time and stored data. To access to the data stored in InfluDB buckets you have to create a new *Data Source* in Grafana. Go to *Configuration → Data sources*, click `Add data source`, select *InfluxDB* in the list and use the data from the image below, fill in `Organization`, `Bucket` and `Token` with the data that you have used in the previous steps:

Create a new panel, select *InfluxDB* in `Data source` and paste here the same Flux script you have generated using the *Query Builder* in the InfluxDB web administrator:

```
from(bucket: "myBucket24h")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "self.environment.outside.temperature")
  |> filter(fn: (r) => r["_field"] == "value")
  |> map(fn: (r) => ({ r with _value: r._value - 272.15 }))
```

Finally modify the display settings of the panel to adapt it to your preferences and you are done.

TWENTYFIVE

MANAGING SERIAL DEVICES

Note: To run this app type this in a terminal:

```
openplotter-serial
```

Help
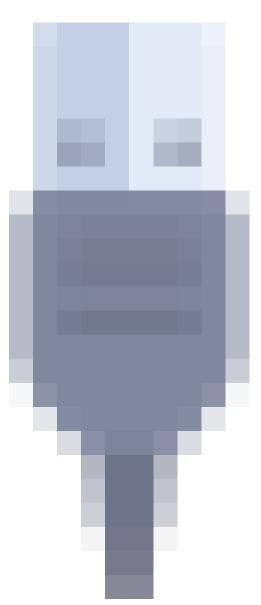
opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

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 device. This *OpenPlotter Serial* app allows you to define an alias for your device that will always be tied to it even after unplugging it.

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

On the
`Devices` tab, you can define your devices by providing the alias, defining the type of data this device manages, and choosing how this device should be remembered: by its vendor, product, and serial number, or by the port where it is plugged in.

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.

Devices are marked in different colors:

white
    not defined yet.

green
    defined as NMEA 0183 device.

blue
    defined as NMEA 2000 device.
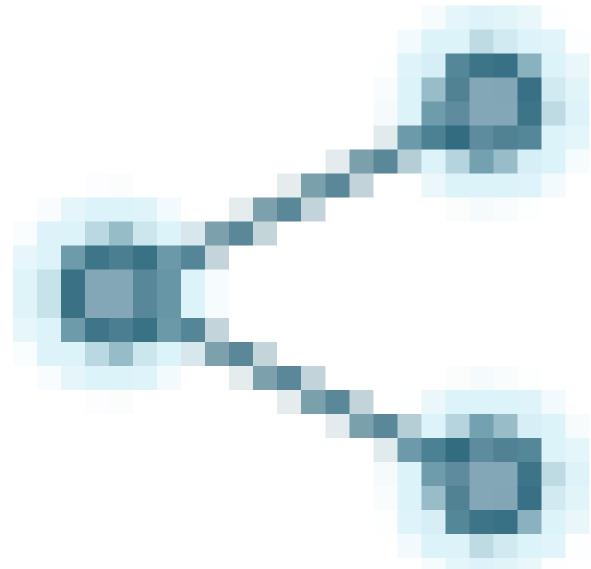
yellow

defined as Signal K device.

red

the device is missing.

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.
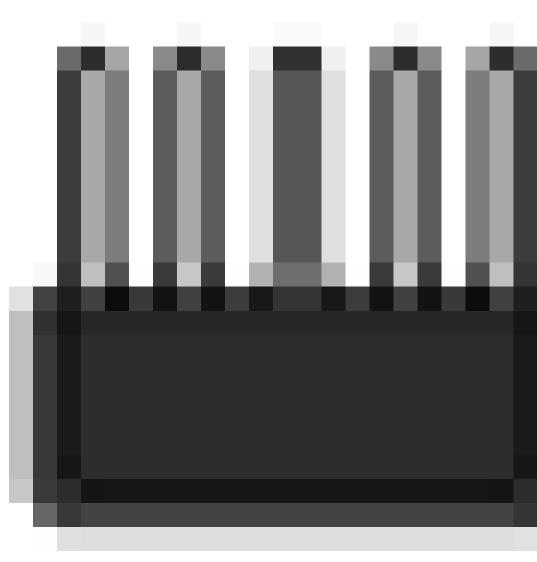
In the rest of this section, you will learn how to define and connect different devices using examples.

25.1 UART

On Raspberry Pi you can also connect serial devices to the GPIOs. On Raspberry Pi model 3 only UART0 is available, but on Raspberry Pi model 4 you have 4 additional UART interfaces.

Interface	RX GPIO	TX GPIO	Model
UART0	15	14	RPI 3, RPI 4
UART2	1	0	RPI 4
UART3	5	4	RPI 4
UART4	9	8	RPI 4
UART5	13	12	RPI 4

In Raspberry Pi 3 and 4 the Bluetooth interface and the UART0 interface share GPIO (GPIO14 for TX and GPIO15 for RX). Bluetooth is enabled and UART0 is dissabled by default. If you enable UART0, Bluetooth will be disabled.

To enable any UART interface, press
`UART0`, `UART2`, `UART3`, `UART4` or `UART5` and after reboot, you will see a new `ttyAMA...` device:
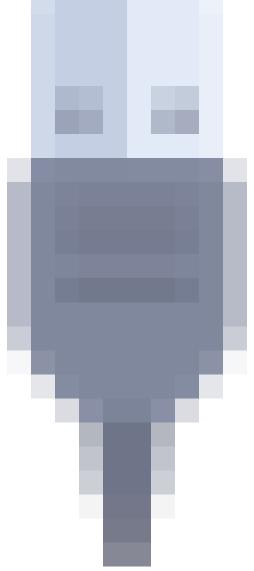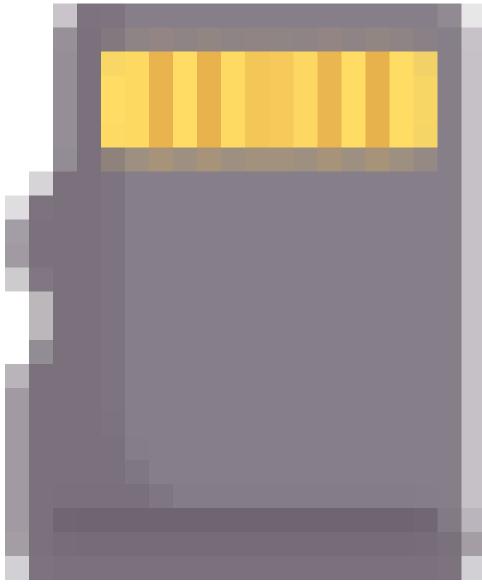
CONNECTING A USB GPS RECEIVER

To see how this all works, we are going to configure the most basic device, a USB GPS receiver. In

Devices tab, 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 plugged in.

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.

To send data from the USB GPS to OpenCPN, you need to first connect the de-

vice to the Signal K server and then connect OpenCPN to the Signal K server. In

Connections tab, select the *ttyOP_gps* device and press ![icon] Add to Signal K:

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:

Note: 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.

Note: Select
`Add to Pypilot` only if you are using a pypilot controller. Pypilot will send data to the Signal K server automatically.

TWENTYSEVEN

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.

27.1 Wiring

Typical RS422 device looks like the one below:
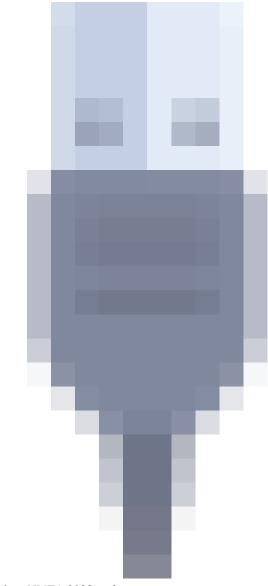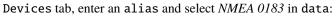
There are normally 4 or 5 connections: TX+, TX-, RX+, RX-, GND.

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-.

27.2 Input data
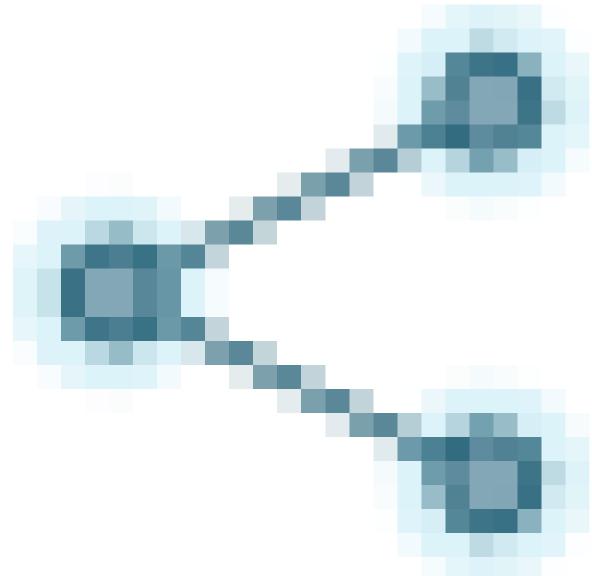
To obtain data from these converters, follow the same steps as for connecting the *USB GPS* in the example of the previous chapter of this section. Below are the summarized steps.

In
Devices tab, enter an alias and select *NMEA 0183* in data:

In
`Connections` tab, create a Signal K connection:

Consult your device manual to find the `Baud Rate`, if you can not find it 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.

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.

27.3 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 OpenCPN control your autopilot. Let's see how to send data to the autopilot using the same USB RS422 converter.

When you activate a route in OpenCPN, RMB and APB sentences start to be generated. As usual we need to send this new data to the Signal K server and you can do this by creating a UDP client connection. Select `Network`, `Protocol`: *UDP*, `Address`: *0.0.0.0*, `DataPort`: *10119* (or any unused UDP port on your system), uncheck `Receive input on this Port`, check `Output on this port` and transmit only sentences *RMB* and *APB* in `Output filtering`:

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 → Data Connections* and click on `Add`. Set `Input Type`: *NMEA 0183*, ID: *opencpn*, `NMEA 0183 Source`: *UDP*, `Port`: *10119* (or whatever you have set in OpenCPN), `Input Event`: *autopilot* and click `Apply`:

Finally, you need to edit the connection you configured for your USB RS422 device input data to specify what data should be sent to your boat. Go to *Server → Data 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. Activate a route in OpenCPN and you will start sending data to your autopilot.
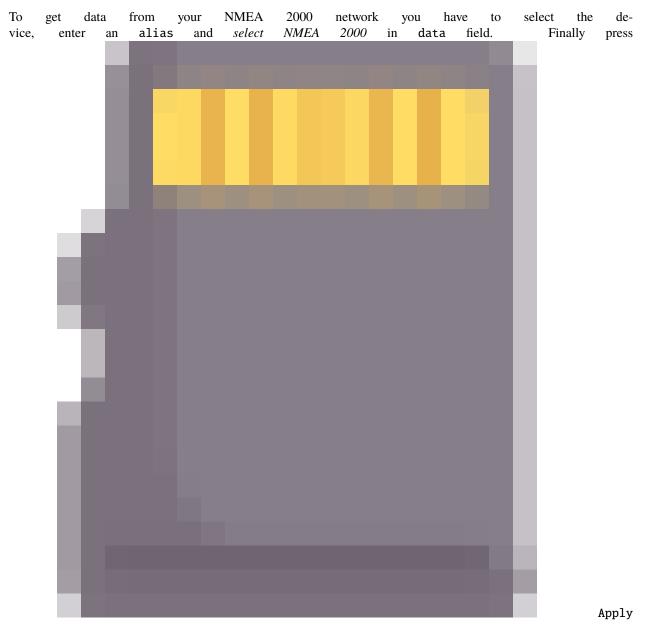
Note: To better understand how events work on the Signal K server, see the chapter *NMEA 0183 multiplexing*.

TWENTYEIGHT

CONNECTING A USB CAN CONVERTER

This tutorial is for any NMEA 2000 converter that can connect via USB such as the *Actisense NGT-1*, the *OpenMarine CAN-USB Stick* (discontinued) and any *slcand* device.

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   Add to CAN Bus:

If you are using a *slcand* device click on `MANUAL` and go to *Input data by slcand* chapter to learn how to configure these devices.

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

The device will be marked blue and you are done:

Open the *OpenPlotter 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:

Note: To learn how to send data through your NMEA 2000 converter, see the chapter *Output data*.

CONNECTING THE DAISY HAT

Specification
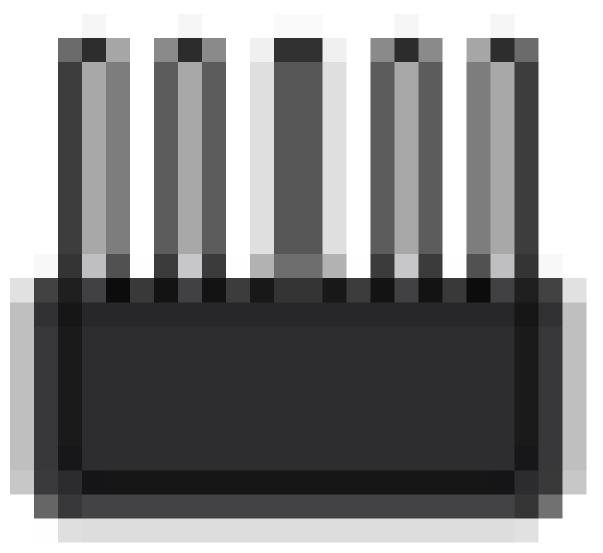
- True two channel receiver, continuously receiving on AIS channels A (161.975 MHz) and B (162.025 MHz)

- 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, Pi 4 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 a serial port in the GPIO header of the Raspberry Pi by click-

ing the
`UART0` 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 a new entry.   Select the line with *ttyAMA0*, give it
an   alias   (for   example   daisy)   and   select   *NMEA   0183*   from   the   data   dropdown,   then   press

Apply:

Now we need to connect the *ttyOP_daisy* device to 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`:

You are done, the Signal K server and any program connected to it, such as OpenCPN, should now receive AIS data.

Check OpenCPN to make sure there is a connection to the Signal K server and that it is getting data from your DAISy HAT:

CAN BUS

Note: To run this app type this in a terminal:

```
openplotter-can
```

Help

opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

To configure a NMEA 2000 converter that can connect via USB such as the *Actisense NGT-1* or the *OpenMarine CAN-USB Stick* (discontinued), see the chapter *Connecting a USB CAN converter data*.

If you want to configure a USB converter type `slcand` or based on the `MCP2515` chip, continue reading this section.

30.1 Wiring

It does not matter what type of converter you have, USB or a Raspberry HAT, all of them should have a L connector and a H connector. In most cases you simply have to connect the CAN_L cable of your NMEA 2000 CAN-BUS to the L connector of your converter and the CAN_H cable of your NMEA 2000 CAN-BUS to the H connector of your converter:

Note: Some converters may also need to connect GND, see the documentation for your device.
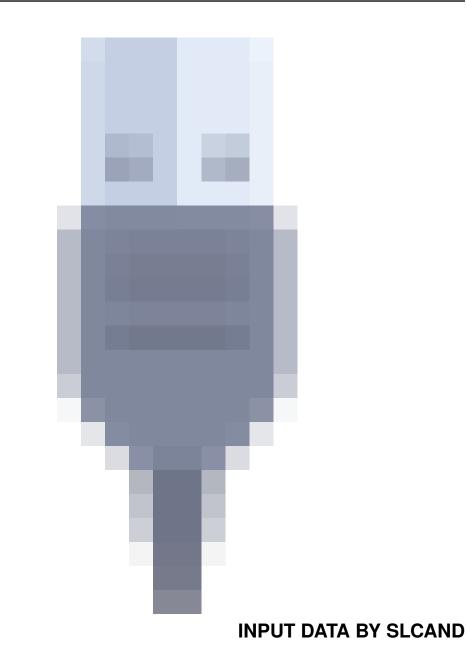
Caution: Do not connect the 12v power cable in any way.

