Projet Industriel 2011

Robot – Radeau Autonome

Projet réalisé par :

COQUELIN Rémi
BECQUET DE MEGILLE François

Promotion ISE 2012

Sommaire

Introduction (p3)

I. Organisation (p4)

a) Cahier des charges et organisation (p5)
b) Emploi du temps (p5)
c) Répartition des tâches (p6)

II. La partie mécanique (p7)

a) Justification des choix mécaniques (p9)
b) Organisation de la construction (p12)
c) Essais et performances (p13)

III. La programmation (p15)

a) Choix de l'électronique et interfaçage (p17)
b) Choix de la stratégie de base (p20)
c) Elaboration des fonctions de base (p21)
d) Elaboration des stratégies de pilotage (p26)

IV. Essais et perspectives (p27)

a) Organisation des essais (28)
b) Améliorations et perspectives (p34)

Conclusion (p36)

Synthèse en anglais (p37)

Remerciements (p39)

Bibliographie (p40)

Annexes (p41)

Introduction

Aujourd'hui la robotique connaît une croissance sans précédent et concerne tous les domaines, du médical au militaire. Pourtant, à l'heure où des drones sont capables de voler des jours entiers sans l'intervention humaine, où des fusées propulsées à des vitesses folles viennent larguer des satellites précisément sur leur orbite de travail, aucun navire n'a jamais traversé l'Océan Atlantique de façon tout à fait autonome.

En effet, il est connu que l'Océan Atlantique est un lieu propice aux tempêtes et où le trafic maritime est particulièrement dense. Il est en outre animé par de nombreux courants marins et aériens, tel le Gulf Stream. Si donc on imagine que le navire puisse résister aux tempêtes, encore faut-il avoir la chance - ou la prévoyance - qu'il n'entre pas en collision avec d'autres navires, ou bien qu'il ne soit emporté par une dérive incontrôlable.

Ainsi, l'importance du défi à relever et la pluridisciplinarité des aspects sous lesquels il faut penser le futur navire pour une telle traversée, sont deux arguments qui ont su persuader les deux élèves ingénieur électroniciens que nous sommes. En effet, il faudra imaginer et concevoir non seulement la structure du navire, mais également l'électronique embarquée et surtout le code qui permettra – ou non – de rallier pour la première fois la vieille Europe à la jeune Amérique avec un navire totalement indépendant.

Ce défi est d'autant plus important que le navire devra être propulsé à la voile, moyen le plus fiable techniquement et le moins consommateur en terme d'énergie, mais qui complique la programmation. La plus grande difficulté sera probablement de devoir tenir compte de la dérive et de gérer les situations à problèmes telles un vent capricieux, une tempête... La robustesse du code, de la plate-forme et de l'électronique embarquée devra donc être sans faille. C'est pourquoi la ligne de vie de notre projet s'appellera : simplicité. Simplicité de la plate-forme, du programme mais également de l'électronique qui sera constituée en tout et pour tout d'un G.P.S, d'une boussole, d'une carte à base de microcontrôleur et d'un servomoteur. En effet, le moins il y a d'éléments intermédiaires, le plus la probabilité d'erreur dans le système est faible.

Cependant notre ambition se limitera dans un premier temps à relier deux points G.P.S. En outre notre navire, plate-forme comme programme, devra être modulable pour pouvoir être facilement développé et complété. Si tout fonctionne comme nous le souhaitons, nous pourrons alors voir plus grand et nous attaquer à des distances de plus en plus importantes, des trajectoires de plus en plus complexes. Amélioration après amélioration, un jour peut-être, l'objectif final, la traversée de l'Atlantique, sera atteint.

Ainsi nous verrons dans un premier temps comment est organisé notre projet et son déroulement global, avant de rentrer plus en avant dans les détails de la plate-forme mécanique et de la programmation. Nous enchaînerons par un descriptif de la campagne d'essais, les résultats finaux auxquels nous avons abouti, puis les perspectives qui s'offrent à notre petit automate sur le chemin de l'Atlantique.

I. Organisation

Nous partons de rien, il y a donc une importante quantité de travail à fournir. L'efficacité est donc la règle et passe par un emploi du temps rigoureusement tenu et une répartition des tâches optimale.

a) Cahier des charges et organisation

Le cahier des charges est le suivant :

– Concevoir un robot autonome à propulsion à voile dimensionné pour une traversée de l'Atlantique;
– Relier avec la plate-forme ainsi réalisée deux points G.P.S de façon automatique. Les seuls capteurs utilisés seront une boussole et un G.P.S. Le seul actionneur sera un servomoteur;
– Faciliter les possibilités de développement du projet par une grande modularité.

En outre le projet contient clairement deux volets :

– la partie mécanique : il s'agit de la définition, de la construction et des essais de la plate-forme mécanique;
– la partie électronique et la programmation : il s'agit du choix des composants électroniques, de leur implantation, de l'écriture des algorithmes et des essais associés.

b) Emploi du temps

L'emploi du temps est disponible en annexe 1. Il se répartit globalement comme suit :

Tâche	% temps affecté	% temps réel
Définition, conception et essais de la plate-forme mécanique	30,00%	38,00%
Implantation de l'électronique embarquée	8,00%	12,00%
Programmation	24,00%	27,00%
Essais	38,00%	23,00%

Tableau 1 : Répartition du temps

L'emploi du temps théorique est prévu de façon à ce que nous passions un temps raisonnable sur la plate-forme mécanique (30%) pour ne pas pénaliser la partie électronique et surtout la programmation. Enfin, conscients qu'il faudra beaucoup d'essais avant que le code fonctionne, nous prévoyons un temps conséquent pour les tests : 38% du temps total.

Dans la réalité, comme on peut s'y attendre, le temps consacré aux essais a été raccourci au bénéfice de la programmation, qui nous a donné plus de fil à retordre que nous ne le pensions. Cependant des essais non mentionnés dans le tableau ont été réalisés tout au long de la programmation pour valider les unes après les autres nos fonctions; les 23% d'essais du tableau ci-dessus correspondent en fait à une période majoritairement centrée vers les essais.

En outre il est difficile de chiffrer le temps total que nous avons passé sur le projet, sachant que nous y avons également beaucoup travaillé hors-séance en fonction des délais et de la quantité de travail à fournir à chaque moment.

c) Répartition des tâches

Le tableau de répartition des tâches est disponible en annexe 2.
Nous avons tous les deux une certaine expérience en modélisme, seulement pour gagner du temps les tâches ont été réparties en fonction des aptitudes de chacun. Il est à noter que toutes les décisions ont cependant été prises conjointement, en particulier la définition des fonctions du code et les choix de l'architecture mécanique et électronique. Ceci étant fait, nous nous sommes répartis les tâches techniques.

Entrons maintenant dans le vif du sujet en étudiant la partie mécanique du projet.

