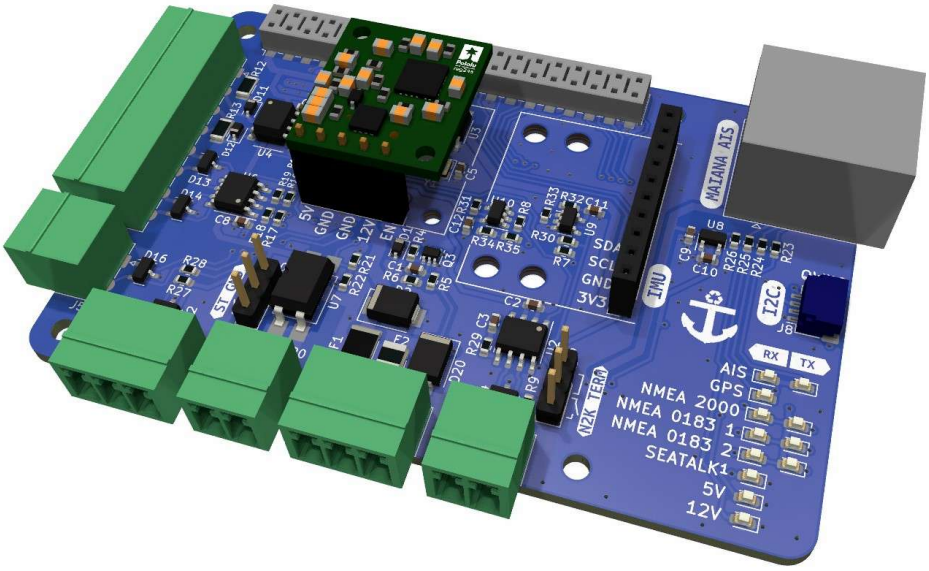


MacArthur HAT v1.0 – Documentation v1.0

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*** [1] (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.

This name is not accidental, we want to honor ***Ellen MacArthur*** [2] who is not only known for being an exceptional sailor but also for her commitment to the circular economy [3]. The *MacArthur HAT* is an electronic circuit that is as difficult to recycle and has an environmental cost to manufacture as any modern circuit, but it is designed to last and stay in your boat forever.

Its main function is to be able to communicate with any old or new marine electronic device using the proprietary and closed protocols ***Seatalk***[1], ***NMEA 0183*** or ***NMEA 2000*** and the free and open protocol ***Signal K***. This means that when an on-board device dies, we are not forced to buy another of the same brand, that uses the same technology or even uses the same protocol because we can mix different devices. We will be able to recycle and reuse old devices giving them a second life or we will be able to gradually replace our old closed and proprietary models with new, cheaper, free and open ones.

You can also power the *Raspberry Pi* directly from the ship's batteries and it has a **smart power management system** to turn it on and off automatically and **protect the SD card**. Power can be taken directly from the CAN or Seatalk[1] bus to simplify installation. This HAT is **stackable** and can be used with other HATs such as the ***Moitessier HAT*** or the ***dAISy HAT***. It can also be connected directly to the ***MAIANA AIS transponder*** without the need for an adapter.

Some of the *MacArthur HAT* features, such as the power management, the AIS receiver/transponder, the **1-Wire sensors** or the **I2C internal and external sensors**, are optional. In this way you only buy what you need saving money and we do not have to manufacture things that will never be used saving natural resources.

MacArthur HAT is fully supported by ***OpenPlotter v3*** [4] and all of its features can be easily configured with just a few clicks. No drivers needed. If you are not using *OpenPlotter*, you can still access all its features, but you have to enable all interfaces and configure the system manually.

[1] https://shop.openmarine.net/home/23-macarthur-hat.html
[2] https://en.wikipedia.org/wiki/Ellen_MacArthur
[3] https://ellenmacarthurfoundation.org
[4] https://openplotter.readthedocs.io/en/latest/description/what_is_openplotter.html

Table of contents

Credits

Development

Adrian Studer – Wegmatt LLC

Sailoog – OpenMarine

Testers

Volksbar,

Under construction

License and sources

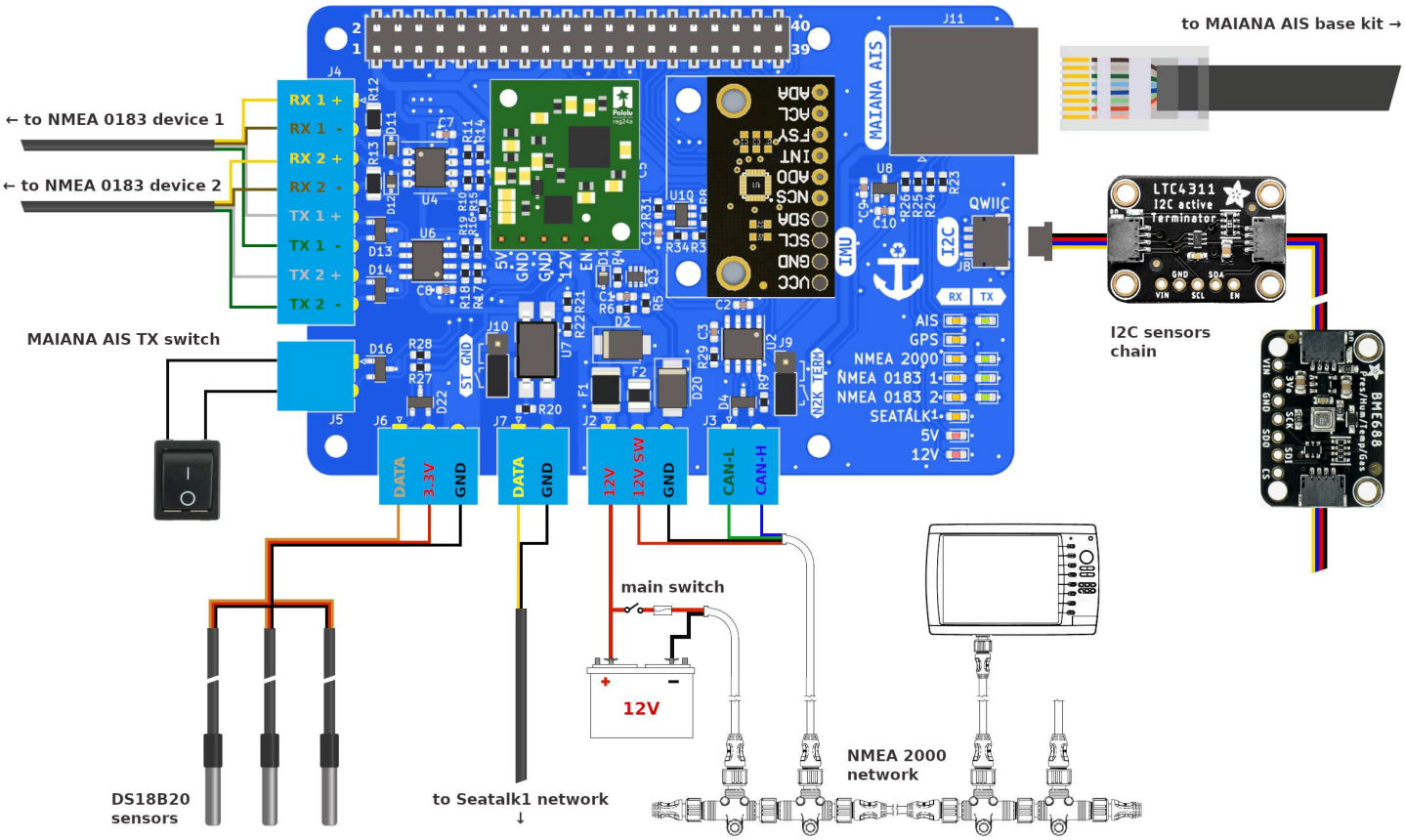
Source location: https://github.com/OpenMarine/MacArthur-HAT [1]

As per CERN-OHL-S v2 section 4, should you produce hardware based on this source, you must where practicable maintain the Source Location visible on the external case of the MacArthur HAT or other products you make using this source.