At each end of the *Trunk Line* a *Termination Resistor* must be installed. Each termination resistor is 120 Ohms and is connected in parallel across the data pair of wires. This provides a total of 60 Ohms resitance across the network and is essential to data integrity. Most converters incorporate a 120 Ohms resistor to be optionally connected in case your NMEA 2000 CAN-BUS does not have a terminator. See the documentation for your device.

THIRTYONE

INPUT DATA BY SLCAND
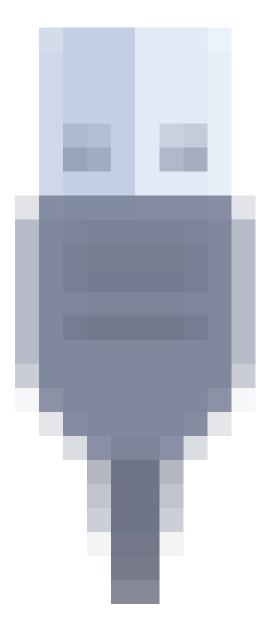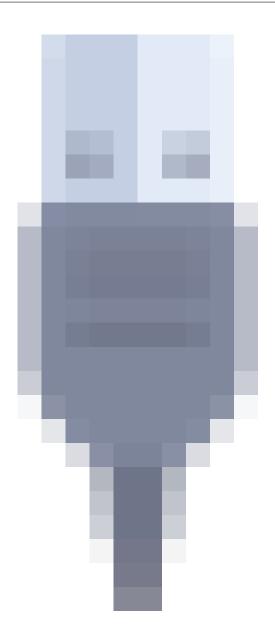
This tutorial is for any NMEA 2000 converter that can connect via USB using *slcand* such as the devices available from CANable.

First of all you have to define the device using the *OpenPlotter Serial* app. Then return to this OpenPlotter CAN Bus app

and click
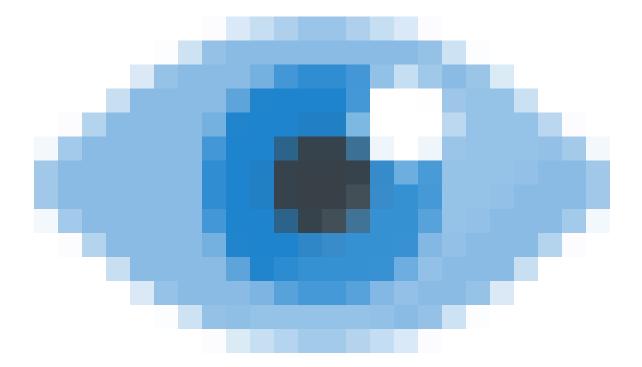
`Add device` in the
`slcand` tab:

Select the device you defined in the *OpenPlotter Serial* app and click OK:

From this moment you should receive data on your device. Select the item from the list and click

Check `device traffic` to confirm data entry:

Now we need to get this data to the Signal K server. Select the connection from the list and click ![icon] **Add Connection**. The Signal K server will restart and you are done:

Go to the Signal K server administration interface to confirm that the connection is now active:

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

THIRTYTWO

INPUT DATA BY MCP2515/MCP251XFD

There are some HATs for Raspberry Pi available with the MCP2515 chip (and its successor MCP251xfd) on board that you can easily configure in OpenPlotter. After mounting the HAT click on

Add

`MCP2515 device` in
`MCP2515` tab:

Interface
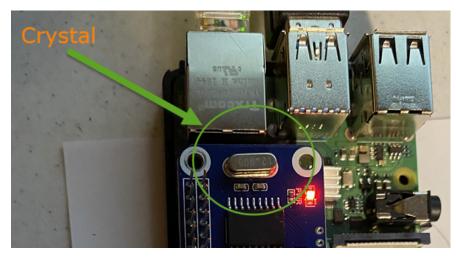
Select *SPI0 CE0* if your HAT connects to GPIO 8 (pin 24) and select *SPI0 CE1* if your HAT connects to GPIO 7 (pin 26).

Oscillator

The oscillator can be found by looking at the crystal chip on the HAT as per the pic below.

Interrupt

Select the GPIO your HAT connects for interruption. See the documentation for your device.

After entering the required data the system needs to reboot to apply the changes:

After a reboot you should receive data on your device. Select the item from the list and click

Check `device traffic` to confirm data entry:

Now we need to get this data to the Signal K server. Select the item from the list and click Add Connection. The Signal K server will restart and you are done:

Go to the Signal K server administration interface using your web browser, 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:

OUTPUT DATA

You may have some sensors connected to OpenPlotter that receive data in Signal K format, or you may have data from NMEA 0183 or Seatalk1 converters that have also been translated to Signal K format. In these cases, you probably want to convert this data to NMEA 2000 and send it to your CAN Bus using the same device you use to send NMEA 2000 data to OpenPlotter.

33.1 Unblocking transmission

To protect your network, the *Actisense NGT-1* and the *OpenMarine CAN USB Stick* (discontinued) devices have most PGNs blocked for transmission. On *slcand* and *MCP2515* devices, PGNs transmission is not blocked, so you can skip this chapter and go to the next one.

To unblock the PGNs you want to send to your NMEA 2000 network, go to ⚓ CAN USB tab, select the device from the list and click ⚓ `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:

33.2 Signal K to NMEA 2000 data conversion

This is valid for all devices. You have to tell the Signal K server what PGNs you need to convert from Signal K format
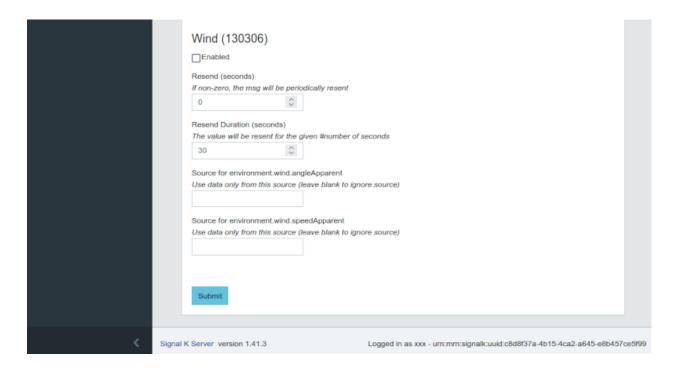
to NMEA 2000 format. To do this we use the Signal K plugin *Signal K to NMEA 2000*. Click ⚓ SK → `NMEA 2000` and you will be directed to the configuration page of this plugin:

Make sure the general setting *Enabled* is set to YES an enable Signal K to NMEA 2000 conversion for all desired PGNs:
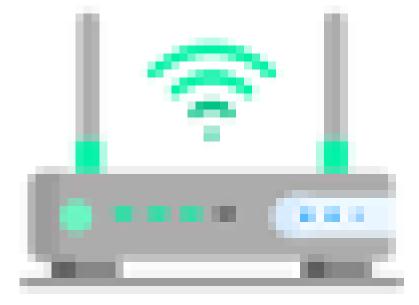
Click Submit at the bottom of the page and you are done:

Important: You do not need to do anything else to start sending data through your device. If you have previously set your device to input data, it will start sending this converted data automatically.
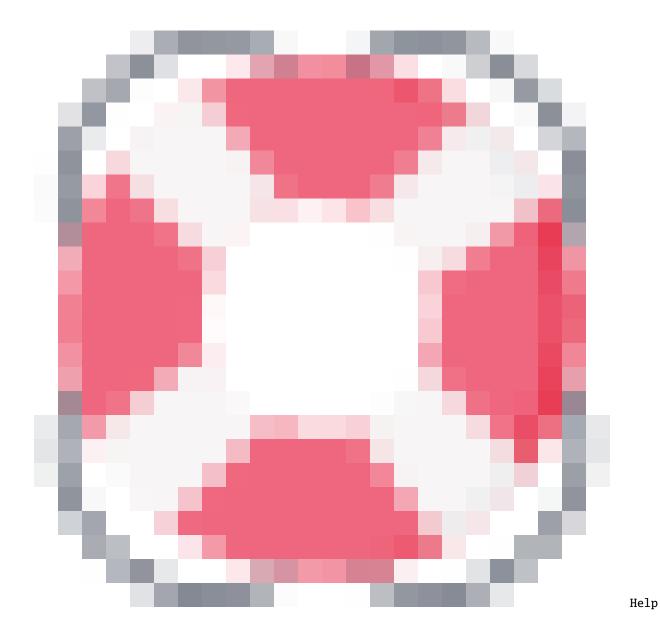
THIRTYFOUR

NETWORK

Note: To run this app type this in a terminal:
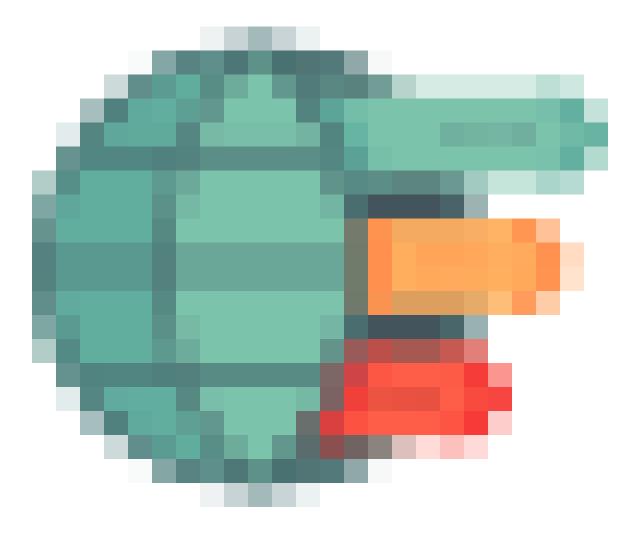
```
openplotter-network
```

Help

opens an offline copy of this documentation in a browser and
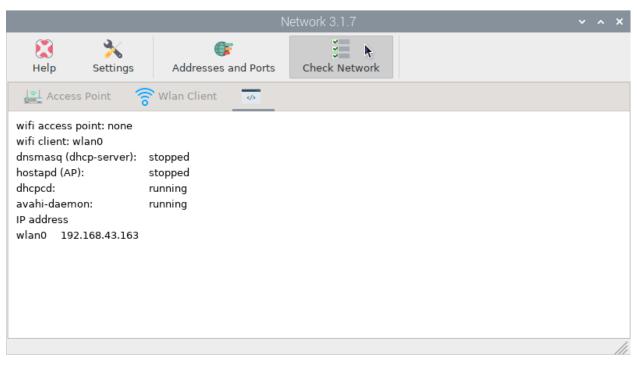`Settings` opens the main app *OpenPlotter Settings*.

The purpose of this app is to provide useful information about your network and create an access point (AP) that other devices can connect to in order to share data.

`Addresses and Ports` is a really useful tool. It will inform you about all the ports that are being used in your system by different applications and the existence of possible conflicts between them:
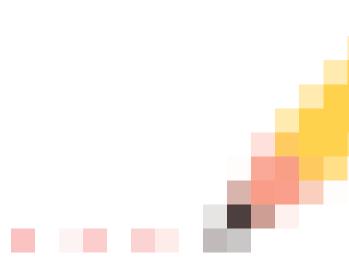
Check `Network` is another useful tool to know the status of your network at any time:
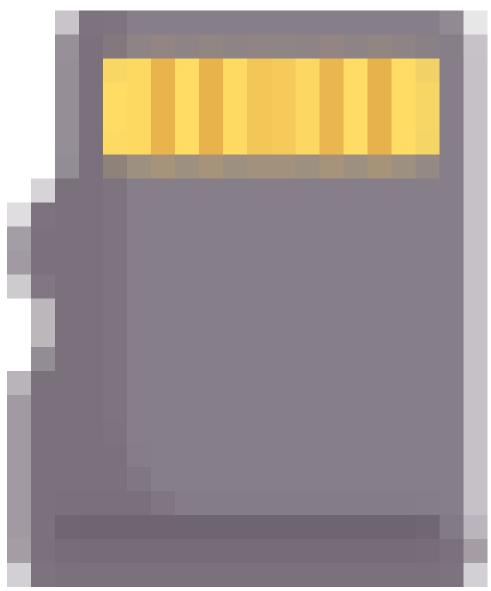
34.1 Security

If you are connected to the marina's public WiFi and that access point is not properly configured or secured, any device connected to the same access point will access your system's open resources like the Signal K server, some data streams, dashboards, etc.

To avoid this, check `activate wlan security (nft filter)`, click

`Edit` and finally
`Save`. After reboot, any device trying to connect to your system will be blocked:

If you trust your marina's public WiFi or you are also connected to a local network using a second WiFi device, such as a USB dongle, you can grant access to your system only to devices connected to that trusted access point. Go to the

Wlan Client tab and add the trusted access point to the *Private Network* by clicking +:
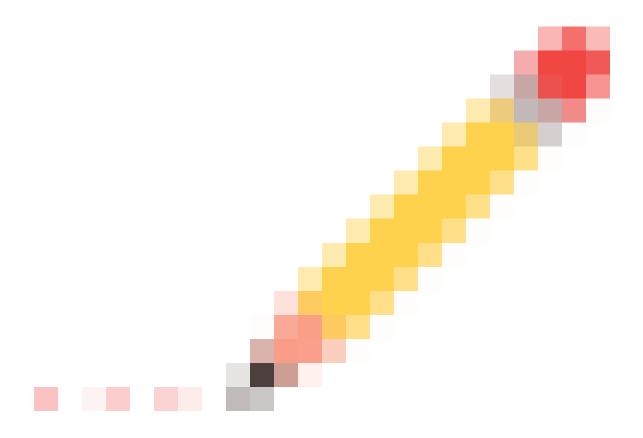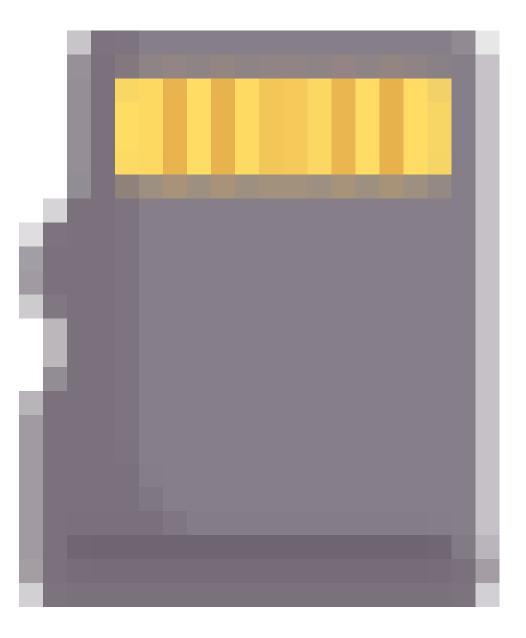
ACCESS POINT MODES

35.1 Client (on board)

If you do not want to set up an access point and just want to connect your system to the marina's public WiFi as a client using your built-in WiFi (on board), you will use the tool provided by your system to make the connection as usual and will set AP to *none*. Enable `activate wlan security (nft filter)` so that no device connected to the same public access point can connect to your system. Click
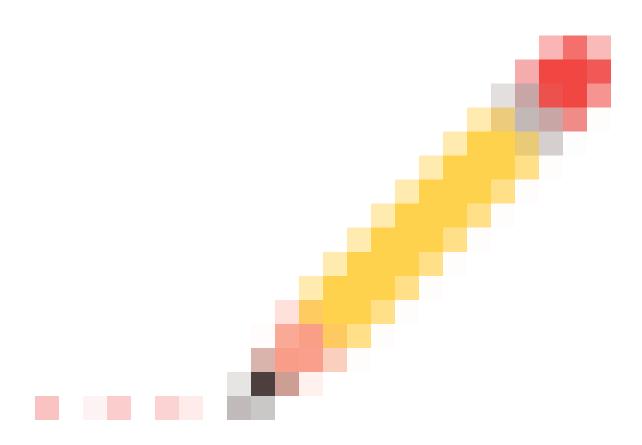
Edit

and finally
Save:

There is an interesting variation of this method to have an Internet connection and at the same time create a local network and be able to connect your onboard devices to OpenPlotter. Create an AP in your smartphone and connect all your devices to it, including OpenPlotter. Disable `activate wlan security (nft filter)` so that any device connected to the same AP can connect to OpenPlotter using the address *openplotter.local* to access to your system's open resources like the Signal K server: *http://openplotter.local:3000*

35.2 AP (on board)

Instead of connecting to an external access point, you can use the built-in WiFi (on board) to create your own access point and connect any onboard device such as tablets, smartphones, laptops…

This time you do not need to enable `activate wlan security (nft filter)`. Select *on board* in `AP` field. By default the AP works at 2.4 GHz but if the built-in WiFi device and the external devices can work at 5 GHz you can check the `5 GHz` field. On your router at home, you can communicate on the same subnet with your Ethernet-connected devices and your WiFi-connected devices. If you want OpenPlotter to treat the AP and Ethernet port this way, check `Add Ethernet port to the AP`. Now you are ready to configure the settings of your AP, click
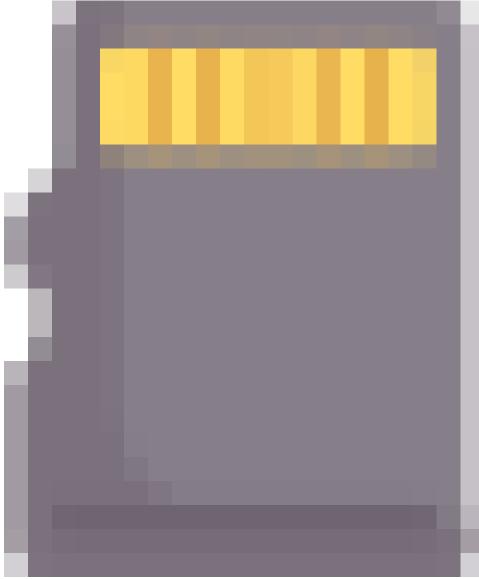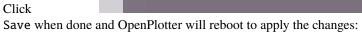
`Edit` to
enable the next window.

