ISEN | école d'ingénieurs

Institut Supérieur de l'Électronique et du Numérique
Tél. : +33 (0)2.98.03.84.00
Fax : +33 (0)2.98.03.84.10
20, rue Cuirassé Bretagne
CS 42807 - 29228 BREST Cedex 2 - **FRANCE**

Projet M1

Année scolaire 2013/2014

Voilier Autonome 1
Partie Software

Proposé par *Yves Auffret*

Thibaut CHARLES — Option : Robotique
Thomas ABOT — Option : Tech. Bio-Médicales

Table des matières

1 Introduction

1.1 Introduction

Le projet de voilier autonome est un projet scolaire M1 2014 réalisé à l'ISEN Brest par deux bînomes. Le premier binôme ayant pour but de s'occuper de la partie Software est composé de Thibaut Charles et Thomas Abot. Le second binôme ayant à charge la partie Hardware est formé de Tangi Pennec de David Jouannic. L'encadrant de projet est M. Auffret.
Ce rapport de projet concerne le travail effectué par l'équipe Software sur le voilier autonome.

L'objectif du voilier est d'effectuer un parcours de manière purement autonome, c'est à dire sans aide ni indication extérieure que celles données par les capteurs embarqués.
Ce projet est une étape qui de préparation, test et expérimentation pour une étape future plus importante, la Microtransat qui consiste à effectuer une traversée de l'atlantique avec un voilier autonome de moins de 4 mètres de long.

1.2 Définition des termes

- **Route directe** : Route la plus courte pour aller d'un point A à un point B (orthodromique)
- **Cap direct** : Cap à tenir pour atteindre la position du prochain waypoint, sans tenir compte du vent
- **Direction vent** : Direction du vent apparent par rapport à l'axe longitudinal du bateau (entre -180˚à 180˚)
- **Direction absolue vent** : Direction du vent apparent par rapport au Nord
- **Direction vent réel** : Direction du vent réel par rapport à la ligne de foi du bateau
- **Allure** : Direction du voilier par rapport au vent. Peut être du "près" (0-30˚), "vent de travers", "Vent arrière", ...
- **Gîte** : Inclinaison du bateau sur le côté sous l'action du vent sur la voile.

2 Définition du projet

2.1 Cahier des charges

Introduction Le voilier autonome est un projet réalisé dans le cadre des études de M1 à l'ISEN. Les étudiants concernés sont Thibaut Charles et Thomas Abot pour la partie Software, Tangi Pennec et David Jouannic pour la partie Hardware.
Le voilier autonome devra réaliser une course au lac de Saint Renan, sans directives ni aides extérieures. Le parcours n'est pas encore défini pour le moment.
Le bateau reçu est un voilier radio-commandé, déjà équipé des actionneurs de voile (grand voile et foc) et de gouvernail. Le but de ce projet est de le coupler à plusieurs capteurs et une intelligence afin de le rendre complètement autonome pendant le temps de la course.

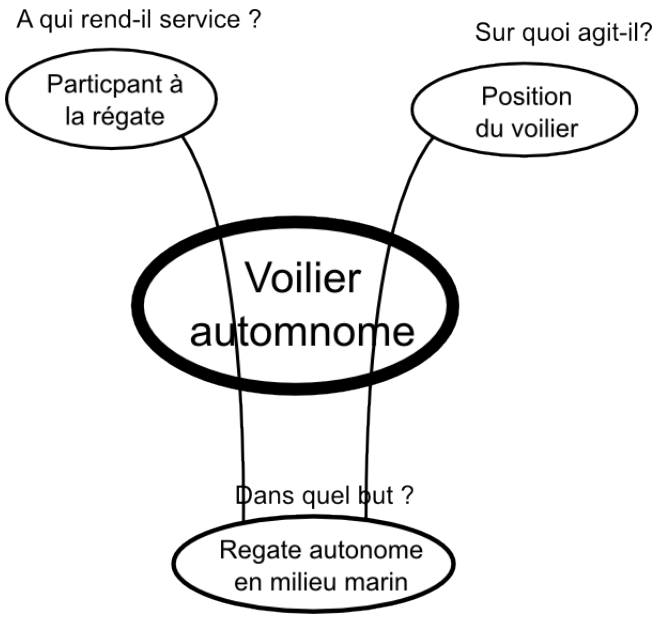


FIGURE 1 – Diagramme bête à corne

Présentation du projet La régate à effectuer aura une durée de moins de 2 heures et consistera à effectuer un parcours entre plusieurs bouées dont la position aura été rentrée manuellement avant la mise à l'eau.
Le bateau sera totalement autonome et ne pourra recevoir aucun ordre de notre part une fois le départ lancé. Il sera cependant connecté à un réseau WIFI tant qu'il sera proche du bord des côtes, et ce uniquement dans le but de recueillir des données relatives à l'étude de son fonctionnement.
Le voilier devra donc récupérer sa position, prendre en compte le milieu, et se diriger en fonction des données reçues par ses capteurs.

fonctionnement Le voilier autonome aura à terme pour seule entrée son programme de fonctionnement. Pour le développement et le prototypage, il est possible de reprendre la main sur le bateau à l'aide de sa télécommande, de façon à pouvoir effectuer nos tests et le contrôler en cas d'avarie ou de problème de

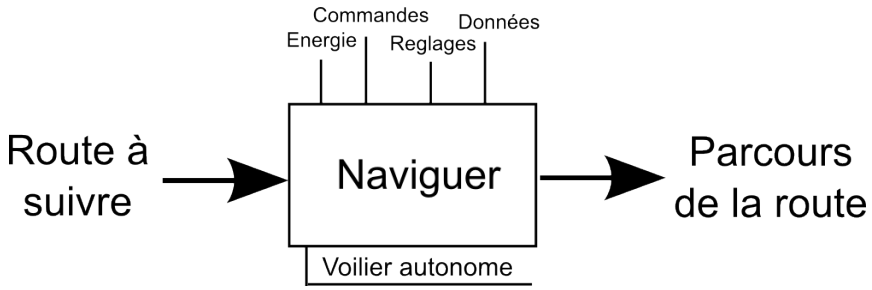


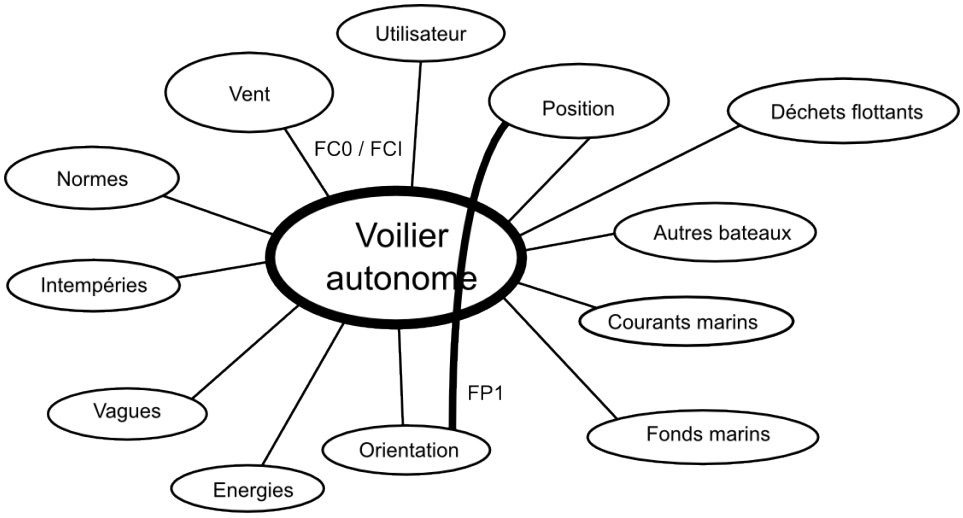
FIGURE 2 – Actigramme A-0

conception.
Le voilier se positionnera donc seul, en fonction des données reçues par ses capteurs. Les réglages interviendrons sur l'étalonnage physique des capteurs et le traitement des données, sur carte ou dans le programme.
Le voilier sera également autonome en énergie et devra se contenter de ses propres moyens d'acquisition ou de réserve d'énergie.

Fonctions et contraintes Dû à un budget limité, nous nous tiendrons dans un premier temps aux fonction principales ainsi qu'aux fonctions contraintes les plus importantes. Nous ajouterons les autres fonction contraintes en fonction du budget et du temps restant.
Le bateau devra donc s'orienter et se positionner automatiquement, quelque soit les contraintes.
Les fonctions contraintes doivent donc permettre à la fonction principale de s'exécuter sans problème.
Nous mettrons donc en place les fonctions contraintes suivantes :

- Connaître le cap actuel du bateau afin de corriger sa trajectoire
- Connaître la position du bateau afin de le diriger vers les objectifs
- Connaître l'orientation du vent afin d'optimiser la trajectoire et l'orientation de la baume.
- Connaître la force du vent afin de gérer l'orientation de la baume et le gîte du voilier.
- Être autonome en énergie. Nous demandons ici de tenir pour la durée de la régate.

L'alimentation fournira l'énergie aux capteurs, cartes et processeurs, ainsi qu'au module WIFI permettant de communiquer avec l'utilisateur afin de lui transmettre des données de navigation. Cette fonctionnalité sera également utilisée durant la conception afin prendre le visualiser les problèmes d'intelligence.



FP1 : Orienter et positionner automatiquement le voilier
FC0 : Gérer l'orientation du vent
FC1 : Gérer la force du vent
FC2 : Respecter les normes (légales et environnelentales)
FC3 : Resister aux intempéries
FC4 : Résister aux vagues
FC5 : Gérer l'équilibre face aux vagues
FC6 : Eviter les hauts-fonds
FC7 : Gérer les courants marins
FC7 : Eviter les autres bateaux
FC8 : Etre visible
FC9 : Eviter les déchets flottants
FC10 : Etre autonome en energie
FC11 : Communiquer avec l'utlisateur

FIGURE 3 – Diagramme Pieuvre

2.1.1 Partie Hardware

Déterminer les contraintes techniques

Résistance à l'environnement Le voilier étant fait pour être en mer, tout ses capteurs, actionneurs et cartes électroniques doivent résister à l'eau de mer. Pour cela, il y a plusieurs solutions. Pour les cartes électroniques, la meilleure solution est l'utilisation d'un caisson étanche. Il faudra faire attention aux dimensions du/des caissons par rapport à la place disponible sur le bateau. Quant aux actionneurs et capteurs, il faudra faire attention à leur résistance à l'eau de mer ainsi qu'à la qualité de leur mesures, peu importe la force du vent.