[1] https://github.com/OpenMarine/MacArthur-HAT

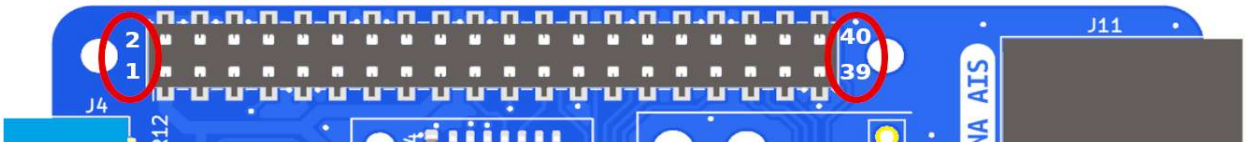
Features

- Only Raspberry Pi 4 is fully supported. If you use a Raspberry Pi 3, the NMEA 0183 inputs and outputs will not work, and you will need extra separation between the Raspberry Pi and the MacArthur HAT.
- 1x NMEA 2000 non-isolated input and output. Data connection by SPI0-1. Optional 120Ω termination resistor included. Compatible with any CAN bus.
- 2x NMEA 0183 opto-isolated inputs and 2x NMEA 0183 non-isolated outputs. Data connection by UART3 and UART5.
- 1x Seatalk[1] opto-isolated input. This connector can be also used as an isolated general-purpose input.
- 1x Connector for multiple 1-Wire temperature sensors such as the DS18B20 (exhaust, engine, fridge...). A 4.7KΩ pull-up resistor is included. This connector can be also used as a non-isolated general-purpose input/output.
- 1x STEMMA QT/Qwiic connector for multiple I2C sensors (IMU, temperature, pressure, humidity, gas...). Compatible with most Adafruit and SparkFun sensors.
- Optional 12V to 5V DC/DC converter via add-on module [1] to power the Raspberry Pi and its peripherals (including touch screens up to 10 inches). When you turn off the main switch of your ship, OpenPlotter will shut down safely. OpenPlotter will start cleanly when the main switch is turned on again.
- Optional GPS reception and AIS reception/transmission with the MAIANA AIS base kit [2]. The MacArthur HAT has all the features of all MAIANA AIS adapters in one. Data connection by UART0.
- Optional compass, heel, and trim via internal or external add-on module (IMU 9DOF).
- Compatible with dAISy HAT [3], Moitessier HAT (hacked) and Pypilot motor controllers [4]. Not compatible with Pypilot HAT.
- Detachable screw connectors for easy mounting and maximum compatibility.
- Includes input and output LEDs to check activity at any time.
- No drivers needed.



[1] https://shop.openmarine.net/home/24-power-module-for-macarthur-hat.html
[2] https://shop.openmarine.net/home/15-maiana-ais-base-kit.html
[3] https://shop.openmarine.net/home/14-daisy-hat-ais-receiver.html
[4] https://pypilot.org/opencart/index.php?route=product/category&path=59

Header pinout



All pins can be accessed by other HATs with a stacking header.
The MacArthur HAT uses the red pins, which can also be used by other HATs but only one at a time.
The MacArthur HAT uses the green pins, which can also be used by other HATs at the same time.
The MacArthur HAT does not use the white pins.

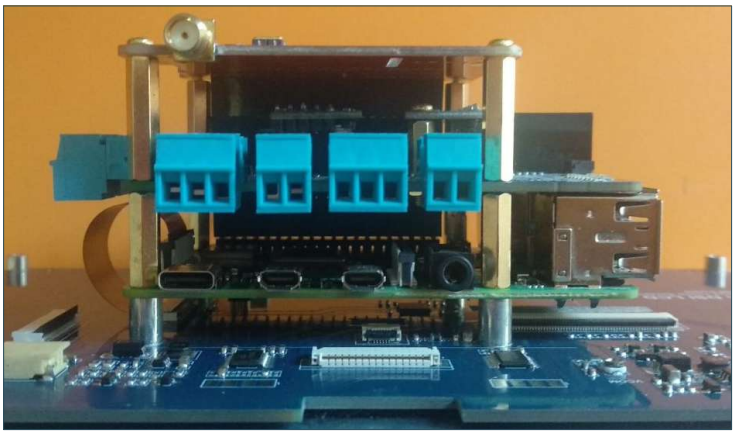
Feature	Interface	BCM	Pin		BCM	Interface	Feature
		3.3V	1	2	5V		
I2C sensors - compass, heel, trim	I2C SDA	GPIO2	3	4	5V		
I2C sensors - compass, heel, trim	I2C SCL	GPIO3	5	6	GND		
NMEA 0183 TX 1 *	UART3 TX	GPIO4	7	8	GPIO14	UART0 TX	MAIANA AIS - dAISy HAT
		GND	9	10	GPIO15	UART0 RX	MAIANA AIS - dAISy HAT
		GPIO17	11	12	GPIO18		
		GPIO27	13	14	GND		
		GPIO22	15	16	GPIO23		
		3.3V	17	18	GPIO24		
NMEA 2000	SPI0 MOSI	GPIO10	19	20	GND		
NMEA 2000	SPI0 MISO	GPIO9	21	22	GPIO25	GPIO	NMEA 2000 INT
NMEA 2000	SPI0 SCLK	GPIO11	23	24	GPIO8		
		GND	25	26	GPIO7	SPI0 CE1	NMEA 2000 CS
		GPIO0	27	28	GPIO1		
NMEA 0183 RX 1 *	UART3 RX	GPIO5	29	30	GND		
		GPIO6	31	32	GPIO12	UART5 TX	NMEA 0183 TX 2 *
NMEA 0183 RX 2 *	UART5 RX	GPIO13	33	34	GND		
1-Wire sensors - GPIO	1W/GPIO	GPIO19	35	36	GPIO16		
Power Off	GPIO	GPIO26	37	38	GPIO20	GPIO	Seatalk[1] RX - GPIO
		GND	39	40	GPIO21	GPIO	Shutdown

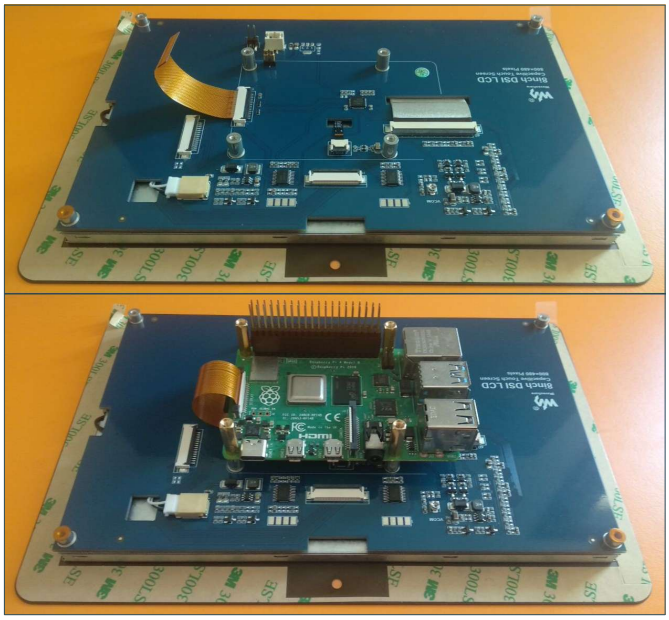
* If you want to use these GPIOs for other purposes, note that they all have pull-up resistors.

Mounting the HAT

The MacArthur HAT is designed to work in conjunction with other boards. In this example, we are going to build a compact 5-floor system with a dAISy HAT on the top floor and an 8-inch touch screen on the ground floor to be installed on a panel inside the boat.

The touch screen is powered from the Raspberry Pi which is powered by the MacArthur HAT power module making installation much simpler. The possibilities are endless and instead of using a touch screen as a ground floor, you could also put the whole set in a sealed electrical box and connect an HDMI monitor.



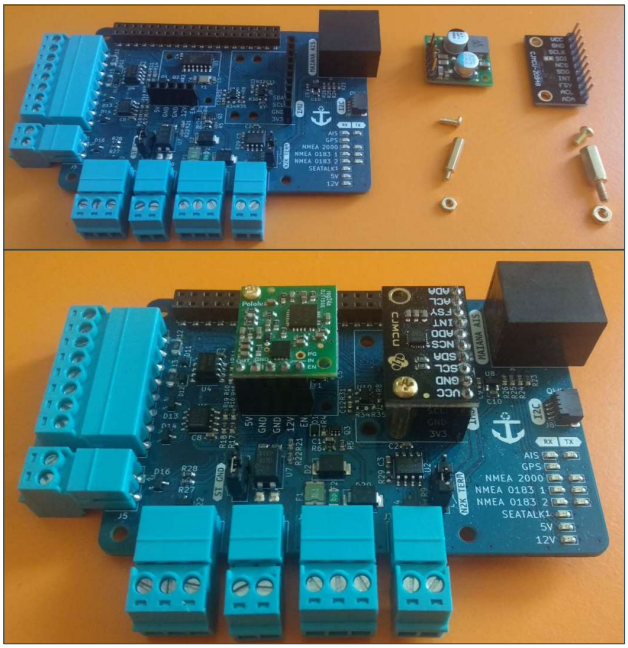


First floor

Screw the Raspberry Pi 4 to the touch screen pillars using the 4 supplied 16mm M2.5 standoffs. Also plug in the extra-long stacking header extender provided.