You are not connected to Internet, so uncheck the `share internet` field. You have to give to your AP a SSID (a name). This name will be listed when your external devices search for a WiFi AP. Give your wlan a secure `Password`. You can choose a channel you like. Not all selectable channels will work depending on the country setup of your system. If you are having network problems, you can sometimes solve them by changing the channel to a less crowded one.

Click
Save when done and OpenPlotter will reboot to apply the changes:

After reboot, you will see the SSID listed as an available AP by searching from your onboard WiFi devices, confirm you can connect them to your new AP. The built-in WiFi device is now busy working as an AP, so it will not be able to connect to the marina AP as you can see in the image below:
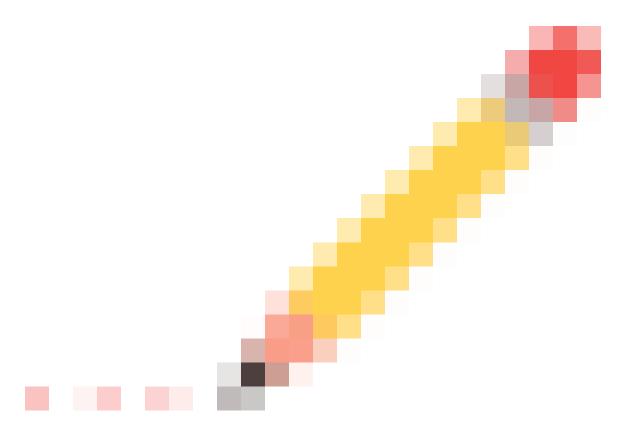

Use the address *openplotter.local* or the IP *10.10.10.1* to access to your system's open resources like the Signal K server: *http://openplotter.local:3000*

35.3 AP + Client (on board)
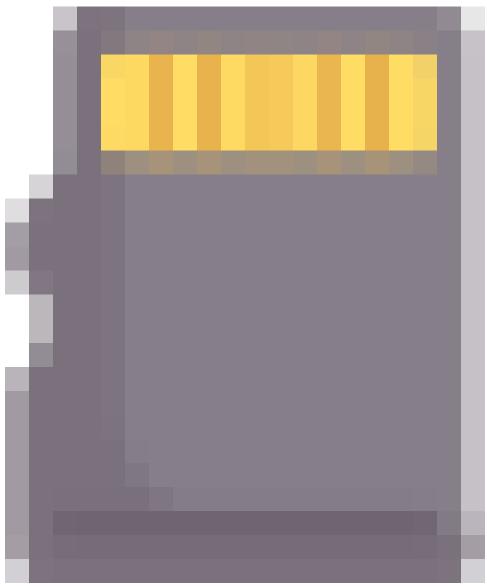
Raspberry Pi models 3 and 4 can use the built-in WiFi device (on board) as an AP and a station at the same time. In this way the devices onboard can connect to OpenPlotter and OpenPlotter can connect to the marina's public WiFi and share its internet connection with the rest of the devices. This is good for saving power, but it is less reliable and slower.

Enable `activate wlan security (nft filter)` so that no device connected to the same public access point can connect to your system. Select *AP and Station* in AP field. Check `5 GHz` and `Add Ethernet port to the AP` if you need it and click

`Edit`.

Check the `share internet` field and fill in the rest of the settings of your AP as usual. Click

Save when done and OpenPlotter will reboot to apply the changes:

After reboot, you will see the SSID listed as an available AP by searching from your onboard WiFi devices, but this time you can also connect to the marina's AP and share internet with your onboard devices:

Use the address *openplotter.local* or the IP *10.10.10.1* to access to your system's open resources like the Signal K server: *http://openplotter.local:3000*

35.4 AP + Client (on board + usb)

This is the most reliable and effective method of creating a local network to connect all devices onboard using the built-in WiFi (on board) and also connect to the marina's public WiFi using an external WiFi device (usb). With this method, you will be able to connect to weak WiFi signals using external WiFi devices with long range antennas.

Enable `activate wlan security (nft filter)` so that no device connected to the same public access point can connect to your system. Select *on board* in AP field. Check `5 GHz` and `Add Ethernet port to the AP` if you need it and click

Edit. Check the `share internet` field and fill in the rest of the settings of your AP as usual. Click

Save

when done and OpenPlotter will reboot to apply the changes.

After reboot, you will see the SSID listed as an available AP by searching from your onboard WiFi devices, and you can also connect to 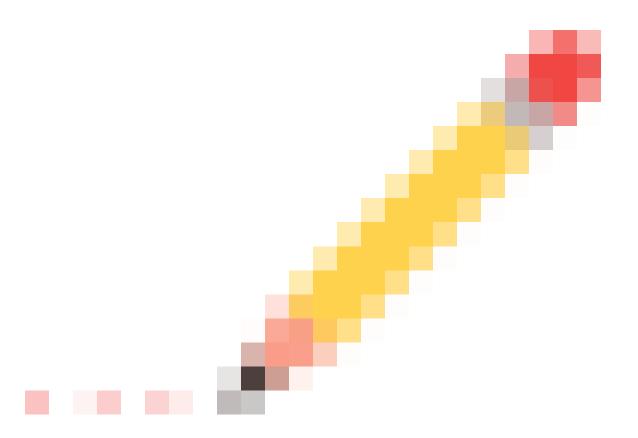the marina's AP and share internet with your onboard devices. Since we are using a second WiFi device, this time we will be able to see our own AP in the list:
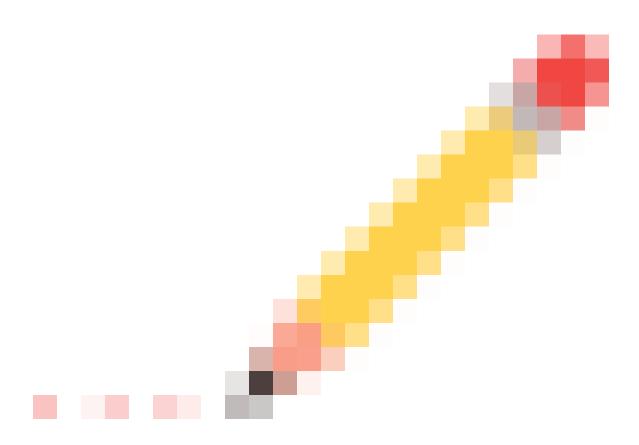
Use the address *openplotter.local* or the IP *10.10.10.1* to access to your system's open resources like the Signal K server: *http://openplotter.local:3000*
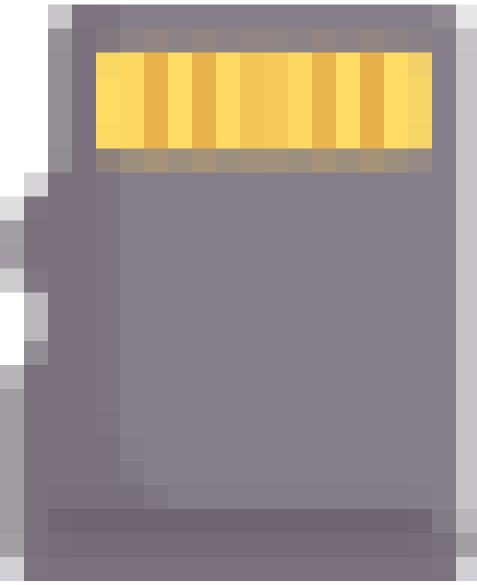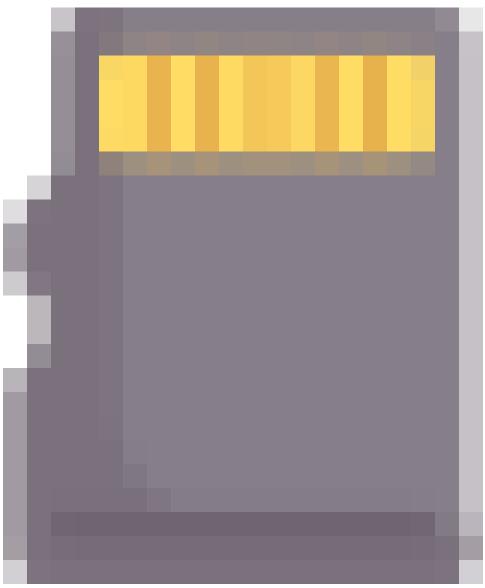
35.5 AP + Client (on board + tethering)

This is the most versatile method and allows you to create a local network, connect to the marina's public WiFi, and also connect to the internet using your mobile data when the marina's WiFi is not available.

Enable `activate wlan security (nft filter)`. Select *on board* in AP field. Check `5 GHz` and `Add Ethernet port to the AP` if you need it and click

Edit. Check the `share internet` field and fill in the rest of the settings of your AP as usual. Click

Save

when done and OpenPlotter will reboot to apply the changes.

Connect your smartphone to OpenPlotter by a data+power USB cable. Enable *share internet connection by USB* in your smartphone (tethering). Connect your smartphone to the marina's public WiFi or use your mobile data to provide Internet to OpenPlotter.

Use the address *openplotter.local* or the IP *10.10.10.1* to access to your system's open resources like the Signal K server: *http://openplotter.local:3000*

PYPILOT

Note: To run this app type this in a terminal:

```
openplotter-pypilot
```

Help

opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

This app is used to integrate the open autopilot, *Pypilot*, into OpenPlotter. You can use Pypilot to install a full autopilot on your boat or simply add an Inertial Measurement Unit (IMU) for heading, heel and trim data. Read on to find out what you need for each of the modes.

Important: Pypilot is a complex environment and you will find many tutorials with instructions for installing programs, editing configuration files or executing commands in the terminal. You do not need to do any of this in OpenPlotter because you can control Pypilot completely from its graphical interface.

The first time you run the
OpenPlotter Pypilot app, you 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`:

Finally go back to the

OpenPlotter Pypilot app and click `Refresh`.

IMU ONLY MODE
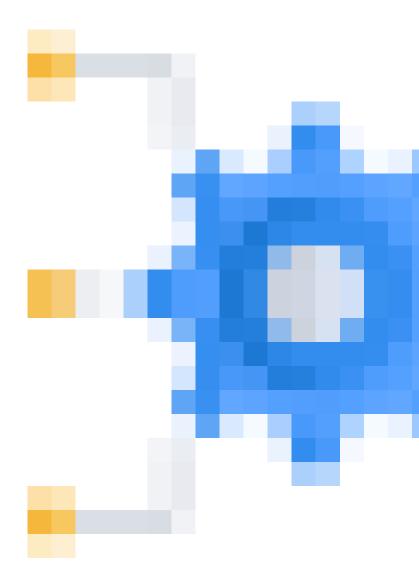
Enable this mode in the
Services tab if you are not using a motor controller and are only interested in adding an Inertial Measurement Unit
(IMU) for heading, heel and trim data. Pypilot currently supports the following I2C-connected IMU models:

- ICM-20948 (recommended)

- MPU-9250

- MPU-9255.
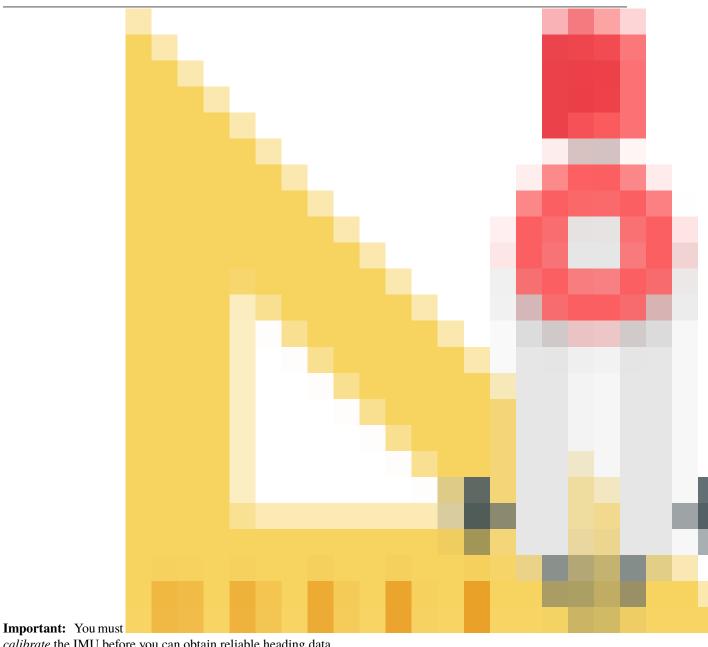
The IMU should be automatically detected and displayed in the *Services* tab:

You do not need to do anything else to start receiving heading and attitude data on the Signal K server:

Important: You must *calibrate* the IMU before you can obtain reliable heading data.

AUTOPILOT MODE

Enable this mode in the
Services tab **only** if you are using a Pypilot motor controller:

The motor controller can be purchased from the official Pypilot store.

You also need one of the IMUs recommended in the previous chapter. Do not forget to *calibrate*.

As a motor you can use the one from your old autopilot if it still works or an industrial equivalent like these replacements for the Raymarine Q047, the EV-100/SPX-5 or the ST4000+. But you can also use other types of motors that are not normally used in marine environments.

