RASPIMUX

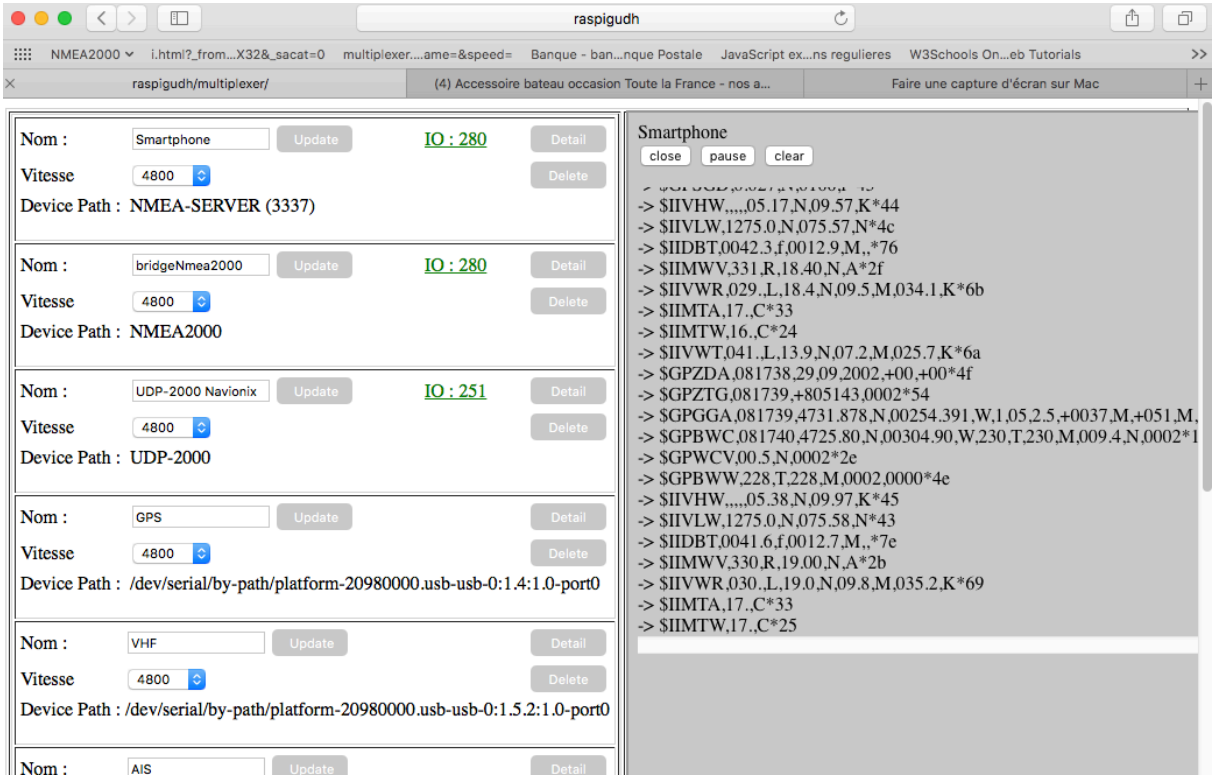
Multiplexeur NMEA0183 bridge NMEA2000

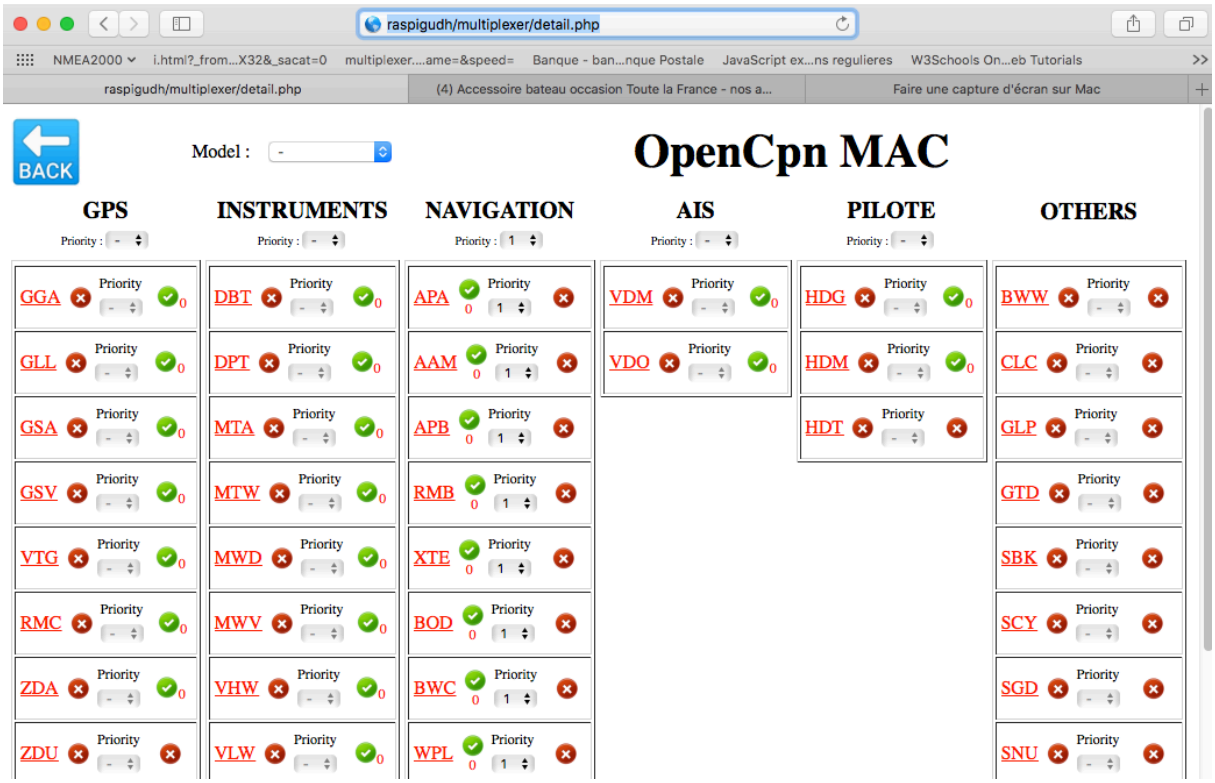
Fonctionalités

Support serial RS232 (supporte pieuvres 2-4-8 ports RS232 et Hub USB)
Chaque port RS232 est une entrée et une sortie du multiplexeur
Plug to play
Pas de paramétrage, détection automatique des connexions
Reconnaissance automatique de la vitesse (GPS / AIS / VHF /CENTRALE)
Multiplexage des données
Sélection des phrases en entrée et sortie
Gestion dynamique des priorités si plusieurs appareils identiques
Hot-spot wifi
Support de connexions sans fils wifi entrées/sorties nmea