II. La partie mécanique

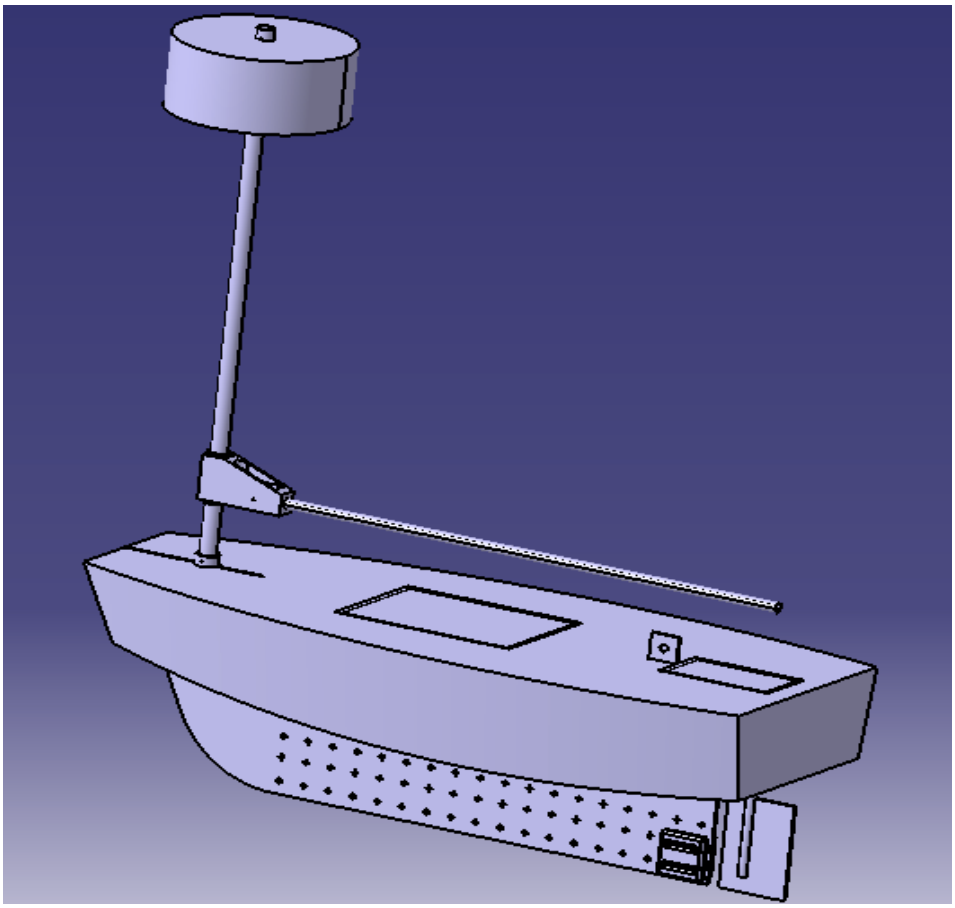


Image 1 : Extrait du logiciel CATIA montrant la plate-forme mécanique assemblée

La partie mécanique est un volet non négligeable de notre projet. Elle consiste en la définition et la construction d'un radeau, qui sera la plate-forme de support pour l'électronique. Pour ne pas compliquer notre programme, il est impératif que cette plate-forme soit crédible mécaniquement et en termes de performances. En effet, imaginons un radeau instable avec un comportement hydrodynamique aléatoire : il faudra alors prendre en compte ces contraintes dans notre programme. Il faut donc une plate-forme au comportement connu.

Le cahier des charges pour la plate-forme mécanique est le suivant :

Impératif	Dimensionnement	Solutions
temps	Être opérationnel pour le 5 avril 2011	1/ simplicité de définition 2/ simplicité de construction
robustesse	Être capable de traverser l'Océan Atlantique	3/ insubmersibilité 4/ solidité
performance	Être autonome en énergie et stable sur un plan d'eau Pouvoir accueillir l'électronique	5/ résistance à la dérive 6/ stabilité en roulis 7/ propulsion à voile et possibilité d'accueil de panneaux solaires 8/ aménagements étanches pour l'électronique

Tableau 2 : Cahier des charges de la plate-forme mécanique

a) Justification des choix mécaniques

La plate-forme telle que nous l'avons imaginée se compose de 4 grands ensembles :

① {mousse}
② {tôle principale}
③ {safran}
④ {mâture}

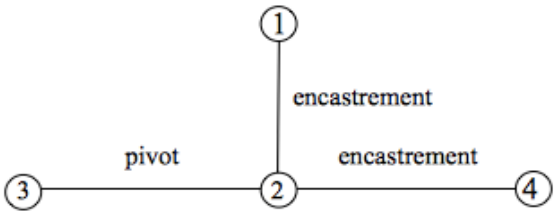


Schéma 1 : Graphe des liaisons

Détail des solutions : Reprenons les différents points vus dans le cahier des charges (tableau 1) et détaillons les un par un.

1/ Simplicité de définition : plutôt que de définir un navire de toutes pièces, nous prenons l'option de reprendre les plans d'une embarcation déjà existante. Ceci nous permet de ne pas perdre de temps au détriment de la programmation et des essais.
Notre choix se porte donc sur l'optimist, probablement le voilier le plus simple et rustique au monde. L'optimiste est un petit voilier destiné à la formation initiale à la voile des plus jeunes. Il est constitué d'une coque de forme grossièrement rectangulaire, d'une dérive, d'un mât et d'une voile.

Image 2 : l'optimist

Nous reprenons donc les plans de l'optimist, disponibles sur le site internet du constructeur, mais choisissons d'en faire une réplique à l'échelle ½ afin qu'il soit transportable, sans être trop petit pour pouvoir tenir face aux assauts de la mer.
Grâce à ces choix, les grandes décisions et la définition en détails sous le logiciel CATIA de la plate-forme nous aurons pris environ 10 heures cumulées en tout et pour tout.

2/ Simplicité de construction : il n'y a que quatre ensembles qui constituent l'embarcation. Trois d'entre eux sont usinables par l'atelier : il s'agit de toutes les pièces en aluminium (safran, tôle principale, mâture). Cela nous épargne une grosse partie du travail. Il ne nous reste plus qu'à tailler la mousse et à assembler le tout.

3/ Insubmersibilité : la mousse taillée selon la forme de coque de l'optimiste, assure l'insubmersibilité de l'embarcation.

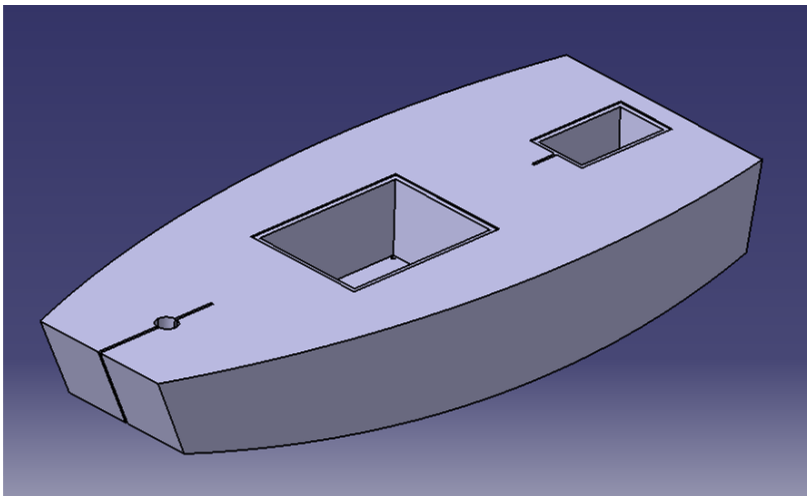


Image 3 : vue extraite du logiciel CATIA montrant la mousse

4/ Solidité : la tôle principale, en aluminium, est en fait le véritable bâti de l'embarcation. Elle assure la cohésion de tous les ensembles.
En outre, mise à part la mousse, les éléments électroniques et la voile, tous les autres éléments du radeau sont en aluminium. Cela garantit la solidité et la durée dans le temps du navire, les seuls points faibles étant donc la mousse et la voile qui peuvent s'abîmer par choc avec des rochers par exemple.