Finally you might also need the Pypilot HAT to control the autopilot although this element is optional because you have other ways to control it as we will see later. The Pypilot HAT can be purchased from the official Pypilot store.
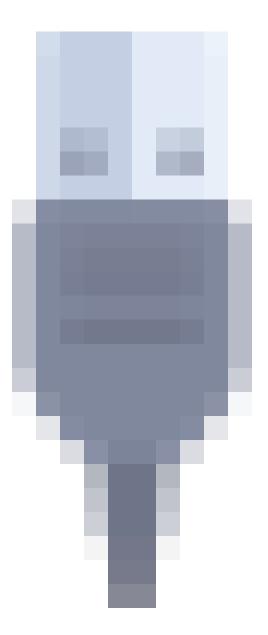
38.1 Connecting the motor controller

On one side you have to connect the controller to the motor and on the other side to one of the UART ports of the Raspberry Pi following this scheme:
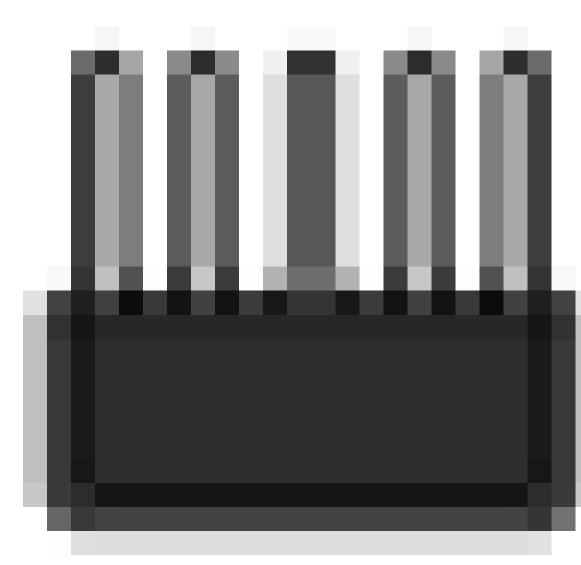
Pypilot motor controller	Raspberry Pi
Vcc	3V3
RxD	UART TX
TxD	UART RX
GND	GND

You can use any of the available UART interfaces on your Raspberry Pi model. See the *UART* chapter to find out which one to choose.

In the following example we are going to use the UART0 interface which is the only one available for both the Raspberry Pi 3 and 4 models. Go to the

Open-
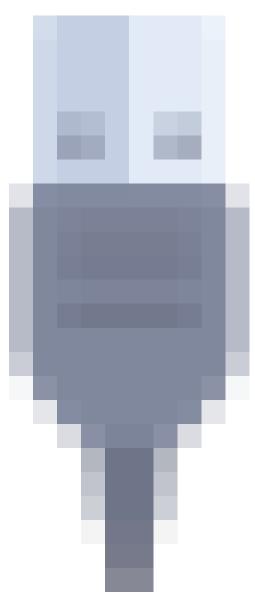
Plotter Serial app and click the
UART0 button, acknowledge the warning and reboot. After the reboot, launch the
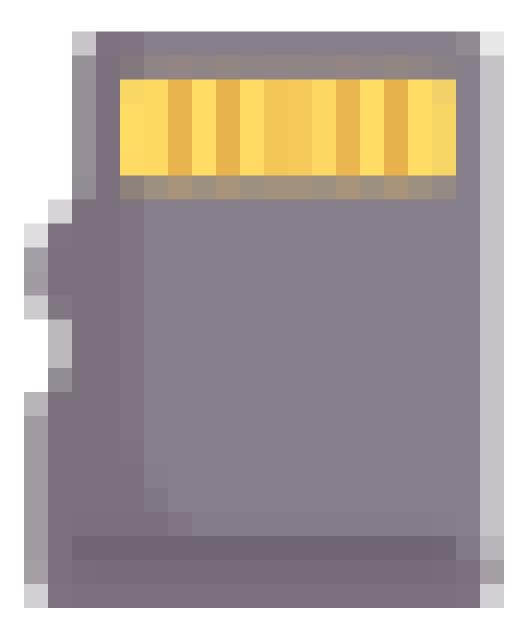
Open-

Plotter Serial app again. On the
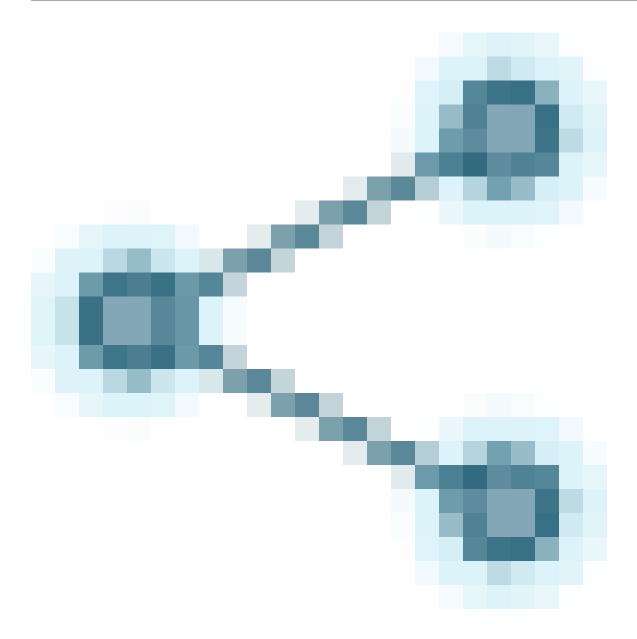
`Devices` tab, you should now see a new entry. Select the line with *ttyAMAx*, give it an *alias* (for example *pypilot*) and select *NMEA 0183* from the *data* dropdown, then press

Apply:
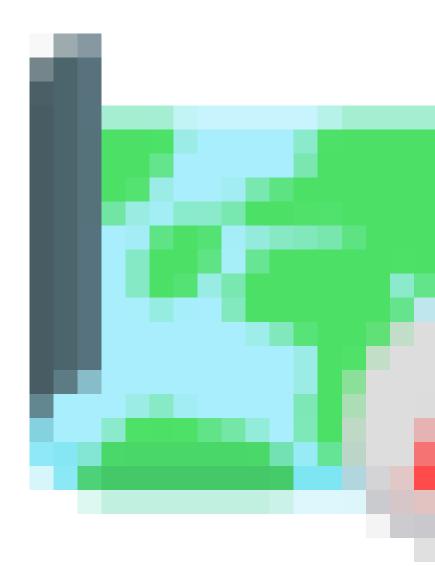
Serial 3.2.0

| Help | Settings | UART0 | UART2 | UART3 | UART4 | UART5 | Refresh |

Devices   Connections

	USB port	device /dev/	alias /dev/	vendor	product	serial	remember
	fe201000.serial	ttyAMA0					

Apply

Remove

	alias	data
/dev/ttyOP_	pypilot	NMEA 0183 ▼

○ Remember device (by vendor, product, serial)

◉ Remember port (positon on the USB-hub)

Finally we need to connect the ttyOP_pypilot device to Pypilot. Switch to the

Connections tab, select the *ttyOP_pypilot* device and click
Add to Pypilot:

In the next window click `AUTO` and that is it. The motor controller will be ready to receive orders from Pypilot.

38.2 Feeding data to autopilot

Pypilot receives data from the IMU and forwards it to the Signal K server automatically when we steer in *compass mode* using the magnetic heading as reference.

We can also steer in *GPS mode*, using the course over ground as reference, or in *wind mode* and *true wind mode*, keeping the apparent wind angle or the true wind angle respectively. In these cases, Pypilot will get the necessary data from the Signal K server automatically when these devices are defined there. If we want to decrease the latency of this data we can also add these devices directly to Pypilot us-

ing the

OpenPlotter Serial app in the same way that we have added the motor controller. Pypilot will take care of sending the data from these devices also to the Signal K server so that other programs can use them.
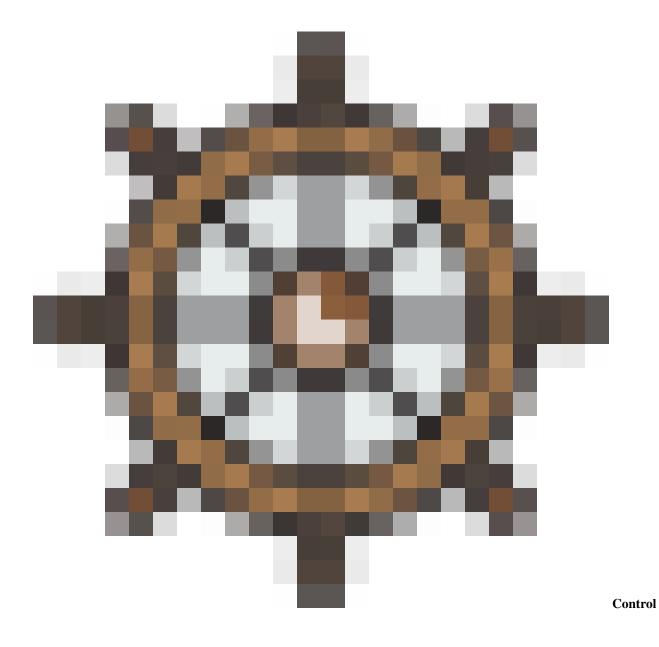
38.3 Autopilot control
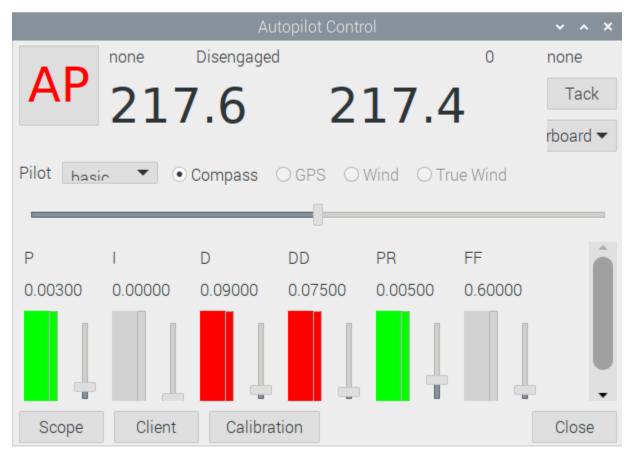
Once we have connected the hardware and we are feeding Pypilot with the necessary data, we can start to control our autopilot using any of the available interfaces. All interfaces can be used in parallel and have all the options you would normally find on any autopilot.
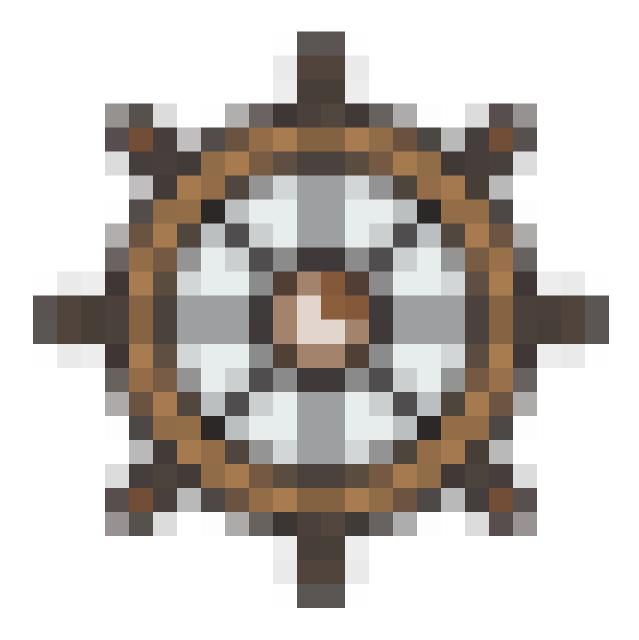
Only one of them is physical and the rest are virtual. The virtual control interfaces can be used from the same device running OpenPlotter and two of them can also be used remotely from other devices, such as a mobile phone connected to the same network as OpenPlotter. The physical control interface, the Pypilot HAT, can also be used remotely via an infrared remote control.

All control interfaces share the same options and parameters that we will briefly explain later.

Control
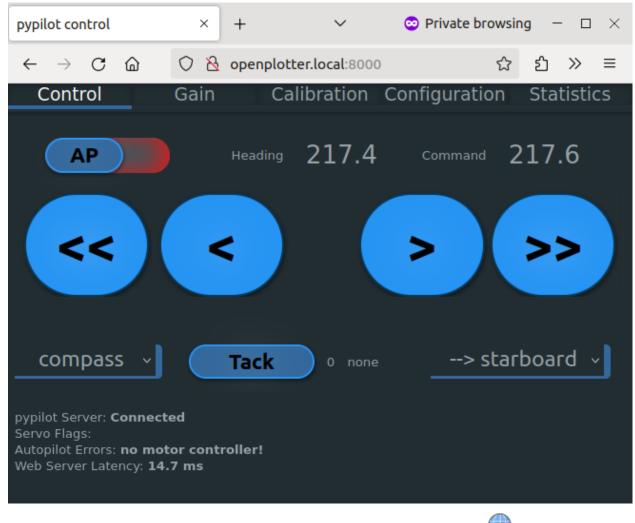
You can access this virtual interface only from the system running OpenPlotter by clicking the

`Control` icon in the
OpenPlotter Pypilot app.
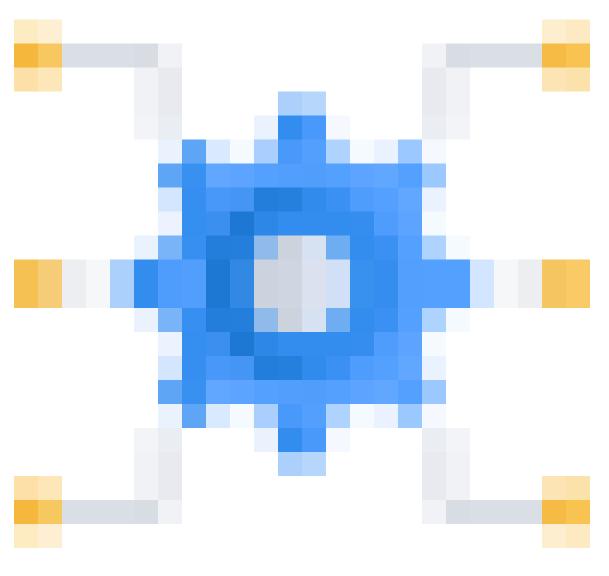
 Web Control

You can access this virtual interface from the system running OpenPlotter by clicking the `Web Control` icon in

the
OpenPlotter Pypilot app or by typing the address *https://localhost:8000* in the browser. Before you can use this interface you need to activate it in the

Ser-
vices tab, checking the `Enable Web Control` option.

You can also use this interface remotely from any device connected to the same network as your OpenPlotter system by typing the address *https://openplotter.local:8000* in the browser.
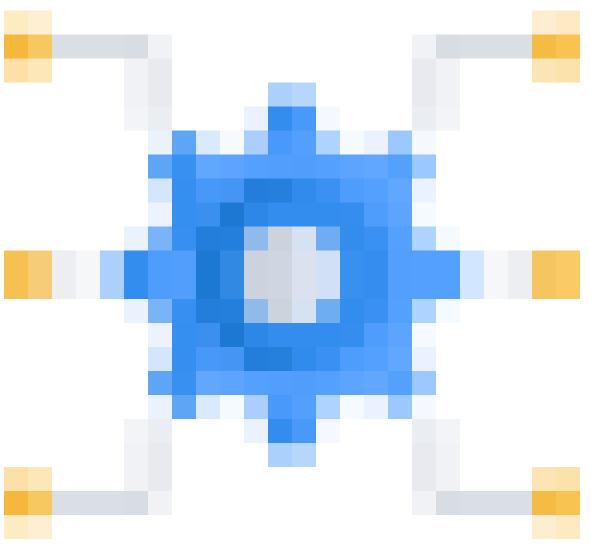
 OpenCPN plugin

You can access this virtual interface from the system running OpenPlotter by installing and enabling the *pypilot* plugin in OpenCPN. Before you can use this interface, you must connect pypilot by clicking `Config` and supplying the *host*. The *host* is *localhost* when your pypilot instance runs on the same system as OpenCPN or *openplotter.local* when your pypilot instance runs on another system on the same network.

Pypilot HAT

This is the physical interface of Pypilot. It basically consists of a small LCD digital screen, a small keyboard and an infrared sensor for a remote control. Before you can use this interface you need to activate it in the

Services

tab, checking the `Enable HAT Control` option.