dongle RS232/WIFI sur appareil NMEA (plus de cablage)
Répétiteur WIFI UDP :2000 (compatibilité navionics)
Support multiples PC wifi (opencpn et autres logiciels)
Bridge NMEA200 dans les deux sens avec filtrage des informations
Serveur web pour la gestion des ports et visualisation du trafic
Fonction répétiteur multi sur smartphones et tablette via WEB

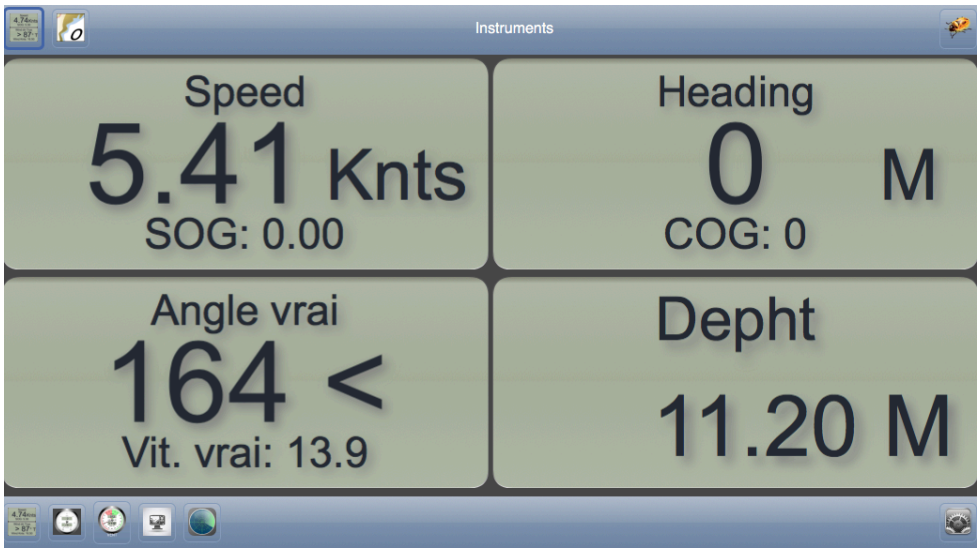
La solution fonctionne parfaitement sur Raspberry PI ZERO, consommation de l'ordre de 55w, elle intègre un paramétrage de linux qui garantit un bon fonctionnement malgrès les démarrages et arrêts intempestifs liés à l'installation sur un bateau.