If you are using a Raspberry Pi 3 you should use an extra GPIO header extender of normal size and standoffs long enough to prevent the MacArthur HAT' RJ45 connector from making contact with the Raspberry Pi 3 USB connectors. Putting electrical tape on top of the USB metal shield might help

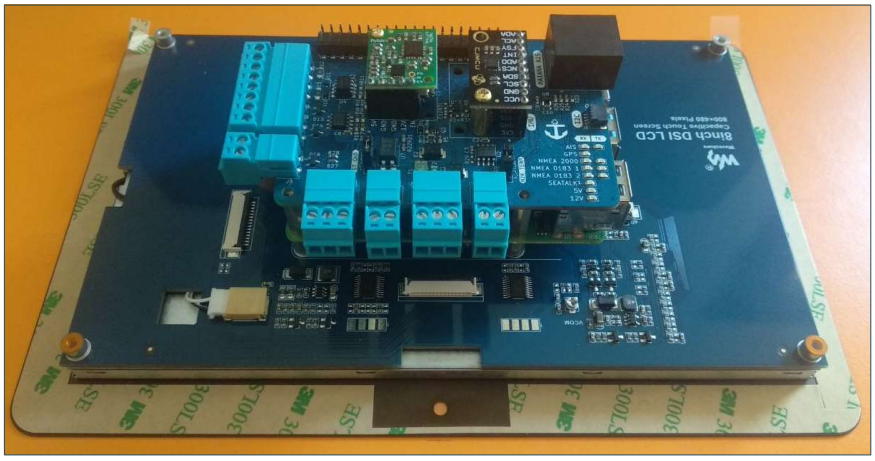
Do not forget to connect the 15PIN FPC cable to the DSI interface of the Raspberry Pi for video and touch screen data.



Third floor

Before adding the second floor to the set we need to add the third floor, which in this case refers to the power module and an internal I2C module with an IMU.

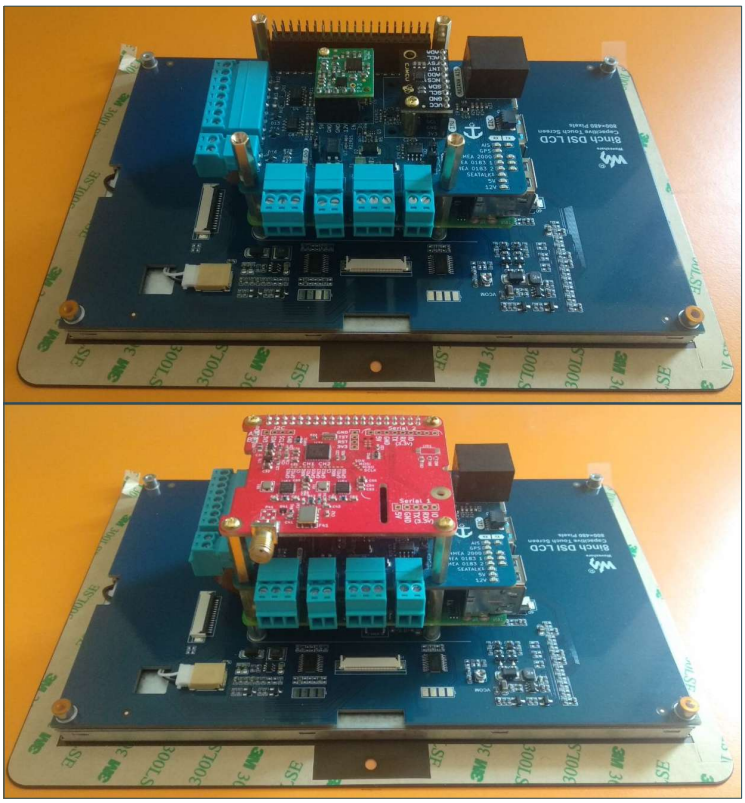
Both modules need to be secured in at least one corner. The power module is supplied with an 11mm standoff and an M2 screw and nut kit. The I2C module needs an 11mm standoff and an M2.5 screw and nut kit.



Second floor

Now we can add the second and third floor to the set. The GPIO header extender pins should stick out enough to connect another HAT or access the pins individually.

If you are not going to connect another HAT, you can now secure the MacArthur HAT with the 4 supplied M2.5 screws.



Fourth floor

If we want to connect another HAT, we must use an extra GPIO header extender of normal size and 4 spacers of 20mm and M2.5. Finally, we will secure the dAISy HAT with 4 M2.5 screws.

Power management

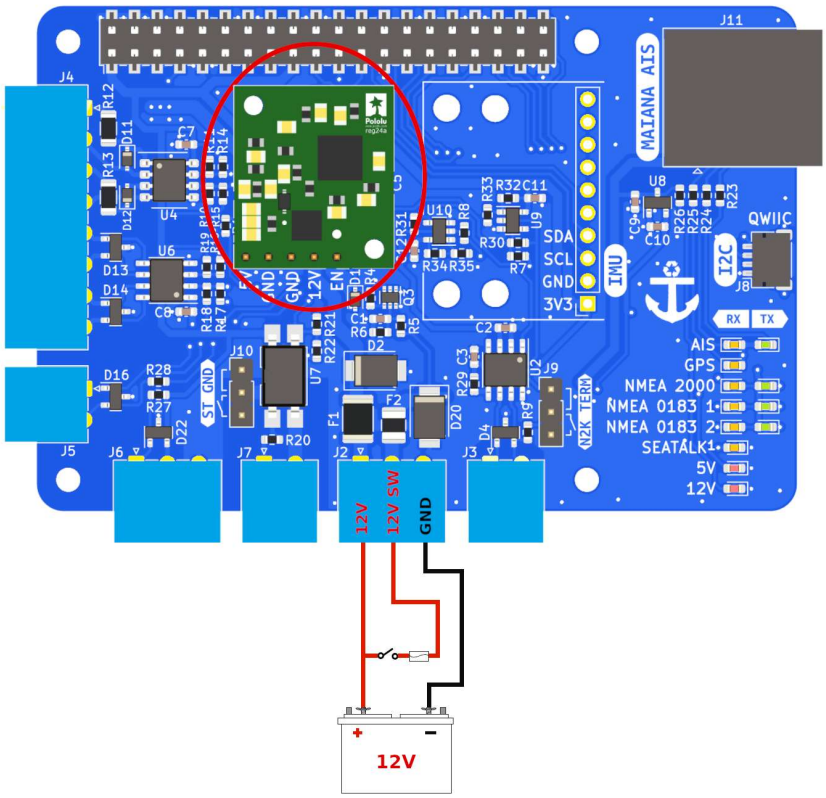
This feature is optional, so you can also use all features of the MacArthur HAT by powering the system from the Raspberry Pi as usual. Do not power the system from the Raspberry Pi and the MacArthur HAT at the same time, you must use only one power supply.

If you choose to power the system from the HAT, you will need to mount the power module which will also provide some extra functionality. One of the main problems with Raspberry Pi based systems is the corruption of SD cards due to power outages. We often turn off the power to our on-board electronics without remembering to shut down the Raspberry Pi OS and the consequences can be critical. There are currently several solutions based on supercapacitors or HATs with its own battery that detect power outages and automatically shut down the system, but they also have their drawbacks. Often the system takes too long to close some programs and needs more time than supercapacitors can offer. And solutions based on extra batteries add unnecessary complexity, why use an extra battery just to keep the system on until it shuts down when we can use our boat's?

Wiring

Once the power module is mounted, we have several options to connect the HAT to the battery and the switch according to the characteristics of your boat. There are resettable fuses integrated, 1.5A for the 12V input and 0.75A for the 12V SW input.

Powering from battery



This is the most basic and simple method. The external circuit consists of a battery, a switch and a fuse that is not essential but is recommended to protect your installation from excessive temperatures and power surges.

