Installation et configuration OpenCPN / Pypilot sur un Raspberry PI 5 avec OS BookWorm hors OpenPlotter

Ce petit guide permet de se passer d'OpenPlotter qui a mon goût est certes d'une grande aide mais trop opaque au niveau des installations. Le but ici est double : fournir une procédure complète d'installation avec tous les bons packages sur un raspbbery pi 5 avec l'os BookWorm 64 bits version Desktop pour utiliser OpenCpn et Pypilot avec le strict nécessaire, comprendre un peu mieux comment fonctionne tout le processus d'un Pypilot installé « proprement ».
Ce manuel s'adresse a des personnes maîtrisant linux et familiers du mode ligne de commande n'ayant pas peur de configurer dans les moindres détails un raspberry pi.

Ce système étant basé sur une configuration matérielle fixe : un PI 5 et un certain type de capteur inertiel, rien n'empêche pour les moins audacieux d'avoir juste l'image stable correspondant à toute cette installation OpenCpn PyPilot sur un système BookWorm. Cette image pourra être fournie au personnes intéressées, pour cela me contacter : yves.curel@gmail.com

Près requis pour OpenCpn et Pypilot disposer sur le PI 5 d'un capteur inertiel ou IMU et disposer d'un contrôleur de moteur.
Contrôleur de moteur de pilote:
Accessible sur le site https://navitop.fr ou sur le site https://pypilot.org/opencart/
Le site navitop.fr a l'avantage de fournir des systèmes pypilot marinisés et bien configurés.
Le site pypilot.org/opencart est pas toujours ouvert et c'est au bon voilier du concepteur génial de ce système Pypilot Sean Depagnier.

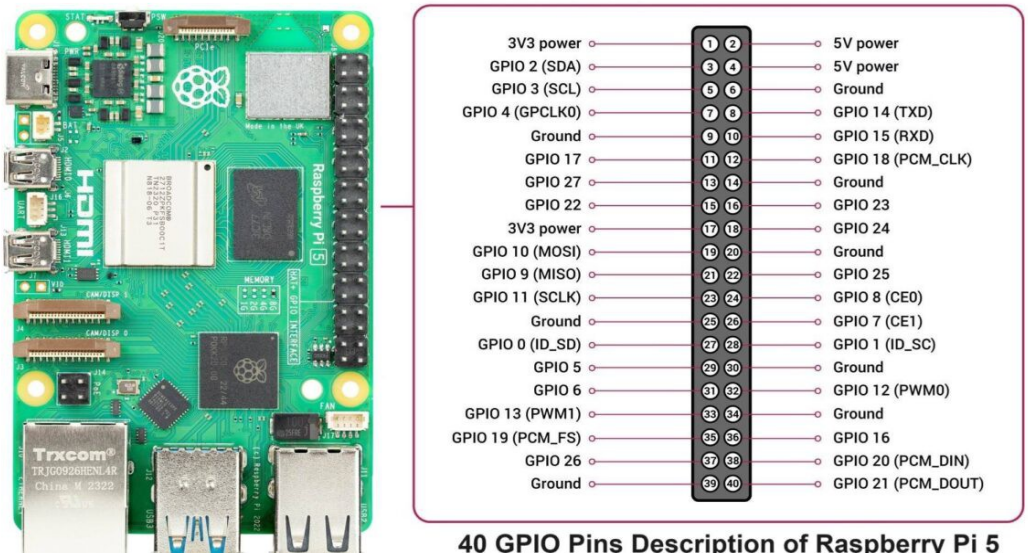
Schéma de câblage sur raspberry PI 5 :
Carte Ctrl moteur RaspBerry PI4

3,3 V + _____ 3,3V+ Pin 17

GND _____ GND Pin 20 ou 30

TX _____ RX pin 10

RX _____ TX Pin 8



40 GPIO Pins Description of Raspberry Pi 5

IMU basé sur TDK icm-20948 :
Mon choix s'est porté sur ce modèle précis:
https://shop.pimoroni.com/products/icm20948
N'étant pas électronicien mais informaticien je vois que l'alimentation VCC va de 2v a 5v ce qui est plutôt rassurant… Si on regarde le site Navitop on voit que le TDK icm-20948 accepte normalement 1,8v sur le pin 8, ici protection en amont sur ce circuit de pimorini.com ?

Schéma câblage sur le raspberry pi 5 :

ICM-20948 Module	**Raspberry Pi 5 GPIO**
VCC	——————→ 3V3 (Pin 1)
GND	——————→ GND (Pin 6 ou Pin 9)
SDA	——————→ SDA (Pin 3)
SCL	——————→ SCL (Pin 5)

Pour commander cet imu plusieurs sites sur internet par exemple :
https://www.gotronic.fr/art-module-icm20948-9-dof-pim448-31595.htm

Installation et configurations des logiciels et packages nécessaires pour faire fonctionner OpenCpn et PyPilot

Installer l'image de bookworm sur une carte SD avec le logiciel Raspberry pi imager :

https://downloads.raspberrypi.com/raspios_oldstable_arm64/images/raspios_oldstable_arm64-2025-10-02/2025-10-01-raspios-bookworm-arm64.img.xz

Une fois cela fait appliquer au démarrage de bookworm : langue et choix du wifi utilisé.