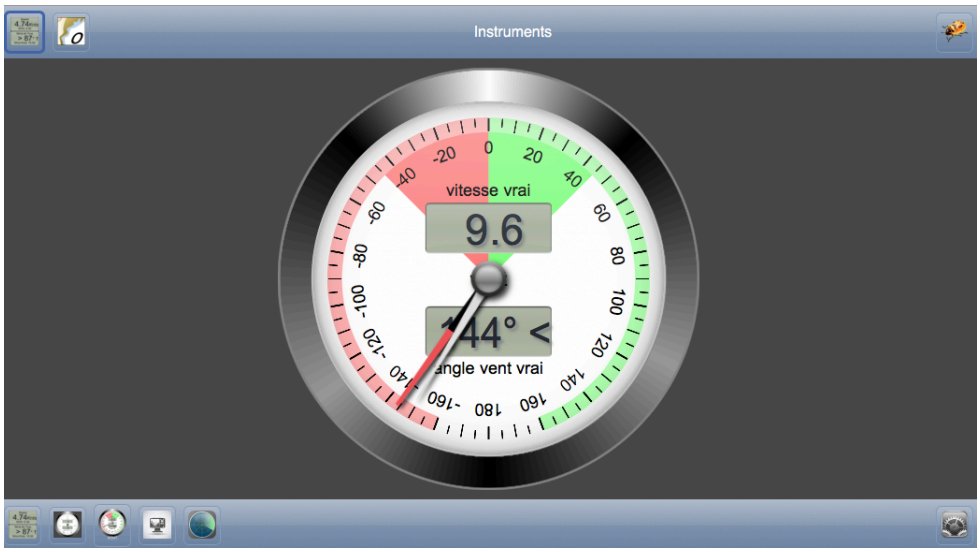
Screen shots web managment

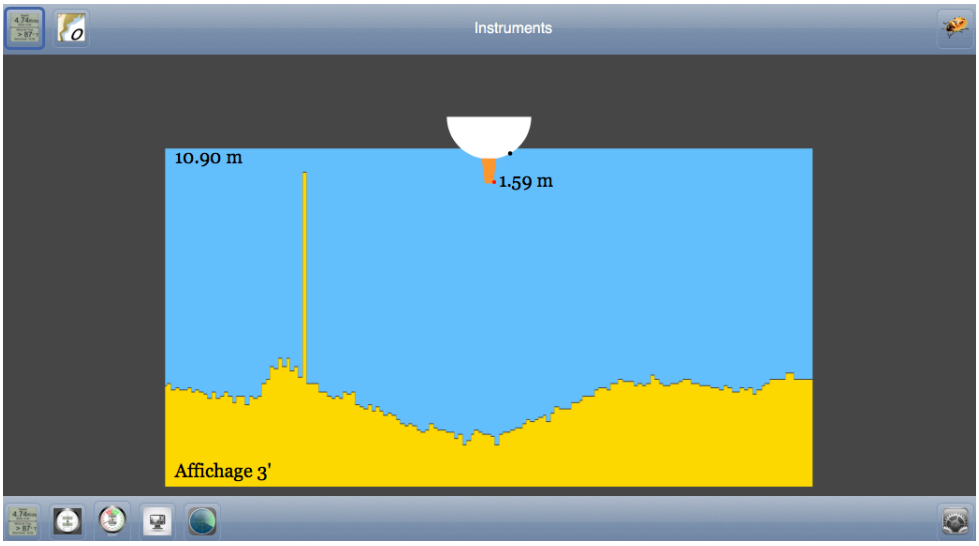




Screen shots repetiteur multi smartPhones tablettes






Instruments
10.90 m
1.59 m
Affichage 3'