Poids Sachant que le voilier mesure environ 1 mètre de long par 30 centimètre de large, a un tirant d'eau de 45 centimètre et un tirant d'air de 1,60 mètre avec un lest de 2,5 kilogramme, il a été estimé que la charge supplémentaire embarquée doit rester en dessous de 1 kilogramme. Il faudra donc faire attention au matériel lourd comme les batteries ou les caissons étanches.

Consommation énergétique La consommation énergétique sera faite plus tard lorsque plusieurs composants auront été sélectionnés, et la batterie fournie.

Budget Limité À la base, au sein de l'école, un budget de 500 euros par binôme est accordé pour un projet M1. Or dans le cadre du projet du voilier autonome, l'école fait construire le bateau et achète quelques équipements, dont les batteries, le servomoteur de grand voile, notre budget est donc de 500 euros environ pour les deux parties du bateau, hardware et software. Il faudra tout de même faire attention à garder un petit budget de côté afin de pouvoir réparer d'éventuelles dégradations lors des essais.

Choisir les capteurs adaptés à la navigation autonome du voilier Afin de naviguer correctement, il faut pouvoir commander avec précision chacun des atouts directionnels du bateau. Ceux-ci dirigés par les actionneurs. Mais avant tout, il faut pouvoir récupérer les données permettant cette navigation optimale. En effet, les calculs permettant la commande préalable du bateau seront fait à l'aide de ces informations récupérées par les capteurs.

Ces informations sont :

— La position du bateau
— Le cap
— La direction du vent
— La gite
— Éventuellement la tension de la batterie

Parallèlement il existe plusieurs solutions (répertoriées ci-dessous), cependant c'est en évaluant les contraintes, au fur et à mesure du travail, qu'une bonne solution pourra être mise en place.

Données à récupérer

— Cap

Le cap est la direction par laquelle pointe le bateau par rapport au Nord géographique. Celle-ci est mesurée sur une boussole de 0 à 360˚entre l'axe longitudinal du bateau et le nord. Un compas serait la bonne solution. En effet, avec le GPS le cap peut être approximée mais le résultat des mesures donnera l'orientation du voilier cumulé à sa dérive. Il faut connaître l'orientation du bateau afin de pouvoir modifier son cap donc avec un compas les mesures seront plus précises et plus réactives.

Solution : Compas électronique

— Direction du vent

Afin de diriger efficacement le bateau en choisissant sa meilleure allure, et régler la tension des voiles, il faut connaître en temps réel la direction du vent.

Solution : la solution de la girouette est envisagée, le problème est de trouver une girouette résistante au milieu marin à un prix abordable. Il est possible d'en réaliser une soi-même en utilisant un codeur optique. Il faudra cependant s'assurer de son étanchéité.

— Gite

La gîte est l'inclinaison transversale du bateau (en fonction des différents axes de l'espace). Dans

des conditions normales (pas de vent, pas de vagues), le bateau est dans une position d'équilibre stable et droite. Cependant, avec l'action du vent sur la voile et les vagues, le bateau s'incline sur le côté, réduisant ainsi l'efficacité de la voile et pouvent entraîner dans les cas extrêmes le chavirement du bateau. Cette inclinaison transversale se mesure en degré : c'est un angle prenant pour origine les axes spatiaux.
La récupération du gîte se fera à l'aide des informations données par les 3 axes spatiaux (x,y et z) du vecteur de gravité, qui permettra de déduire l'angle d'inclinaison du bateau.

Solution : Accéléromètre 3-axes

Systèmes de pilotage adaptés à la navigation autonome du voilier

Safran Le safran la partie immergée du gouvernail d'un bateau, constitué d'un plan vertical pouvant pivoter afin de dévier le flux d'eau sous la coque pour changer la direction du voilier. Cette partie de commande du pilotage du bateau est donc indépendante et nous agirons dessus à l'aide d'un ou plusieurs actionneurs afin de contrôler la barre du voiler et donc son cap.
Le servomoteur de direction est fournit avec le bateau . Sa référence est : HXT 6.9kg / 39.2g / .16sec Twin bearing.

Position de la baume pour la Grand-Voile Idem que pour la partie du safran ci-dessus, pour le réglage de la baume et donc de la grand voile, nous allons utiliser un actionneur qui sera relié à une carte électronique. Pour rappel, la grand voile est la voile principale d'un voilier et sert à propulser ce dernier grâce à la force du vent. La grand voile est attachée au bateau par l'intermédiaire du mât (vertical) et de la baume (horizontal).
Le servomoteur de treuil est fournit avec le bateau. Sa référence est : SW4808-6PA Sailwinch Servo 6,13kg/45g/0,70 .

Réglage du Foc Idem que pour la grand voile ou le safran, le foc à un réglage indépendant. Il sera réglé à l'aide d'un actionneur, que nous déterminerons plus tard. Pour rappel, le foc est la voile à l'avant du bateau, rattachée au mât.
Le servomoteur de treuil treuil est fournit avec le bateau. Sa référence est : SW4808-6PA Sailwinch Servo 6,13kg/45g/0,70 .

Choix adapté de l'alimentation électrique du bateau Le choix de l'alimentation du bateau se fera en fonction des différents composants utilisés. En effet, l'utilisation de batteries autonomes permettra aux composants la pleine disposition énergétique pour l'utilisation qu'il en sera faite (une autonomie de 4h est prévue). Cependant, une étude énergétique préalable de chacun des composants doit être faite. En effet, l'utilisation des batteries doit être garantie par les moyens qui seront mis à leurs dispositions. Ensuite, celles-ci doivent avoir un poids total en adéquation avec l'utilisation que l'on veut faire du bateau (voir paragraphe sur l'installation des équipements).
Les batteries à lithium polymère sont déjà commandées, ainsi le bilan énergétique de l'ensemble du bateau servira à la meilleure utilisation des batteries dans le bateau.

Visibilité du bateau Lorsque l'on parle de visibilité, on parle d'être vu. Ainsi cette contrainte physique permet la pleine sécurité du bateau sur les eaux. En effet, un accident est vite arrivé si l'on envisage une navigation par mauvais temps. Pour cela, on pourrait envisager d'y installer une lampe ou une diode luminescente. En plus, plusieurs solutions peuvent nous être suggérées : la couleur de la coque, la couleur des voiles, bandes réfléchissantes, lumières obligatoires...

Installation des équipements sur le bateau L'installation des équipements sur le bateau se fait en fonction de la place que prend le composant (carte, capteur, actionneur) dans le bateau dont les dimensions sont déjà prédéfinies. Aussi, le poids de chacun des composants a une influence sur le poids global du bateau. En effet, une masse trop importante ralentira considérablement le bateau. Il faut donc faire attention aux contraintes de poids pour pouvoir naviguer correctement (voir paragraphe Poids).
Afin de palier à cela, il faudra faire une étude de taille mais surtout de poids de chacun des composants par rapport à la stature du bateau. L'installation pure ne sera plus qu'une contrainte mécanique ou architecturale à laquelle une étude de place sera faite.

2.1.2 Partie Software

Choix de la technologie pour contrôler le bateau Le Beaglebone est doté d'un processeur ARM de 1GHz, ce qui permet largement d'exécuter facilement le programme du voilier. Chargé avec un système Linux, il est possible de le programmer à distance via un accès SSH via un réseau filaire/wifi. Le Beaglebone possède de nombreuses entrées et sorties numériques et analogiques, ce qui correspond à notre besoin pour recevoir les données des capteurs et contrôler les servomoteurs.
De plus le Beaglebone consomme peu, ce qui permet au voilier de rester autonome en énergie plus longtemps.
Concernant la partie logicielle, il est prévu d'utiliser le langage D, compilé avec GDC pour processeur ARM sous Linux. Le D est un langage compilé, donc rapide. Inspiré du C et C++, sa syntaxe est améliorée, plus claire et sa bibliothèque standard permet de réaliser la quasi-totalité du programme.

Mettre en place un système de test/simulation des capteurs

récupération des données des capteurs La récupération des données des capteurs se fera via les entrées analogiques et numériques du Beaglebone.

Simulation des capteurs La simulation des capteurs se fera via un programme qui simulera les valeurs retournées par les capteurs. L'utilisateur pourra aussi prendre la main sur les capteurs, signalant au voilier de remplacer la valeur reçue d'un capteur, par la valeur donnée par l'utilisateur.

Communication entre le bateau et un terminal La communication entre le bateau et un terminal se fera via un serveur HTTP. Le programme du voilier sera en fait un serveur HTTP, renvoyant au client (PC, portable de l'opérateur) sous forme de page HTML stockée sur le serveur. Les commandes en provenance de l'opérateur se feront via une API REST ou une connexion avec une Websocket, et permettront de prendre le contrôle manuel du voilier. Ce contrôle pourra être au choix de l'opérateur, total ou partiel, laissant ou non le voilier gérer certain servomoteurs ou interpréter les données de certains capteurs. Il pourrait aussi

être envisageable de créer une page pour les portables afin de s'en servir comme d'une télécommande. Pour cela, nous aurons besoin de la bibliothèque *vibe.d*.

Gérer les différentes stratégies de navigation (route, allure)

Suivre un cap Pour la progression suivant un cap, nous utiliserons un algorithme de calculs de cap à tenir pour rejoindre une destination en fonction du vent. Il faudra prendre en compte la position actuelle, la direction du vent ainsi que la position de fin. Nous obtiendrons le cap temporaire (cap intermédiaire en cas de près, par exemple).

Suivre un objectif Pour suivre un objectif, nous pourront utiliser l'algorithme de calcul de cap, mais en tenant compte de la direction du vent, afin d'optimiser la vitesse.

Gestion des pannes et imprévus En cas de gîte trop élevée, les voiles sont relâchées progressivement jusqu'a ce que la gîte revienne dans la zone de sécurité.
En cas de panne, on peut choisir une attitude à adopter pour :

- Continuer la course si la panne est mineure.
- Sauver le matériel

Ceci implique d'effectuer une vérification du matériel pour déterminer la gravité de la panne : les valeurs sont-elles toujours cohérentes ? Varient-elles correctement ? Le capteur répond-il ?
L'opérateur peut aussi reprendre le contrôle complet sur le voilier afin de le ramener au bord ou le dégager d'une situation bloquante.