Configuration de l'IMU

activer i2c :
soit en mode ligne de commande taper :
sudo raspi-config nonint do_i2c 0
soit aller dans le menu préférences configuration raspberry pi et activer I2C
activer aussi : Serial Port et Serial Console et accessoirement : SSH et VNC

éventuellement pour mise a jour des dépendances taper:
sudo apt update

taper :
sudo apt install -y python3-pip python3-dev i2c-tools

pour testee l'imu taper:
i2cdetect -y 1
on obtient normalement un grille d'adresse avec 0x68

installer la bibliothèque python pimoroni présente dans

https://github.com/pimoroni/icm20948-python

suivre les instruction d'installation présentes sur ce site github taper ces commandes :

git clone https://github.com/pimoroni/icm20948-python
cd icm20948-python
./install.sh

pour installer la librairie taper : pip install icm2094

Pour mise a jour de l'environement virtuel python correspondant taper :
python3 -m venv --system-site-packages $HOME/.virtualenvs/pimoroni
Pour switcher sur cet environement taper:
 source ~/.virtualenvs/pimoroni/bin/activate

Pour tester l'imu taper :
source ~/.virtualenvs/pimoroni/bin/activate
pip show icm20948
si bibliothèque python bien chargée on doit avoir:
Name: icm20948
Version: 1.0.0
Summary: Python library for the ICM20948/AKA09916 9-DOF IMU
…...

A partir du répertoire ou on a lancé l'installation exemple ici :
/home/yves/install_navyc, un sous repertoire est crée avec toutes les sources et exemples:
/home/yves/install_navyc/icm20948-python

Pour vérifier et éventuellement calibrer l'imu taper :

cd /home/yves/install_navyc/icm20948-python/examples

lancer:
python3 read-all.py
on alors l'affichage successif des valeurs de l'imu toutes les 0.25 secondes

Accel: -0.05 00.01 00.99
Gyro: -1.22 01.47 00.47
Mag: -42.00 20.85 21.30

Accel: -0.05 00.01 00.99
Gyro: -1.06 01.63 00.63
Mag: -42.30 18.90 22.80

Accel: -0.05 00.02 01.00
Gyro: -1.31 01.44 00.53
Mag: -41.70 21.00 23.25
.....
.....
.....

pour calibrer l'imu on peut lancer soit :
python3 bargraph.py
python3 magnetometer.py
python3 magnetomer-to-rgb5x5.py

Installation OpenCPN :
Taper :
sudo apt install python3 python3-pip python3-serial git python3-numpy
git clone https://github.com/pypilot/pypilot.git
cd pypilot
sudo python3 setup.py install

Normalement à ce jour une version récente d'OpenCPN est alors installée : la version 5.12.4
Puis installer les Cartes et le plugin PyPilot

Éventuellement installer SignalK amis cela n'est pas obligatoire si on part sur une version légère avec le strict minimum d'autant plus que signalK rajoute une couche de communication et donc des temps de latence supplémentaires ce qui n'est jamais bon par rapport aux trames NMEA0183 qui peuvent être paramétrées et envoyées ici directement vers OpenCpn et son Plugin Pypilot.
Selon moi dans cette configuration là l'intérêt de signalK réside dans le fait que si notre pi 5 est point d'accès on peut ensuite envoyer les données de nos capteurs et nos informations de navigation via SignalK sur notre « web local » avec d'autres appareils et applications consommatrices de ces ressources.

Attention SignalK est basé sur un NodeJS récent, taper :
curl -fsSL https://deb.nodesource.com/setup_22.x | sudo -E bash -
sudo apt install -y nodejs

pour vérifier version taper :
node -v
npm -v

Pour installer signalK taper :
sudo npm install -g --unsafe-perm signalk-server

pour lancer signalK taper :
signalk-server

On accède alors à SignalK en tapant :
http://localhost:3000
ou depuis un autre appareil du réseau local :
http://<adresse_IP_du_Pi>:3000

Pour le rendre permanent (démarrage au boot) taper:
sudo npm install -g --unsafe-perm signalk-server-setup
sudo signalk-setup enable

Cela crée un service systemd nommé signalk.service
On peux vérifier son état en tapant:
sudo systemctl status signalk
On doit voir alors :
Active: active (running)

Et l'activer au démarrage si ce n'est pas déjà fait , taper:
sudo systemctl enable signalk
sudo systemctl start signalk