Pypilot should automatically detect the HAT. You can modify some parameters by clicking the `Configure` button or by typing the address *https://localhost:33333* in the browser. Note that the web interface at port 33333 has a startup delay of 30 seconds.

38.4 Parameters

These are some of the parameters that you will find in all control interfaces:

AP

The AP button is the equivalent of the physical button panel known from boat autopilots. When AP is green, the autopilot is on and when the AP button is red, the autopilot is in standby mode.

Heading

[Heading - Command] On all control interfaces you will see two numeric values from 0 to 360 expressed in degrees. One of them is the *current heading*, which can be calculated differently depending

---

on the selected mode as we will see in the next point, and the other is the *command heading*, which is the desired heading. When *AP* is on and green, pypilot will steer until the *current heading* is similar to the *command heading*.

<< < > >>

[-10 -1 1 10] When *AP* is in standby mode you will see four arrow buttons which can be used to align your actuator with your rudder so you can engage it and if you are lazy you can also steer the boat with them. When AP is enabled, these buttons will have numerical values -10, -1, 1 and 10 to add those amounts in degrees to the *command heading*.

Mode

[*Compass* - *GPS* - *Wind* - *True Wind*] In *Compass* mode the *current heading* is based on magnetic north, in *GPS* mode the *current heading* is based on true north, in *Wind* mode the *current heading* is the direction the wind comes from relative to the bow and in *True Wind* mode the *current heading* is the direction the wind comes from relative to the true north. You will find a more detailed explanation in the Pypilot functions chapter of the workbook.

Tack

[starboard - board] Sailing with a limited crew sometimes requires a system to automate tacking. Clicking the *Tack* button initiates the maneuver to port or starboard using the predefined settings:

- **ap.tack.angle**: In wind mode it is automatically determined from current course. Unit is degrees.

- **ap.tack.delay**: How many seconds to wait to tack after hitting tack button.

- **ap.tack.rate**: How quickly to tack. Unit is degrees/sec.

- **ap.tack.threshold**: When to revert back to normal filter. Unit is percentage. Typically half the tack angle but adjusting this would be useful to prevent overshoot.

Gain

[P - I - D - DD - PR - FF] Gains of the steering mechanism determine reactiveness of the steering and suppression of overshoot. The values of these gains are different for each boat, so they need to be set properly. To get started retuning from scratch (or on a new boat) set all of the gains to zero, except the P and D gains. It is possible to have a fully usable (but less efficient) autopilot using only these two gains. You will find a more detailed explanation in the official docs.

Calibration

The compass (IMU) must be *calibrated*. If you have a rudder feedback sensor, with analog input to the motor controller, it must also be calibrated. The rudder range should first be set to the maximum angle the autopilot is allowed to move the rudder. The angle is needed to provide accurate rudder feedback measurements. For example the opencpn dashboard display. Next, manually turn the rudder for, centered, starboard range, port range and press the button each time.

Settings

In addition to the tack settings there are many other parameters that can be modified. You can access these settings from the *Configuration* option of the control interfaces or from within the *Calibration* option. You will find a more detailed explanation in the parameters chapter of the workbook.

Client

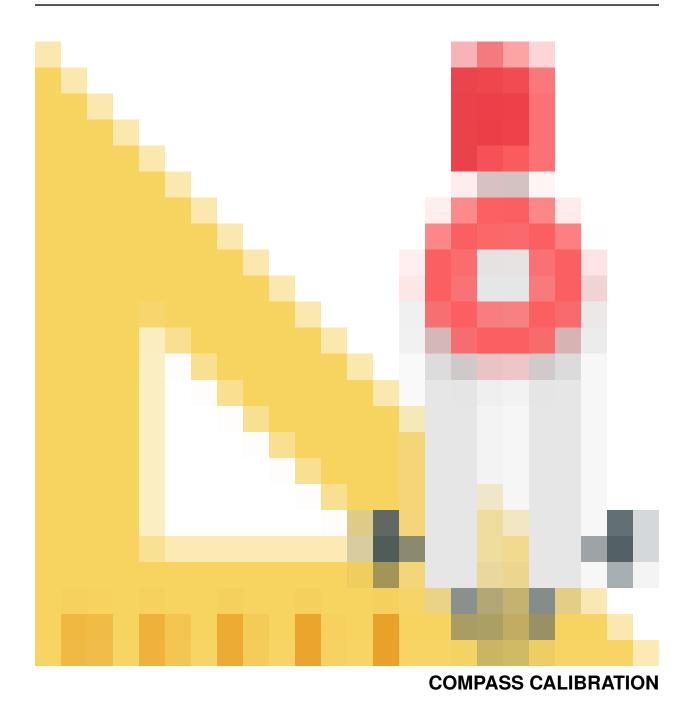In some control interfaces you will find a *Client* button to access a list of parameters with their current values to know what is happening in your system at any time. In this list you can also modify some values.

Scope

In some control interfaces you will also find a *Scope* button to access a graphical representation of all Pypilot parameters and their current values.

COMPASS CALIBRATION

Follow these steps in order:
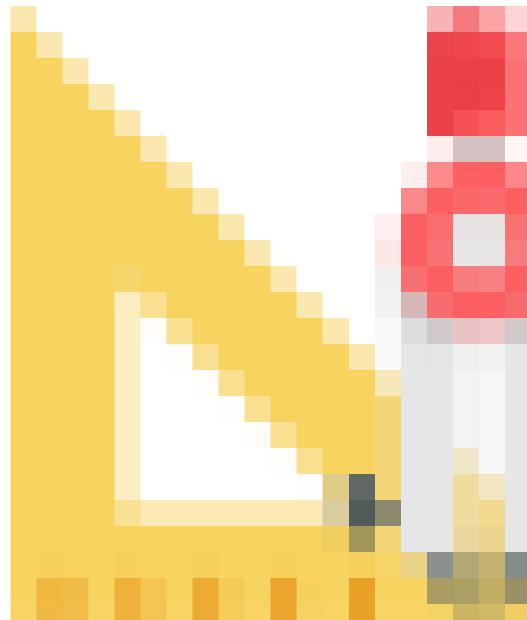
39.1 1. Accelerometer bias

IMUs require accelerometer bias calibration. Without it, there will be significant pitch and roll errors. Most of them are factory calibrated, which means you could skip this step, but it is recommended to calibrate the accelerometer bias, even if it is factory calibrated, as it will slightly improve the factory calibration.

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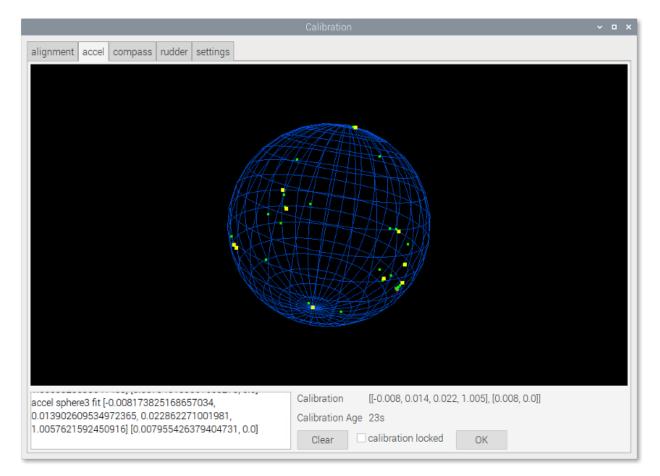
OpenPlotter Pypilot app and click on
`Calibration`. In Calibration window click on `accel` tab. Make sure *calibration locked* is not enabled.

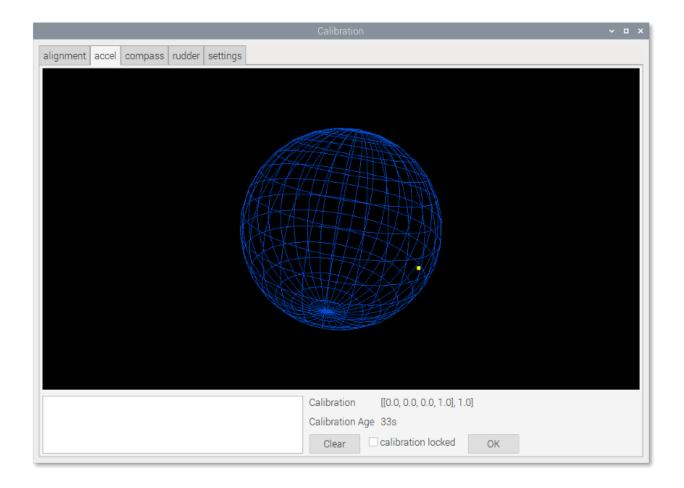Carefully place the sensor on each of the 6 sides of a box (+- 10 degrees will do) the actual orientation is not critical, so long as enough measurements can be taken to fit a sphere. Leave the sensors in each position for a few seconds.

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.

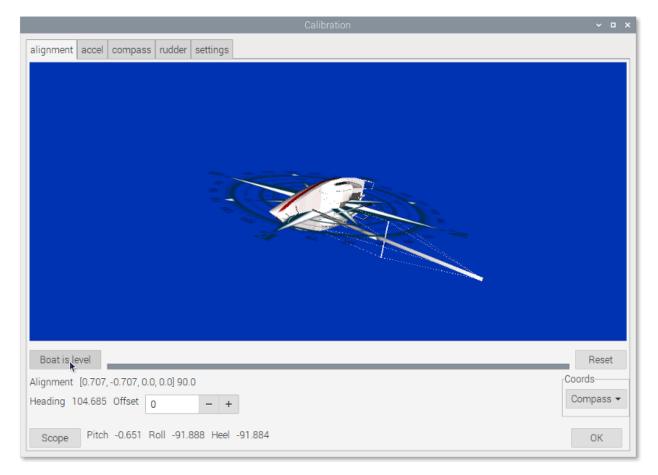If you use the cheapest sensors, sometimes they have bad accelerometers. Either one axis will always read zero, or they will saturate because the bias is greater than 1g. This is easy to determine from the accelerometer calibration plot in calibration window.

39.2 2. Alignment

Once the accelerometer is 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.

39.3 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.
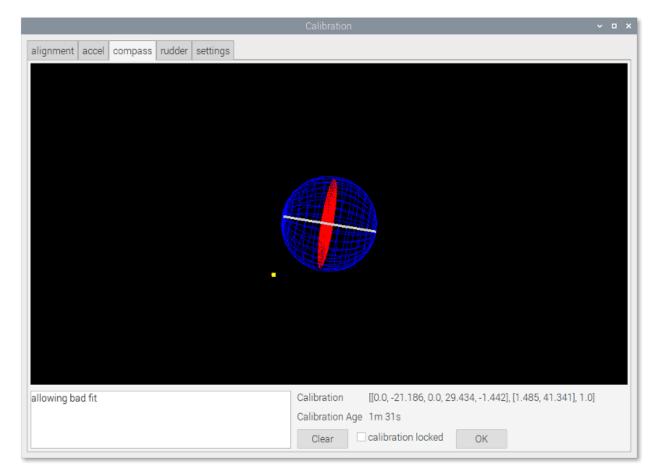
The compass calibration is mostly automatic. If the accelerometer and alignment are calibrated, you just need to sail turning more than 180 degrees to calibrate the compass.

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 want to lock it to prevent future calibration updates when you know there is compass interference, but unlocking it is recommended.

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.
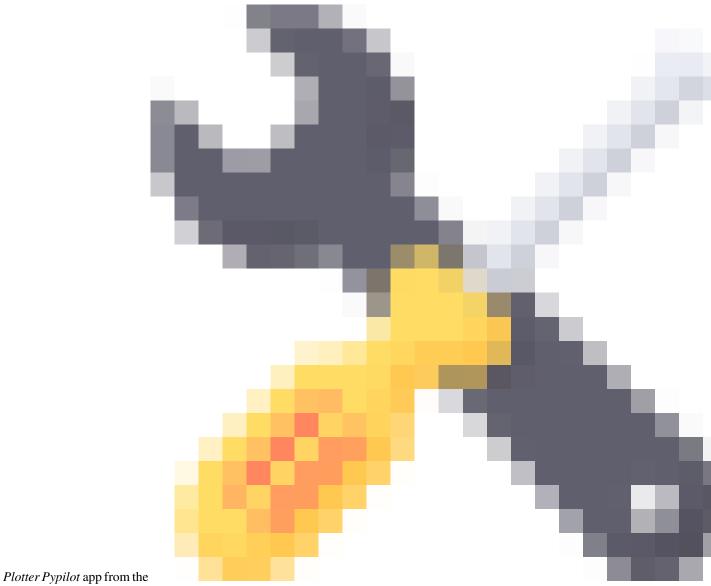
UPDATE PYPILOT CODE

The
OpenPlotter Pypilot app always installs the latest version of Pypilot but it may happen that the Pypilot code is updated later. To update the Pypilot code, just reinstall the

Open-

Plotter Pypilot app from the
OpenPlotter Settings and the installer will perform all necessary tasks keeping the current settings.

MORE INFO

Important: Pypilot is a complex environment and you will find many tutorials with instructions for installing programs, editing configuration files or executing commands in the terminal. You do not need to do any of this in OpenPlotter because you can control Pypilot completely from its graphical interface.

Official site
https://pypilot.org

Official docs
https://pypilot.org/wiki/doku.php

Official store
https://pypilot.org/opencart

Support
https://forum.openmarine.net/forumdisplay.php?fid=17

Danger: This workbook is perhaps the most comprehensive document on Pypilot, but beware! it contains a lot of outdated information about OpenPlotter v2 and some instructions that are no longer needed in OpenPlotter v3 and could make your system unstable.

Workbook
https://github.com/pypilot/workbook/wiki

I2C

Coming soon
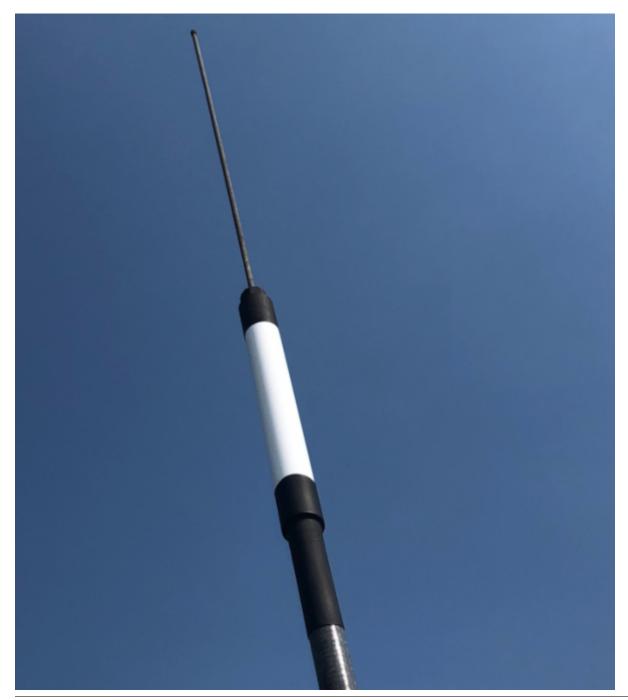
FORTYTHREE

GPIO

Coming soon

FORTYFOUR

IOB

Coming soon

FORTYFIVE

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.
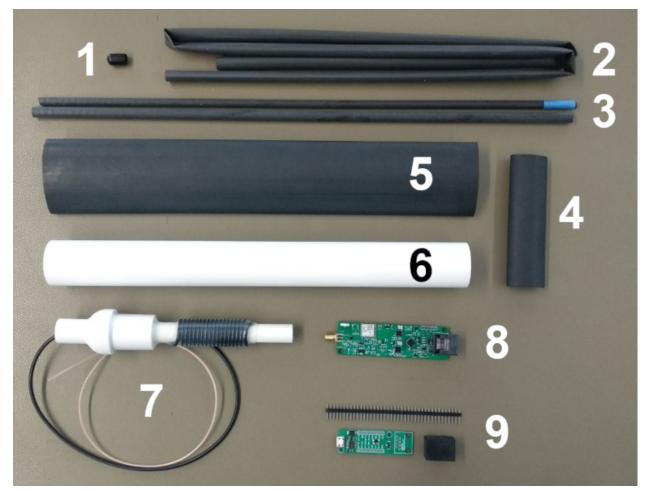
Note: This product is available in the OpenMarine Shop.