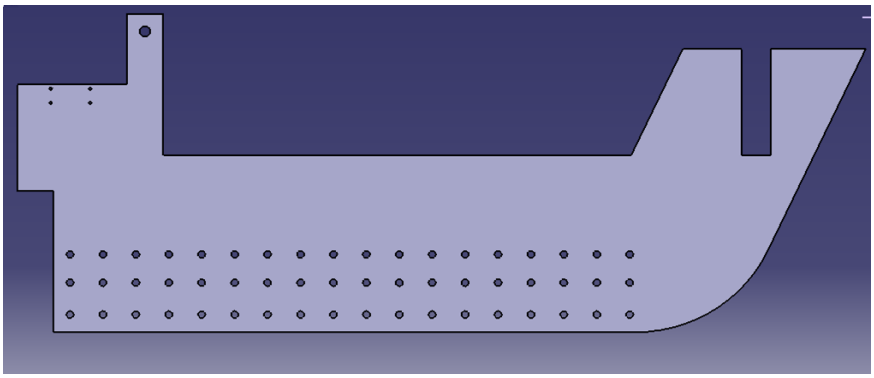


Image 4 : vue extraite du logiciel CATIA montrant la tôle principale

5/ Résistance à la dérive : la tôle principale a également le rôle de dérive, car elle dépasse largement en-dessous de la ligne de flottaison. Sa forme arrondie sur l'avant permet d'éviter que d'éventuelles algues ne viennent s'y accrocher et ralentir le radeau.

6/ Stabilité en roulis : comme tout navire digne de ce nom, il faut que l'embarcation soit stable sur l'eau, c'est à dire qu'elle ne puisse pas se retourner à la moindre vague. Pour cela, la tôle principale sert également de quille : des trous y sont prévus pour accrocher des plombs de plongée de 1kg (cf. Image 4). On peut y en accrocher 18 au total de part et d'autre (9 sur bâbord, 9 sur tribord), soit 18 kg donc. Bien entendu nous n'accrocherons que la masse qui nous semblera suffisante pour une navigabilité optimale, lors des essais. En passant, ces trous peuvent également servir à fixer des plaques de zinc pour éviter la corrosion, ou bien des capteurs supplémentaires (salinité de l'eau, …) dans le cas où l'embarcation serait détournée de sa vocation première.

En outre, si le navire est fortement déstabilisé par un fort coup de vent par exemple, il retrouvera naturellement sa position d'équilibre. Une bouée de tête de mât en mousse est en effet prévue, qui évite à l'embarcation de se retourner. Cette bouée est dimensionnée pour pouvoir supporter le poids du bateau. La quille, lestée, génère ensuite un couple qui le ramène à la verticale.

7/ Propulsion à voile et possibilité d'accueil de panneaux solaires : la propulsion du navire se fait à la voile. Celle-ci a une surface équivalente à la voile tempête d'un optimiste classique. Il n'est pas prévu qu'elle soit commandée, mais laissée libre de changer de bord au grès du vent. Une écoute, fixée à la tôle principale grâce à un système d'attache qui émerge de la mousse, va cependant limiter son débattement. La longueur d'écoute, et donc le débattement optimal, seront déterminés lors des essais.

L'embarcation peut de plus recevoir des panneaux solaires : la surface supérieure de la mousse est volontairement dépourvue de tout élément, pour pouvoir disposer d'une grande surface exposée au soleil (cf. Image 3). Cette surface est de 0.5 m², suffisamment grande pour pouvoir fournir une intensité d'environ 1 Ampère. Sachant que la consommation de l'électronique embarquée sera inférieure à 1 Ampère, cela garantit l'autonomie énergétique du bateau.

8/ aménagements pour l'électronique : Taillé dans la mousse, un logement permet d'y placer une boîte étanche suffisamment large pour pouvoir contenir notre carte électronique, la batterie ainsi que les capteurs. Un trou dans la mousse permet de communiquer avec le servomoteur, situé au niveau de la poupe, via différents fils (alimentation, signal). Le servomoteur est lui même installé dans un logement similaire. Pour éviter que l'eau n'y pénètre trop, ces deux logements sont fermés : celui où se trouve l'électronique, par une plaque de plexiglas (pour ne pas brouiller le G.P.S) et celui contenant le servomoteur, par une plaque en aluminium. Cependant afin d'éviter une accumulation d'eau due à une fuite ou à de la condensation, ces compartiments sont volontairement semi-étanches (le bas des compartiments est percé et communique avec l'eau, les compartiments étant au dessus de la ligne de flottaison).

Nous n'entrerons pas plus dans les détails de la réalisation de la partie mécanique (liaisons pivots, support du servomoteur...), ceci n'étant pas le but principal du projet. En outre, nos choix n'ont rien de révolutionnaire et sont fait dans un esprit de simplicité. Toutefois des photos des liaisons en question sont disponibles dans l'annexe 3.

b) Organisation de la construction

La construction de la plate-forme est un véritable sous-projet en soi. Elle pose plus de problèmes que nous ne le pensions, plus à cause des commandes auxquelles il faut procéder et des délais de livraison, que des imprévus mécaniques.

La structure en aluminium : Afin de gagner du temps, nous confions l'usinage de toute la structure en aluminium à l'atelier de l'ENSTA Bretagne. Il faut alors commander la matière première, sous forme de tubes et plaques de tôle, car le budget de l'atelier n'est pas le même que celui qui est consacré à notre projet. En outre, l'atelier a d'autres pièces en cours d'usinage et nous ne sommes pas sa priorité : ce pourquoi nous nous y prenons à l'avance pour pouvoir être certains d'avoir notre plate-forme mécanique à temps pour les essais.

La voile : La voile est fabriquée par un chercheur du département électronique, artisan à ses heures, et qui nous a aimablement proposé son concours. Le matériau de base, du tissu en « dacron » (matériau spécialisé pour les voiles), nous est fourni par « Voiles Océan », un spécialiste situé sur le port de commerce à Brest. Une fois la voile terminée, « Voiles Océan » se charge également de la pose des 10 œillets qui serviront à maintenir la voile sur le mât et la bôme. « Voiles Océan » nous fournira également le bout pour l'écoute.

La mousse : Le club robotique possède de gros blocs de mousse résistants à l'eau à disposition de tous ses membres. Il nous suffit alors de tailler directement ces blocs de mousse. Cependant, nous sommes contraints de les coller entre eux car aucun n'a la taille suffisante. La colle choisie est résistante à l'eau. En outre, pour vérifier la bonne résistance à l'eau de cette mousse, nous immergeons un échantillon dans la piscine d'essais de l'ENSTA Bretagne deux semaines durant. Au bout de ce temps, la mousse est presque intacte, légèrement plus molle sur les bords : cela suffira pour les premiers essais mais pour des traversées plus longues il faudra probablement penser à recouvrir l'ensemble de fibre de verre.
Pour usiner correctement la mousse, nous commençons par imprimer grandeur nature les plans de la coque, que nous collons sur le gros bloc de mousse obtenu donc par combinaison de deux blocs plus petits. Nous choisissons ensuite la technique du fil chaud pour découper les formes. De cette manière nous avons un usinage beaucoup plus fin et continu que si nous avions utilisé le cutter et la scie (Image 5).