Interface de supervision Pour l'interface de collecte des données (gérée par les groupes de supervision), ceux-ci doivent s'entretenir avec leur référent afin de préciser leurs objectifs et les technologies qu'ils vont utiliser (désaccord entre les différents référents).

2.2 Problèmes soulevés

Certains cas critiques ont été évoqués, et il faut trouver un moyen de les gérer.

Arrêt du bateau En cas de perte de vent, le bateau peut s'immobiliser et ne plus être manœuvrable. Il faut donc définir un moyen de détecter ce cas et une procédure à suivre pour reprendre du vent et de la vitesse.

Panne matérielle Il se peut que un des capteurs/actionneurs tombe en panne. Il est donc préférable de définir des comportements à adopter en cas de panne.

Voies d'eau Il est possible que de l'eau entre petit à petit dans le bateau. Il faut donc la détecter et éventuellement l'évacuer. D'autres solutions sont possibles comme rendre le bateau insubmersible et l'électronique étanche.

2.3 GANTT

Un diagramme de GANTT a été établit afin de planifier les tâches à effectuer.
Les semaines affichées dans le GANTT sont les semaines de projet.

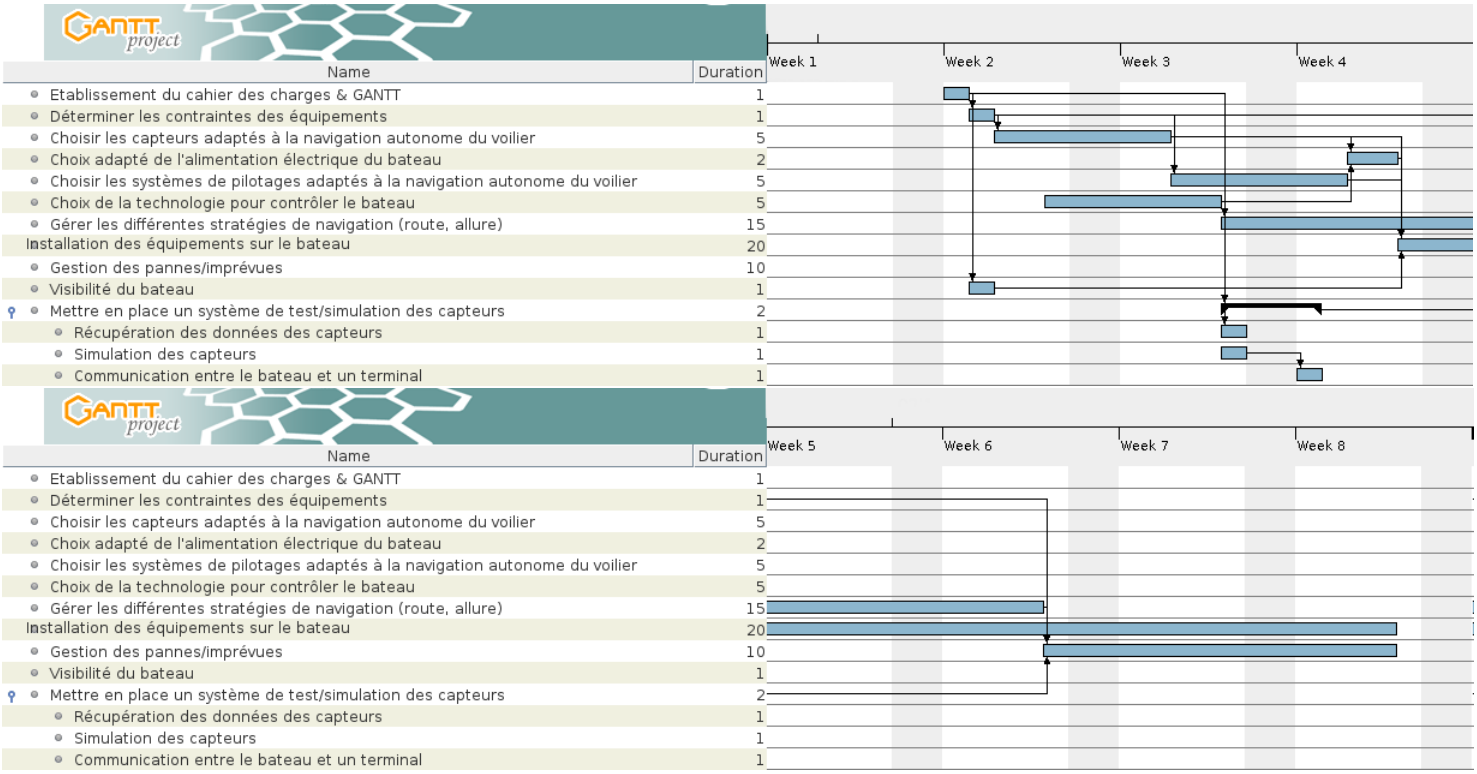


FIGURE 4 – Diagramme Gantt prévisionnel

2.4 Répartition des tâches

2.4.1 Hardware / Software

La partie Hardware porte sur :

- Choix des capteurs et actionneurs
- Mise en place des composants sur le bateau
- Modification du bateau suivant besoins
- Programme Hardware :
  - Communication avec les composants électroniques embarqués (acquisition des données capteurs, commande des actionneurs)
  - L'extraction des données utiles des capteurs
  - L'envoi des donnés à l'intelligence via une API simplifiée

La partie Software porte sur :

- Choix des stratégies de navigation
- Interprétation des données des capteurs
- Envoi au hardware des commandes des actionneurs
- Configuration du système d'exploitation du Beaglebone
- Gestion des crash des programmes hardware/software

2.4.2 Software

Thibaut CHARLES

- Structure générale du programme
- API de développement intelligence
- Calculs de coordonnées géographiques
- Filtrage basique
- Communication inter-process Hardware-Software

Thomas ABOT

- Serveur Web
- Interface web de debug
- Polaires de navigation
- Navigation avec polaires de navigation
- Filtrage des données angulaires

Tâches partie Hardware Suite au retard de la partie Hardware, les tâches suivantes ont été effectuées par le binôme Software :

- Programmation du logiciel d'acquisition hardware (hwdaemon)
- Configuration et programmation des composants :
  - GPS
  - Capteur de gîte
  - Servomoteurs

    - Girouette
    - Tension batterie
- Modifications du bateau :
    - Préparation des caissons étanches pour accueillir les cartes électroniques
    - Réduction de la voilure
    - Fabrication de capots spéciaux pour éviter les coincements de corde
    - Fabrication et installation de la Girouette en tête de mât
    - Fabrication d'une fixation pour porter l'antenne Wifi
    - Correction de l'étanchéité du bateau

2.5 Problématiques de conception du programme

Le programme doit être évolutif et particulièrement fiable tout au long de sa conception.

2.5.1 Évolutivité

Le programme étant assez complexe, il est important de commencer en développant des fonctionnalités basiques, afin de valider la cohérence de l'ensemble, puis de progressivement implémenter des fonctions plus complexes. Il faut donc penser le programme de façon à ce qu'il génère le moins de contraintes possible concernant de futures évolutions.

2.5.2 Fiabilité

Étant donné que les tests réels seront difficiles à mettre en place, il faut impérativement mettre en place des outils pour tester un maximum le code écrit. Il faut utiliser les solutions adoptées par les gros projets existants :

- Tests unitaires
- Calculs de coverage
- Programmation par contrat

En plus de cela, il serait intéressant de développer une interface graphique afin de relever les états de chaque élément du programme, et de simuler le fonctionnement des capteurs du bateau.

2.6 Communication software/hardware

Pour des soucis d'indépendance, la partie logicielle traitant directement avec les composants hardware du Beaglebone (GPIO, UART, I2C, PWM, ...) et la partie intelligence ont été séparées.

Il y a donc le programme **HardwareDaemon** (ou HWDaemon) qui traite les différents composants hardware, et l'**Intelligence** (Intel) qui interprète les données du Hardware Daemon.

La communication entre les deux est faite via une API simplifiée utilisant une socket UNIX, écrite par le groupe software pour le groupe hardware.

La récupération des données des capteurs et l'envoi des commandes sont faites via la classe Hardware, qui communique avec le Hardware Daemon. Il faudra donc définir une trame de données pour communiquer ces informations.

2.7 Interface Web

L'interface web sert à :

— Visualiser l'état de chaque capteur/actionneur
— Visualiser l'état des parties importantes du programme (Prochain objectif, cap à suivre, ...)
— Emuler les capteurs et désactiver certaines parties de l'intelligence pendant les phases de test du programme
— Visualiser à distance les logs émis par le programme.