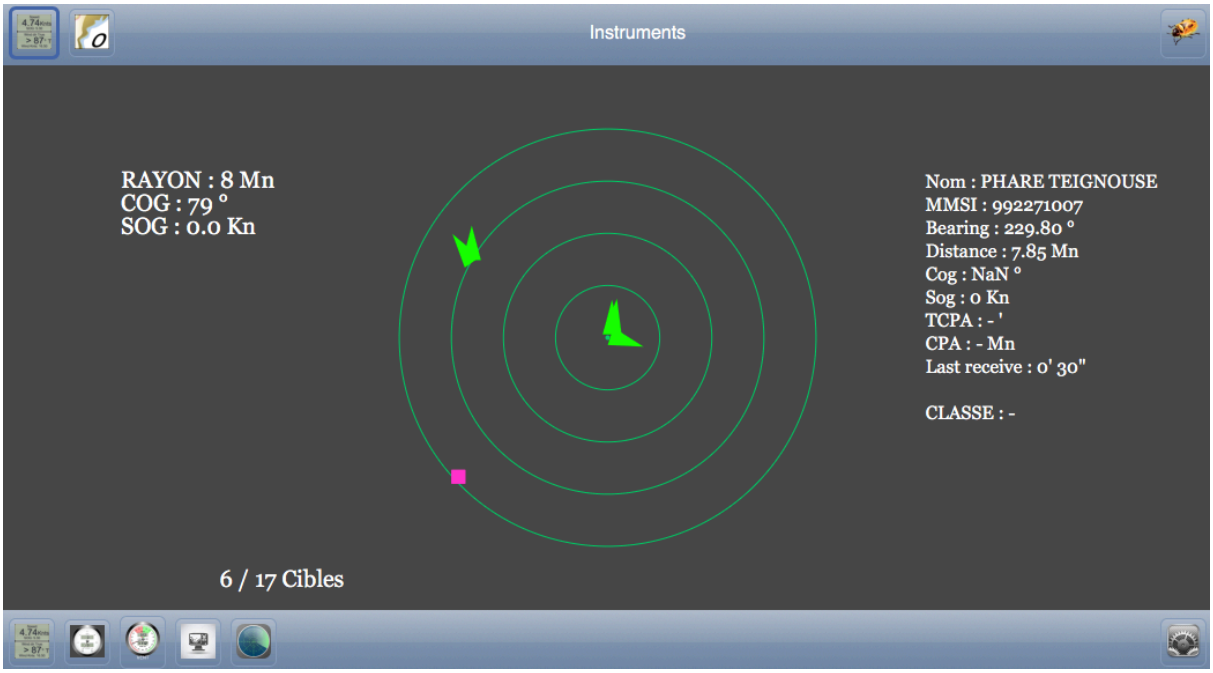

Instruments
RAYON : 8 Mn
COG : 79 °
SOG : 0.0 Kn
Nom : PHARE TEIGNOUSE
MMSI : 992271007
Bearing : 229.80 °
Distance : 7.85 Mn
Cog : NaN °
Sog : 0 Kn
TCPA : - '
CPA : - Mn
Last receive : 0' 30"
CLASSE : -
6 / 17 Cibles

Diagramme fonctionnel nmeaHub

Flux NMEA0183

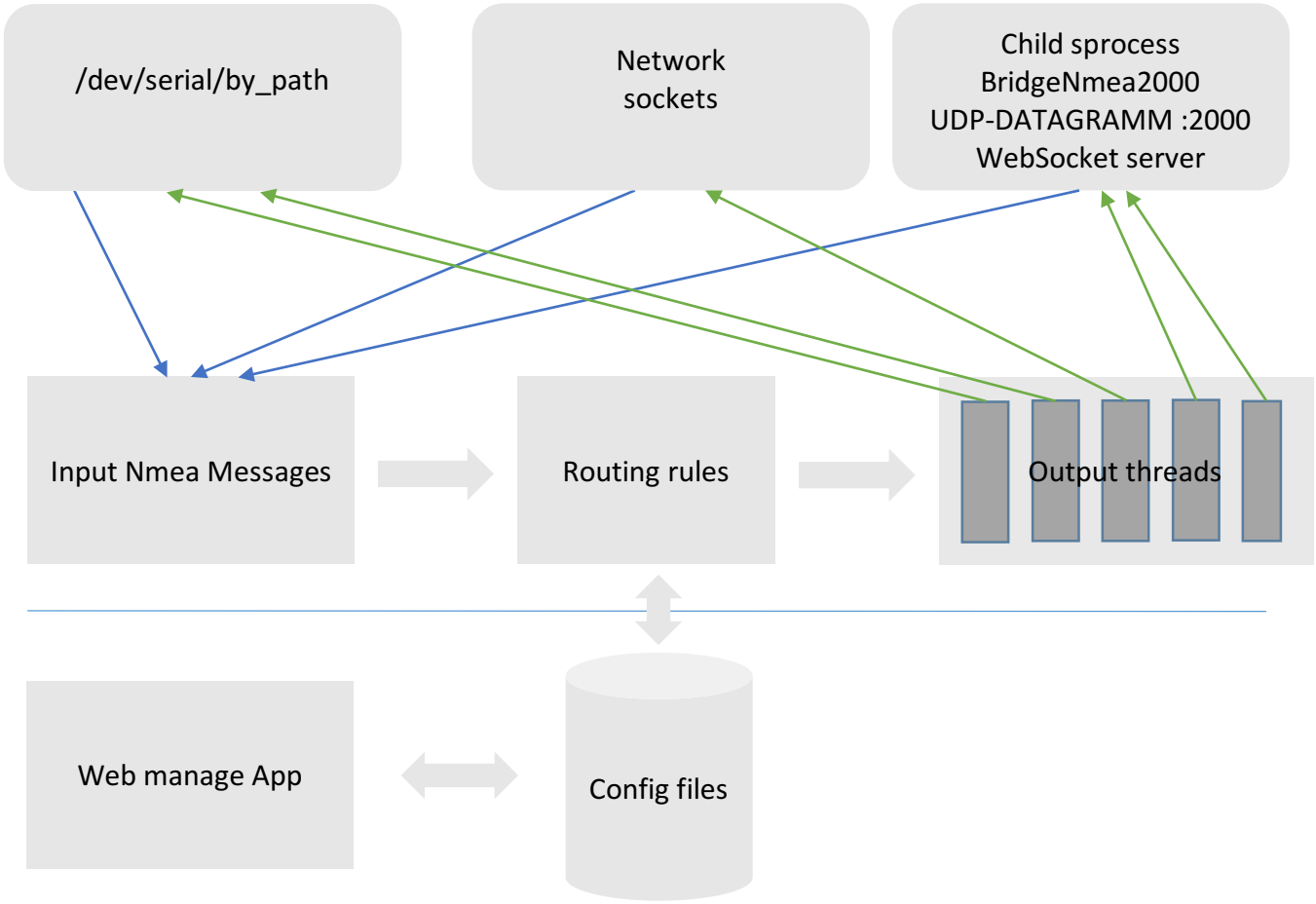
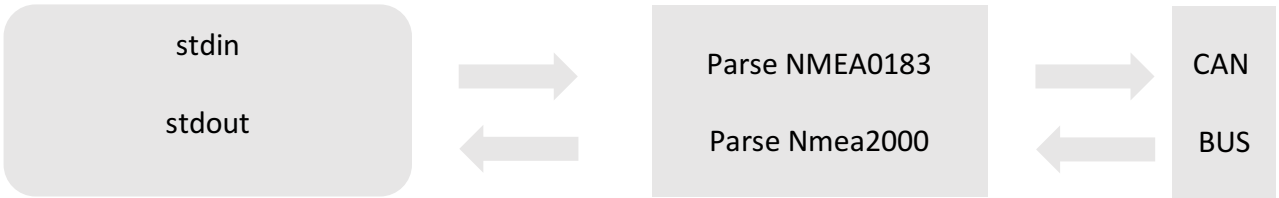