Image 5 : Découpe de la mousse au fil chaud en s'appuyant sur des tiges métalliques

L'assemblage final : Nous voici en possession de la mousse, de la tôle et de la voile, il ne reste plus qu'à les assembler. Cette étape ne pose pas de problèmes et peu d'obstacles sont rencontrés, mise à part sur le logement du servomoteur qu'il faut agrandir. Nous nous chargeons également de la fabrication de certaines pièces de dernière minute (tôles pour maintenir la mousse) ainsi que des usinages finaux. En outre, des perçages supplémentaires sont effectués par nos soins pour pouvoir notamment relier l'écoute à la bôme. Ceci étant fait, il nous reste à essayer le prototype ainsi obtenu.

c) Essais et performances

Notre plate-forme mécanique est prête le 05 avril. C'est le début des essais, l'objectif étant de savoir comment l'embarcation se comporte, ce qui va conditionner la programmation.

Méca Essai 1 : première mise à l'eau

Date : mardi 5 avril 2011

Objectif : observation de la flottabilité et correction éventuelle; observation de la stabilité.

Mise en œuvre : La première mise à l'eau s'effectue à la piscine d'essais de l'ENSTA Bretagne. Une grue permet de soulever le bateau.

Résultats : A vide, c'est à dire sans son lest, le navire flotte mais « pique du nez », la proue étant bien plus enfoncée dans l'eau que la poupe. Ceci est logique, étant donné que le mât en aluminium, de masse non négligeable par rapport à l'ensemble, se situe sur la proue et engendre donc un déplacement du centre de gravité vers l'avant. Or il s'agit de faire coïncider le centre de gravité avec le centre géométrique du bateau afin d'obtenir une navigabilité optimale. De façon empirique, nous ajoutons finalement 12 kilogrammes au niveau de la poupe, de façon à obtenir une ligne de flottaison horizontale. Ce rajout se fait par l'intermédiaire de plombs de plongée de 1Kg chacun, fixés sur la dérive par le biais de colliers en plastique aux emplacements prévus à cet effet. Par manque de place sur la dérive, nous rajoutons provisoirement deux plombs dans le compartiment servomoteur.

Nous tentons ensuite de retourner le bateau ainsi lesté. Les plombs situés au niveau de la dérive font également office de quille et engendrent un couple stabilisateur. En pratique, lorsque l'on tente de retourner le bateau, celui-ci résiste et retrouve très rapidement sa position d'équilibre. La stabilité en roulis est donc satisfaisante. En forçant beaucoup il est possible que la tête de mât touche l'eau, la bouée joue alors pleinement son rôle en empêchant la voile d'aller sous l'eau malgré nos efforts.

Méca Essai 2 : premier essai dynamique

Date : mardi 12 avril 2011

Objectif : observation des performances dynamiques en fonction du vent; réglage de l'écoute.

Mise en œuvre : comme nous ne connaissons pas le comportement du navire, ce premier essai dynamique doit donc se faire sur un plan d'eau calme par jour de beau temps. Le navire doit en outre être facilement récupéré. Notre choix se porte alors sur le lac de Ty-Colo, à quelques kilomètres de Brest. Le jour de l'essai, le vent est assez faible (force 2 à 3 avec rafales). Il y a quelques petites vaguelettes sur le lac, engendrées par le vent. Les conditions sont donc presque

optimales. Cependant, le vent est irrégulier est tourne fréquemment, ce qui complique les essais.

Le navire est transporté par monospace au lac, puis mis à l'eau à partir d'un ponton. Spécialement pour cet essai, nous avons adapté une radiocommande qui permet de contrôler le safran à distance. Cela permet de tester les différentes configurations de vent sans perdre de temps à devoir aller chercher le navire aux quatre coins du lac.

Les essais se déroulent de la manière suivante : le navire est lancé avec une petite vitesse initiale (poussé à la main), testé dans toutes les configurations de vent puis ramené au ponton. Nous changeons ensuite la longueur d'écoute et renouvelons l'expérience, l'idée étant de trouver la longueur optimale.

Résultats : malgré une voile que nous pensions sous-dimensionnée et le vent faible, le navire avance. La dérive liée au vent semble quasiment nulle.

Pour comprendre ce qui va suivre, il est nécessaire de rappeler les différentes configurations de vent (Schéma 2).

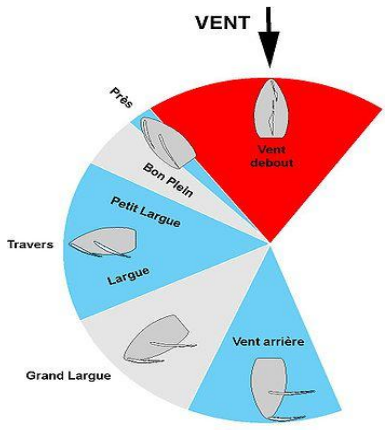


Schéma 2 : Les différentes configurations du vent

Le premier essai se fait en donnant beaucoup de mou à l'écoute de façon à ce que la voile ne soit presque pas bordée (écoute choquée). Dans cette configuration le navire tiens très bien le vent arrière mais beaucoup moins bien le travers et le près, ce qui est logique.
Au fur et à mesure que nous bordons la voile, les performances au près et en vent de travers s'améliorent. Force est de constater que malgré cela, la tenue en vent arrière reste correcte même si ainsi le navire va moins vite qu'avec une voile choquée.
Dans tout les cas, la situation la plus critique pour le navire est lorsqu'il remonte au vent; le mieux est donc de border au maximum la voile pour avoir une vitesse maximale, tout en laissant suffisamment de mou pour qu'il puisse tenir le vent arrière.
Après plusieurs tentatives de virement de bord avortées, nous constatons également qu'il faut une certaine vitesse au navire pour qu'il puisse virer de bord.
L'empannage ne pose pas de problème.

Cet essai marque la fin de la partie mécanique de notre projet. Maintenant que nous avons une plate-forme crédible, il est temps de s'atteler à la mise en place de l'électronique et à sa programmation.

Nous appelons notre plate-forme mécanique l' « Improbable » en pensant au comportement hasardeux qu'il risque d'avoir lorsque nous ferons nos premiers essais en automatique.

III. La programmation

Image 6 : programmation de l' « Improbable » à partir d'un ordinateur

Rappelons le cahier des charges: il faut être capable de relier deux points GPS sur des distances océaniques. A l'image de la plate-forme que nous avons conçue, il faut donc que le programme qui la dirige soit robuste, c'est à dire simple, efficace et modulable. Cela n'est pas évident car il est difficile d'avoir conscience à l'avance de tous les problèmes physiques à prendre en compte :

- provenant des caractéristiques propres aux capteurs (cap, ...)
- provenant de l'environnement (vent, houle, dérive...)

Le défi est donc de faire simple tout en tenant compte de tous ces paramètres.

En ce qui concerne l'organisation de notre travail, la logique exige que nous suivions le plan suivant :

a) choix de l'électronique et interfaçage
b) choix de la stratégie de base
c) élaboration des fonctions de base
d) élaboration de la stratégie de pilotage

a) Choix de l'électronique et interfaçage

Il y a un large choix de cartes électroniques embarquées disponibles dans le commerce, avec leurs avantages et leurs inconvénients. Notre choix se porte sur la carte « Arduino uno » (Image 7), de conception italienne. Les arguments qui justifient cette décision sont les suivants :
- possibilité d'y ajouter des capteurs avec une grande facilité d'assemblage
- aucune programmation d'interface n'est à faire avec les différents périphériques (bibliothèques fournies)
- faible consommation électrique
- programmation aisée en langage C
- aucune conception de carte nécessaire contrairement aux cartes à base de PICs

En plus de la carte, nous commandons donc un G.P.S, une boussole et un lecteur de carte micro SD qui nous permet de disposer de mémoire afin de stocker des données telles que la trace G.P.S et les décisions que prend le bateau.