Installer pypilot

éventuellement pour mise a jour des dépendances taper:
sudo apt update
sudo apt install python3 python3-pip python3-serial git python3-numpy

git clone https://github.com/pypilot/pypilot.git
cd pypilot
sudo python3 setup.py install

laisser se dérouler l'installation jusqu'a la fin du script d'installation on a ensuite ce message :
running install_scripts :
Installing pypilot script to /usr/local/bin
Installing pypilot_boatimu script to /usr/local/bin
Installing pypilot_calibration script to /usr/local/bin
Installing pypilot_client script to /usr/local/bin
Installing pypilot_client_wx script to /usr/local/bin
Installing pypilot_control script to /usr/local/bin
Installing pypilot_hat script to /usr/local/bin
Installing pypilot_scope script to /usr/local/bin
Installing pypilot_servo script to /usr/local/bin
Installing pypilot_web script to /usr/local/bin

Si on va dans le répertoire /usr/local/bin où sont installés tous les scripts des différents modules de pypilot (pypilot_*) il ne suffit pas de les lancer pour faire marcher Pypilot cela serait trop simple…. ! Une étape de configuration de tout cela va être nécessaire.

Configurer PyPilot :

Après installation dans le répertoire /home/<user>/.pypilot se trouvent les fichiers de configuration de pypilot.
Nmea0device fichier de configuration des autres device communiquant avec OpenCPN
exemple ici mon GPS sur le port USB :
["\/dev\/serial\/by-id\/usb-u-blox_AG_-_www.u-blox.com_u-blox_7_-_GPS_GNSS_Receiver-if00",38400]
Ce fichier ne doit pas être renseigné et il le sera au premier démarrage correct de pypilot.

persist_fail fichier log des erreurs. Ce fichier ne doit pas être renseigné.

pypilot_client.conf ce fichier concerne les réglages des clients pypilot, il ne doit pas être renseigné il le sera automatiquement au premier démarrage correct de pypilot. Son contenu après premier démarrage correct : {"host":"127.0.0.1","port":23322}

servodevice ce fichier ne doit pas être renseigné il le sera automatiquement au premier démarrage correct de pypilot. Son contenu après premier démarrage correct : ["\/dev\/ttyAMA0",38400] (Cela correspond aux pins RX (pin 8) et TX (pin 10) sur lesquels sont branchés les fils TX et RX du contrôleur du moteur de pilote).

Pour une prise en compte des bons paramètres au démarrage de pypilot avec le plugin Pypilot dans OpenCPN il est nécessaire de mettre a jour le **fichier de configuration principal de pypilot pypilot.conf**

Exemple :/home/yves/.pypilot/pypilot.conf

```
#general
autostart = true
log = true
#network
# Adresse et ports NMEA pour OpenCPN
nmea_port = 20220
nmea_host = 0.0.0.0      ; écoute sur toutes les interfaces
nmea_broadcast = true
# Permet la connexion du plugin OpenCPN
client_port = 20223
web_port = 8080
#IMU
device = icm20948
bus = 1
address = 0x68
#rate = 20              ; fréquence en Hz
orientation = 0, 0, 0  ; à ajuster après montage fixe de l'imu
#servo
# optionnel pour le moteur de barre
servo_min = 0.05
servo_max = 0.95
servo_center = 0.5
invert = false
#ap
# Réglages de base de l'autopilot
mode = compass
pid.Kp = 1.0
pid.Ki = 0.1
pid.Kd = 0.02
```

Copier / coller cette configuration minimum de démarrage dans ce fichier de configuration principal pypilot.conf a pour effet de permettre à OpenCpn via son plugin Pypilot de communiquer correctement avec Pypilot qui peut peut être ici lancé la premier fois par la commande :

```
cd /usr/local/bin
sudo pypilot
```

Ce fichier ne sera plus a modifier car il sera mis ensuite a jour de façon dynamique par les réglages que l'on appliquera a notre pypilot via OpenCpn et son plugin Pypilot : calibration imu, réglages pilote etc.…

Une fois ce script **pypilot** (ici version 0.56) qui est cœur du système lancé, lancer OpenCpn et les paramétrages plus fins de pypilot vont pouvoir commencer…..

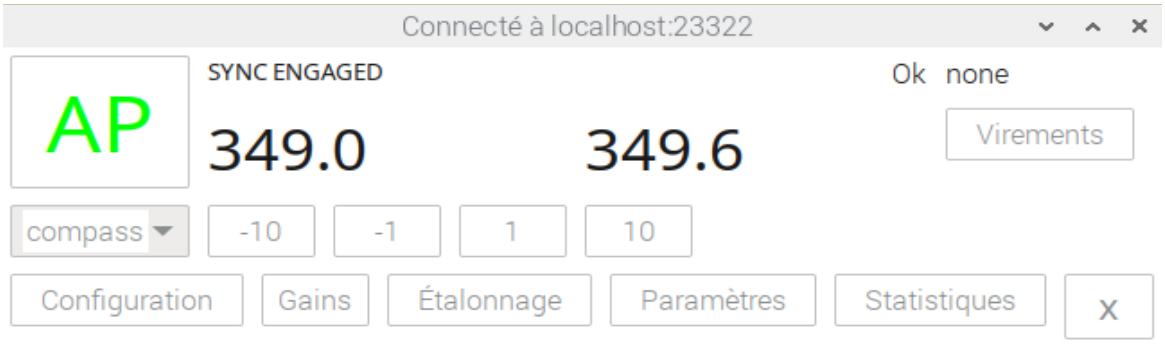
Paramétrages Pypilot dans OpenCpn :