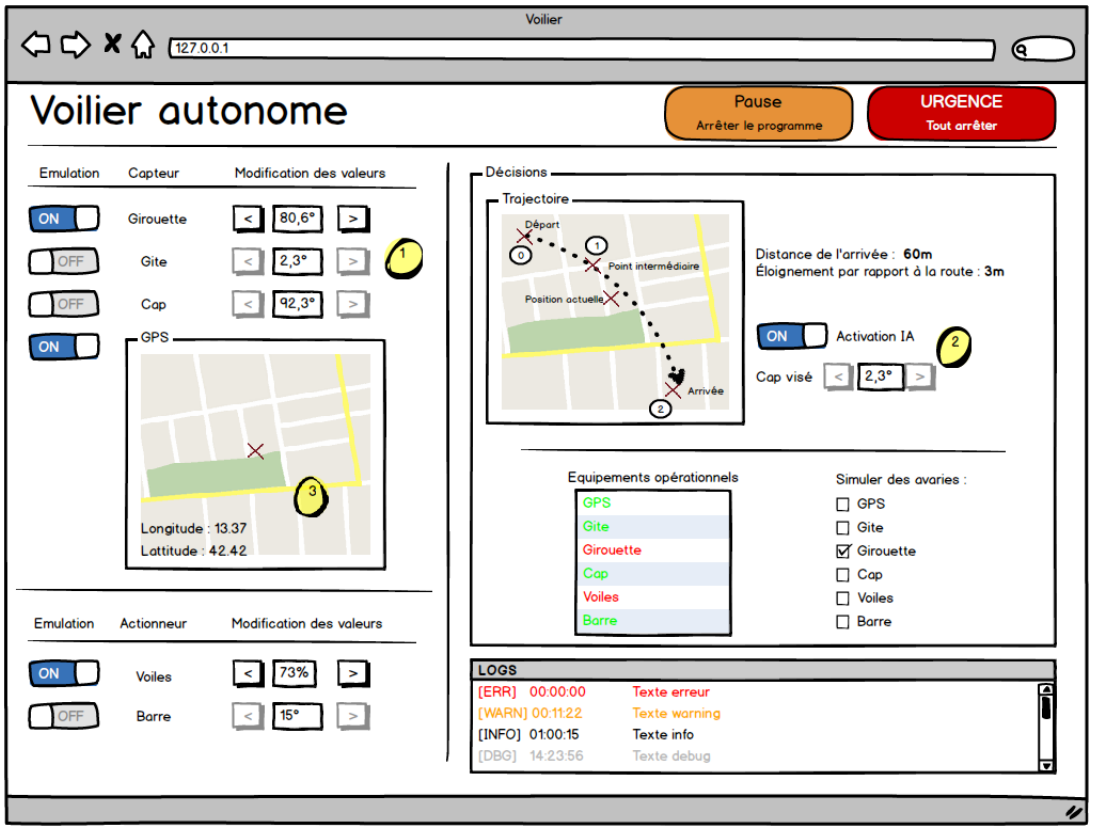


FIGURE 5 – Maquette interface Web

2.8 Diagramme UML

2.8.1 Diagramme

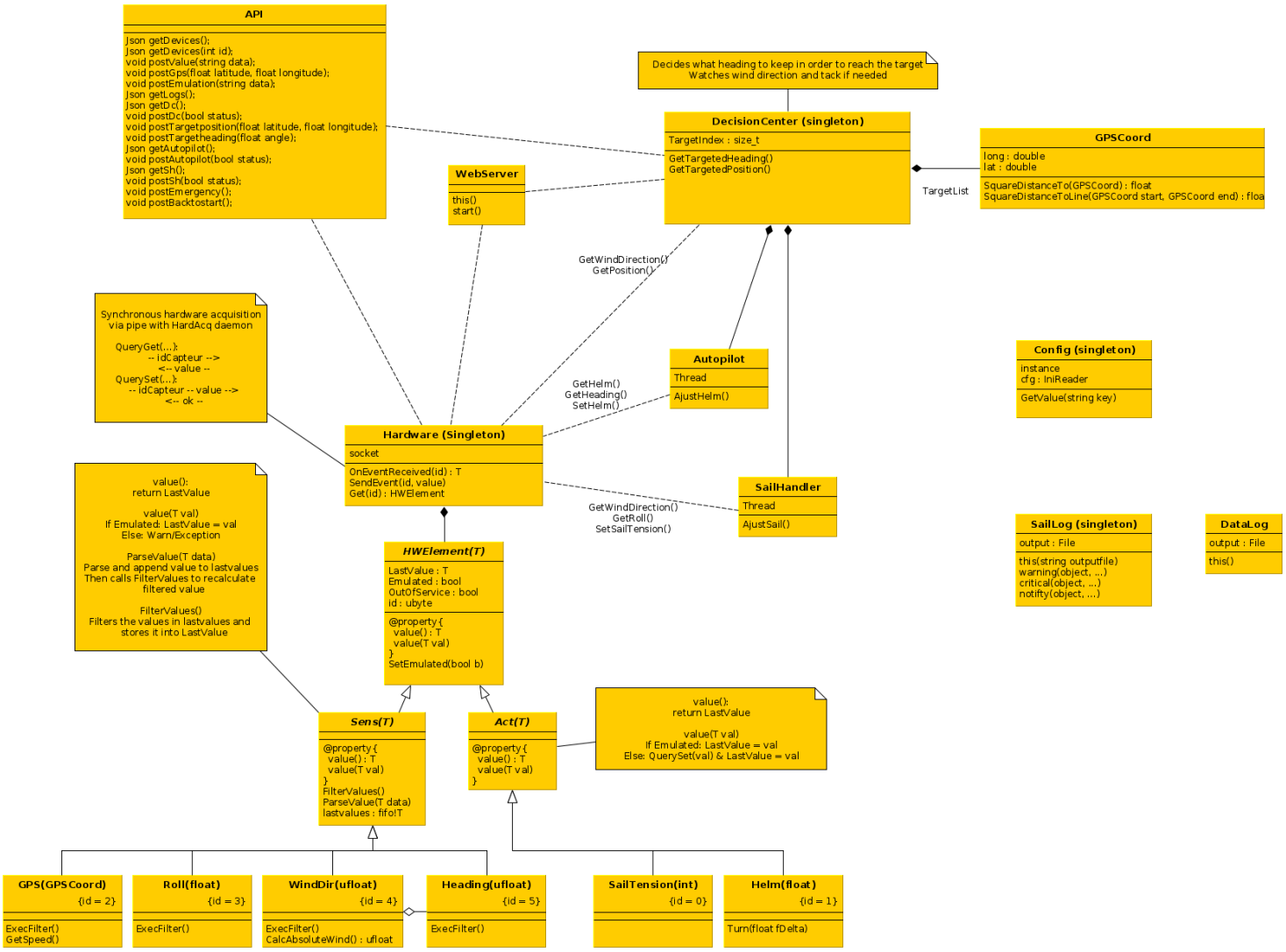


FIGURE 6 – Diagramme UML

2.8.2 Classes utilitaires

Logger Gère l'écriture des messages de log
- Formate les messages en fonction de leur importance (Warning, Critical, Success, ...) et donne les dates de lancement du programme
- Écrit les messages dans stdout, dans le fichier log et sur l'interface web

Config Gère le fichier de configuration du programme (res/config.ini)
- Parse et enregistre les différentes valeurs pour un accès ultérieur rapide
- Contient une liste des entrées par défaut dans Config.CONFIG_DEFAULT (fichier : config.d)

GPSCoord Calculs de coordonnées GPS (distances, angles, ...)
- Parse les chaines de caractères via regex pour extraire les données utiles suivant le format (DMS, GPS, DD, ...)
- Calculs de distance et caps en utilisant les routes orthodromiques (plus courtes)

Polar Calculs liés aux polaires
- Initialise les polaires via tableaux associatifs ou lecture d'un fichier
- Extrapole avec les valeurs stockées les valeurs pour tout angle.

2.8.3 Classes hardware

Hardware Gère la communication avec les composants hardware du bateau.
- Accesseur pour les capteur/actionneurs embarqués
- Initialise et gère la communication avec le HWDaemon via une Socket UNIX
- Se reconnecte automatiquement au HWDaemon en cas de plantage
- Gère l'émulation des capteurs en leur interdisant l'accès à la pipe

HWElement Représente un composant hardware
- Peut être émulé ou non
- En mode non émulé, les valeurs sont récupérées/envoyées aux composants hardware
- En mode émulé, les données sont stockées dans l'objet et ne sont jamais récupérées/envoyées aux composants hardware

2.8.4 Classes Intelligence

Trois organes régissent indépendamment l'intelligence du voilier :

DecisionCenter Le cerveau du bateau, celui qui commande
- Contient la liste des objectifs GPS à atteindre
- Utilise des polaires de vitesse pour déterminer le meilleur cap à suivre en fonction des capteurs embarqués
- En cas de capteur/actionneur défaillant, il émulera le capteur afin de limiter les répercussions sur le reste du système

Autopilot Il tient la barre pour suivre un cap donné

— Agit sur la barre par deltas pour se rapprocher du cap à suivre
— Si la barre arrive en butée sans que le bateau se rapproche du cap (le bateau n'avance plus), la barre revient au centre quelques temps afin de laisser le bateau reprendre de la vitesse

SailHandler Gère la tension de la voile

— Il ajuste les voiles en fonction de la direction du vent
— Si le bateau gîte trop, il choque progressivement les voiles jusqu'à ce que le bateau se redresse.

2.8.5 Interface Web

Server Gère les connexions HTTP sur le bateau.
— Sert les fichier statiques de l'interface web du bateau.
— Route vers l'API REST.

API API REST d'accès en lecture et écriture aux données du bateau.
— Activer ou désactiver l'émulation des capteurs et actionneurs.
— Obtenir les valeurs des capteurs et actionneurs.
— Affecter une valeur aux capteurs et actionneurs.
— Obtenir les logs du bateau, avec le niveau de log et la date.
— Activer ou désactiver le DecisionCenter, le SailHandler, l'Autopilot.
— Obtenir les données du DecisionCenter.
— Arrêt d'urgence : désactivation de l'intelligence et émulation de tous les capteurs.
— Retour au point de départ.

3 Environnement et choix

3.1 Choix des technologies

3.1.1 Beaglebone, Debian

Pour faciliter le développement et le débugging, il fallait de préférence un Linux embarqué. Les deux solutions retenues étaient le RaspberryPi ou le Beaglebone.

Le RaspberryPi pose deux problèmes :

— Consommation électrique : 900mA en 5v (4.5W) pour le RaspberryPi, contre 460mA (2.3W) pour le Beaglebone

— Hardware pins : Le Beaglebone offre beaucoup plus de GPIO, de PWM, ... que le RaspberryPi

3.1.2 Langage D

Le langage D est un langage récent, encore peu connu. Malgré cela, il dispose de nombreuses fonctionnalités qui le rendent très pratique à utiliser.

C'est en quelques sorte une évolution du langage C++, car il dispose à peu près des mêmes fonctionnalités de base, avec de nombreuses simplifications de syntaxe, ce qui en fait un langage facile à apprendre et très puissant.

En quelques mots le langage D est :

— **Natif** : Le D se compile vers du code machine, pouvant tourner sur ARM

— **Performant** : Le compilateur permet d'optimiser le code en l'exécutant lors de la compilation (cf CTFE)

— **Maintenable** : Le langage inclue de quoi faire des tests unitaires, des calculs de coverage, des assertions à la compilation, de la programmation par contrat, ...

— **Minimaliste** : Grâce au meta-programming très présent en D, Le code écrit est beaucoup moins verbeux et plus générique que du code C++ ou (pire) Java.

3.1.3 Dub

Dub est un logiciel de gestion de paquets et de projet pour D. Il permet d'abstraire le compilateur utilisé (DMD, GDC, LDC, ...) et de compiler rapidement un projet D.

Il est généralement plus performant qu'un Makefile avec compilation par parties (génération de .o puis linkage).