Image 7 : la carte « Arduino » (en bleu) et son module GPS (bloc noir)

Les fonctions permettant la communication entre l'Arduino et les différentes cartes sont fortement inspirées du code disponible sur internet:
- « **setup »** pour l'initialisation des différentes cartes
- utilisation de la bibliothèque « Servo » pour la commande du servomoteur.
- « **GetDouble** », « **GetSingle** », « **GPSHeading** », « **GPSSpeed** », « **GPSLatitude** » et « **GPSLongitude** » pour le G.P.S
- « **Boussole** » pour la boussole.

Se pose ensuite le problème de l'agencement de ces différents capteurs sur la carte. Par exemple, il faut éviter de placer la boussole au-dessus du GPS au risque de gêner la communication du GPS avec le satellite.

En outre, au fur et à mesure de la programmation nous prenons conscience que nous aurions pu faire un meilleur choix si nous avions travaillé avec une carte « ArduPilot ». Sur ces cartes sont en effet déjà implantés les capteurs dont nous avons besoins ainsi que certaines fonctions de navigation qui simplifient énormément la programmation.
Lors de la programmation nous avons en effet rencontré les problèmes suivants:

Problème 1 : Problème de manipulation de caps lié au passage au Nord : lorsque l'on passe au Nord, le cap passe de 360° à 0° ou vice-versa. Il devient alors très délicat de manipuler les caps. Ainsi par exemple, si nous souhaitons calculer la différence *cap* du *cap1* avec le *cap2*, cela ne reviens plus à la simple soustraction :

cap = cap1- cap2

Car si : *cap2* = 323°, *cap1* = 030°, alors *cap* = -253° ce qui est impossible.

Dans un premier temps, comme nous ne pouvons ignorer le problème, nous élaborons une méthode qui distingue tous les cas de figure (code 1).

```
if(cap1<cap2){
if(cap2-cap1<((360+cap1)-cap2)){
cap=cap2-cap1;
}
else{
cap=(360+cap1)-cap2;
}
}
else{
if(cap1-cap2<((360+cap2)-cap1)){
cap=cap1-cap2;
}
else{
cap=(360+cap2)-cap1;
}
}
return cap;
}
```

Code 1 : méthode de calcul de la différence de cap

Ces lignes sont bien sûr lourdes mais la méthode est testée et validée. Cependant, la méthode ci-dessus ne permet que de calculer la différence entre un *cap1* et un *cap2*. Pour calculer par exemple le cap de la bissectrice d'un angle, il faut rajouter de nouveaux cas. Donc la méthode ne peut pas être généralisée en une seule fonction à laquelle on ferait appel, mais il faut la reprendre et la modifier au cas par cas. En conséquence, le programme devient de plus en plus lourd. Or plus il y a de lignes dans le programme, plus la probabilité d'erreur devient importante. Nous abandonnons donc finalement cette méthode (plus de 700 lignes pour les fonctions de base uniquement).

Il existe en effet un moyen beaucoup plus simple de s'affranchir du problème du Nord : utiliser une fonction périodique de période 2π. Nous remercions d'ailleurs Monsieur JAULIN qui nous a mis sur cette voie, le problème étant déjà bien connu.

La solution réside dans la vectorisation du cap : au lieu de donner le cap thêta sous forme d'un entier de 0 à 360, on le retourne sous la forme d'un vecteur cap de norme 1 et de coordonnées (cosinus (thêta), sinus (thêta)). Les fonctions sinus et cosinus étant 2π périodiques, le problème du Nord est donc réglé. La seule lourdeur réside dans le fait que nous utilisons désormais un tableau de deux coordonnées, plutôt qu'un simple entier, pour exprimer le cap.
Désormais, ce que nous appellerons par la suite « cap » désignera donc le vecteur cap.

Nous verrons par la suite comment l'utilisation du calcul vectoriel appliqué au vecteur cap permet d'obtenir facilement des données sur la dynamique du navire.

Ces conversions de caps sont visibles dans les méthodes « **GPSHeading** » et « **Boussole** » de notre programme.

Problème 2 : Problème d'utilisation des données de position GPS : le GPS retourne la vitesse, le cap vrai (par rapport au fond marin) et la position du navire. La position en particulier est constituée d'un certain nombre de données : degrés, minutes, dixièmes, centièmes et millièmes de minutes, orientation géographique (Nord, Sud, Est, Ouest), et cela pour la latitude et pour la longitude. Au final il faut donc gérer 6 nombres et 2 lettres, ce qui risque d'être assez lourd dans la programmation. Le mieux est donc de ramener tout cela à 2 nombres, un pour la latitude et un pour la longitude de telle sorte que :

1. La latitude évolue entre $-\pi/2$ et $\pi/2$ de part et d'autre de l'équateur;
2. La longitude évolue entre $-\pi$ et π de part et d'autre du méridien de Greenwich;

Pour cela nous convertissons donc toutes les données d'angle en radians et les additionnons.

Ces conversions de position sont visibles dans les méthodes « **ConversionLatitude** » et « **ConversionLongitude** » de notre programme. Elles sont utilisées pour chaque manipulation de la position des waypoints GPS entrés manuellement au programme pour la navigation.
Les méthodes « **GPSLatitude** » et « **GPSLongitude** » retournent directement la position dans le système décrit ci-dessus.

Problème 3 : Problème de dépassement de la valeur limite pour la commande du servomoteur : la position neutre du servo est définie en variable réglable par « neutreservo »; c'est une valeur déterminée expérimentalement pour laquelle le safran est parfaitement aligné avec la ligne de foi du navire (83°). Le servo a un débattement maximal « courseservo » autour de cette position neutre.

Le danger est que le servo reçoive un ordre de débattement supérieur à la limite fixée, ce qui peut donner des résultats imprévisibles. On élabore donc la petite fonction suivante :

Si l'ordre donné au servo > débattement maximal :
Alors nouvel ordre = débattement maximal;

Cela permet de s'affranchir du problème. Cette méthode de saturation de commande servo est visible dans la fonction **« servocmd »** de notre programme.

b) Choix de la stratégie de base

Deux grandes options s'offrent à nous pour piloter le navire:

Option 1 : optimisation du vecteur vitesse
Dans cette option nous ne connaissons pas la direction du vent. Le navire teste différents caps pour déterminer celui où sa vitesse est optimale. Il choisi ensuite le meilleur rapport cap/vitesse qui lui permettra d'atteindre le plus rapidement possible son point GPS de destination (schéma 3).

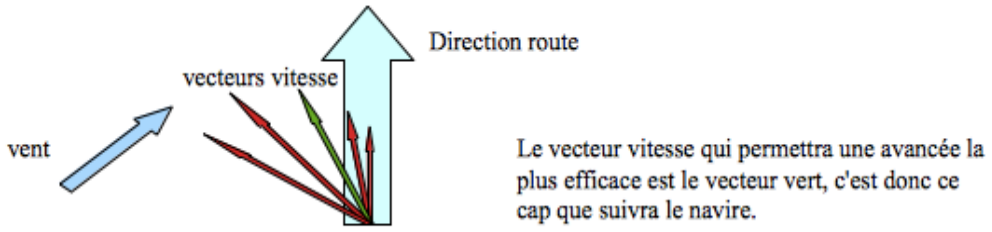


Schéma 3 : choix du meilleur vecteur vitesse

Avantages : simple et efficace théoriquement: le navire optimise au mieux le vent.