	Protocole	Entrée/Sortie	Port des données	Statut		
☑	nmea0183	Entrée	TCP:localhost:20220	✓	⚙	🗑
☑	nmea0183	Entrée	GPSD:localhost:2947	✓	⚙	🗑

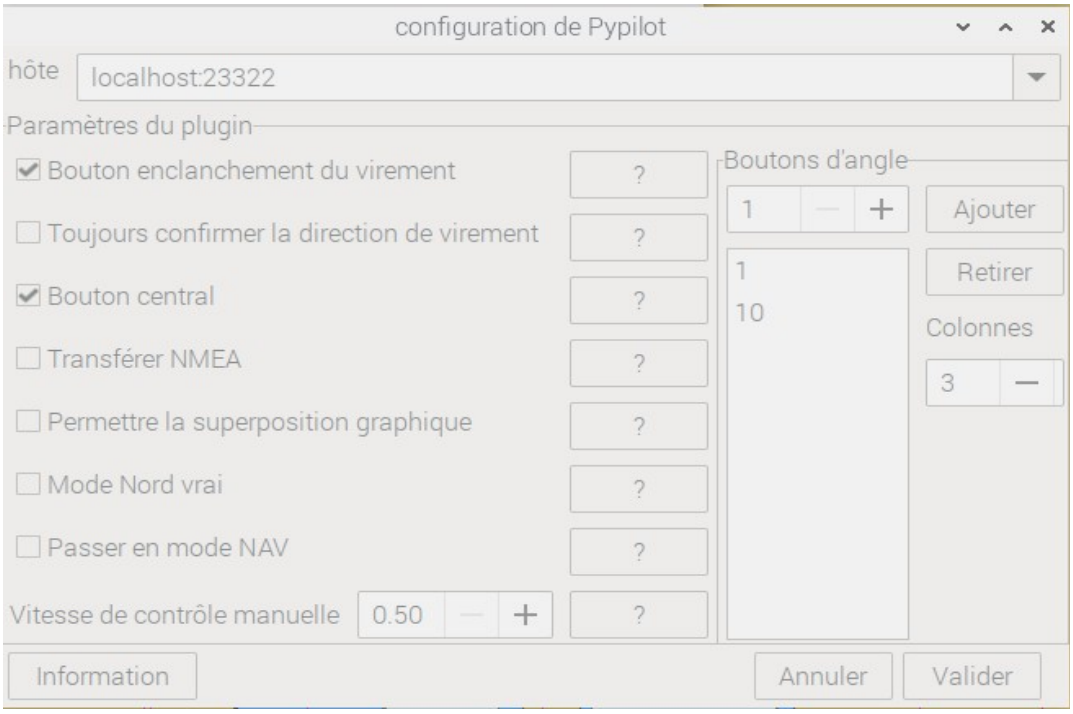
Ici mon GPS est installé sur le port USB : dev/ttyACM0 et TCP:localhost:20220 correspond à l'IMU et a toutes les data NME0183 qui viennent du service Pypilot qui doit être actif.

Attention pour le GPS il est préférable d'utiliser le service GPSD qui a l'avantage de sortir le trames NMEA0183 du GPS sur son port TCP associé 2947 cela présente plein d'avantages par rapport à Pypilot on verra cette configuration du GPS dans un prochain chapit

Lancement du plugin OpenCPN Pypilot (version 0.53.2)