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

Disclaimer
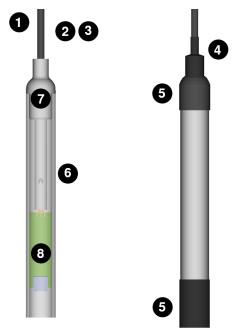
MAIANA™ devices are not finished products, they need to be assembled by end users to function. MAIANA™ devices are distributed for research and development purposes. MAIANA™ devices are delivered with the ability to transmit disabled by default and have not been tested for compliance with regulations 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).

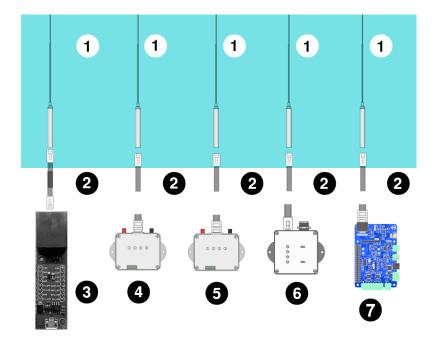Easy assembly. To complete the installation you will need:

- 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.

CONNECTING THE BASE KIT
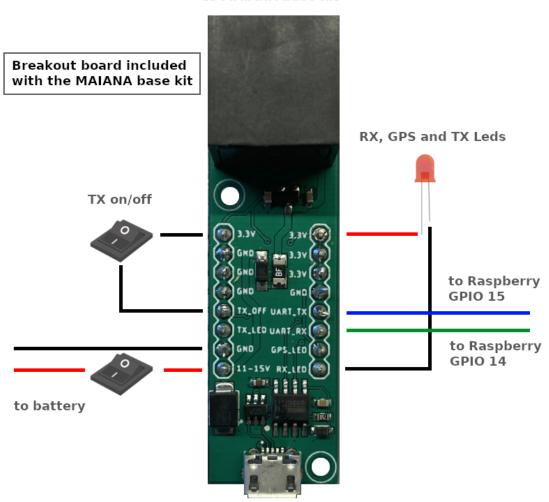

1. MAIANA base kit.

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.

7. MacArthur HAT. 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 5 options: the Breakout board, the USB+UART, USB+RS422 and USB+CAN adapters or the MacArthur HAT.

46.1 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.
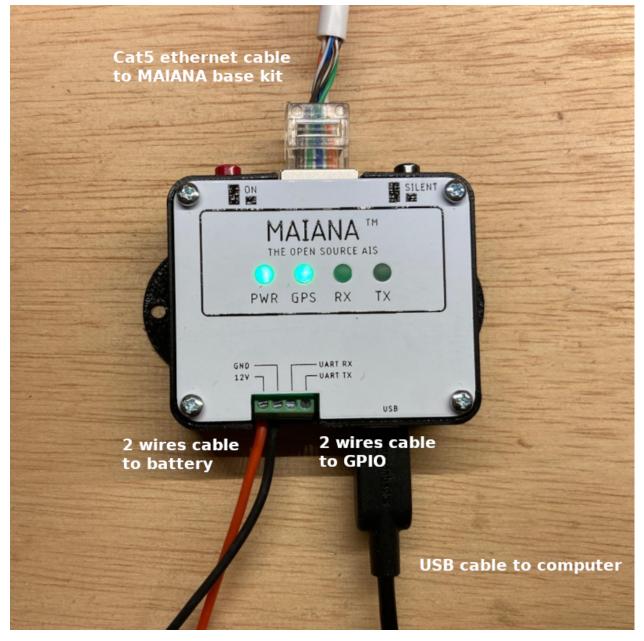
If you want a power and/or TX switch, you can use simple rocker switches. A 1A-rated SPST can simply interrupt the main 12V supply.

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.

46.2 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.
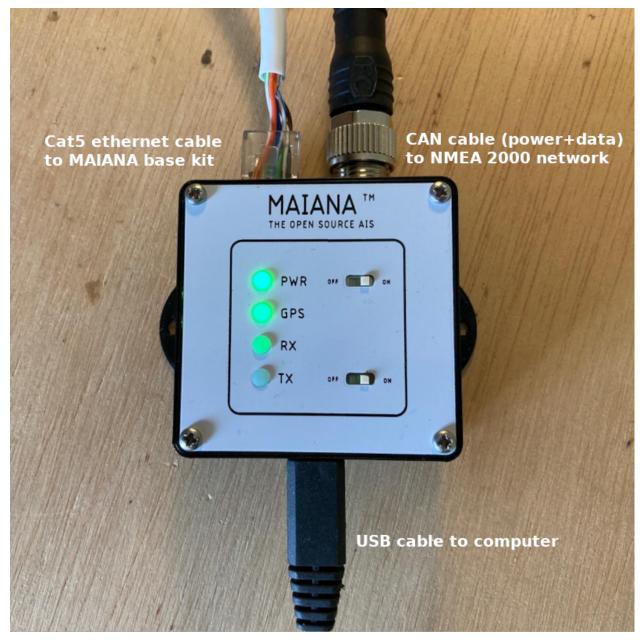
46.3 USB + RS422 Adapter

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.
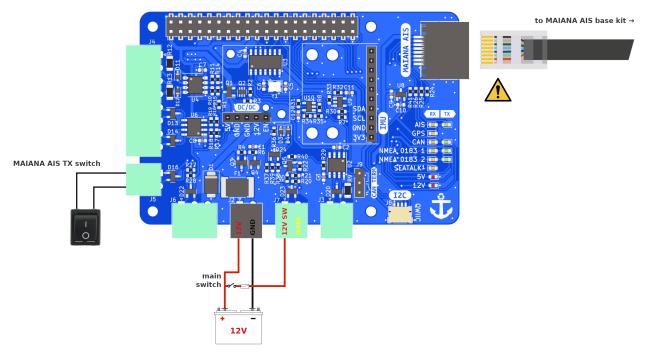
46.4 USB + CAN Adapter

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.

46.5 MacArthur HAT

This HAT has the same features as all MAIANA AIS adapters together. You can receive AIS and GNSS data and configure the MAIANA transponder in OpenPlotter through the UART0 interface. You can also send NMEA 0183, NMEA 2000 or Signal K data to any device on your boat.

Read the "MAIANA AIS" chapter of the MacArthur HAT manual to see all connection options.
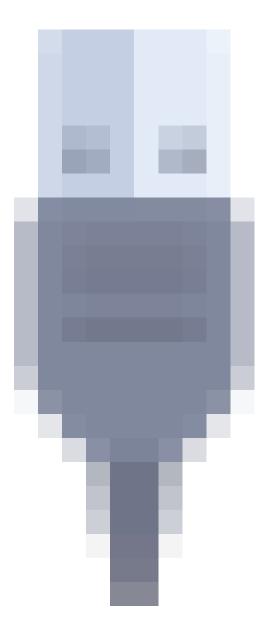
FORTYSEVEN

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.
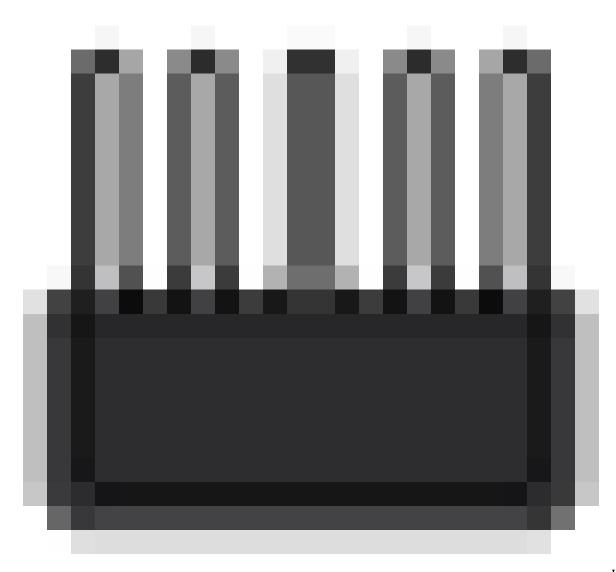
47.1 Getting AIS and GNSS data

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 us-

ing 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
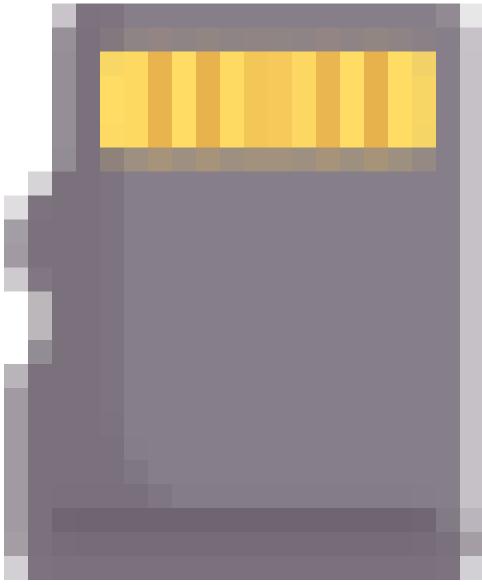
UART0

and then click `Yes`. Remember that enabling the UART interface will disable Bluetooth. If you are connected by USB, skip this step.
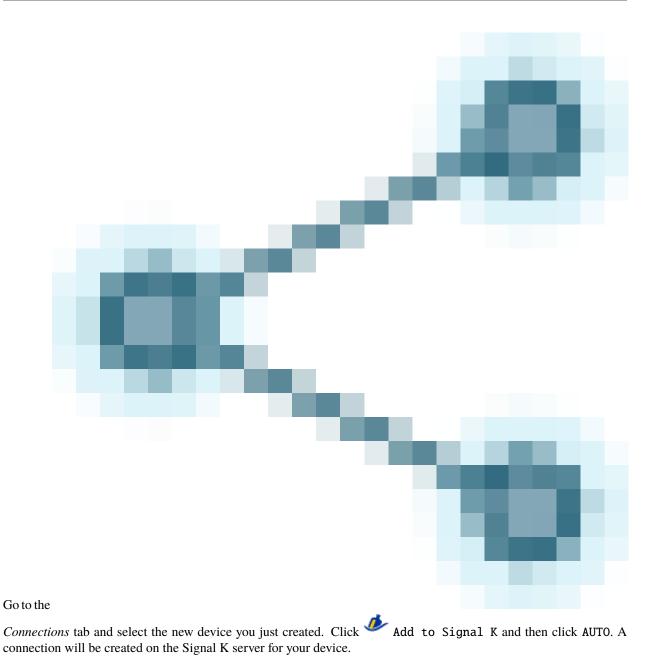
After enabling UART or just plugging in the USB and clicking Refresh, you will see a new device listed. Select this new device and provide a short name for the *alias* and select NMEA 0183 under *data*. If it is connected by USB check *Remember device* and if it is connected by UART check *Remember port*. 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.

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

47.2 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:

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`.
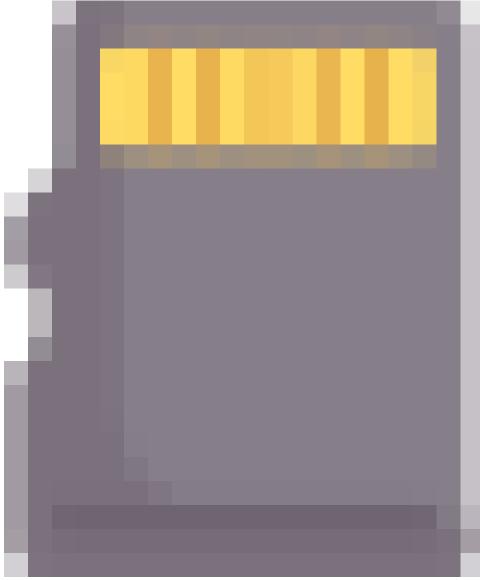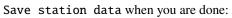
47.3 Enabling transmission

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)
- Vessel name (up to 20 alphanumeric characters, no punctuation. Use all caps)
- Call sign (may be empty if you don't have one)
- Vessel type (this is the numeric type of the vessel, see below)
- LOA - 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 the adapters. The breakout board also has a pin for this. This switch has priority over the Software switch.

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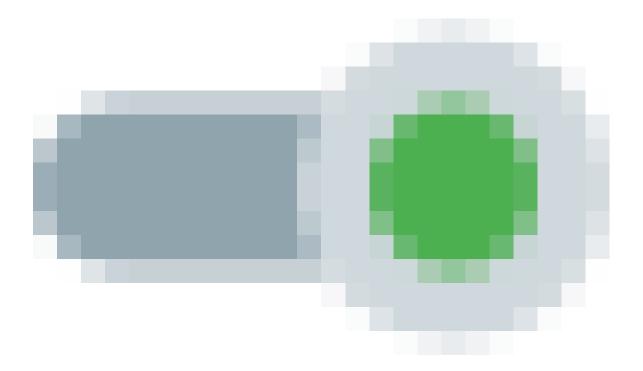
You will find a button *Software TX switch* in *OpenPlotter MAIANA AIS transponder* app.

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!

47.4 Notifications actions

If you have the *OpenPlotter Notifications* app installed, you will see two new actions added to the list to automatically turn the software TX switch on and off upon receiving a specific notification:

47.5 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.
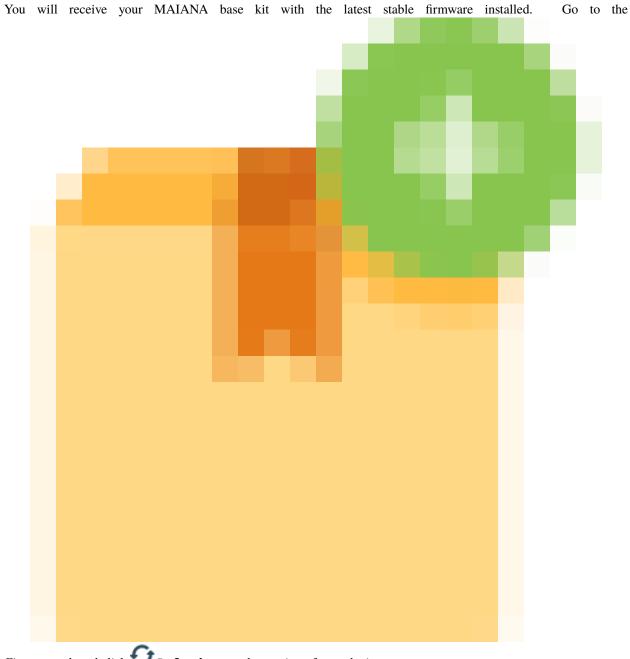
If you have the `OpenPlotter Notifications` app installed, you will see an alert win-

dow like this one:

47.6 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:

Click `Download Firmware` to find the bin file that corresponds to your MCU and hardware revision from the project page:

Ignore the last digit of your hardware revision – it doesn't matter. So if you have board 11.3.0 with an STM32L422 processor, the right binary is maiana-stm32l422-hw11.3-fwXXX.bin where XXX is the latest revision you see here. If you already have this firmware on your board, there is no update:

Once the correct file is downloaded click
`Update firmware` to start the firmware update process:

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:

FORTYEIGHT

NOTIFICATIONS

This app is already stable, docs are coming soon

AVNAV INSTALLER

Note: To run this app type this in a terminal:

```
openplotter-avnav
```

Help

opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

This app installs the web chart plotter AvNav. Enable/Disable the AvNav server at startup by toggling `Autostart`. You

can also

Stop,

`Start` and
`Restart` the AvNav server at any time.