Diagramme fonctionnel bridgeNmea2000



NmeaHub

Parametres de lancement :

-h help
-r génère RMB et XTE sur réception RMC (Compatibilité répétiteur GPS NASA)
-w génère VWV sur réception VWR (compatibilité répétiteur SIMRAD GMI 20)
-g génère GLL sur réception GGA (compatibilité Raymarine S2000)
-f n [1,10] filtre SOG etc COG sur n valeurs pour lissage
-d n niveau de debug de 0 a 9
-p n port d'écoute IP (3333 par défaut). 4 ports successifs sont crées (n, n+1 … n+3) de manière à pouvoir différencier plusieurs process clients disposants d'une même adresse IP.
Un port en listen : UDP 0xBEE est par ailleurs crée pour la gestion des interfaces wifi XBEE voir https://www.digi.com/products/xbee-rf-solutions/2-4-ghz-modules/xbee-wi-fi

Les fichiers de configuration :

Ils se trouvent sous le répertoire /var/www/data/nmeaHub afin d'être partagés avec l'outil de configuration WEB.
nmeaHub.conf : est maintenu par nmeaHub et l'outil WEB normalement ne pas y toucher.
nmeaHub.def : comporte une brève description de chaque message NMEA0183, la définition de quelques modèles d'instruments, la définition des processus fils ainsi que la gestion des ports RS232 :

Exemple model générique GPS :

```
GENERIC|GPS|GGA|1|0|GLL|1|0|GSA|1|0|GSV|1|0|VTG|1|0|RMC|1|0|ZDA|1|0|ZZU|1|0
```

Pour chaque phrase NMEA :

- Niveau de priorité pour la phrase entrante.
- La phrase sera transmise ou non au périphérique.

Par exemple, si votre GPS est en priorité 1 et que votre AIS dispose d'un GPS il transmettra aussi les phrase GPS. Si l'est paramétré en priorité 2, les phrases de l'AIS seront ignorées tant que le GPS sera opérationnel. Si vous coupez votre GPS, l'AIS prendra automatiquement le relais.