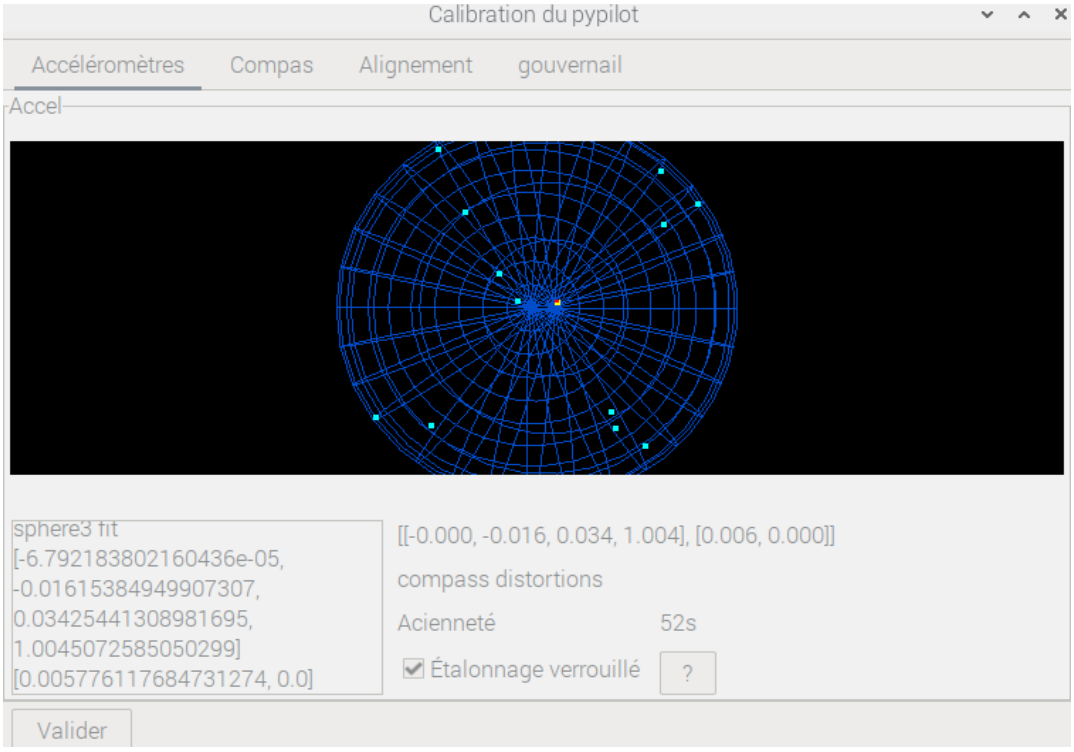


Clic sur Configuration

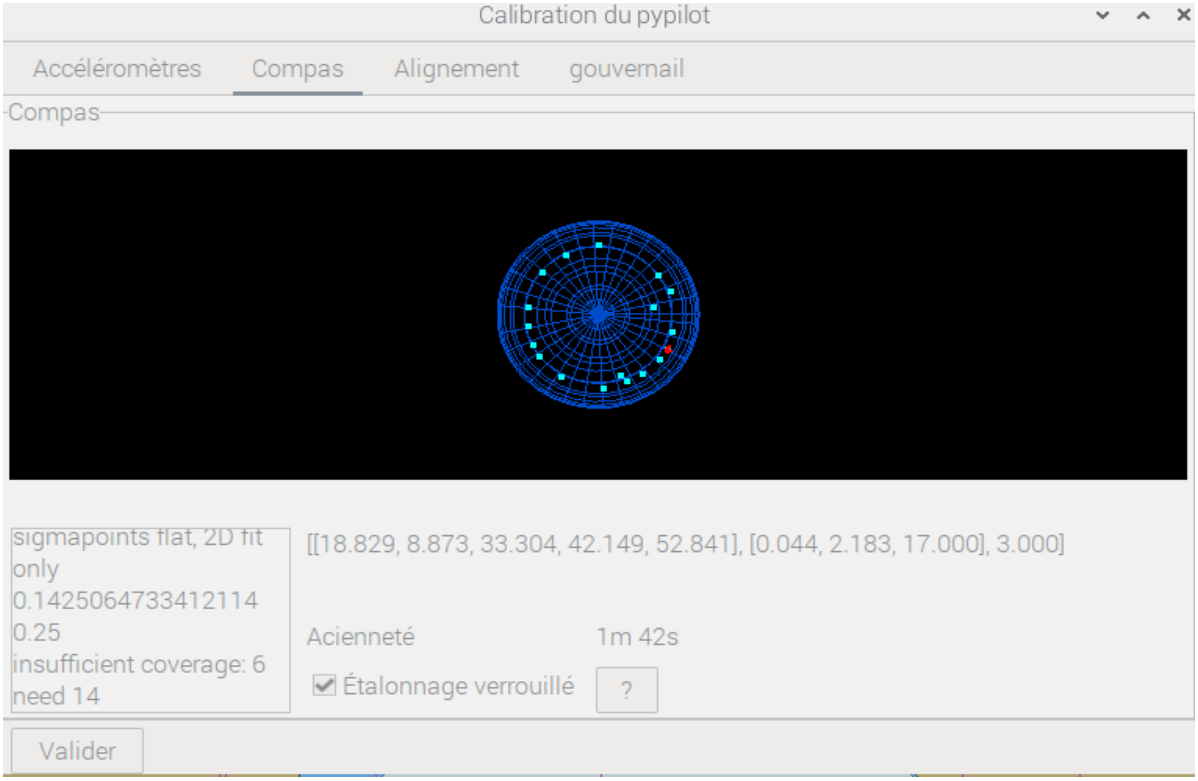


Clic sur Étalonnage

Étalonnage accéléromètre : Déplacer doucement l'imu sur tous ses axes une fois ses points bleus à l'intérieur de la sphère, verrouiller l'étalonnage et valider



Étalonnage Compas



Premier test du moteur du pilote
Une fois validés les étalonnages de l'IMU on clique sur AP pour engager le contrôleur avec le moteur du pilote