Open an instance of AvNav in your local browser by clicking `Avnav` or click `Avnav split` to run two instances in the same window.

You can also run AvNav from any device connected to the same network as OpenPlotter using this address: *http://openplotter.local:8080*
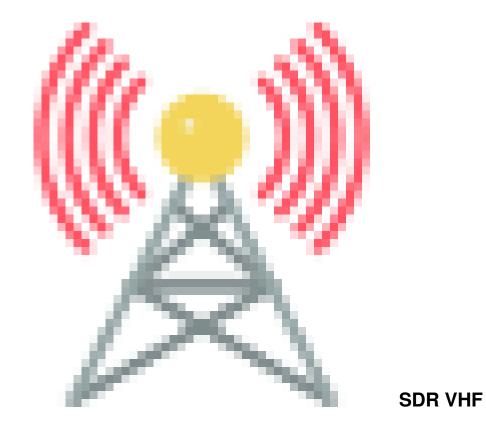
By default, AvNav listens on port 8080, but you can change this, and related ports in the `Settings` tab if the default ports conflict with other programs:

Note: Read here the full documentation.

FIFTY

SDR VHF

Note: To run this app type this in a terminal:

```
openplotter-sdr-vhf
```

Help

opens an offline copy of this documentation in a browser and
`Settings` opens the main app *OpenPlotter Settings*.

This app 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:
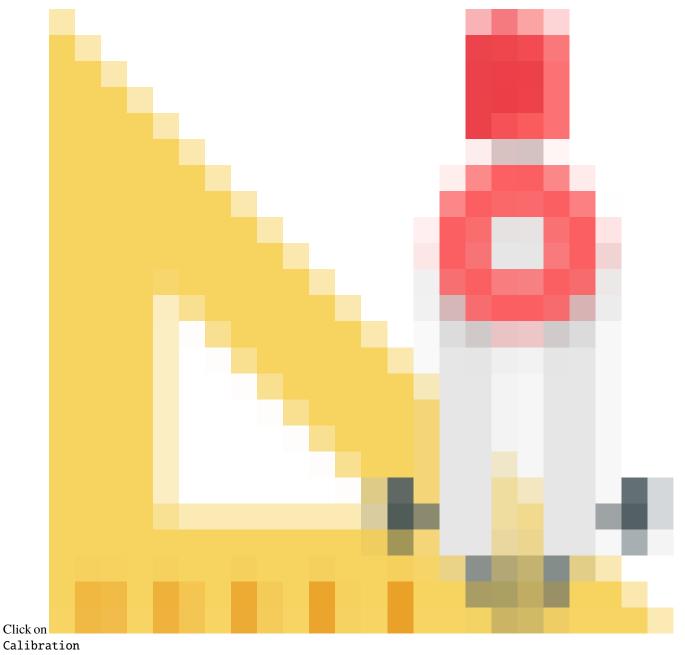
- Use as a police radio scanner.

- 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 this app we include some of these interesting tools for maritime use. Before you start using any of these bundled tools, there are a few steps you should take.

50.1 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.
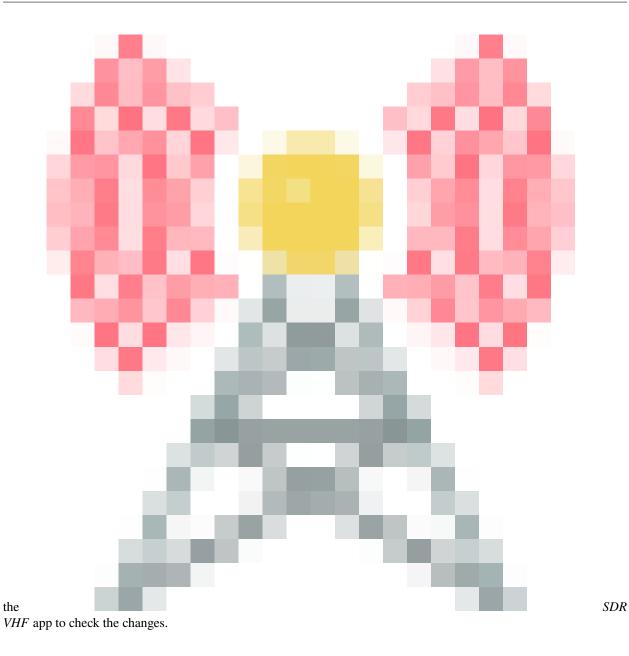
Click on
Calibration

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

the *SDR VHF* app to check the changes.

```
                                    bash                              ∨  ∧  ✕

  File  Edit  Tabs  Help
 Found 2 device(s):
   0:  Generic RTL2832U OEM
   1:  Generic RTL2832U OEM

 Using device 0: Generic RTL2832U OEM
 Detached kernel driver
 Found Rafael Micro R820T tuner

 Current configuration:
 _____
 Vendor ID:            0x0bda
 Product ID:           0x2838
 Manufacturer:         Realtek
 Product:              RTL2838UHIDIR
 Serial number:        00000001
 Serial number enabled: yes
 IR endpoint enabled:   yes
 Remote wakeup enabled: no
 _____

 New configuration:
 _____
 Vendor ID:            0x0bda
 Product ID:           0x2838
 Manufacturer:         Realtek
 Product:              RTL2838UHIDIR
 Serial number:        mySDR
 Serial number enabled: yes
 IR endpoint enabled:   yes
 Remote wakeup enabled: no
 _____
 Write new configuration to device [y/n]? y
```

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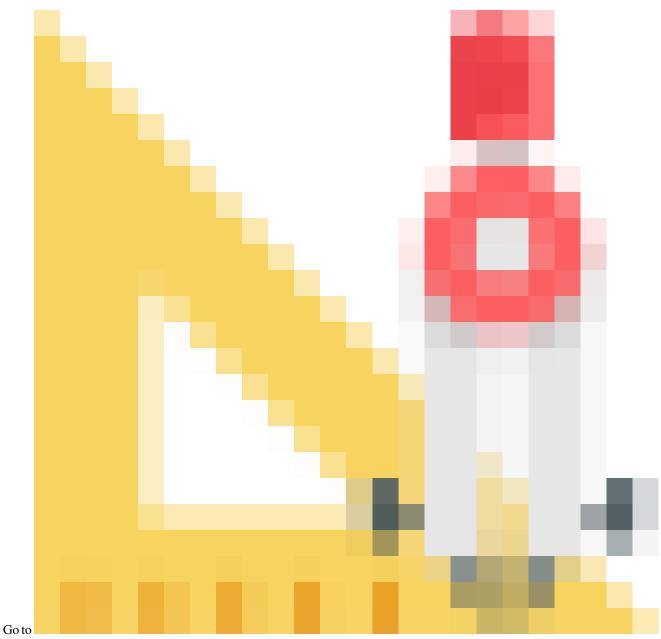
Every SDR device will have a small frequency error as it is cheaply mass produced and not tested for accuracy. This frequency error is linear across the spectrum, and can be adjusted in most SDR programs by entering a **PPM** (parts per million) offset value.

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).

```
                                    rtl_test                              ∨  ∧  ✕
 File  Edit  Tabs  Help
Found 2 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: mySDR
  1:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Detached kernel driver
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7 20.7 22.9 25.4
28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.
Reporting PPM error measurement every 10 seconds...
Press ^C after a few minutes.
Reading samples in async mode...
Allocating 15 zero-copy buffers
lost at least 132 bytes
real sample rate: 2047753 current PPM: -121 cumulative PPM: -121
real sample rate: 2047980 current PPM: -10 cumulative PPM: -65
real sample rate: 2047948 current PPM: -25 cumulative PPM: -52
real sample rate: 2048022 current PPM: 11 cumulative PPM: -36
real sample rate: 2047994 current PPM: -3 cumulative PPM: -29
real sample rate: 2047988 current PPM: -6 cumulative PPM: -25
real sample rate: 2047947 current PPM: -26 cumulative PPM: -25
real sample rate: 2048036 current PPM: 18 cumulative PPM: -20
real sample rate: 2047998 current PPM: -1 cumulative PPM: -18
real sample rate: 2048001 current PPM: 1 cumulative PPM: -16
real sample rate: 2047994 current PPM: -3 cumulative PPM: -15
real sample rate: 2048013 current PPM: 7 cumulative PPM: -13
real sample rate: 2047997 current PPM: -1 cumulative PPM: -12
real sample rate: 2048009 current PPM: 5 cumulative PPM: -11
real sample rate: 2047939 current PPM: -30 cumulative PPM: -12
real sample rate: 2048081 current PPM: 40 cumulative PPM: -9
real sample rate: 2048021 current PPM: 10 cumulative PPM: -8
```

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:

```
                                    bash                          ⌄  ^  ✕

 File  Edit  Tabs  Help
Found 2 device(s):
  0:  Generic RTL2832U OEM
  1:  Generic RTL2832U OEM

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
[R82XX] PLL not locked!
Setting gain: 49.6 dB
kal: Scanning for GSM-900 base stations.
channel detect threshold: 106146.075677
GSM-900:
    chan:     5 (936.0MHz +   0Hz)     power:  116750.60
    chan:     6 (936.2MHz - 380Hz)     power:  229446.70
    chan:    10 (937.0MHz - 254Hz)     power:  465368.43
    chan:    15 (938.0MHz - 693Hz)     power:  111906.26
    chan:    19 (938.8MHz - 349Hz)     power:  151554.81
    chan:    22 (939.4MHz - 678Hz)     power:  243993.92
    chan:    75 (950.0MHz -  75Hz)     power:  192543.66
    chan:   101 (955.2MHz -  21Hz)     power:  178506.18
    chan:   102 (955.4MHz - 378Hz)     power:  133635.92
    chan:   104 (955.8MHz + 613Hz)     power:  158769.13
    chan:   107 (956.4MHz - 637Hz)     power:  225573.48
    chan:   113 (957.6MHz + 437Hz)     power:  196972.16
    chan:   117 (958.4MHz - 246Hz)     power:  192052.90
    chan:   121 (959.2MHz - 105Hz)     power:  175866.60
...chan 124
Difference of offsets between channels is >1kHz. This likely means that the correct PPM is
too far away and you need to provide a rough estimate using the '-e' option. Try tuning aga
inst a local FM radio or other known frequency first.
----------------------------------------------------------------------
Take note of the channel with the highest power value and press Enter to close this window.
----------------------------------------------------------------------
```

Put the strongest channel into *Channel* field and press Get PPM:
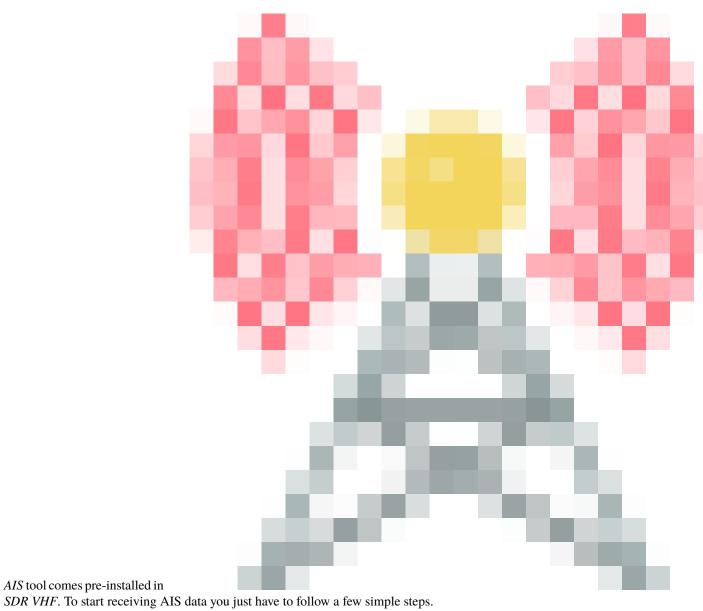
Write down the final PPM value and close the window:

```
                                    bash                     ∨  ∧  ✕
File  Edit  Tabs  Help
        offset   87: 3983.21
        offset   88: 3985.28
        offset   89: 3985.28
        offset   90: 3982.17
        offset   91: 4001.81
        offset   92: 3995.61
        offset   93: 4000.78
        offset   94: 3996.64
        offset   95: 3996.64
        offset   96: 3949.12
        offset   97: 3978.04
        offset   98: 3958.42
        offset   99: 3948.08
        offset  100: 3993.54
average          [min, max]        (range, stddev)
+ 3.980kHz                 [3962, 3997]    (35, 9.809118)
overruns: 0
not found: 0
average absolute error: 60.752 ppm
-----------------------------------------------------------
Take note of the final ppm value rounded to the nearest whole number an
d press Enter to close this window.
-----------------------------------------------------------
```
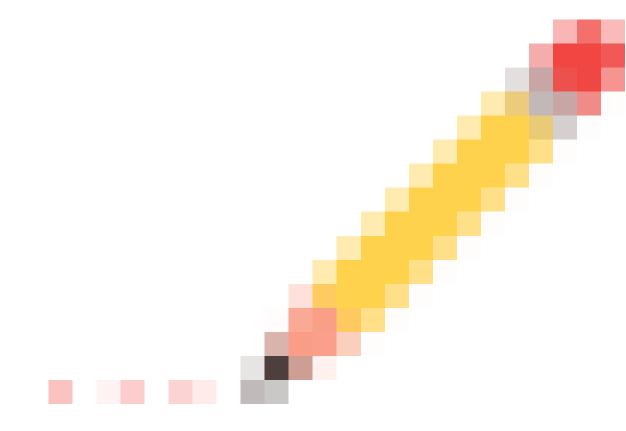
Put the final PPM into the *PPM* field without decimals rounding the value to the nearest integer number, click on
Change and you are done:

FIFTYONE

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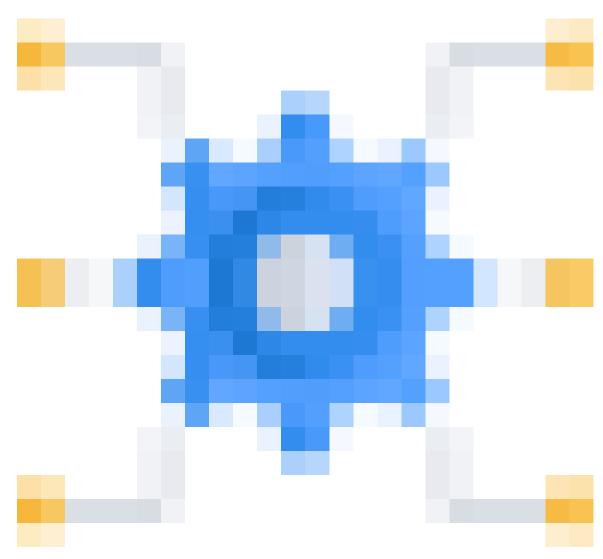
Set the *PPM* value for your device. If you have calibrated your device, you should see this value in the device list. You can also set any value to test.