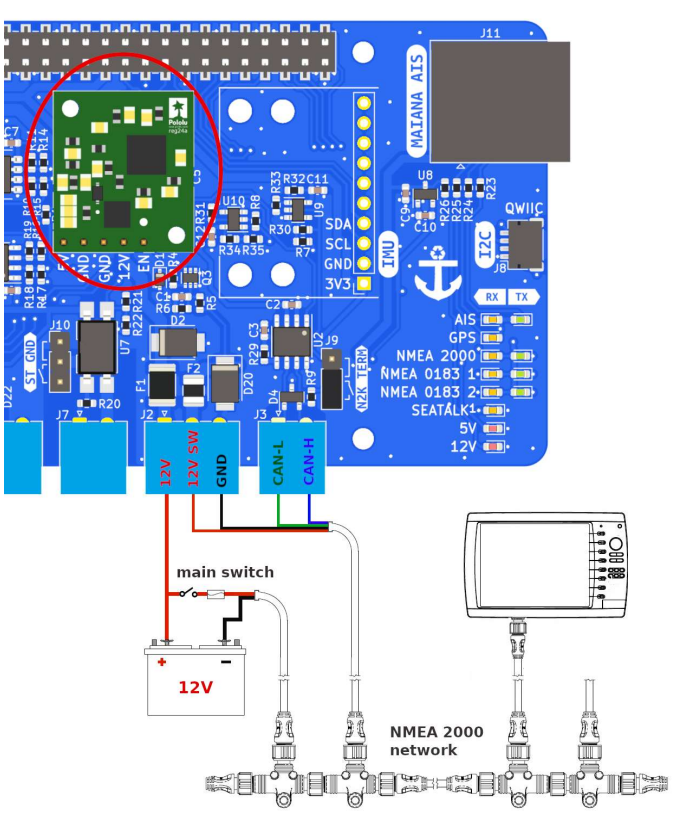
After the software configuration just close the switch to start OpenPlotter and open the switch to shutdown OpenPlotter cleanly no matter how much time the system needs. When the power module cuts the current, it is still connected to the battery, but the consumption is only 0.1 mA.

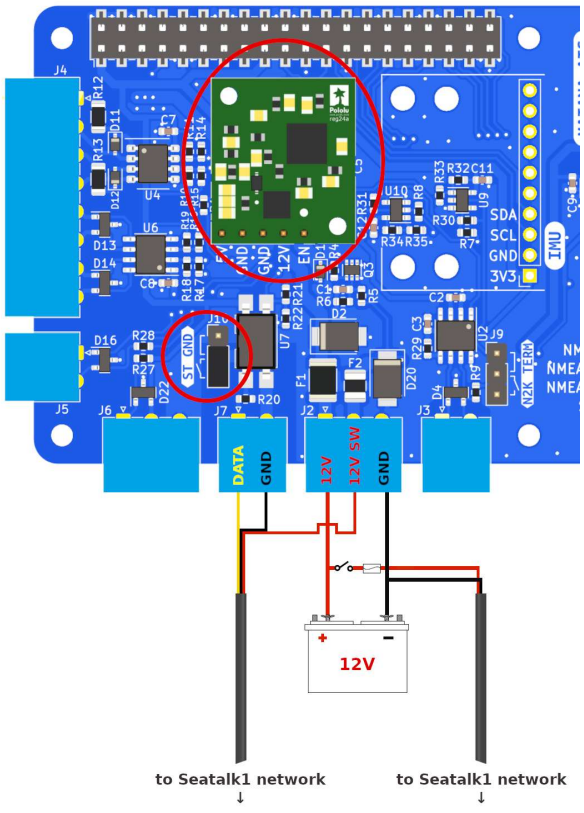
Powering from NMEA 2000 bus

If your boat has a NMEA 2000 network, it probably already has a main switch and fuse. In this case you can cut off the connector at one end of a drop line of the CAN bus and directly connect the wires as shown in the diagram on the right.

When you close the main switch, all the devices powered from the NMEA 2000 bus will be turned off (plotter, sensors, displays...) but OpenPlotter will stay on until the system shuts down safely.

If you want to power the Raspberry Pi without sending or receiving data on the NMEA 2000 network, just ignore the CAN-L and CAN-H wires. To learn more about NMEA 2000 wires and connectors see the chapter *NMEA 2000*.





Powering from Seatalk[1] bus

If your boat has an old Seatalk[1] network, it probably already has a main switch and fuse. Make the connections as shown in the diagram on the left and place the **ST GND** jumper in the **open** position.

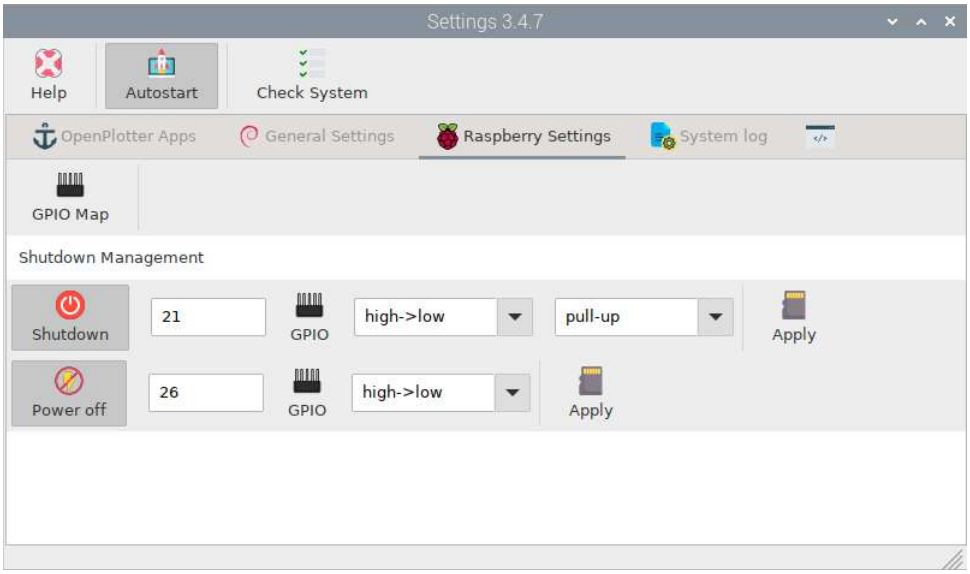
When you close the main switch, all the devices powered from the Seatalk[1] bus will be turned off (plotter, sensors, displays...) but OpenPlotter will stay on until the system shuts down safely.

If you want to power the Raspberry Pi without receiving data from the Seatalk[1] network, just ignore the DATA and GND wires of the Seatalk[1] connector. To learn more about Seatalk[1] wires and connectors see the chapter *Seatalk[1]*.

Configuration

When the external switch is turned off, the MacArthur HAT tells OpenPlotter via a GPIO that it must initiate the shutdown immediately. OpenPlotter will use another GPIO to tell the HAT when the shutdown is complete so that the HAT can cut power to the Raspberry Pi. From that moment on, the HAT keeps listening to the external switch and when it is turned on it provides power to the Raspberry Pi again, starting OpenPlotter.

We need to define in OpenPlotter which GPIOs are going to perform these tasks and we can do this easily using the *Settings* application:



Go to ***Raspberry Settings*** tab. Click the ***Shutdown*** button. Select the *GPIO 21*. Select *high→low* as the event to trigger the shutdown action. Select *pull-up* to define the pull resistor and click ***Apply***.

Click the ***Power off*** button. Select the *GPIO 26*. Select *high→low* and finally click ***Apply***. Reboot OpenPlotter to apply the changes and turn the external switch off/on to confirm everything works.

LEDs

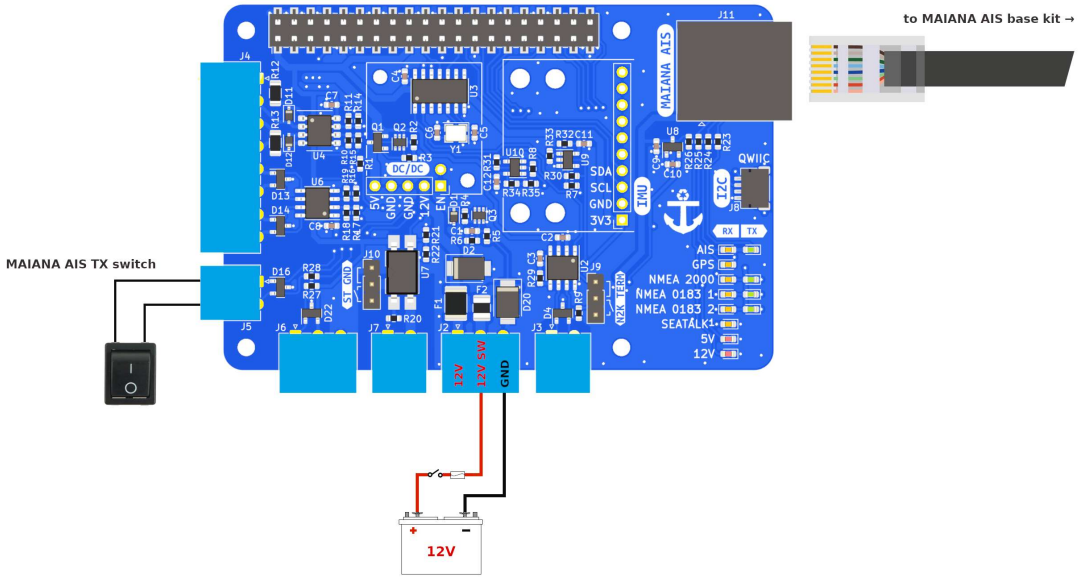
□ off | ■■■ blinking | ▬▬ fixed

LED	RX	TX	Description
5V	□		The power module is not present or there is no 12V input yet
5V	▬		The power module is present and is powering the Raspberry Pi
12V	□		There is no 12V SW input
12V	▬		There is 12V SW input

MAIANA AIS

You can connect a MAIANA AIS base kit directly to the MacArthur HAT using a Cat5 cable (Ethernet cable with RJ45 connectors). The 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 or NMEA 2000 data to any device on your boat.