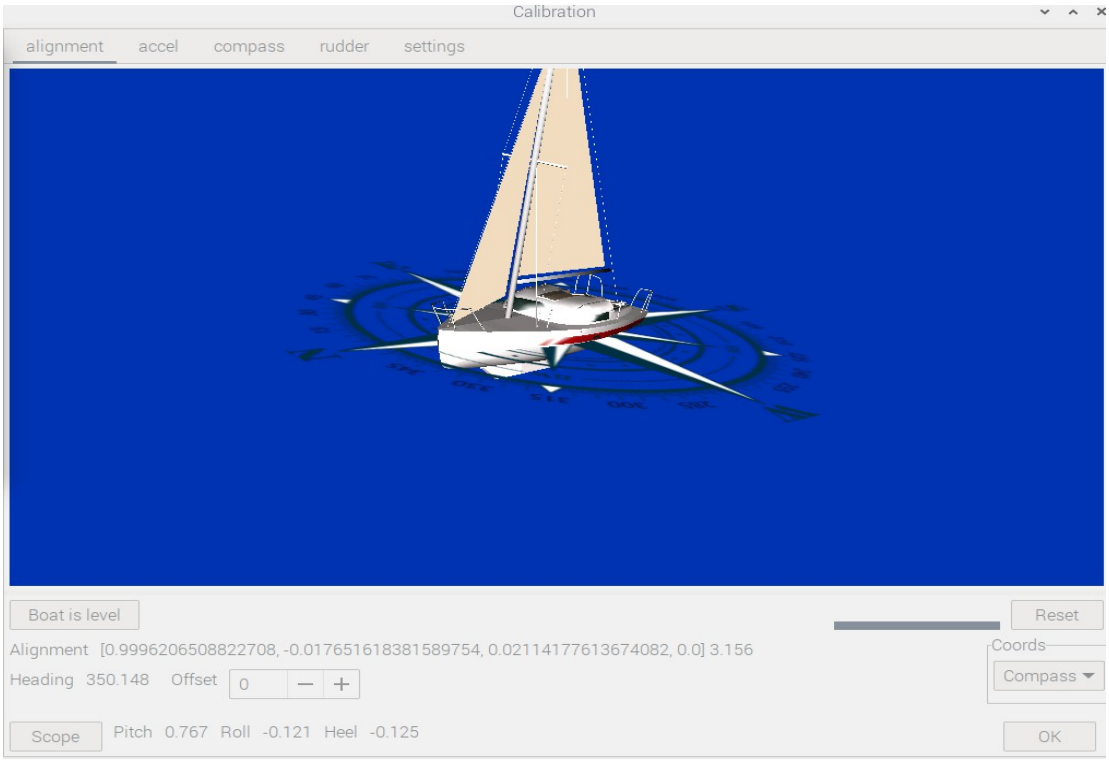
Si on déplace l'imu ou qu'on clique sur +10 ou -10 le moteur tourne. Ça marche.
Cela est la première étape de validation, après il faudra bien entendu régler plus finement tous les paramètres du pypilot en fonction de son voilier de son moteur de pilote etc...

Dans le répertoire /usr/local/bin rôles des différents scripts pypilot

pypilot_boatimu
Réalise une partie spécialisée du système : plutôt que de faire tout (pilotage + IMU + servo), il isole la partie capteurs/IMU pour permettre la calibration, l'évaluation, et l'intégration à l'ensemble. Avant de lancer l'autopilot, **vérifiez que l'IMU est bien calibrée** via pypilot_boatimu. Cela permet de s'assurer que les valeurs de roulis, tangage, cap sont correctes.

Conseils : Montez l'IMU dans une position stable, bien fixée, afin d'éviter vibrations ou mouvements parasites. L'alignement magnétique dépend de la stabilité. Effectuer la calibration à l'ancre ou au mouillage donc quand le bateau est immobile. Assurez-vous que l'IMU n'est pas trop proche d'éléments perturbants (métal, câbles de forte puissance) pouvant influer sur le magnétomètre.

pypilot_calibration
Accessible dans le Plugin OpenCpn Pypilot via le menu Etalonnage
Permet la calibration de l'imu redondant par rapport au menu Etalonnage accessible dans le plugin OpenCpn Toutefois ici l'écran alignement est pratique pour vérifier rapidement si on déplace l'imu on voit bouger le bateau. A noter que ici les menu accel et compass ne marchent pas mais fonctionnent via les même menus du Plugin OpenCpn Pypilot Etalonnage.



pypilot_client
Mode ligne de commande des différentes valeurs internes de pypilot, (gain, servos, etc.). Pour des réglages sur imu utiliser cela plutôt quand le voilier est immobile. Peut être utilisé via ssh.
Principales option mode ligne de commande :
`-c` monitoring continu d'une ou plusieurs valeurs (rafraîchi en boucle).

`-i` sortie verbose/détaillée pour la(les) valeur(s) demandée(s).