Inconvénients : aléatoire; on ne connait pas la direction du vent, donc on ne sait pas vraiment comment le navire va réagir dans des situations particulières: face au vent, tempête, dérive...
Cet algorithme est difficile à mettre en œuvre et long lors de la navigation car il faut alors tester un certain nombre de directions différentes. De plus, même si un vecteur a pu être déterminé, comment être sûr qu'il ne s'agissait pas juste d'un saut de vent?

Option 2 : navigation en fonction du vent
Dans cette option, on détermine la direction du vent. Une fois que celle-ci est connue, on a alors recours aux techniques classiques de navigation à la voile : remonter au vent, prendre un vent de travers...

Schéma 4 : quelques stratégies d'évolution au vent

Avantages : prévisibilité; connaissant le vent on contrôle mieux le navire et on se couvre contre d'éventuels imprévus.

Inconvénients : compliqué; il n'est pas simple de trouver la direction du vent sans girouette. Il faut ensuite élaborer toute la bibliothèque de stratégies pour arriver au point voulu en fonction de la direction du vent. De plus, si l'on se trompe sur la direction du vent, le comportement du navire risque d'être très aléatoire, ce pourquoi cette fonction devra être particulièrement robuste.

Intuitivement cette dernière solution semble plus logique que sa rivale : il paraît difficile de naviguer lorsque la direction du vent n'est pas connue.

Il faut donc mettre en place un ensemble de fonctions de base qui vont retourner des actions simples. Ces actions, nous les utiliserons par la suite pour élaborer les stratégies de navigation. Sans entrer trop dans les détails, nous allons décrire une par une ces fonctions de base en donnant leur utilité et leur principe global de fonctionnement.

c) Elaboration des fonctions de base

Le comportement du navire repose sur un ensemble de fonctions sans lesquelles rien n'est possible.

La fonction « BoussoleLissee » :

Action : donne le cap moyen suivi par le navire.

Principe : dans le cas d'un navire, il y a trois caps à prendre en compte (Schéma 5) :
- le cap vrai : cap du vecteur vitesse du navire par rapport au fond marin (compte tenu de la dérive liée aux courants). Il s'agit du cap donné par le GPS.
- le cap relatif : cap du vecteur vitesse du navire par rapport à la masse d'eau qui le supporte (sans prendre en compte la dérive liée aux courants)
- le cap boussole : cap de la ligne de foi du navire (l'axe longitudinal). Il s'agit du cap donné par la boussole.

On a les relations suivantes : cap vrai = cap relatif + dérive masse d'eau
cap relatif = cap boussole – erreur éléments

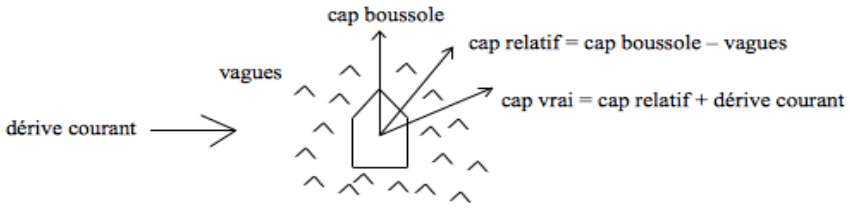


Schéma 5 : les trois différents caps

Le cap boussole n'est pas un cap fiable : il est très instable car directement lié aux éléments. En effet, sur l'eau, les vagues constituent pour le navire de nombreuses perturbations d'amplitude variable, qui font osciller le cap boussole autour de ce qu'on peut appeler un cap moyen. Ce cap moyen est le cap relatif, c'est donc ce cap que le navire suit réellement par rapport à la masse d'eau sur laquelle il se déplace. Or pour l'asservissement il est plus intéressant pour nous de suivre un cap moyen stable qui demandera de petites corrections ponctuelles, que le cap boussole, instable, qui exigera des corrections vives et incessantes.

Dans la fonction, nous faisons donc appel à un filtre passe bas qui permet d'avoir une valeur lissée de nos caps boussole, donnant en sortie notre cap vitesse relative.

La fonction « initialisation » :

Action : trouve la direction du vent.

Principe : trouver les deux caps de limite de virement de bord et calculer la bissectrice.
1- On fait tourner très doucement le navire. Tant qu'il n'est pas face au vent, celui-ci avance et continue de tourner. Lorsqu'il se trouve presque face au vent, il ne vire pas de bord puisque sa vitesse de rotation est trop faible pour cela. Mais le safran incliné l'empêche de repartir dans le sens inverse. Il se trouve donc dans une position d'équilibre (stabilité à la cape), son cap est quasi-constant. On prélève alors ce cap.
2- On fait ensuite tourner le navire dans le sens inverse et on répète ce qui est décrit ci-dessus.
3- On dispose donc de 2 caps, on prend la bissectrice entre ces 2 caps et on obtient ainsi la direction d'où souffle le vent.

Remarque 1 : Il faut tenir compte de l'erreur due aux éléments : les vagues et les rafales de vent, même de petite amplitude, peuvent fausser l'évolution des caps. Ceci est d'autant plus vrai que le navire tourne doucement et il est donc plus sensible à l'état de la mer. Ainsi si on se contente de dire par exemple au navire « si le cap diminue alors qu'il augmentait, alors nous sommes à la limite du virement de bord » et qu'une vague vient faire diminuer le cap de façon ponctuelle, le navire va croire qu'il a trouvé son cap. Ce n'est donc pas fiable.

Remarque 2 : Pour être sûrs d'avoir trouvé le cap limite de virement de bord, il faut donc s'assurer que le navire n'évolue pas durant un certain temps. On prélève donc régulièrement les caps et on en fait la différence. On défini un écart de tolérance "diffmin" défini en valeur de réglage. Si de façon continue, cette différence est inférieure à "diffmin", alors on considère que le navire n'évolue presque plus :

```
diffmin << 360;
entier n;
cap1;
attente;
cap2;
diff = cap1 – cap2;
si diff < diffmin : le navire est immobile à diffmin près;
cap3 = cap du moment;
```

Code 2 : haut niveau de l'initialisation

Remarque 3 : Choix du cap G.P.S ou du cap boussole : rappelons que ces deux capteurs renvoient un cap. L'ennui du cap G.P.S est qu'il renvoie le cap du vecteur vitesse vraie du navire, qui est d'autant plus imprécis pour notre méthode que :

1. le navire n'avance pas vite
2. le vecteur vitesse vraie n'est pas égal au vecteur vitesse relative s'il y a de la dérive

Or ce qui nous intéresse, c'est l'évolution du cap relatif pour détecter l'immobilité par rapport à la masse d'eau. Mieux vaut alors prélever pour cela le cap boussole qui donne une valeur du cap relatif.

Remarque 4 : Division de la fonction en sous-fonctions : comme il faut répéter la même action dans un sens puis dans l'autre pour avoir nos deux caps, on fait appel à trois sous-fonctions :

1. « diffcap » renvoie la valeur absolue du déterminant de la différence vectorielle des caps;
2. « capimmobile() » renvoie le cap pour lequel « diffcap() » n'évolue plus sur un certain temps;

3. « initialisation » fait tourner le servo dans un sens puis dans l'autre et appelle « capimmobile() » pour les deux cas, puis calcule la bissectrice des deux caps obtenus;

La fonction « dist » :

Action : donne la distance qui sépare deux points G.P.S.

Principe : il s'agit d'une fonction purement mathématique. Cette fonction renvoie la distance qui sépare deux points G.P.S en utilisant la formule « Haversine » créée pour la manipulation de coordonnées G.P.S.