Le paramétrage des fils :
Les processus fis sont des processus qui reçoivent en entrée standard les phrases filtrées et peuvent transmettre sur leur sortie standard des phrases NMEA (par exemple issue du bus N2K).
nmeaHub lance les processus fils au démarrage, s'ils tombent il seront relancés.
Le paramétrage comporte le mot clé CHILD, un commentaire et les arguments de chaque process.

```
CHILD|NMEA2000|bridgeNmea2000
CHILD|UDP-2000|socat|-|UDP-DATAGRAM:255.255.255.255:2000,broadcast
```

CHILD|NMEA-SERVER (3337)|/var/www/data/bin/nmeaServer|3337

La gestion des ports RS232 :

Les ports sont régulièrement scannés de manière à assurer une configuration plug and play. Par défaut, les fichiers scrutés sont /dev/ttyUSB* et /dev/serial/by_path/*. Ce dernier répertoire offre l'avantage de nommer le périphérique selon la hiérarchie matérielle des connexions physiques. Il en résulte qu'un équipement connecté sur un port physique donné de votre raspberry sera toujours connu sous le même nom. Cela évite les problèmes classiques de nommage des ports série sous linux où ttyUSB1 peut devenir ttyUSB0 au prochain reboot si un équipement est déconnecté.

Les ports sont ouverts par défaut en 4800 baud. La vitesse peut être forcée à une autre valeur par l'outil de gestion WEB. Si nmeaHub détecte des ereurs en lecture sur un port donné, il bouclera successivement avec les vitesses suivantes : 38400, 9600, 19200, 56700 4800 … finissant ainsi par trouver la bonne vitesse.

Pour modifier la liste des ports scrutés ou interdire l'utilisation d'un port, il est possible d'ajouter des règles dans le fichier nmeaHub.def. Les deux entrées réservée à cet usage sont SCANTTY et NOSCANTTY.

Exemple :
SCANTTY:/dev/serial/by_path
SCANTTY:/dev/USB*
NOSCANTTY:/dev/USB0

Dans cet exemple, tous les devices présents dans /dev/serial/by_path ou dont le nom est /dev/USBxxx sauf le port /dev/USB0 seront pris en compte.

Si par exemple, vous souhaitez gérer votre gps avec gpsd, il vous faudra définir le port RS232 correspondant au GPS dans la configuration gpsd et utiliser une règle NOSCANTTY pour ce port.

nmeaHub.stat : est maintenu par nmeaHub et permet d'afficher dans l'outil de gestion WEB le nombre de phrases recues et transmises pour chaque périphérique.

La gestion de l'interface réseau :

nmeaHub écoute sur 4 ports ip , par défaut 3333,3334,3335,3336. Sur demande de connexion, celle-ci est acceptée et le flux NMEA0183 démarre. A la première connexion, une ligne avec la clé adresse-ip :port est enregistrée dans le fichier de configuration nmeaHub.conf avec le profil générique DEFAULT (uniquement phrases sortantes).
Le périphérique est immédiatement accessible avec l'outil WEB de configuration pour être paramétré selon vos préférences.

La particularité des modules WIFI-XBEE est la suivante, ces modules sont paramétrés pour envoyer un message broadcast sur le port 0xBEE (3054). A la réception de ce message, le serveur nmeaHub établit une connexion de type socket avec le périphérique. Le module WIF-XBEE travaille alors en mode transparent avec l'instrument auquel il est connecté. Il apporte donc une couc he réseau à n'importe quel instrument NMEA, permettant ainsi de s'affranchir du câblage en étoile nécessaire au système NMEA0183.

Gestion des files d'attente :

Une file d'attente en écriture est gérée pour chaque périphérique par un thread dédié. Celle-ci est dimensionnée à un maximum de 8 messages. Si la file d'attente est pleine, elle est tout simplement purgée pour éviter de stocker des messages qui ne pourront probablement pas être transmis au récepteur.
Dans ce cas, le log comportera un message indiquant quel périphérique est en défaut et détaillant l'état de toutes les files d'attente.
Les conditions d'engorgement peuvent avoir plusieurs origines :

- Trop de messages sur une ligne RS232 à 4800bd.
- Congestion du réseau.

L'état des files d'attentes peut être visualisé en envoyant le signal USR1 au precessus nmeaHub (kill –USR1 pid), Le message est envoyé sur la sortie erreur, voir le fichier /var/log/nmeaHub.log.

Le mode safe de muxberry

Le mode safe permet de faire fonctionner linux en préservant l'intégrité de la carte sd du raspberry. Pour cela, linux est paramétré de manière à protéger les files systèmes de la carte / et /boot, mode read only. Le raspberry peut être arrêté sauvagement sans subir de dommages, il peut donc être alimenté par l'interrupteur de la centrale de navigation.
Le système est basé sur quelques scripts qui utilisent unionfs-fuse , capable de présenter un système de fichier virtuel composé d'une base en lecture seule et d'une partition en débordement pour l'écriture.
Dans des cas d'utilisation tel que le multiplexeur, une simple ram disk est suffisante, il n'y a pas à sauvegarder les paramètres de session.
Dans le cas d'une application plus complexe comme openCpn ou zyGrib, les préférence utilisateur et différents fichiers de session nécessitent de sauvegarder les données de manière durable afin de les retrouver lors du prochain reboot.
Le système utilise pour cela une clé USB sur laquelle seront stockées toutes les données de session.