Wiring



You do not need the power module to make the MAIANA AIS work, but you need to power the MAIANA AIS with 12V as shown in the image above. Of course, MAIANA AIS will also work when the power module is present.

There are times when you are not interested in sharing your position. To turn the AIS data transmission off quickly, the HAT has a connector for you to add a switch. MAIANA AIS has two switches for transmission, a *software TX switch*, and this *hardware TX switch*. Both switches must be on to transmit AIS data. When this physical switch is open, the *hardware TX switch* is on, when it is closed, the *hardware TX switch* is off.

Configuration

To configure the MAIANA AIS, follow the steps in the OpenPlotter manual [1]. Remember that when using the MacArthur HAT, the MAIANA AIS connects via UART0.

LEDs

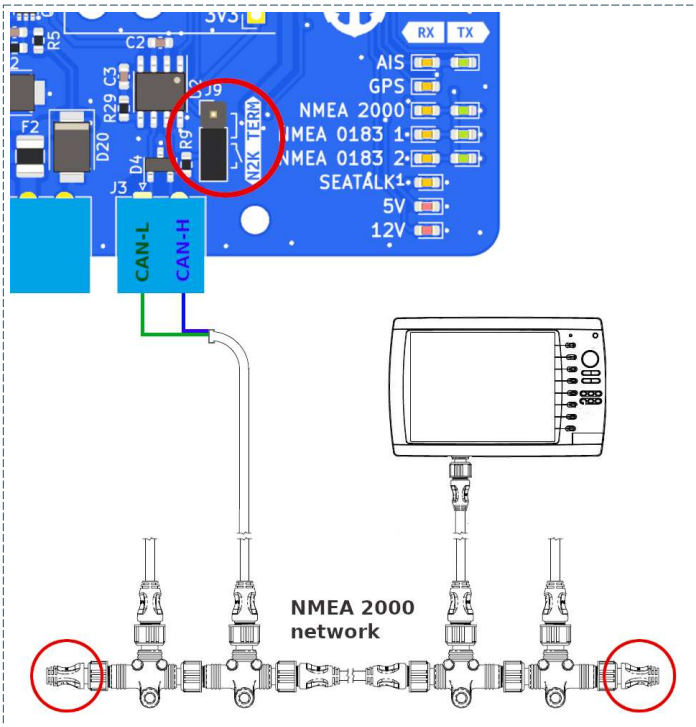
□ off | ■■■ blinking | ▬▬ fixed

LED	RX	TX	Description
AIS	□		MAIANA AIS is not connected or is not receiving GNSS or AIS data
AIS	■■■		MAIANA AIS is receiving AIS data from ships around you
AIS		□	The hardware or software TX switch is off and there is no AIS transmission
AIS		▬▬	The hardware and software TX switches are on and there is AIS transmission
GPS	□		There is not GPS fix yet
GPS	▬▬		There is GPS fix

[1] https://openplotter.readthedocs.io/en/latest/maiana/configuring.html

NMEA 2000

Raymarine SeaTalk[2], Raymarine SeaTalk[NG], Simrad Simnet and Furuno CAN are rebranded implementations of NMEA 2000, but they use physical connectors different from the standardised DeviceNet 5-pin A-coded M12 screw connector. This is intentional to keep you tied to a brand, but they are all electrically compatible.



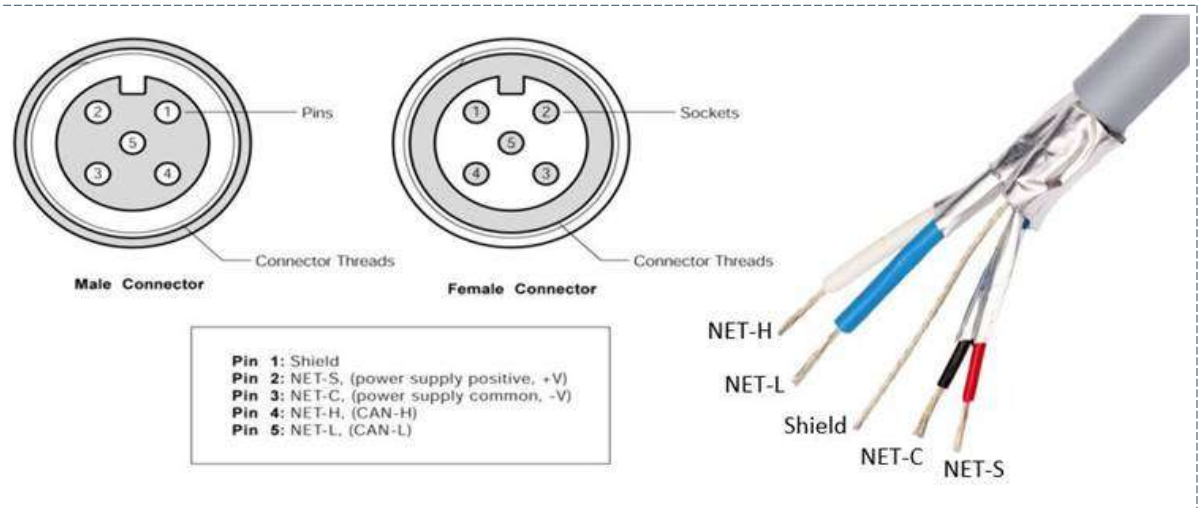
Wiring

Cables that convert exclusive formats to the standard NMEA 2000 format are extremely expensive. Using simply screw connectors we make the MacArthur HAT easy to connect to all brands by simply cutting any of the cables used in their drop lines and connecting the **CAN-H** and **CAN-L** wires as shown in the image on the right. If you also want to power the system from the NMEA 2000 network, read the *Powering from NMEA 2000 bus* section.

The MacArthur HAT NMEA 2000 circuit is not isolated but you should have no problems as long as the Raspberry Pi and the NMEA 2000 network are powered from the same source.

NMEA 2000 networks must have a terminator at each end. Without these terminators there will be no data traffic, so the MacArthur HAT has an optional termination resistor in case one of the network terminators is missing or you are connecting the HAT directly to another NMEA 2000 device. If termination is missing in your network, the N2K TERM jumper must be in the *closed* position. If your network is correctly terminated, the **N2K TERM** jumper must be in the ***open*** position.

The image below shows the pinout of the standard NMEA 2000 connectors and the colors of the wires, but each brand will have different pinout and colors. To identify the CAN-H, CAN-L, V+ and V- wires of our drop line we must consult the documentation of the device or ask the manufacturer.



Configuration

Once the MacArthur HAT is connected to the NMEA 2000 network we need to configure OpenPlotter to receive and send data. Open the *CAN Bus* app and go to the ***MCP251xfd*** tab:

Click ***Add MCP251xfd device*** and select *SPI0 CE1* as *Interface*, *20000000* as *Oscillator*, *GPIO 25* as *Interrupt* and click ***OK***.

After rebooting OpenPlotter, go back to the *CAN Bus* app at ***MCP251xfd*** tab, select the new entry in the device list and click ***Add Connection*** to create a connection to the Signal K server. The Signal K server will restart, and you should start receiving data from the NMEA 2000 network.

To send data from the Signal K server to the NMEA 2000 network follow the steps in the OpenPlotter manual [1].