Remarque : Cette fonction est utilisée dans un certain nombre de fonctions que nous allons voir, telles « SuiviRoute », « ecartroute » et « Navigation ».

La fonction « SuiviRoute » :

Action : fait suivre au navire une route définie par deux points G.P.S.

Principe : on calcule l'écart du navire part rapport à la route. On applique alors une commande proportionnelle à cet écart et à l'erreur de cap entre l'axe du navire et l'axe de la route pour le faire revenir sur la route (Schéma 6).

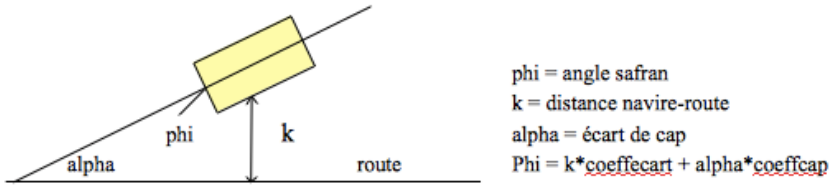


Schéma 6 : asservissement angle et distance

Remarque 1 : Division de la fonction en deux sous-fonctions :

1. « ecartroute » donne l'écart du navire par rapport à une route en mètres;
2. « SuiviRoute » agit sur le servo pour faire suivre la route au navire en agissant proportionnellement à « ecartroute » et à l'écart de cap.

Remarque 2 : Pour calculer l'erreur de cap étant donné que nous avons des vecteurs à disposition nous utilisons directement le calcul vectoriel. Nous utilisons en particulier la propriété suivante : le déterminant de deux vecteurs donne l'aire entre ces deux vecteurs avec un signe négatif ou positif selon la position de ces vecteurs l'un par rapport à l'autre.
Il suffit donc de calculer le déterminant entre le vecteur unitaire cap vrai du navire et le vecteur unitaire cap route. L'erreur de cap avec la route est proportionnelle à la valeur du déterminant, et le sens de correction dépend du signe du déterminant. Ceci nous donne donc l'erreur de cap.

En outre, il faut envisager le cas où le navire aurait tourné de plus de 90° par rapport à la route suivie. On détermine cela en calculant le produit scalaire des deux vecteurs : s'il est positif, c'est que l'on est dans la bonne direction. S'il est négatif, c'est que l'on fait marche arrière. Dans ce cas il faut faire demi-tour, on vire alors de bord.

Remarque 3 : Choix du capteur : nous avons vu que le cap donné par le G.P.S est plutôt imprécis aux faibles vitesses. En revanche, il est plus fiable que le cap boussole aux grandes vitesses, car il donne la direction du Nord vrai (Nord géographique), tandis que le cap boussole indique celle du Nord magnétique, de plus il permet de prendre en compte la dérive. Pour une optimisation de la précision de la méthode, il faut donc faire un compromis entre cap boussole et cap G.P.S en fonction de la vitesse. On distingue donc deux cas : lorsque le navire est en dessous d'une certaine vitesse V, on s'asservit sur le cap boussole, et lorsqu'on est au-dessus, on s'asservit sur le cap G.P.S. V sera déterminée lors des essais.

La fonction « remontervent » :

Action : fait suivre au navire une route vers le vent.

Principe : « remonter vent » s'applique lorsque le navire doit suivre une route parallèle à la direction du vent, avec le vent de face. Pour avancer, il doit alors tirer des bords tout en restant sur la route.
Le navire prend donc un cap intermédiaire de remontée au vent, entre le vent de face et le vent de travers. Il ne suit donc pas exactement la route mais se rapproche du bord de celle-ci. Une fois arrivé au bord avec une marge prédéfinie (pour éviter qu'avec l'inertie ou un éventuel contretemps il ne sorte de la route), on le fait virer de bord. Il repart vers l'autre côté de la route. Et l'on continue ainsi de suite (Schéma 7).

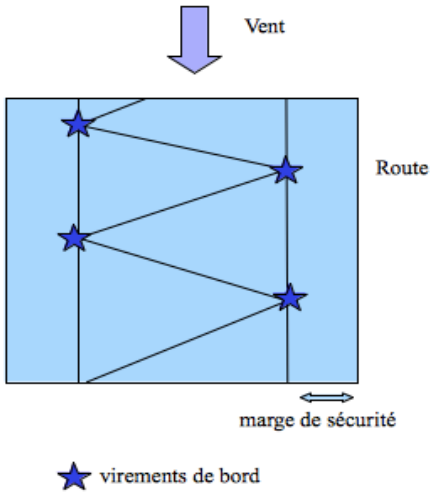


Schéma 7 : principe de « remontervent »

Remarque 1 : autre méthode possible : plutôt que de détecter l'intersection avec le bord de la route, la première idée était de définir de nouveaux waypoints dits « waypoints intermédiaires ». Le navire empruntait donc des routes intermédiaires entre ces waypoints, qui le faisaient converger vers le waypoint final de la route principale. Cela permettait d'affiner la trajectoire moyenne du navire en tirant des bords de plus en plus courts au fur et à mesure que l'on se rapprochait du waypoint final, pour ne pas le manquer. Cependant notre choix s'est finalement porté sur le principe décrit ci-dessus, moins précis mais plus simple.

Remarque 2 : Choix du capteur : même problématique que pour la fonction « SuiviRoute ».

La fonction « cgtpreviousWP » :

Action : actualise la position du waypoint de départ de la route à la position actuelle du navire.

Principe : on remplace les valeurs du waypoint de départ par la position actuelle. Ceci permet de redéfinir une nouvelle route lorsque l'on est sorti de celle que l'on essayait de suivre.

La fonction « navigation » :

Action : détecte un waypoint.

Principe : un waypoint est un cercle de diamètre réglable. On fait appel à la fonction « **dist** » pour savoir quelle distance sépare la position actuelle du navire par rapport au waypoint visé. Si celle-ci est inférieure au diamètre du waypoint alors on considère le waypoint atteint et on le défini comme waypoint de départ de la nouvelle route. Le waypoint suivant devient le nouveau waypoint cible.

Remarque : les latitudes et les longitudes sont en fait des tableaux : on peut donc y stocker plusieurs coordonnées qui correspondent à autant de waypoints. Il en est de même pour le diamètre des waypoints, qui est également un tableau de façon à pouvoir régler la précision souhaitée en fonction de la position du waypoint visé : en plein océan par exemple, on peut fixer un grand diamètre car on exige qu'une faible précision, mais sur un lac ou aux abords des côtes la précision doit être plus fine donc on mettra un plus petit diamètre de validation.

d) Elaboration des stratégies de pilotage

Nous sommes maintenant en possession des fonctions de base qui vont nous permettre de créer la fonction principale de pilotage. Nous l'appelons « **stratégie** », c'est en fait le « main » de notre programme.

La stratégie est élémentaire et se décompose en deux phases :

INITIALISATION : Le navire s'initialise et trouve la direction du vent.

COMPARAISON : Il compare la direction du vent avec celle de la route à suivre :
- si le vent vient de face (avec un certain angle de tolérance), on fait appel à la méthode « remontervent ».
- s'il vient d'ailleurs, on fait appel à la méthode « SuiviRoute ».