3.1.4 Vibe.d

Vibe.d est une bibliothèque écrite en D permettant de créer son propre serveur web personnalisé. Cela apporte beaucoup d'avantages, dont l'intégration au programme d'intelligence (pas besoin de CGI), la légèreté (pas besoin de serveur Apache), et la performance.

3.1.5 Angular.js

Angular.js est un framework Javascript développé par Google afin de concevoir rapidement des sites web, avec certaines fonctionnalités très pratiques comme le data-binding ou la recherche dans une liste. Le projet ne portant pas sur l'interface web, il était important de choisir le bon outil pour la réaliser le plus rapidement et simplement possible.

3.2 Environnement

3.2.1 Beaglebone : Debian

Le système d'exploitation utilisé sur la Beaglebone White est un Linux Debian 7.4. Le programme d'Intelligence est indépendant de la distribution installée (compatible avec tous les Linux), contrairement au HWDaemon qui dépend du noyau installé : les chemins vers les pseudo-fichiers du beaglebone peuvent varier d'une version à une autre.

Le système utilisé est précisément : Debian 7.4 (bone-debian-7.4-2014-04-18-2gb.img), re-dimensionnée à 4Go par un outil de partitionnement. Kernel 3.8.13-bone49 patché EHRPPWM (voir "Beaglebone : Réparation des drivers Kernel PWM")

Paquets requis ca-certificates git libevent-2.0-5 libevent-pthreads-2.0-5

Paquets de développement build-essential

Paquets optionnels zsh vim gdb

3.2.2 Beaglebone : Réparation des drivers Kernel PWM

Le Kernel distribué avec les Debian récentes (3.8) possède un bug dans le driver Kernel EHRPPWM : on ne peut pas utiliser qu'une des deux sorties des modules EHRPWM.

Le problème vient du fait que deux sorties d'un même module PWM (sorties A et B) doivent avoir la même fréquence de fonctionnement, et qu'elles doivent concorder lors de l'activation des PWM. On ne peut régler la fréquence qu'à l'activation des sorties, ce qui fait que l'activation de la seconde sortie échoue car par défaut les PWM ont une période de 500us et diffère de la période de la première sortie.

La correction consiste à recompiler le Kernel en patchant les drivers PWM du Beaglebone. Le patch proposé vérifie si la fréquence du PWM a été initialisée auparavant avant de générer le signal. Bien qu'il ne fonctionne pas/est incomplet, il a permit d'établir un patch fonctionnant sur le système.

Procédure :

— Préparer l'environnement de build en suivant les instructions de LinuxOnArm :

```
wget -c https://raw.github.com/RobertCNelson/tools/master/pkgs/dtc.sh
chmod +x dtc.sh
./dtc.sh

git clone https://github.com/RobertCNelson/linux-dev.git #Attention,
    l'ISEN semble bloquer la commande
```

```
cd linux-dev/
git checkout origin/am33x-v3.8 -b tmp
./build_kernel.sh
```

— Modifications du driver : fichier KERNEL/drivers/pwm/pwm_test.c, ligne 273. Remplacer :

```
/* polarity is already set */
rc = pwm_config(pwm_test->pwm, pwm_test->duty, pwm_test->period);
if (rc) {
      dev_err(dev, "pwm_config() failed\n");
      return rc;
}
```

Par :

```
* Only set the config if the period is > 0. Otherwise the period will
   be set later dynamically */
if ( pwm_test->period > 0 )
{
      /* polarity is already set */
      rc = pwm_config(pwm_test->pwm, pwm_test->duty, pwm_test->period);
      if (rc) {
            dev_err(dev, "pwm_config() failed\n");
            return rc;
      }
} else {
      pwm_test->period = 0;
}
```

— Modification des fichiers du SYSFS Modifier les fichiers : KERNEL/firmware/capes/bone_pwm_P8_19* KERNEL/firmware/capes/bone_pwm_P8_13* KERNEL/firmware/capes/bone_pwm_P9_14* KERNEL/firmware/capes/bone_pwm_P9_16* en changeant

```
pwms         = <&ehrpwmX 1 500000 1>;
[...]
enabled      = <1>;
```

par

```
pwms         = <&ehrpwmX 1 0 1>;
[...]
enabled      = <0>;
```

Installation du Kernel Via la carte SD du BeagleBone

```
mount /dev/mmblck0p1 /mnt #Mount boot partition
cp deploy/*.zImage /mnt/uboot/zImage
tar xvf deploy/*-dtbs.tar.gz -C /mnt/uboot/dtbs
```

```
umount /mnt

mount /dev/mmblck0p2 /mnt #Mount rootfs partition
tar xvf deploy/*-firmware.tar.gz -C /mnt/lib/firmware
tar xvf deploy/*-modules.tar.gz -C /mnt
umount /mnt
```

3.2.3 Dev : Linux (ArchLinux/Ubuntu 13.10)

Il est préférable de développer le programme sur une machine puissante, car le compilateur D est gourmand. Le langage D étant fortement multi-plateforme, il on peut sans soucis travailler sur une machine Linux classique.

ArchLinux est la distribution de choix pour travailler sur ce programme. D'autres distributions fonctionnent aussi, comme Ubuntu 13.10.

3.2.4 Git

Git est un logiciel de gestion de version très largement utilisé dans le monde de l'OpenSource.
Git doit être installé sur la machine de dev.

3.2.5 Cross-Toolchain GNU GDC

La compilation du toolchain pour compiler pour ARM sur x86_64 est une étape qui a pris beaucoup de temps.

Le toolchain compilé est téléchargeable sur thibautcharles.net/ftp et les instructions pour le compiler sont disponibles sur le wiki du projet

Il doit être accessible via le PATH de la machine de dev. (arm-unknown-linux-gnueabi/bin)

3.2.6 Génération de la documentation

La documentation du programme est générée via la cible "doc" du Makefile, et utilise DMD et DDOX.

4 Réalisation technique

4.1 Structure du programme

La structure du programme se veut simple et souple pour que le système soit évolutif. Le nombre réduit de classes permet un prise en main rapide des sources afin de les modifier.

L'utilisation de test unitaires permet la validation automatique du fonctionnement de chaque classe et de leurs interactions. Ces tests sont effectués régulièrement, afin de repérer les problèmes causés par l'ajout de fonctionnalités.

Des calculs de coverage sont ajoutés aux tests unitaires afin de vérifier que les tests unitaires couvrent l'ensemble des fonctionnalités du programme. Dans le programme d'intelligence, le coverage est supérieur à 60% (60% du code est testé par les tests unitaires)

La programmation par contrat permet de vérifier que les fonctions programmées s'exécutent correctement en effectuant systématiquement des vérifications avant et après leur appel. Cela permet aussi de vérifier la cohérence des objets manipulés dans le programme lors de leur modification.

4.2 Coordonnées GPS

La classe GpsCoord fournit une implémentation des calculs de base de coordonnées géographiques.

Routes orthodromiques/loxodromiques Comme la Terre est ronde, le chemin le plus court entre deux points n'est pas une droite mais un arc de cercle. On fait donc la différence entre la route la plus courte entre deux points (orthodromique ou great circle) et la route suivant un cap constant (loxodromique)

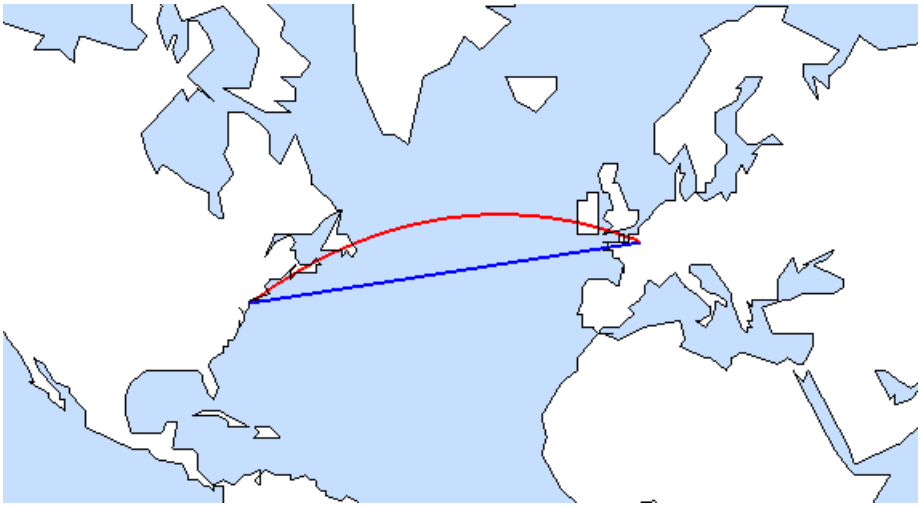


FIGURE 7 – Route Loxodromique (Bleue) / Orthodromique (Rouge)

Il existe généralement plusieurs façons d'effectuer ces calculs, et généralement la précision augmente au détriment de la complexité algorithmique. La classe GpsCoord fournit ainsi plusieurs solutions, à choisir en fonction de la précision requise.

Il existe plusieurs formats pour représenter une coordonnée GPS : degrés décimaux, degrés et minutes décimales, et degrés minutes secondes sont les plus utilisés. La conversion de l'un à l'autre est effectuée par la class GpsCoord qui formate et parse via regex les données de latitude et longitude.

4.3 Polaires

4.3.1 Théorie

Les polaires se représentent via un graphe de coordonnées polaires. L'ensemble des points de la courbe se modélise via la relation $f(\theta) = re^{i\theta}$ (figure 8).

Nous pouvons ainsi modéliser l'évolution d'un système en fonction d'un paramètre d'entrée grâce au polaires. La valeur d'entrée est θ . La valeur de r étant donnée par le modèle.

Attention : Il est important de noter que les données en entrée du modèle sont périodiques. Il s'agit d'une condition d'utilisation du modèle polaire.

Nous pouvons donc avoir n'importe quelle valeur en entrée du modèle, il aura toujours une valeur associée. Nous n'avons donc pas à nous soucier des bornes des valeurs d'entrée.