Les scripts se trouvent dans le répertoire /usr/local/bin.

Le script format_usbkey format une clé usb en ext3 et la nomme seaBerry.

Attention, format_usbkey travaille sur /dev/sda1, il ne fera pas de différence s'il y a plusieurs clés connectées au raspberry. Assurez-vous bien que seule la clé à formater est connectée aux ports du rasberry avant de lancer le script.

Mount_usbkey DIR (ou DIR est /tmp ou /home ou /var ou autre)
Vérifie la présence d'une clé
Effectue fsck sur la clé (au premier appel)
Monte la clé sous le répertoire /usbkey
Renome si necessaire $DIR en $DIR_orig et crée (vierge) $DIR pour le point de montage.
Test le mode de montage de la partition /
    Si mode rw => bind entre $DIR_orig et $DIR
    Si mode ro
        Si clé USB non présente
            Crée $DIR_RW = /ramdisk/$DIR
            Crée ramdik montée sur $DIR_RW
        FinSi
    Si non
        Crée si nécessaire /usbkey/$DIR
        $DIR_RW = /usbkey/$DIR
    finSi
    Fusionne sous $DIR $DIR_orig (readOnly) $DIR_RW (readWrite)

L'appel du script mount_usbkey se fait dans le fichier /etc/fstab :

```
proc            /proc           proc    defaults            0   0
/dev/mmcblk0p6  /boot           vfat    ro,defaults         0   2
/dev/mmcblk0p7  /               ext4    ro,defaults,noatime 0   1
/usr/local/bin/mount_usbkey    /var     fuse    cow,user    0   0
/usr/local/bin/mount_usbkey    /tmp     fuse    cow,user    0   0
/usr/local/bin/mount_usbkey    /home    fuse    cow,user    0   0
```

Attention, eviter de se contenter d'un mount –o remount,rw / pour être en mode lecture criture et effectuer des tâches diverses. J'ai eu un problème avec apt-get qui à désynchronisé sa base car une partie s'est retrouvée sur la clé USB.

Pour revenir en mode readWrite temporaire, utiliser la commande « modeProtect stop »
Pour revenir en mode readOnly utiliser la commande « modeProtect start »

Pour passer de manière permanente en readWrite :
sudo d'un mount –o remount,rw /
Editer le fichier /etc/fstab, changer les attributs ro de / et /boot en rw
sudo reboot.

Le process bridgeNmea2000

Ce module effectue le correspondance des phrases nmea0183 en messages N2K et vice versa.
Il est basé sur l'excellente librairie https://github.com/ttlappalainen/NMEA2000/wiki

bridgeNmea200 est un processus indépendant de nmeaHub.

Les logs sont enregistrées dans /tmp/bridgeNmea2000.log
Le mode trace debug peut être activé par la commande kill –USR1 pid(bridgeNmea2000) et désactivé par la même commande.
Le mode trace est particulièrement bavard.

Script d'installation

Le fichier muxberry.tar comporte l'ensemble des modules du projet muxberry et un script d'installation conçu pour être installé à partir d'une version raspbian stretch lite https://www.raspberrypi.org/downloads/raspbian/ mais peut être installé sur une autre installation.

Compte tenu des problèmes rencontrés lors des premières livraisons, l'installation est non destructive de l'environnement existant, elle sauvegarde tous les fichiers de configuration modifiés et prépare l'exécution des modules installés mais laisse à l'utilisateur le choix de les démarrer.

Recopier sur le raspberry le fichier muxberry.tar.gz

gunzip muxberry.tar.gz
sudo tar xvf muxberry.tar
cd muxberry
sudo ./install.sh

Tous les fichiers impactés sont recopiés en « fichier-orig ».
Le script install.sh peut être lancé plusieurs fois de suite sans dommage, dans ce cas les fichiers de sauvegarde file–orig sont enregistrés séquentiellement en file-orig-n.
Les modifications de configuration réseau (gestion du mode AP) ne sont pas appliquées par l'installation, le script hostAP s'en chargera dans un deuxième temps.
Les modifications de configuration du fichier fstab pour le mode safe ne sont pas appliquées par l'installation, vous devez vous-même recopier /etc/fstab.safe en /etc/fstab et rebooter pour activer les modifications.
L'arrêt du swap est recommandé si votre raspberry est dédié à la fonction multiplexeur et sans interface graphique.