`-h` aide

Exemples d'utilisation :

#Activer / désactiver l'autopilote : pypilot_client ap.enabled=true ou false

Lire le gyro de l'IMU : pypilot_client imu.gyro

Suivre les gyros en continu : pypilot_client -c imu.gyro

Journaliser l'angle de safran pour analyse : pypilot_client -c rudder.angle > rudder_log.txt

Limiter le courant maxi du servo : pypilot_client servo.max_current=10

Passer en mode compas : pypilot_client ap.mode=compass

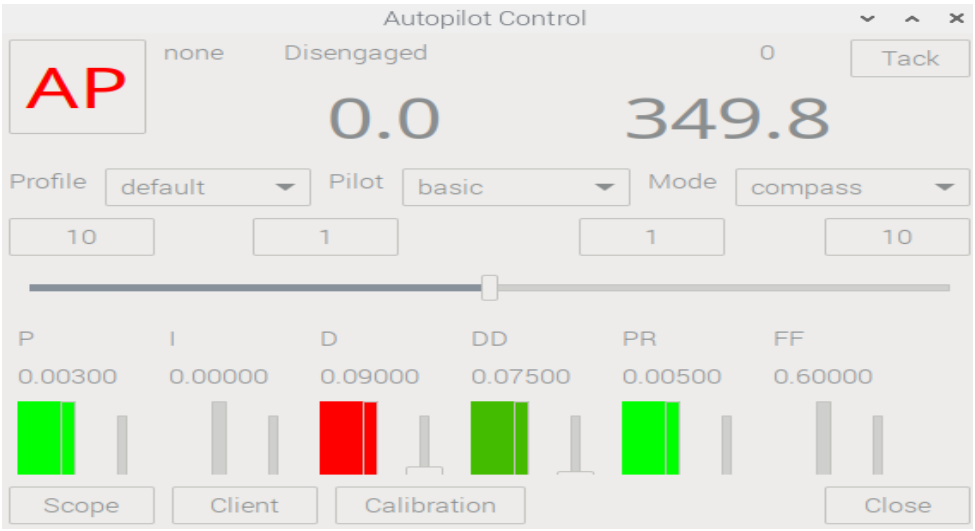
passer en mode GPS / vent apparent (si sources dispo)

pypilot_client ap.mode=gps ou pypilot_client ap.mode=wind

etc……..

pypilot_client_wx
Accessible via le Plugin OpenCPN pypilot Paramètres / Client Pypilot
interface graphique du pypilot_client

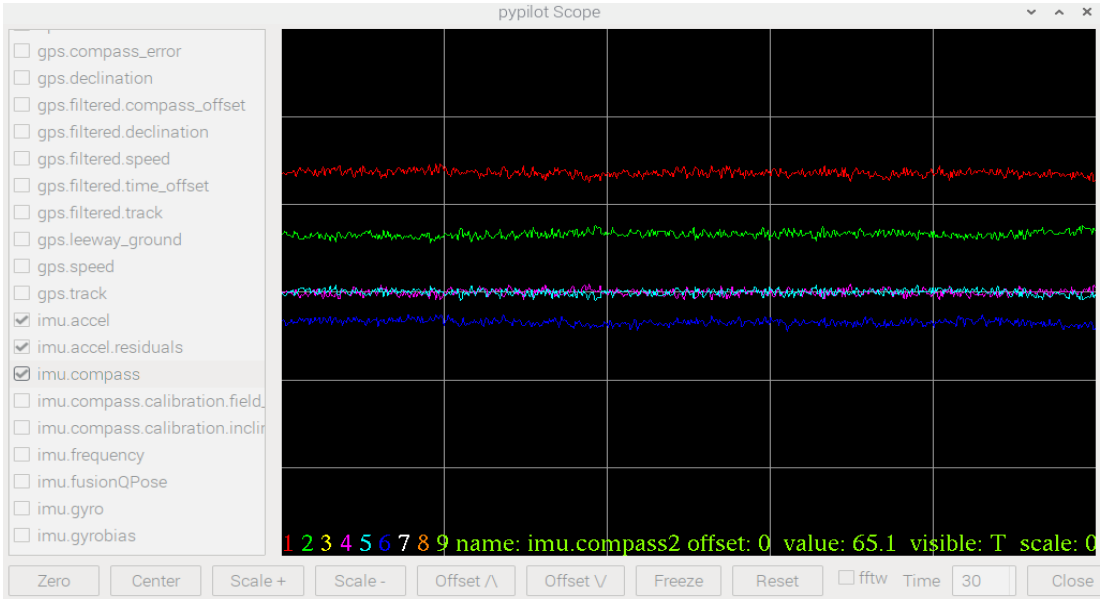
pypilot_control
Accessible via le Plugin OpenCPN Pypilot (bouton Gains) qui ouvre par défaut cette interface de contrôle