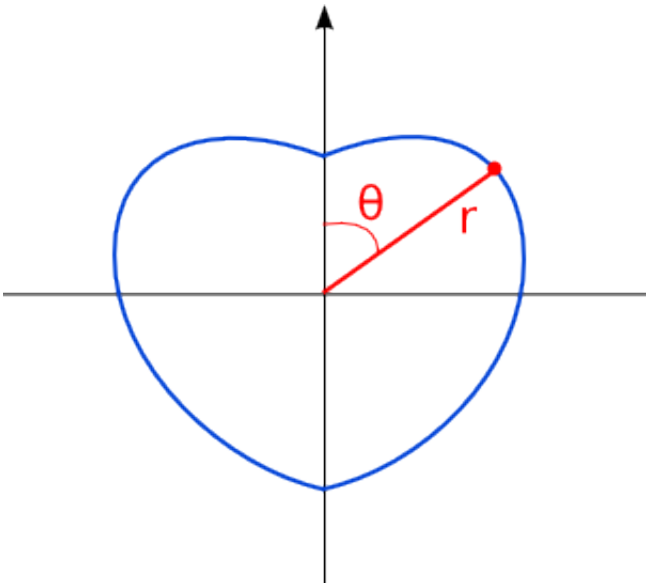


FIGURE 8 – Exemple de polaire basique et symétrique

L'ensemble des valeurs r fait partie du modèle polaire. La forme de stockage de ces valeurs dépendra de votre modèle. Ce modèle peut être une relation de type $f(\theta) = re^{i\theta}$ ou purement numérique comme une suite de points (figure 9), par exemple.

Règles polaires La règle polaire correspond à une représentation d'une polaire dans un tableau de mémoire numérique. Pour cela θ devient une graduation d'axe d'abscisse et r devient une graduation d'axe des ordonnées (figure 10). Nous conservons les rapports entre θ et r pour chaque point de la courbe.

Cette approche permet non seulement une représentation d'un tableau de mémoire numérique plus aisée, mais également une meilleur visualisation graphique. Il suffit de faire coulisser les règles polaires les unes sous les autres en fonctions des données précédentes pour visualiser les mesures possibles, comme si nous manipulions des règles de mesure, d'où le nom de règle polaire.

La figure 11 illustre ce principe. Chaque élément attaché à un signe de masse est un élément fixe, immutable à un instant t correponsant à la mesure en cours. Nous pouvons faire varier θ_1, mais cela fera bouger l'ensemble de la seconde règle. La règle 1 et le vecteur en θ_2 ne bougeant pas, ces vairations font varier r_1 et r_2.

Nous allons donc mesurer les valeurs de r1 et r2 afin de trouver la meilleur valeur de la combinaison de ces deux valeurs. C'est ici que l'AIUR vient jouer son rôle.

4.3.2 Application au modèle numérique

Nous allons à présent utiliser l'AIUR dans un contexte numérique, d'où l'utilisation de règles polaires est appropriée.

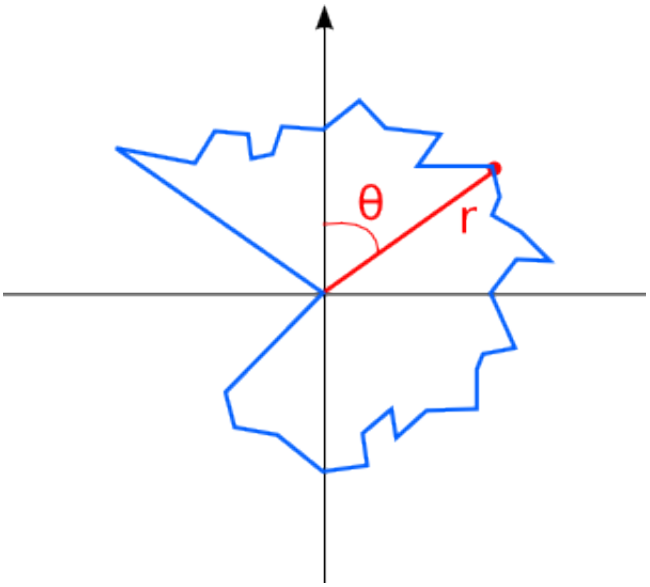


FIGURE 9 – Exemple de polaire dont le modèle est numérique

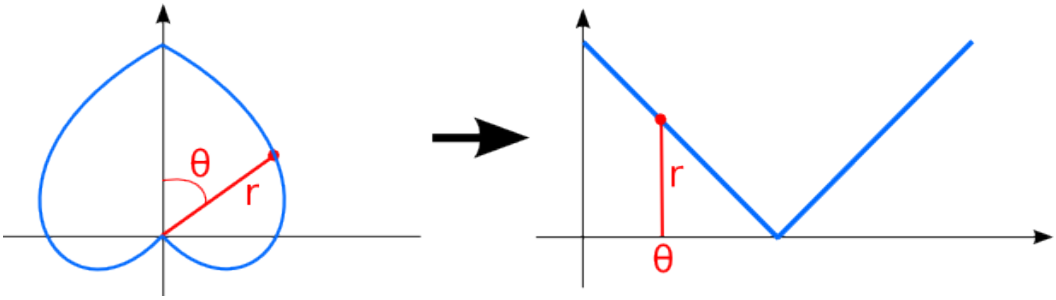


FIGURE 10 – Passage du res polaire à la règle polaire

Stockage du modèle Le modèle en règle correspond à un tableau de paires de valeurs (abscisse, ordonnée). Ces points seront stockés dans un tableau ou une liste, de manière ordonnée ou non. L'important est de pouvoir stocker les points les plus importants. Les valeurs manquantes pouvant se retrouver par extrapolation.

Nous allons donc stocker nos valeurs dans un tableau. Ces valeurs peuvent prendre toutes les formes. Il faut juste se conformer à une forme pour le clé du tableau (l'abscisse) et une forme pour la valeur associée (l'ordonnée).

Par exemple, nous utilisons ici un tableau de clé et valeur associé en réels à virgule flottante (Table 1) :

Clé	0.0	45.0	...	360.0
Valeur	0.0	0.8	...	0.249

TABLE 1 – Un exemple de tableau clé-valeur

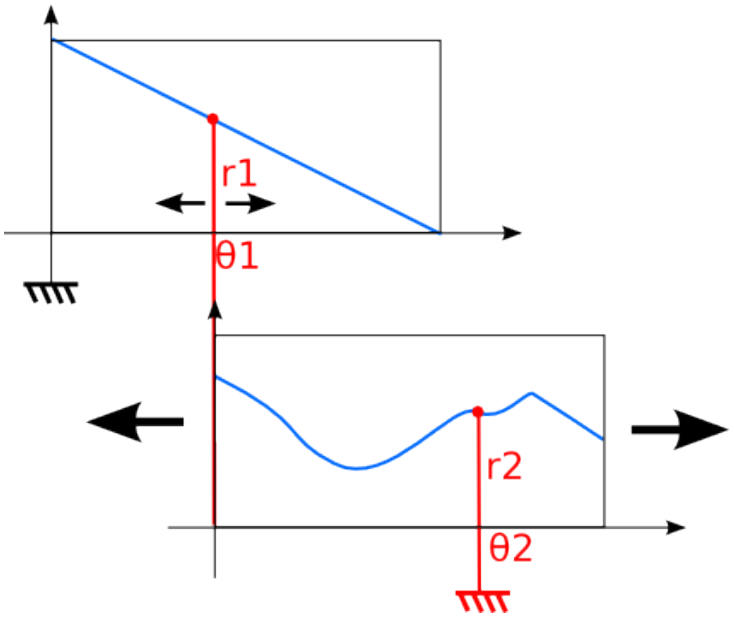


FIGURE 11 – Principe de visualisation des règles polaires

Comme notre modèle numérique possède un nombre fini de points, il faudra extrapoler pour trouver une la valeur associée à une clé non définie. L'algorithme d'extrapolation dépend de votre besoin de précision.

Sachez toutefois que si vous avez besoin d'extrapoler une courbe sur une polaire, ceci se traduit par une droite sur la règle polaire.

Algorithme Imaginons un système constitué de N règles (Pour information, le système de la figure 11 comporte 2 règles). Nous devons trouver la clé (valeur d'entrée) et la valeur associée (valeur de sortie) pour chaque règle.

A présent, nous devons trouver des relations entre les règles afin de créer une chaine de règles, comportant des entrées fixées :

règle 1 fixée $\rightarrow$ règle 2 $\rightarrow \ldots \rightarrow$ règle N $\leftarrow$ Autre entrée fixée

Par exemple, sur la figure 11, nous observons que la première règle est fixée à un instant t, il s'agit de la référence du système.Il en est de même pour l'entrée de la deuxième règle qui est fixée par rappor à la référence du système.

Ces références vont nous permettre de fixer les conditions imposées au système. Ces points fixes peuvent varier en fonction du temps, mais restent fixe pour une mesure donnée. La référence du système restera fixe à moins que le système varie de lui même.

Exemple Lors de la réalisation de notre voilier, la règle fixe correspondait au cap théorique du voilier, autrement dit, une chose immutable. L'entrée de cette règle est le cap effectif du bateau. Il s'agit d'une des valeurs que nous voulons optimiser.

La deuxième règle se fixe par rapport à l'orientation du bateau par rapport au vent. Ce rapport est directement lié au cap théorique, d'où le lien entre ces deux règles. L'entrée de cette règle est l'angle que forme le vent par rapport au cap du bateau. Nous voulons également optimiser cet angle. Cependant, nous ne pouvons décider de l'orientation du vent. Donc ce dernier est fixé.

La seule chose mobile de ce modèle est la deuxième règle, liée à l'entrée de la première règle : les deux bougent en même temps, et de la même manière.

Nous pouvons passer à la phase de recherche de la solution optimum. A chaque entrée, nous pouvons lire une valeur de sortie sur la courbe de la règle polaire. Nous relevons alors une valeur par règle, soit N valeurs au total.

Nous allons bouger les règles mobiles. Le but étant de récupérer la valeur maximale ou minimale (suivant le système recherché) en appliquant un coefficient de pondération sur chaque valeur suivant sont importance dans le choix de la valeur de sortie. Ces différentes valeurs pondérées sont alors sommées.

Nous pouvons alors choisir les valeurs des sorties pour la position des règles ayant la somme trouvée précédemment maximale ou minimale, suivant le système.

Chaque règle nous a alors fourni une sortie optimale pour obtenir le comportement souhaité.