LEDs

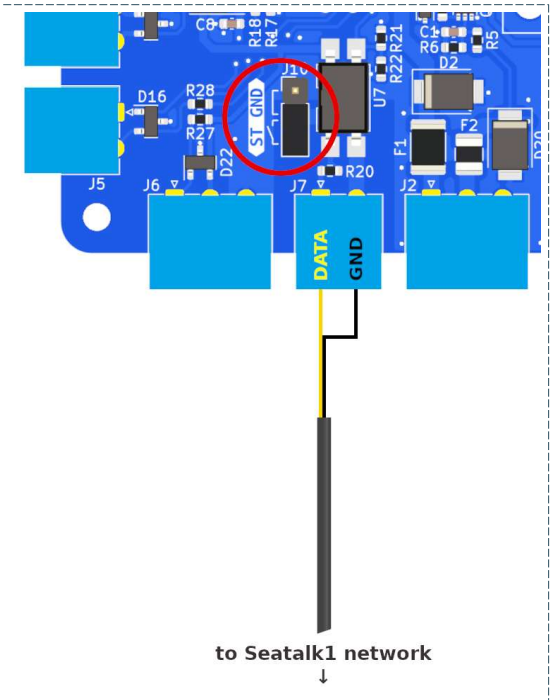
□ off | ■■■ blinking | ▬ fixed

LED	RX	TX	Description
NMEA 2000	□	□	OpenPlotter is not configured and the HAT is not connected to the CAN bus
NMEA 2000	▬	▬	OpenPlotter is configured but the HAT is not connected to the CAN bus
NMEA 2000	▬	□	OpenPlotter is not configured but the HAT is connected to the CAN bus
NMEA 2000	■■■	□	OpenPlotter is configured, the HAT is connected to the CAN bus and is receiving data
NMEA 2000	■■■	■■■	OpenPlotter is configured, the HAT is connected to the CAN bus and is sending data (when data is sent the 2 LEDs blink at the same time)

[1] https://openplotter.readthedocs.io/en/latest/can/output.html

Seatalk[1]

Seatalk[1] is an old and outdated communication protocol, but it is still quite present on many boats.



Wiring

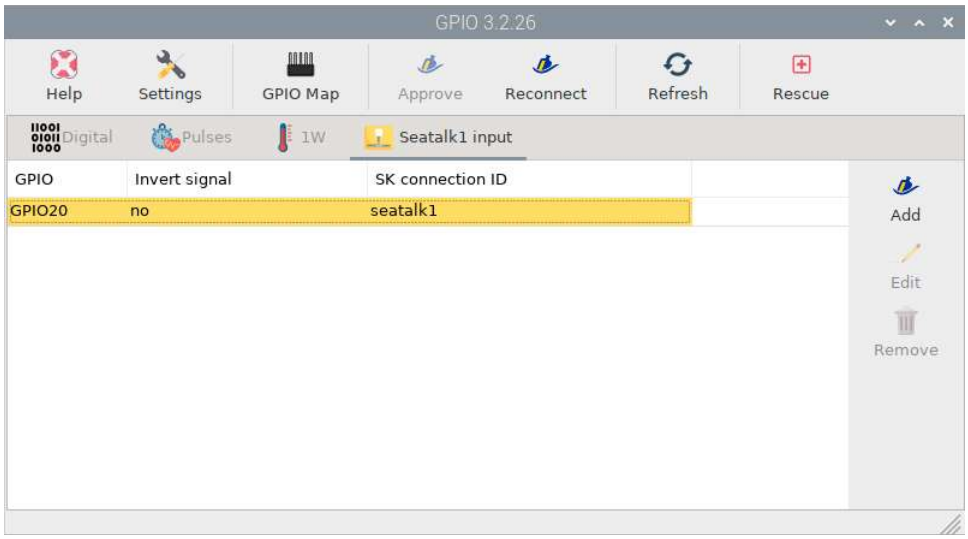
The Seatalk[1] bus has 3 wires: **black** for GND, **yellow** for data, and red for power. You just need to connect yellow and black as shown in the image above and ignore the red one.

The **ST GND** jumper must be in the ***open*** position. If you also want to power the system from the Seatalk[1] bus, read the *Powering from Seatalk[1] bus* section.

Configuration

To receive data via Seatalk[1] you need the *GPIO* app. Go to the **Seatalk1 input** tab and click **Add**. Provide an ID for this connection, e.g. *seatalk1*, select the *GPIO 20* and click **OK**.

The Signal K server will restart, and you should start receiving data from the Seatalk[1] bus.



LEDs

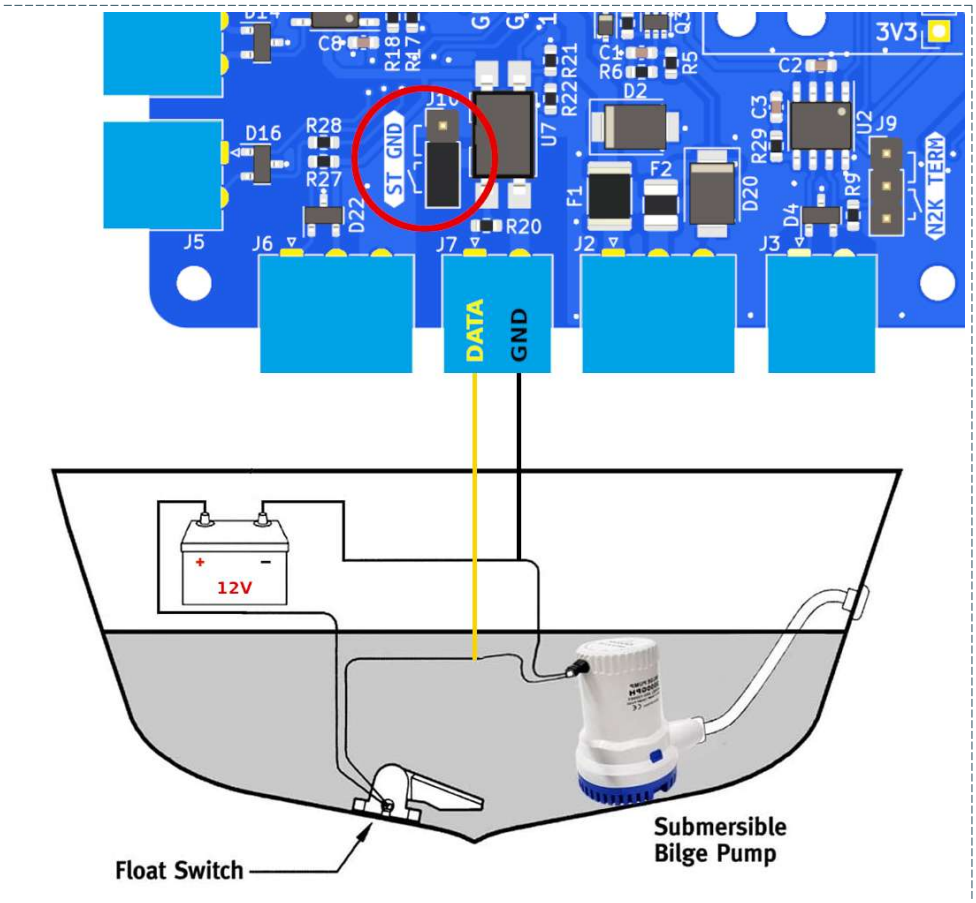
□ off | ■■■ blinking | ▬▬ fixed

LED	RX	TX	Description
Seatalk[1]	□		Seatalk[1] wires are not connected
Seatalk[1]	▬▬		Seatalk[1] wires are connected

Seatalk[1] - alternative uses

This connector is basically an optocoupler connected to the GPIO 20 of the Raspberry using a pull-down resistor so that you can connect signals of up to 12V without damaging the Raspberry. In this scenario make sure that the **ST GND** jumper is in the ***open*** position. Some possible uses are:

- Measure engine RPM by connecting the 'W' terminal or tachometer signal wire. This would be configured in the **Pulses** tab of the *GPIO* app.
- Detect when a *Bilge Pump* has been activated. After wiring as below picture, go to the *GPIO* app, **Digital** tab and click **Add input**. Select **GPIO 20** in *GPIO* field and **none** in *internal pull resistor* field. Select a **state** and write a *message* for each of the possible *high* or *low* states. Finally select **visual** in *method* and click **OK** to check the operation of the detector. In addition to visual and sound warnings, you can use the *Notifications* app to assign **Actions** to each of the defined states for the Signal K key “notifications.GPIO20”.