Module	Fonctionnalité	Fichier impacté
Support module CAN	Modifie les paramètres de configuration pour le support d'une carte CAN	/boot/config.tx
Support mode acces point	Transforme le raspberry en Acces Point SSID muxberry password muxberry	/etc/hostapd/hostapd.conf /etc/default/hostapd /etc/network/interfaces /etc/dnsmasq.conf /etc/hosts /etc/hostname
Installation fichiers	Copie des exécutables dans /usr/local/bin Copie des fichiers www Droits d'exécution sudo de /var/www/data/bin/remountRead et remountWrite	/ets/sudoers

Arret du swap	Désactive le swap	
rc.local	Verrue pour le bon fonctionnement du service canBus. Création d'une route default en mode AP	/etc/rc.local
Mode safe	Permet de faire tourner le raspberry avec la carte sd en mode readOnly, les débordements en écriture sur /home /var et /tmp sont gérés soit en ramdisk soit sur une clé usb.	/etc/fstab.safe

Après l'installation de ces différents modules :

Vous devez rebooter pour prendre en compte le support de la carte CAN.

Pour démarrer nmeaHub :

Démarrage temporaire (pour test) service nmeaHub start
Attention, nmeaHub s'approprie tous les ports USB SERIAL.
Démarrage persistant systemctl enable nmeaHub

Pour lancer une simulation :
/var/www/data/simulNmea.sh
Deux sources de simulation sont disponibles, vous pouvez éditer le script et changer la source. Le fichier traces-raspi-mux.log est une trace statique enregistrée au port du Crouesty et comporte des messages AIS lus par l'application répétiteur multi. Le fichier tracesGps.log ne comporte pas de messages AIS, il simule un parcours de régate en baie de Quiberon.

Paramétrer votre multiplexeur :
http://muxberry/multiplexer

Accéder au module répétiteur multi :
http://muxberry

Visualiser les données :
Lancer un telnet depuis un autre ordinateur vers le raspberry sur le port 3333
Ex : telnet muxberry 3333

Paramétrer votre OpenCpn :
Choisir port TCP 3333 ou 3334 ou 3335 ou 3336 (mode read et write)
Ou Port UDP 2000 (en mode readOnly)

Le mode AP (access point).
Ce mode de fonctionnement vous permet d'avoir un bord un réseau wifi privé, dédié à la navigation.
Le paramétrage de base lance simultanément deux interfaces :

wlan0 : en mode wpa
uap0 : en mode AP SSID MUXBERRY passwd muxberry

Plus d'informations sur le paramétrage :
https://blog.ithasu.org/2017/11/raspberry-pi-0w-as-both-wifi-client-and-access-point/

Passer votre muxberry en mode safe
sudo cp /etc/fstab.safe /etc/fstab
reboot

Attention, sans clé usb dédiée à l'enregistrement des données utilisateur, toutes les informations de session (sauf paramétrage du multiplexeur) seront perdues.
Les données sur /tmp /home et /var sont enregistrées en ramdisk.

Si le swap est resté actif, ou si vous avez une interface graphique avec openCpn, il est recommandé d'utiliser une clé usb sur laquelle seront enregistrés tous les fichiers applicatifs.

Formatage de la clé :
Insérer la clé dans un port usb du raspberry, et veiller à ce que seule cette soit connectée.
Lancer l'utilitaire format_usbkey

Si vous êtes en mode safe, le prochain reboot prendra en compte votre clé pour les répertoires /tmp /var et /home.

Permettre temporairement la modification de fichiers lorsque l'on est en mode safe :
sudo modeProtect stop
Revenir dans le mode safe
sudo modeProtect start

Passer de manière permanente en mode rw :
sudo mount –o remount,rw /
Editer le fichier /etc/fstab et modifier les attributs ro des filesystem / et boot en rw.
rebooot
Une fois le mode safe installé, les répertoires /tmp /home et /var sont renommés en /tmp_orig /home_orig et /var_orig. Ne jamais supprimer ces répertoires.
Le script mount_usb se charge de faire le lien entre les répertoires d'origine et les répertoires system.

Pour revenir au mode d'origine de raspbian, il vous faudra monter votre sd carte sur une autre machine linux, renommer les répertoires –orig et reinitialisr le fichier fstab.

Inérer la carte sd dans le port usb d'une machine linux.
mount /dev/sda2 /mnt

cd /mnt
rm –rf home var tmp_orig
mv home_orig home
mv var_orig var
cd etc
cp fstab-orig fstab
cd /
umount /mnt

Votre carte est prête à redémarrer dans sa configuration d’origine.