4.3.3 Dans l'Intelligence

Le voilier base une partie de son intelligence sur des polaires de données.
Ces polaires représentent des valeurs en fonction d'un angle allant de 0 à 360 degrés. Pour des valeurs d'angles autres, un modulo 360 est appliqué car ces valeurs sont considérées comme périodiques.

Le stockage des données est un tableau de points représentant un angle associé à une valeur. Les données entre chaque point sont extrapolées linéairement afin d'obtenir une valeur même si celle-ci n'a pas été enregistrée. Toutes ces données dont enregistrées dans un ficher de polaire, au format JSON.

Un exemple simple de fichier JSON de polaire :

```
[
   {
      "key": 0.0,
      "value" : 1.0
   },
   {
      "key" : 180.0,
      "value" : 0.0
   },
   {
      "key": 360,
      "value" : 1.0
   }
]
```

Ces polaires étant pour la plupart symétriques par rapport à l'angle des 180 degrés, une fonctionnalité de désactivation d'une ou toutes les moitiés de la polaire à été ajoutée. Cette fonctionnalité a pour but de désactiver tout ou partie des choix faits par l'intelligence, une valeur de zéro pour un angle donnée équivalant à une situation à éviter. La désactivation d'une moitié de polaire est ici utilisé pour forcer le bateau à virer de bord (le cap actuel n'étant plus autorisé, il lui faut trouver un nouveau cap optimal en choisissant un cap sur la plage des cap autorisés (dont la valeur est supérieure à zéro).

En superposant des polaires, le système est à même de les orienter les unes par rapport aux autres de manière à trouver la position optimale, c'est à dire la position ou les valeurs associées aux angles données sont maximales en tenant compte de points fixés par le système. Cet algorithme est associé et choix du cap dans le système. La première polaire représentant le cap à privilégier par rapport au cap direct vers le point à rejoindre (tout droit étant préférable à une position déviée), et la deuxième polaire représentant

4.4 Filtrage des données capteurs

Raw Le filtre dit "Raw" renvoie la dernière valeur reçu, autrement dit, il n'effectue aucun filtrage.
Il est utilisé pour traiter les tensions de la batterie qui varient lentement.

Moyenne glissante Le filtre de moyenne glissante est un filtre qui additionne l'ensemble des valeurs et divise par le nombre de valeurs.
Ce filtre est défini, mais n'est pas utilisé par le système.

Moyenne glissante pondérée du temps Le filtre de moyenne glissante pondéré de temps est une variante du filtre à moyenne glissante, mais qui pondère en plus chaque valeur par sa durée par rapport à la durée totale des valeurs du filtre.
Ce filtre est utilisé pour le GPS et l'accéléromètre, dont les données varient rapidement et avec beaucoup de bruit lié au mouvement du bateau.

Moyenne glissante d'angles pondérée du temps Ce filtre est une déclinaison du filtre de moyenne glissante pondérée du temps. La différence est qu'il gère de façon périodique des valeurs comprises entre -180˚et 180˚.
Par exemple, la moyenne entre 179˚et -179˚n'est pas 0˚mais bien 180˚.
Ce filtre est utilisé pour filtrer les valeurs émises par la girouette et le compas, qui varient rapidement et de manière périodique.

Kalman Le filtre de Kalman est un filtre très utilisé car il effectue une prédiction des prochaines valeurs pour effectuer un filtrage. Il est notamment utilisé pour le filtrage de données GPS.
Faute de temps, le filtre de Kalman n'a pas été implémenté dans le programme.

4.5 Communication HWDaemon - Intelligence

Les deux programmes ont été séparés afin de les développer séparément et pour que si l'un crash, il puisse se relancer et ne pas gêner l'autre.

La communication entre les deux processus est gérée par une socket UNIX (comme une socket TCP-IP contenue dans la RAM et donc plus rapide).

La trame de communication est définit ainsi :

8bit ID	64bit Data0	64bit Data1

— ID : Identifiant du capteur/actionneur
— Data0, Data1 : Valeur du capteur/commande de l'actionneur

4.6 Serveur Web

Le serveur web est séparé en deux parties : le serveur HTTP et l'API REST.
Le serveur HTTP est créé via vibe.d et sert au client des fichiers statiques, tels que le contenu et les scripts de la page web. Il redirige les requêtes entre les fichiers statique et l'API REST.
Sur ce serveur, l'API REST fournit une liste d'adresses de requêtes REST à appeler pour accéder à différentes fonctionnalités du programme : visualiser les capteurs, actionneurs et les parties de l'intelligence, modifier les valeurs, émuler des capteurs et actionneurs, visualiser les logs.
Le port 80 habituellement utilisé pour le protocole HTTP étant utilisé par l'équipe de supervision, le serveur web utilise le port 1337.

4.7 Interface Web

L'interface web est une application JavaScript utilisant AngularJS, sur une seule page, et permettant de visusaliser à distance les données du système. Les fonctionnalités de l'interface sont :

— Visualiser les valeurs des capteurs et actionneurs.
— Émuler les valeurs des capteurs et actionneurs.
— Visualiser les logs émis par le programme, avec colorisation des différents niveau de log, affichage des dates, et champs de tris des logs par mot clé et niveau de log.
— Activer et désactiver les parties de l'intelligence : SailHandler, Autopilot, et DecisionCenter.
— Visualiser et modifier les paramètres de l'intelligence : cap et position GPS du point à atteindre.
— Option de réglage de la fréquence de rafraichissement des données de l'interface, avec possibilité de désactiver ce rafraichissement ou de de n'effectuer qu'un seul rafraichissement.
— Bouton de retour au point de départ.
— Bouton d'arrêt d'urgence : désactiver toutes les parties de l'intelligence et émuler toutes les valeurs des capteurs et actionneurs. Le système n'est plus autonome, l'utilisateur prend la main dessus via l'interface.

Cette interface est avant tout un outil de test et de debug, elle n'est pas conçue pour télécommander à distance le bateau. Elle peut être utilisée pour visualiser à distance des données, ainsi que pour simuler certaines situations telles de des avaries logicielles ou matérielles. En cas d'avarie réelle, cette interface permet de déactiver et d'émuler la partie logicielle ou matérielle défaillante.

4.8 Gestion des logs

Classe de logging La classe SailLog permet d'enregistrer des logs en les écrivant dans un fichier en plus de les afficher dans la console et l'interface web, et de les classer en fonction de leur importance.

— Post : information d'importance très faible (n'est pas écrite dans le fichier)
— Notify : Information utile
— Success : Réussite d'un test, d'une action
— Warning : Comportement suspect, pouvant causer des bugs
— Critical : Erreur grave pouvant entraîner un dysfonctionnement partiel ou total du bateau.

Classe GPS La classe GPS, en plus de gérer l'acquisition des données géographique, log les positions du bateau dans un fichier séparé, afin de retracer le chemin sur une carte après la navigation.

Visualisation des logs via le web Les logs générés par le SailLog sont copiés dans un cache de log de la classe API, contenant les 256 derniers logs du système, avec leur niveau de log et leur date.
Ces logs sont ensuite récupérés via l'API REST en effectuant une requête Ajax, puis affichés sur l'interface. L'interface fournit une fonctionnalité de recherche de logs par mot clé et par niveau de log.

4.9 Mise en place des capteurs et actionneurs

4.9.1 Gouvernail

Le gouvernail est actionné par un servomoteur fourni avec le voilier. Le servomoteur produit une rotation qui est transmise au gouvernail et qui lui permet de pivoter pour diriger le bateau.
Ce servomoteur s'alimente en 5V et utilise un PWM de période de 20ms. Le duty time (temps à l'état haut) varie généralement entre 1 et 2ms suivant la position à adopter.

4.9.2 Voiles

La grand-voile et le foc sont actionnées par un servomoteurs propre à chaque voile. Ces servomoteurs sont indépendants, mais l'intelligence envoie les mêmes ordres à ces deux actionneurs pour des raisons de simplicité. L'envoi de deux ordres différents définis grâce à des polaires est cependant possible et pourrait être l'objet d'améliorations futures.
Ces servomoteurs s'alimentent en 5V et utilisent un PWM de période de 20ms. Le duty time varie généralement entre 1 et 2ms suivant la tension voulue.

4.9.3 Accéléromètre

L'accéléromètre utilisé est l'ADXL335.

Cet accéléromètre s'alimente normalement en 3.3V, mais accepte une tension minimale d'alimentation de 1.8V. Pour des raisons de compatibilité avec les CAN du BeagleBone (qui ne supportent pas une tension supérieure à 1.8V), il faut l'alimenter en 1.8V. Pour cela, des broches d'alimentation 1.8V sont prévues sur le BeagleBone. Il fonctionne de manière analogique, renvoyant théoriquement une tension entre 0 et 1.8V (pour une alimentation en 1.8V) pour des accélérations respectivement de -3g à +3g. Le symbole g correspondant au vecteur gravité.
Trois données sont récupérées : l'accélération selon x, y et z dans un repère orthonormé. Ces valeurs sont exploitées pour calculer la gite du bateau.
Le programme récupère les valeurs émises par les CAN, qui correspondent à des entiers représentant la tension en entrée du CAN en millivolts. Ces valeurs vont de 0 à 1799. Il faut adapter ces valeurs pour les transformer en vecteur d'accélération sur -3G à +3G, en tenant compte des éventuelles imprécisions du composant, notamment sur l'axe Z qui comporte le plus d'imprécisions, selon la documentation.

4.9.4 GPS

Le GPS utilisé est le BU-353-S4. Il s'agit d'un GPS peu consommateur qui possède une interface de connexion USB.

L'interface entre le daemon de lecture des capteurs et l'équipement GPS se fait via un daemon linux nommé gpsd. Le package utilisé ici est en version 3.9-4. La visualisation en console de ses données est possible grâce aux logiciels gpsmon ou cgps, le dernier étant le plus complet. Ce daemon possède une librairie nommée libgps pour exploiter ses données. Deux headers peuvent être inclus pour utiliser cette librairie : gps.h pour la librairie en C, gpsmm.h pour cette même librairie, mais en C++. Lors de la compilation, il faudra lier la librairie libgps, ainsi que la librairie math qui peut être nécessaire pour faire fonctionner libgps. Les flags de compilation à utliser sont donc : -lm -lgps. la librairie traite les trames NMEA et les convertis au

format JSON. La libairie les interprète ensuite en les stockant dans une structure, qui est lue pour récupérer les données du GPS.