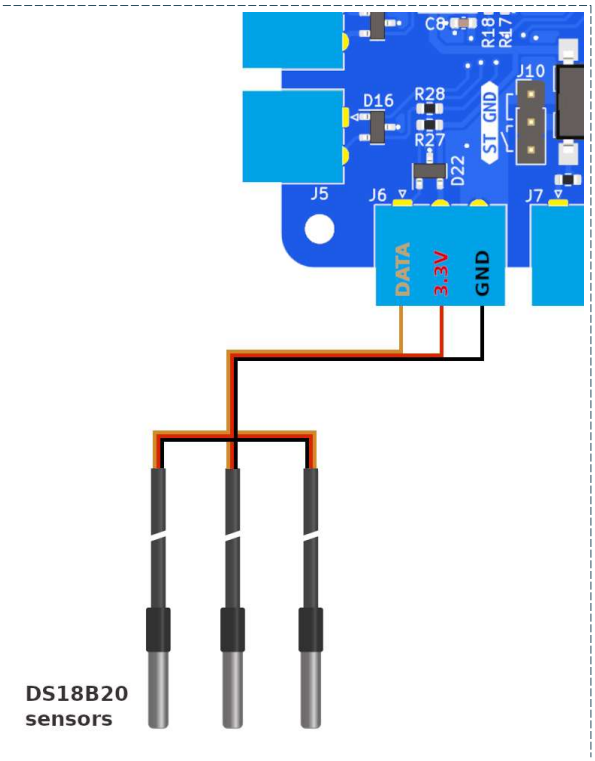
LEDs

□ off | ■■■ blinking | ▬▬ fixed

LED	RX	TX	Description
Seatalk[1]	□		Input is low
Seatalk[1]	▬		Input is high

1-Wire sensors

1-Wire sensors like the DS18B20 are digital sensors specially designed to measure temperature in extreme conditions. That makes them perfect for your engine or fridge. Because they are digital, you do not get any signal degradation even over long distances (30-40 meters). You can connect multiple sensors to the same connector because each one has a unique 64-bit ID burned in at the factory to differentiate them.



Wiring

The MacArthur HAT includes a 4.7K pull-up resistor for this connector, so all you have to do is screw the wires into their correct place.

The color of the wires of these sensors are usually the same, yellow for **data**, red for **power** and black for **ground**. You may also find a white wire that you can ignore.

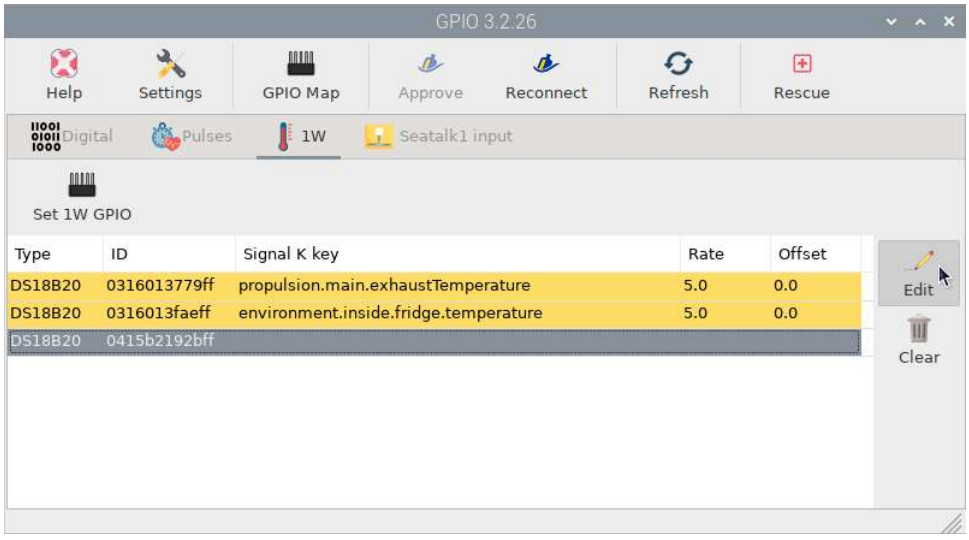
Configuration

After wiring, you can identify your sensors and assign them to a Signal K key using the *GPIO* app. First you need to enable the 1-Wire interface in Raspberry OS. Go to *Preferences – Raspberry Pi configuration – Interfaces*, enable the *1-Wire* slider and click **OK**.

After rebooting you may get an error at startup warning you about a conflict in *GPIO - 1W*. This means that by default the 1-Wire interface is set to GPIO 4 and you may already be using that GPIO (usually for UART3). Do not worry, in the next step we will configure the GPIO used by the MacArthur HAT for 1-Wire and the conflict will be resolved.

Go to the *GPIO* app, **1W** tab and click **Set 1W GPIO**. Select *GPIO 19* and click **OK**. After reboot, the conflict will be resolved, and we should get the list of connected sensors when we go back to the *GPIO* app.

Select one of the detected sensors and click **Edit**. Type or select a *Signal K key*, select the sample rate, set an *Offset* if you need it and click **OK**. You should start receiving data on the Signal K server for those keys. Remember that the temperature on the Signal K server is always displayed in Kelvin.

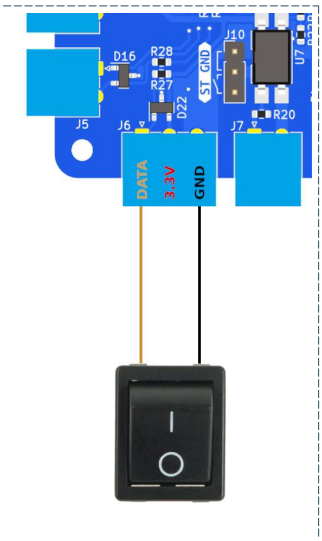


1-Wire - alternative uses

This connector basically consists of a 4.7K pull-up resistor connected to the GPIO 19 of the Raspberry Pi, so you can configure this GPIO as an input and connect any switch or any digital sensor such as float switches, door detectors, motion sensors… You can also configure this connector as an output and activate low power external devices such as LEDs, relays, buzzers…

ATTENTION – If you supply this connector or any GPIO in the header with more than 3.3V you will damage the Raspberry Pi. Do not connect high consumption devices such as motors either.

Input



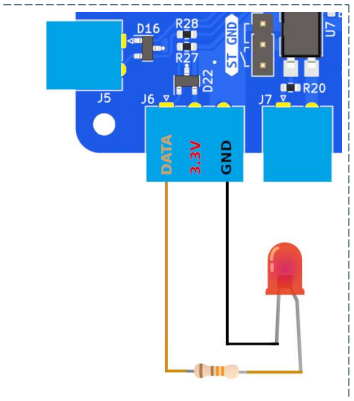
By connecting a switch as shown in the image to the right, GPIO 19 will go **high** when the switch is **open** and **low** when the switch is **closed**.

After wiring, go to the *GPIO* app, **Digital** tab and click **Add input**. Select **GPIO 19** in *GPIO* field and **none** in *internal pull resistor* field. Select a **state** and write a *message* for each of the possible *high* or *low* states. Finally select **visual** in *method* and click **OK** to check the operation of the switch.

In addition to visual and sound warnings, you can use the *Notifications* app to assign **Actions** to each of the defined states for the Signal K key “notifications.GPIO19”.

Output

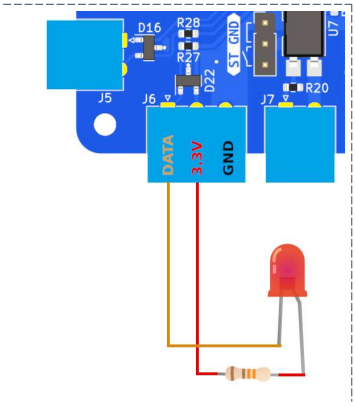
Depending on how you connect the external device, in this case a LED, it will behave differently. To treat this connection as a digital output you have to go to the *GPIO* app, **Digital** tab and click **Add output**. Select **GPIO 19** in *GPIO* field and automatically two new **Actions** will we created in the *Notifications* app to set high (3.3V) or low (0V) the GPIO 19.



When the external device is connected to **DATA** and **GND**, GPIO 19 will be high and the LED will turn on at startup (~3V).

When you run the action “localhost-GPIO19: turn it high”, the LED will turn on (3.3V).

When you run the action “localhost-GPIO19: turn it low”, the LED will turn off (~0V).