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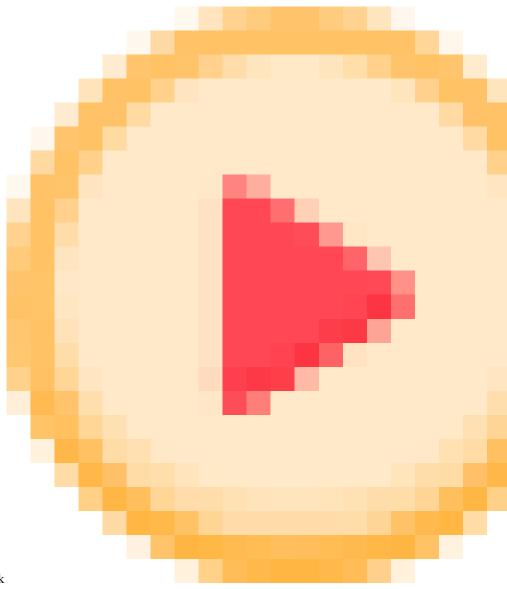
Go to the
Processes tab and check `Autostart` on AIS process to start getting data at system startup:

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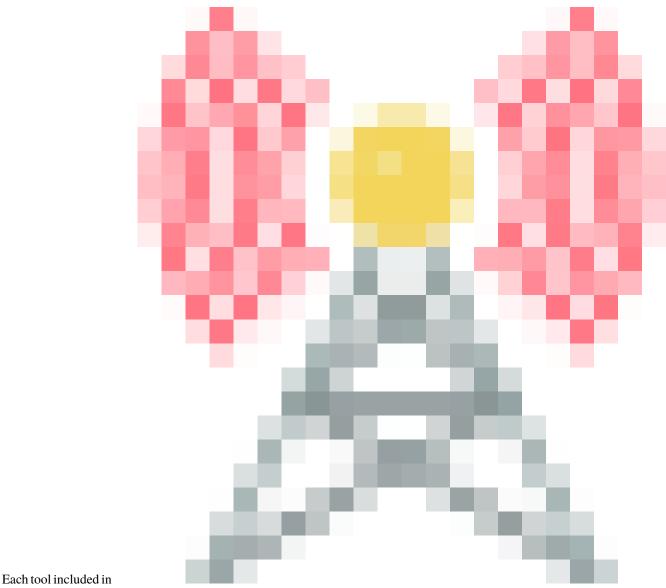
Then, go to OpenCPN and confirm that a connection with the Signal K server exists and is getting AIS data:

51.1 Antennas

Each tool included in
SDR VHF uses a specific frequency range and you will need a specific antenna for each of them.

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.
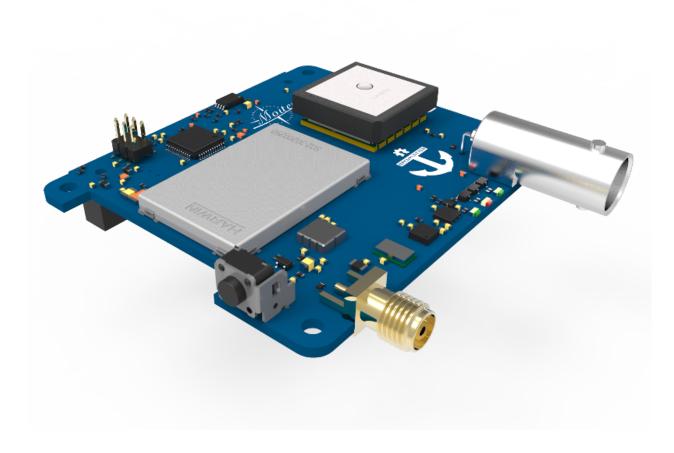
GQRX

DAB

DVB-T

MOITESSIER HAT 2

Important: The Moitessier HAT developers dropped support for the drivers leaving hundreds of users with unusable devices on Raspberry OS bullseye. Despite the promise of transferring manufacturing to interested companies or opening up the sources of hardware and software, this has never happened. At OpenMarine we consider this unacceptable and we promise to never again support closed and proprietary products.

In this document, we offer a way to hack the Moitessier HAT 1 and 2 models without damaging the hardware or software to get rid of the drivers and give it a second life using most of its features in OpenPlotter v3.x.x.

55.1 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.

- 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 .
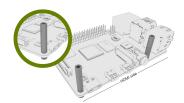
The shutdown button needs the abandoned driver and it will not work when used in "hacked" mode.

55.2 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
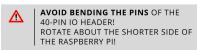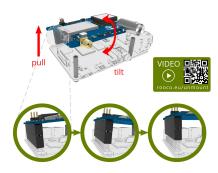Screw Raspberry Pi and HAT together

55.3 Removing the HAT

Removing

Step 1: Remove the screws

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

> ⚠ **AVOID BENDING THE PINS** OF THE 40-PIN IO HEADER!
> ROTATE ABOUT THE SHORTER SIDE OF THE RASPBERRY PI!

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

55.4 Mounting the HAT into the case

Step 2: Assemble HAT and Pi

2.1: Connect pin header and HAT with
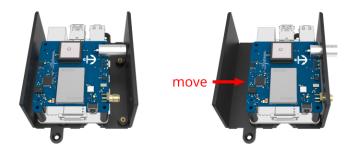Raspberry Pi.

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.
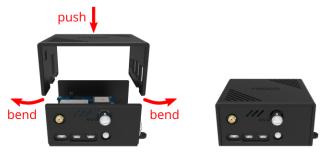

3.2: Screw together Pi, HAT and case with the 4 screws.

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

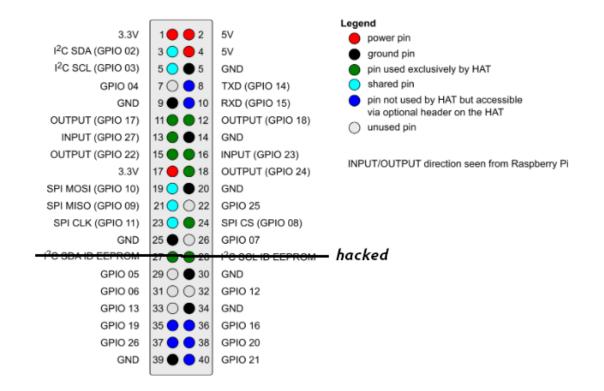Case (two-part), 4 spacers (19mm),
4 screws (M2.5x25mm)

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
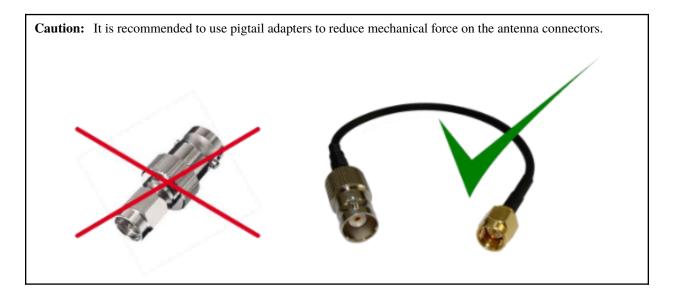
Contact:
support@rooco.eu

55.5 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	*hacked*
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

55.6 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.

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.
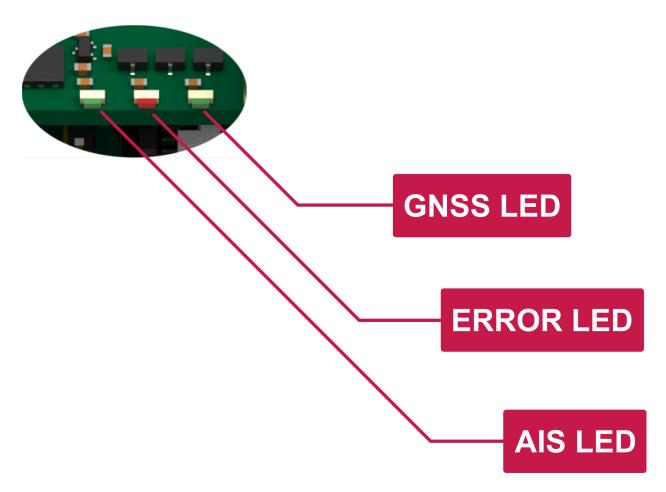

A suitable splitter also enables the Moitessier HAT to share the VHF antenna of other radio equipmenton a ship.

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

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.

55.7 Status LEDs

GNSS LED

ERROR LED

AIS LED

HACKING THE MOITESSIER HAT 1/2

Important: The Moitessier HAT developers dropped support for the drivers leaving hundreds of users with unusable devices on Raspberry OS bullseye. Despite the promise of transferring manufacturing to interested companies or opening up the sources of hardware and software, this has never happened. At OpenMarine we consider this unacceptable and we promise to never again support closed and proprietary products.

In this document, we offer a way to hack the Moitessier HAT 1 and 2 models without damaging the hardware or software to get rid of the drivers and give it a second life using most of its features in OpenPlotter v3.x.x.
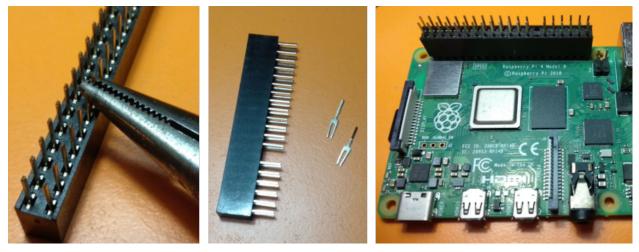
This hack will make any Moitessier HAT 1 or 2 work again without drivers. The goal is to trick the HAT into thinking it is running without being connected to a Raspberry (standalone mode) and forward the AIS/GNSS data output to UART0 instead of SPI0.0. The rest of the data (compass, heel, trim, temperature and pressure) will work as usual. The shutdown button is the only thing that will not work with this hack.

56.1 What do you need?
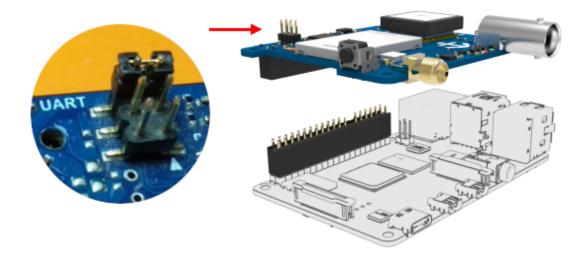
You just need a jumper, the pin header extender that comes with your Moitessier HAT and some small pliers to pull out some pins.

With the help of the pliers we have to pull hard on pins 27 and 28 (GPIO0 and GPIO1) until we extract them. In this way we prevent the EEPROM from connecting to the Raspberry and the HAT thinks that it is working in standalone mode.
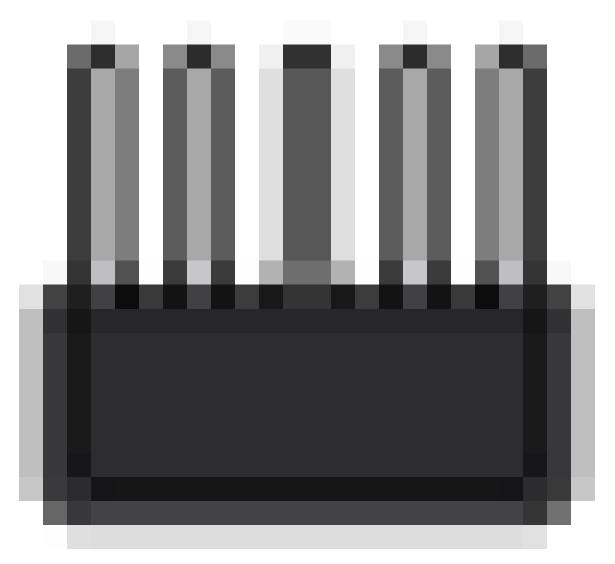
Finally, we need to bridge the pins that you see in the picture below using a jumper. In this way we send the data from a back door, which all Moitessier HAT models have, to the pin 10 of the Raspberry (GPIO15 UART0 RX). And that is it, now you just have to configure OpenPlotter to receive data in this new situation.

56.2 Configuring AIS and GNSS reception

After   mounting   the   Moitessier   HAT   using   the   modified   header   extender,   go   to   the

OpenPlotter Serial app and enable a serial port in the GPIO header of the Raspberry Pi by clicking the

UART0

icon:

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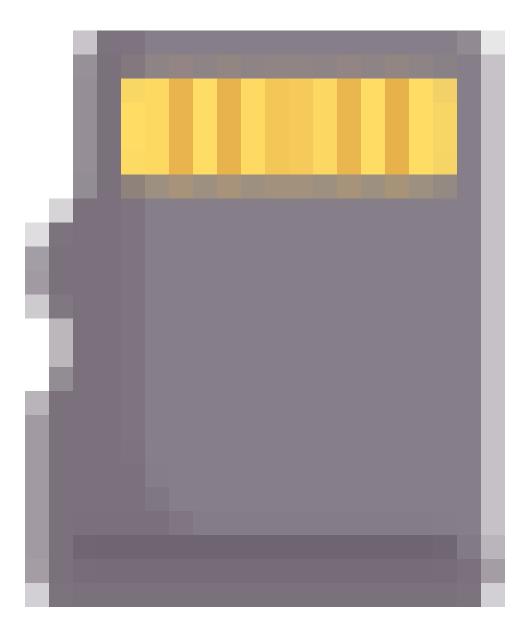
After the reboot, launch the *OpenPlotter Serial* app again. On the
Devices tab, you should now see a new entry. Select the line with *ttyAMA0*, give it
an alias (for example aisgnss) and select *NMEA 0183* from the data dropdown, then press

Apply:

Now we need to connect the *ttyOP_aisgnss* device to the Signal K server, the central data processing hub of OpenPlotter.

Switch to the

Connections tab, select the *ttyOP_aisgnss* device and click `Add to Signal K:`

Select *921600* as the *Baud Rate* and press AUTO.

You are done, the Signal K server and any program connected to it, such as OpenCPN, should now receive AIS data.
Check OpenCPN to make sure there is a connection to the Signal K server and it is getting data from your Moitessier
HAT:

Fig. 1: AIS and GNSS

56.3 Configuring compass, heel and trim reception

Go to the *OpenPlotter pypilot* app. You should see *Detected IMU: MPU-925x* in

Ser-vices tab. Select *Enable IMU Only* and you are done:

The Signal K server and any program connected to it, such as OpenCPN, should now receive magnetic heading, heel and trim data:
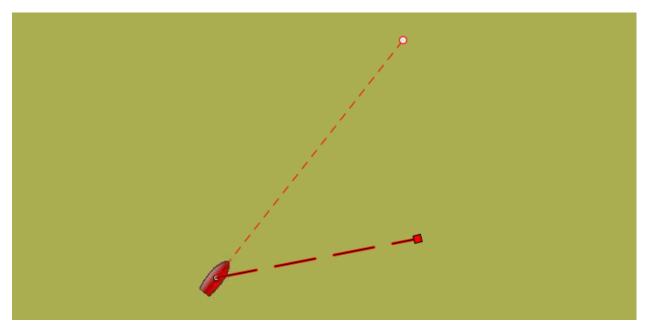
Fig. 2: Magnetic Heading (circle), Course Over Ground (square)

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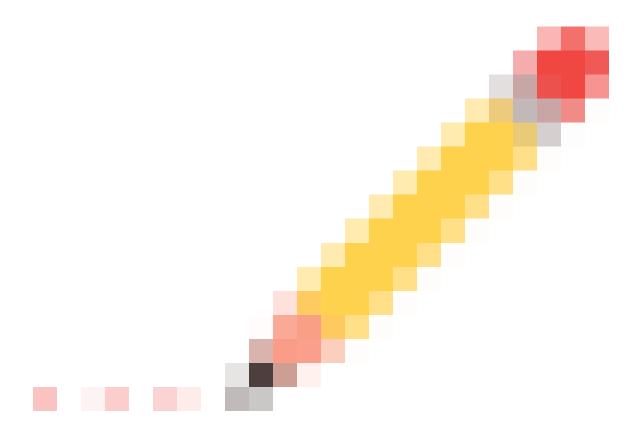
56.4 Configuring pressure reception

Go to the ▪ *OpenPlotter I2C* app and click ▪ Add:

Select *MS55607-02BA03* sensor, *0x77* address (0x68 is the IMU) and press OK:

A Signal K key will be created for pressure by default. You can assign another one for temperature. Select the *temperature* item and click

Edit:

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.inside.moitessier.temperature. We do not need to check this temperature every second, so we will select 5 seconds as *Rate*. Click OK when you are done:

We do not need to check the pressure every second either, so we will edit the *pressure* item to put 30 seconds in *Rate*:

You are done, the Signal K server and any program connected to it, such as OpenCPN, should now receive pressure data:

Fig. 3: Heel, Pitch and Pressure