Le programme de lecture des données GPS est donc composé d'une classe GpsHandler qui possède deux parties : les fonctions de lecture des données stockées par la classe, et un thread d'actualisation de ces données par interrogation du daemon gpsd. L'utilisation de la librairie pthread s'est imposée ici. Un bug est présent dans une des librairies du compilateur g++ sur système ARM, il empêche l'utilisation de la librairie thread du C++11. Pour utiliser la librairie pthread, il faut ajouter le flag -lpthread lors de la compilation.

4.9.5 Girouette

Le capteur angulaire de la girouette est un codeur incrémental optique. Ce codeur est une combinaison 6 bits binaires répartis entre codage binaire pur et code gray. Le code source permettant de décoder cette information sur 5 bits a été fourni par M. Reboux, et a été amélioré par l'équipe pour le faire fonctionner sur 6 bits. Le quantum est alors de 5,625°, ce qui est suffisamment précis pour le système.
La girouette a été fixée en tête de mât pour une prise au vent maximum. La distance entre la girouette et le capteur a été réduite afin de limiter les frottements sur l'axe de rotation. Le poids a été minimisé en isolant la girouette de l'humidité à l'aide d'un fin film plastique, afin de ne pas augmenter la gite du bateau (effet de balancier avec un poids en tête de mât).
Note : Le codeur optique est sensible à la lumière, il est donc conseillé de l'isoler de toute interférence lumineuse extérieure.

4.9.6 Boussole

La boussole utilise un bus I2C pour communiquer avec le Beaglebone.
Les valeurs envoyées par la boussole ne sont pas très fiable car elles sont influencées par la gite du bateau. il faudrait utiliser l'accéléromètre pour corriger ces valeurs.
Le capteur est également très sensible aux perturbations de champs magnétique, il faut donc veiller à ne pas placer d'éléments électromagnétiques qui pourraient altérer les mesures.

5 Cahier de tests

Autopilot Avec un cap à suivre de 90˚ :

- Quand le compas est <87˚, la barre doit se déplacer de -1˚toute les secondes : **OK**
- Quand le compas est >93˚, la barre doit se déplacer de +1˚toute les secondes : **OK**
- Quand le compas est entre 93˚et 87˚, la barre doit rester immobile : **OK**
- Quand le compas est >93˚, la barre doit aller jusqu'à -45˚puis retourner à 0˚ : **OK**

SailHandler (Le vent est par rapport au bateau, 0˚étant un vent de face)

- Quand le vent est à 0˚, la tension est maximale (255) : **OK**
- Quand le vent est à 25˚, la tension est maximale (255) : **OK**
- Quand le vent est à 45˚, la tension est presque maximale : **OK**
- Quand le vent est à 90˚, la tension est moyenne : **OK**
- Quand le vent est à 180˚, la tension est minimale (0) : **OK**
- Quand le vent est à 25˚, que la gîte est de 30˚, la tension est maximale : **OK**
- Quand le vent est à 25˚, que la gîte est de 50˚, la tension diminue progressivement : **OK**
- Quand le vent est à 25˚, que la gîte est de 20˚, la tension augmente progressivement : **OK**

DecisionCenter

- Quand la coordonnée GPS du bateau se rapproche de la cible à 5m, le DecisionCenter change de cible : **OK**
- Quand la coordonnée GPS du bateau s'éloigne de la route, les polaires font virer le bateau de bord : **OK**

GPS

- Le GPS a une valeur de longitude et de latitude valide, il renvoie ces valeurs : **OK**
- Le GPS a une valeur de longitude valide mais pas de latitude valide, le GPS ne renvoie rien : **OK**
- Le GPS a une valeur de latitude valide : **OK**
- Le GPS n'a pas de valeur de latitude et de longitude valide, le GPS ne renvoie rien : **OK**
- Position en salle 210 ISEN Brest en (48.407127˚, -4.495195˚) : **OK** avec quelques erreurs de mesure.
- Position en salle 239 ISEN Brest en (48.406699˚, -4.495766˚) : **OK** avec quelques erreurs de mesure.

Gite

- Quand il n'y a pas de gite (0˚), l'accéléromètre renvoie une valeur de 0˚ : **KO**
- Quand il y a de la gite vers tribord (valeur positive), l'accéléromètre renvoie la valeur positive correspondante : **KO**
- Quand il y a de la gite vers bâbord (valeur négative), l'accéléromètre renvoie la valeur négative correspondante : **KO**

Boussole La boussole doit être initialisée vers le Nord avant les tests.

- Le bateau est orienté vers le Nord, la boussole renvoie 0˚ : **OK**
- Le bateau est orienté vers l'Est, la boussole renvoie 90˚ : **OK**
- Le bateau est orienté vers le Sud, la boussole renvoie 180˚ : **OK**

— Le bateau est orienté vers l'Ouest, la boussole renvoie 270° : **OK**

Girouette

— Le vent vient de face, la girouette renvoie 0° : **OK**
— Le vent vient de l'arrière, la boussole renvoie 180° : **OK**
— Le vent vient de la droite, la boussole renvoie 90° : **OK**
— Le vent vient de la gauche, la boussole renvoie -90° : **OK**

Servomoteurs

— Le servomoteur s'arrête normalement quand il ne reçoit pas de commande (pas de tremblements, de forcements) : **OK**
— La position minimale dans le programme correspond à la position minimale réelle : **OK**
— La position maximale dans le programme correspond à la position maximale réelle : **OK**

6 Problèmes rencontrés

La réalisation du projet ne s'est pas faite sans soucis, différentes difficultés sont apparues :

— Le retard de la partie hardware a considérablement retardé les tests du bateau en conditions réelles, notamment les réglages de l'intelligence.

— La stabilité du matériel Hardware a posé quelques problèmes, notamment au niveau de la connectique qui n'était pas adaptée et générait des faux contacts. Il a fallu refaire une partie de la connectique.

— Lors des premiers test, il est apparu qu'il restait des fuites, il a fallu mettre la coque dans un bassin en le tapissant de papier absorbant pour détecter l'origine des fuites. Celles ci ont été rebouchées avec du silicone.
Il est important de boucher toutes les fuites, la sécurité du matériel électronique est en jeu.

— Le fonctionnement de la girouette a fait perdre quelques jours à l'équipe, ne savant pas que le codage du capteur étant un mélange de binaire pur et de code gray. De plus, la première solution proposée de placer la girouette dans la coque et de faire remonter un fil de fer jusqu'à la girouette causait trop de frottements. Il a donc fallu placer le capteur et la girouette en tête de mât, en la rendant étanche et imperméable à la lumière qui interfère avec les capteurs.

— Certaines assertions (erreurs du programme) ne sont pas remontées à la console lorsque le programme est exécuté sur une cible ARM. L'origine du bug est donc difficile et oblige de reproduire le bug sur un processeur x86.

— Un bug dans le driver PWM du Kernel de base fourni avec la distribution rend impossible l'utilisation de plus de deux PWM distincts. Pour palier à cela, il a fallut corriger le driver PWM et recompiler le Kernel. (Procédure détaillée dans la partie 3.2.2)

— Les fils permettant aux servomoteurs d'actionner les voiles ont tendance à se coincer dans les équipement du bateau (couvercle, gouvernail, servomoteurs). Pour remédier à cela, de nouveaux couvercles en styrodur on été fabriqués. Ils recouvrent tous les éléments susceptibles de gêner les fils et remplacent les capots d'origine en bois et les élastiques en se bloquant par simple pression.

7 Évolution du programme

7.1 Détection des pannes

Faute de temps, cette fonctionnalité n'a pas été implémentée. Cependant des pistes de réflexion existent. Les critères de base pour la détection de pannes seraient :

— La détection de valeurs trop éloignées les unes des autres. Si cette suite de valeurs est trop absurde, c'est que le capteur a un dysfonctionnement. Il peut s'agir de valeurs erronées en provenance d'un capteur défaillant ou de bruit dû à une patte du capteur non connectée.

— La non mise à jour d'un capteur dans un laps de temps prédéterminé peut correspondre à un capteur mécanique bloqué ou à un composant électronique déconnecté. Si cette période sans mesure devient trop importante, il y a peut être une avarie.

7.2 Gestion des pannes

La gestion des pannes peut être améliorée. Théoriquement, il est possible de naviguer uniquement avec le GPS et la barre.

Pour définir ces comportements en cas de panne, l'API du programme permet deux solutions :

— Émulation des capteurs avec des valeurs calculées par le DecisionCenter

— Changement de tactique de navigation dans le DecisionCenter

7.3 Apprentissage et auto-étalonnage

Grâce au système d'intelligence par polaires, le bateau pourrait se mettre en mode d'apprentissage où il pourrait tester les différentes vitesses en fonction de l'orientation et redéfinir une nouvelle polaire de vitesse. Ce scénario pourrait être envisagé en cas d'avarie où il faudrait redéfinir une nouvelle façon d'avancer).

8 Conclusion

Tous les objectifs fixés par l'équipe Software n'ont pas pu être réalisé car ses membres ont du basculer sur la partie Hardware afin de la faire avancer et avoir une base de système opérationnelle. Cependant, tous les objectifs de base du Software ont été réalisés et les pistes d'améliorations explorées.

En simulation, le programme d'Intelligence fonctionne et passe tous les tests effectués. Les tests en milieu réel (sur le lac de Saint Renan) n'ont pas permis de régler et débugger l'Intelligence à cause de l'instabilité de la partie hardware (faux contacts, pistes sectionnées, capteurs incorrectement étalonnés, ...).

9 Bibliographie

D Programming Language Tutorial par Ali Çehreli
`http://ddili.org/ders/d.en/` (Version web et PDF)

JavaScript Les bons éléments par Douglas Crockford
Numéro ISBN-13 : 978-2-7440-2582-2
Edité par Pearson

Movable Type Scripts par Chris Veness
`http://www.movable-type.co.uk/scripts/latlong.html`

Wiki de GDC `http://wiki.dlang.org/GDC`

10 Liens utiles

Site officiel du langage D
`http://dlang.org/`

Bibliothèques standard D (Phobos)
`http://dlang.org/phobos/index.html`

Site officiel de la bibliothèque Vibe.d
`http://vibed.org/`

Site officiel d'AngularJS
`https://angularjs.org/`

Document sur la navigation par polaires par Thomas ABOT `https://raw.githubusercontent.com/triskell/voilier-doc/master/docs/AIUR.pdf`