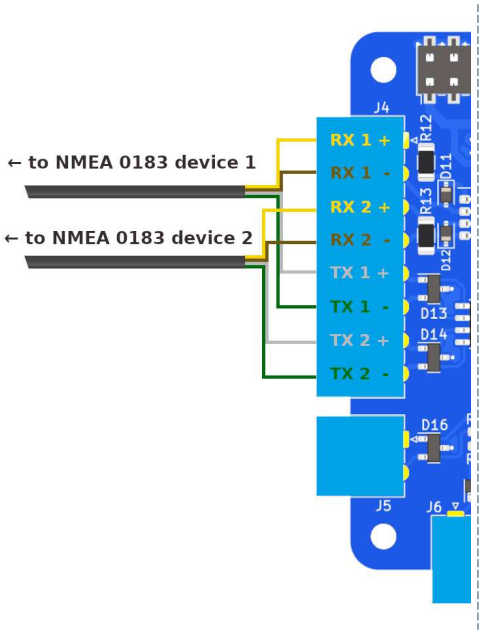
When the external device is connected to **DATA** and **3.3V**, GPIO 19 will be low and the LED will stay off at startup (~0.3V).

When you run the action “localhost-GPIO19: turn it high”, the LED will turn off (0V).

When you run the action “localhost-GPIO19: turn it low”, the LED will turn on (~3.3V).

NMEA 0183

Most GNSS, AIS or autopilot devices still accept this old protocol, so we can say that it will be on our boats for a long time. The MacArthur HAT can send and receive NMEA 0183 data from two devices at the same time, but it can also communicate with a total of four devices if they are only sending or receiving data.



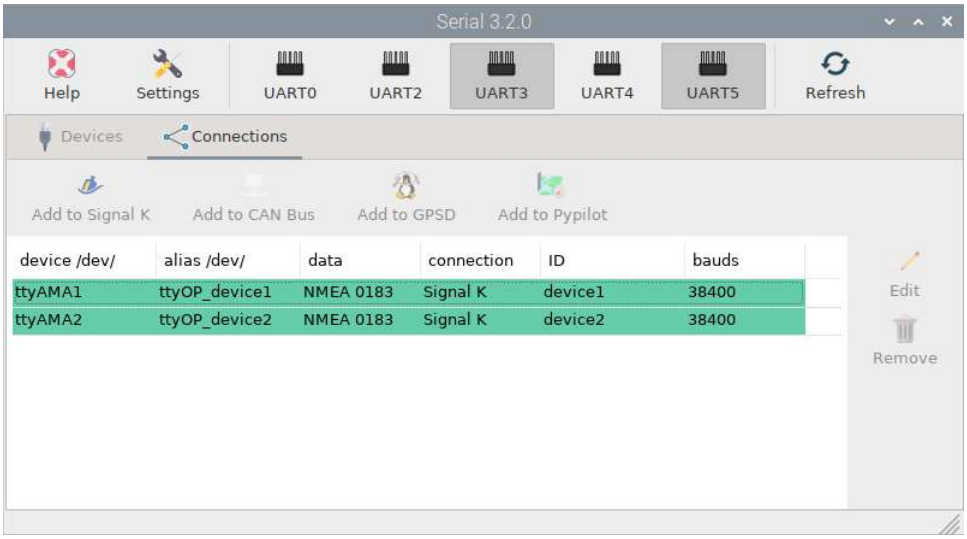
Wiring

Your NMEA 0183 device usually will have four connections: TX+, TX-, RX+ and RX-. TX means *transmit* and RX means *receive*.

You have to connect the TX connectors of your device to the RX connectors of the MacArthur HAT and the RX connectors of your device to the TX connectors of the MacArthur HAT. However, there is little consistency between devices as to what is positive and what is negative, so if the connection between TX and RX connectors with the same sign does not work, try connecting opposite signs. Do not worry, you won't damage the device or the MacArthur HAT by reversing the polarity.

Configuration

In order to send and receive data via TX/RX 1 and TX/RX 2, we must first enable the UART3 and UART5 serial interfaces respectively in OpenPlotter. We can easily do this using the *Serial* app. Click the **UART3** and **UART5** buttons and reboot. After booting you should see two new items in the list of detected serial devices. Select the first one, provide an *alias*, for example *device1*, and select *NMEA 0183* under *data*. Click **Apply** and repeat the process for the second device. Then go to the **Connections** tab, select the first item, and click the **Add to Signal K** button. In the next window, select the *Baud Rate* of your device and click **AUTO**. Repeat the process with the second item and from then on you are ready to get NMEA 0183 data to the Signal K server from both devices (RX 1 and RX 2).



To send data through TX 1 and TX 2 we need to use Signal K events. For example, if we wanted to relay to TX 2 all the data coming in through RX 1, we would need to edit the connection for *device1* in the Signal K server and add an event name in the *Input Event* field, for example *RX1input*. Finally, we need to edit the connection for *device2* and add the *RX1input* event in the *Output Events* field. After editing both connections, restart the Signal K server and data coming in from *device1* will start flowing to *device2*. To learn more about Signal K events, read the *NMEA 0183 Multiplexing* chapter in the OpenPlotter manual [1].

[1] https://openplotter.readthedocs.io/en/latest/signalk/multiplexing.html

LEDs

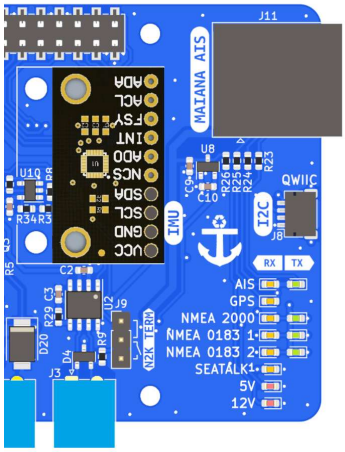
□ off | ■■■ blinking | ▬▬ fixed

LED	RX	TX	Description
NMEA 0183 1/2	□		The device is not connected or is not sending us data
NMEA 0183 1/2	▬		The device is connected but you should probably try reversing polarity
NMEA 0183 1/2	■■■		The device is connected and is sending us data
NMEA 0183 1/2		□	OpenPlotter is not yet configured to send data or there is nothing to send
NMEA 0183 1/2		■■■	OpenPlotter is configured and sending data to the device

I2C sensors

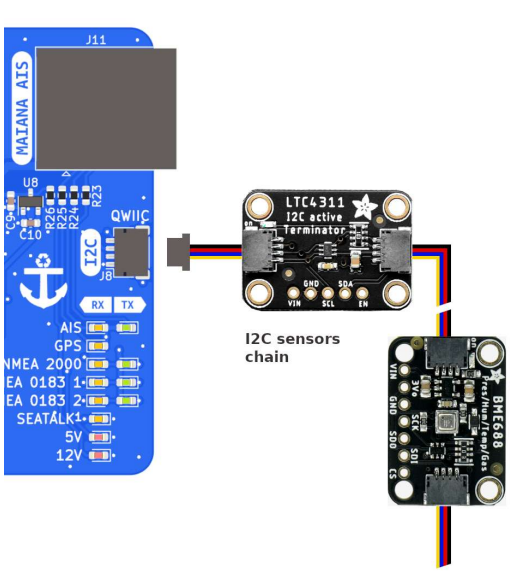
I2C digital sensors are perfect for collecting data on multiple environmental parameters such as temperature, pressure, humidity, air quality or presence of gases, light intensity, etc. and orientation parameters such as heading, heel and trim. You can also use sensors to measure the voltage and current of your batteries or analog-to-digital converters to be able to use any analog sensors and infinitely expand the list of parameters to be measured.

There are two ways to connect I2C sensors to the MacArthur HAT, externally via STEMMA QT/Qwiic connectors or internally via a female socket on the board, each one has its disadvantages and advantages.



The **internally** mounted sensors are compact and very easy to install but they are affected by the temperature of the Raspberry and make it impossible to read some parameters such as pressure when our system is inside a sealed box. Maybe it would not be the perfect place for environment sensors, but it is the perfect place for an IMU sensor and get heading, heel and trim as shown in the image on the left.

Externally mounted sensors will be less affected by other devices since we can move them away. We can also connect multiple sensors in line to the same bus. However, the I2C bus is not designed to support long distances and we will start getting invalid readings when the cable we use is longer than one meter. If you need long distances, you must add an I2C amplifier like the LTC4311 module as shown in the image on the right. By connecting the amplifier at the beginning of the sensors chain, you can get up to 30 meters at the default frequency using quality cables.



Configuration

I2C sensors are configured using the *I2C* app. Learn more about supported I2C sensors and their configuration in the OpenPlotter manuals [1]. IMU sensors for heading, trim and heel are an exception and are configured using the *Pypilot* app, learn more about it in the OpenPlotter manual [2].

[1] https://openplotter.readthedocs.io/en/latest/i2c/i2c_app.html
[2] https://openplotter.readthedocs.io/en/latest/pypilot/pypilot_app.html