Les zones en bas en couleur avec des gauges vertes et rouges concernent le réglage du PID (Proportionnelle, Integrale, Dérivée) Cela est important pour régler plus finement les gains et donc la vitesse et réactivité du pilote automatique. Nous verrons cela dans un prochain chapitre spécifique.
Les autres boutons ouvrent les écrans déjà évoqués : pypilot_client_wx et pypilot_calibration et pypilot_scope (moniteur d'utilisation du pypilot)

pypilot_hat
Ne fonctionne pas ici, s'interface avec un petit ecran LCD typiquement utilisé par le tynipilot et permet de lire les boutons physiques et afficher les informations et paramètres essentiels du Pypilot

pypilot_scope
Visualisation en temps réel des données pour tuning et diagnostic
Affiche des graphes défilants (courbes dans le temps) des menus pour choisir quelles variables afficher, des boutons de zoom, pause, effacement, etc.… Permet de diagnostiquer un comportement anormal(biais, dérive, etc ..) et mieux appréhender comment pypilot interprète différents capteurs.



pypilot_servo
Ce script communique avec la carte contrôleur moteur (module basé sur un pont H, un arduino Nano et des commandes PWM).

Il envoie donc les ordres de direction et de puissance :

exemples : “tourne à tribord à 20 %”, “tourne à bâbord à 50 %”, “arrête le moteur”

Selon le matériel utilisé, la communication se fait via GPIO (c’est le cas ici sur le raspberry Rx et Tx via port série Pin 8 et 10) ou via UART / I2C / CAN bus ou via un contrôleur dédié.

Il reçoit donc les consignes de barre, exemple : servo.command = -0.3 # tourner 30% à bâbord

et il renvoie les mesures :
servo.position = 0.25
servo.current = 1.2
servo.voltage = 12.1
servo.engaged = True

`pypilot_servo` intègre plusieurs sécurités logicielles :

Limite de courant (`servo.max_current`)

Limite de tension (`servo.max_voltage`)

Timeout de commande (si plus de consigne reçue → arrêt)

Arrêt d’urgence en cas de surchauffe ou de court-circuit

Il désactive le moteur si une anomalie est détectée pour éviter de griller le driver ou de forcer la barre.

Paramètres de configuration typiques :

servo.max_current=3.0

servo.max_voltage=14.0

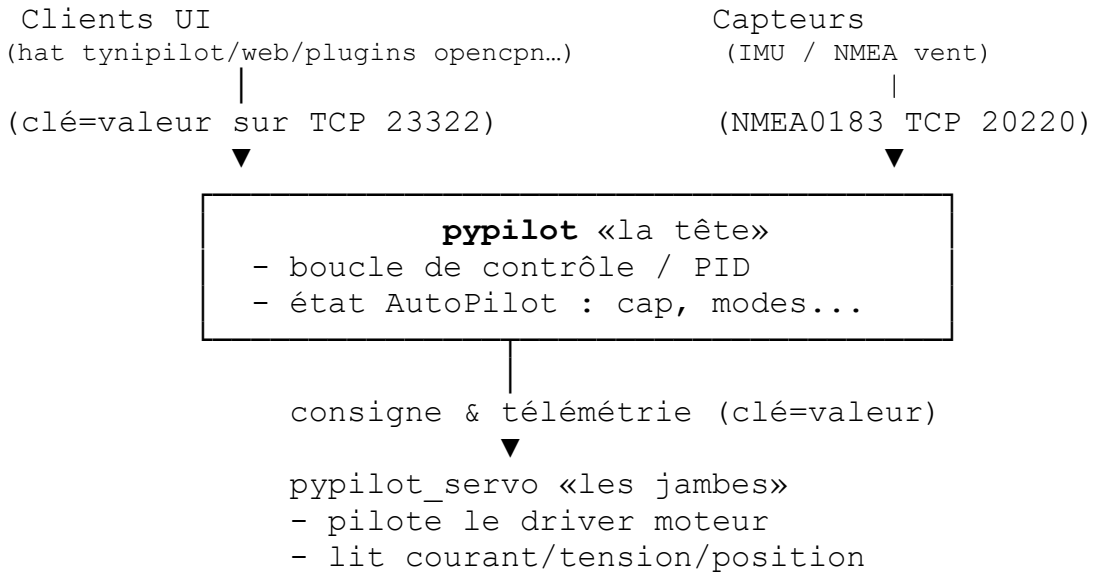
servo.min_voltage=10.0

servo.engage=True

servo.position_offset=0.0

pypilot_web
Lance un mini serveur web http non utilisé ici qui permet via une page web locale d’utiliser un smartphone une tablette ou pc pour dialoguer et paramétrer le Pypilot comme le fait le plugin OpenCpn ou les différents scripts énumérés ci dessus. De plus après vérification ce service pypilot_web fonctionne avec une vieille version du mini serveur web flask le rendant incompatible avec le mini serveur flask installé par défaut dans bookworm basé sur une version beaucoup plus récente.

Schéma de principe de fonctionnement où le script pypilot est le cœur du processus



Avant d'être installé au voilier comment tout ça dialogue à la maison ?