Gestion des cas non conformes :
- sortie de route
- impossibilité de suivre la route (dérive trop importante, …)
- changement de la direction du vent

Dans tous ces cas, le navire ne peut pas suivre la route. Une variable booléenne appelée « initdone » passe alors à « false » ce qui a pour conséquence de réinitialiser le navire, c'est à dire trouver à nouveau la direction du vent. Le navire recalcule alors sa route à partir de sa position actuelle.
De plus, dans le cas où les conditions pour naviguer ne seraient pas réunies (vent ou dérive trop importants, houle de trop forte amplitude, vent nul), le navire va avoir un comportement aléatoire probablement constitué d'un enchaînement d'initialisations-comparaisons jusqu'à ce que les conditions soient à nouveau favorables.

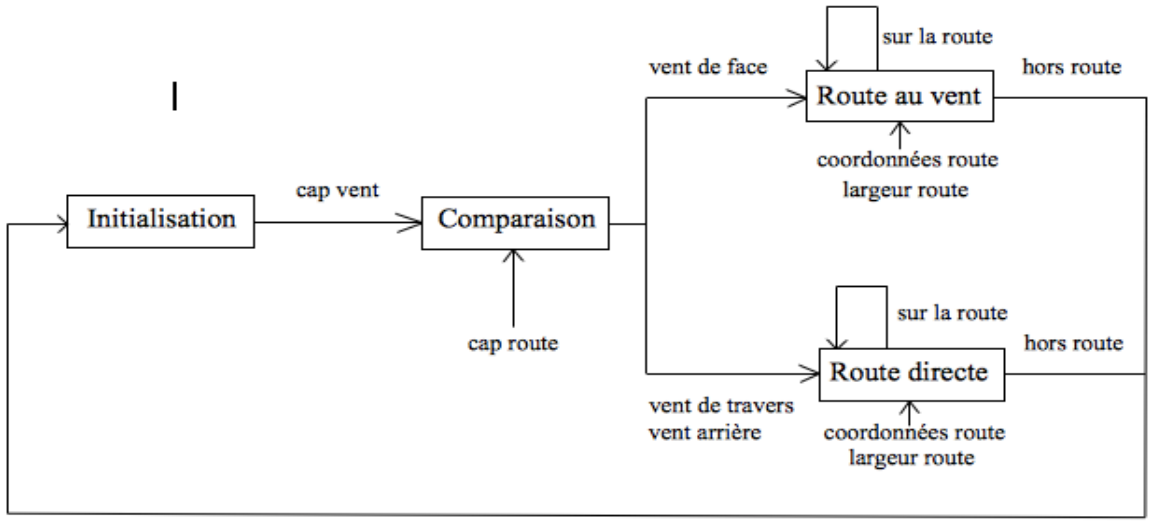


Schéma 8 : principe de la stratégie

IV. Essais et perspectives

Image 8 : configuration en vent arrière sur le lac de Ty-Colo

a) Organisation des essais

Les essais ont pour objectif de valider la stratégie, en commençant par les fonctions élémentaires, puis en montant en puissance au niveau de la complexité du code testé.

nom	elec essai 1	elec essai 2	elec essai 3	elec essai 4
date	11/05/11	18/05/11	25/05/11	31/05/11
lieu	ENSTA Bretagne	lac du Ty Colo	lac du Ty Colo	Moulin blanc
objectif	Validation fonctions de base	Validation fonctions de base	Validation fonctions de base	Validation stratégie

Tableau 3 : déroulement de la campagne d'essais

D'autres essais non mentionnés ci-dessus se sont déroulés à l'ENSTA Bretagne sur le stade tout au long de la période de programmation. Ceci permet de gagner du temps grâce à une mise en œuvre allégée, comme nous allons le voir par la suite.

Elec essai 1 :

Date : mardi 11 mai 2011

Objectif : valider les fonctions de base hors milieu maritime

Mise en œuvre : le terrain de rugby de l'ENSTA Bretagne constitue un lieu d'essais idéal. Les buts du terrain sont en effet visibles depuis le logiciel « Google Earth » et leurs coordonnées sont disponibles. On peut donc assimiler le terrain de rugby à une route cap au 173, d'une longueur de 100 m, de largeur réglable (20m dans notre cas), de waypoint de départ le but le plus au nord et de waypoint d'arrivée celui le plus au sud (Image 9).

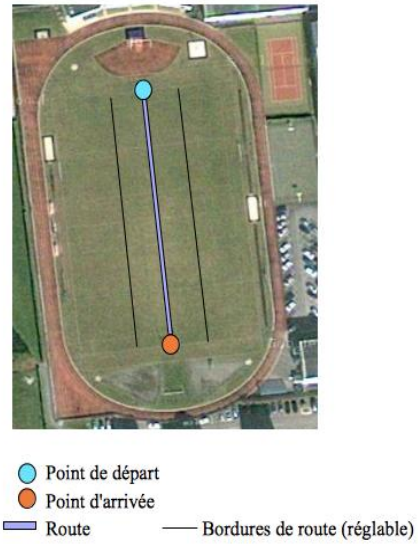


Image 9 : terrain de rugby de l'ENSTA Bretagne

De plus, pour éviter d'avoir à déplacer l'ensemble du navire, on ne fait les essais qu'avec l'électronique, un servo annexe à la place du servo du safran et un ordinateur connecté à la carte qui permet de lire en temps réel les informations. Les essais consistent à se déplacer avec ce matériel réduit sur le terrain de rugby et simuler le comportement du navire sur la route qui relie les deux buts entre eux.

Observations : nous n'entrerons pas dans les détails de toutes les corrections sporadiques qu'il a fallu faire pour aboutir au résultat final. Toutes les fonctions ont été testées et validées. Le navire est donc prêt pour des essais en conditions réelles.

Résultat : toutes fonctions validées.

Elec essai 2 : premier essai de la plate-forme automatisée

Date : mardi 18 mai 2011

Objectif : test de la fonction « SuiviRoute » sur l'eau.

Conditions : vent moyen du nord, 8 nœuds avec courtes rafales à 15 nœuds. Houle : nulle. Mer du vent : nulle.

Mise en œuvre : les essais de la plate forme mécanique ont déjà été réalisés au lac de Ty-Colo. Ce lieu proche de Brest présente l'avantage d'être calme et par nature sans houle et présente donc des conditions optimales pour les premiers essais. Plus tard lorsque nous aurons validé le programme dans ces conditions calmes nous pourrons envisager de passer à un milieu plus réaliste tel la rade de Brest.

En outre, afin de gagner du temps et de parer aux situations imprévues, nous mettons au point un système mécanique de prise de contrôle à distance du navire. Il s'agit d'un servo qui permet de passer de la carte électronique au récepteur radio sur le même principe qu'un interrupteur, pour le pilotage du safran. Pour actionner ce servo il suffit d'appuyer sur la gâchette de la radiocommande.

Pour ce premier essai grandeur nature, nous sélectionnons deux point G.P.S situés sur le lac : le point de départ étant le ponton et celui d'arrivée l'extrémité d'en face. Le vent viendra de travers. En théorie le navire doit naturellement suivre la route qui relie ces deux points. On le lance du ponton avec une légère vitesse initiale, la fonction « SuiviRoute » tournant en boucle.

Observations : les deux premiers essais sont surprenants : le navire parcourt quelques mètres puis fait volte-face et reviens précisément au ponton. Il se trouve que notre commande de barre était inversée.

Après correction, le troisième essai semble satisfaisant. Le navire corrige régulièrement et semble suivre la direction du point G.P.S sur l'autre rive de lac. Nous tentons de reprendre le contrôle avant que le navire ne s'éloigne trop, mais ne parvenons pas à le ramener vers la rive, étant probablement hors de portée radio. A un moment donné le navire commence à décrire d'incessants cercles. C'est à la nage que nous récupérons finalement notre prototype.

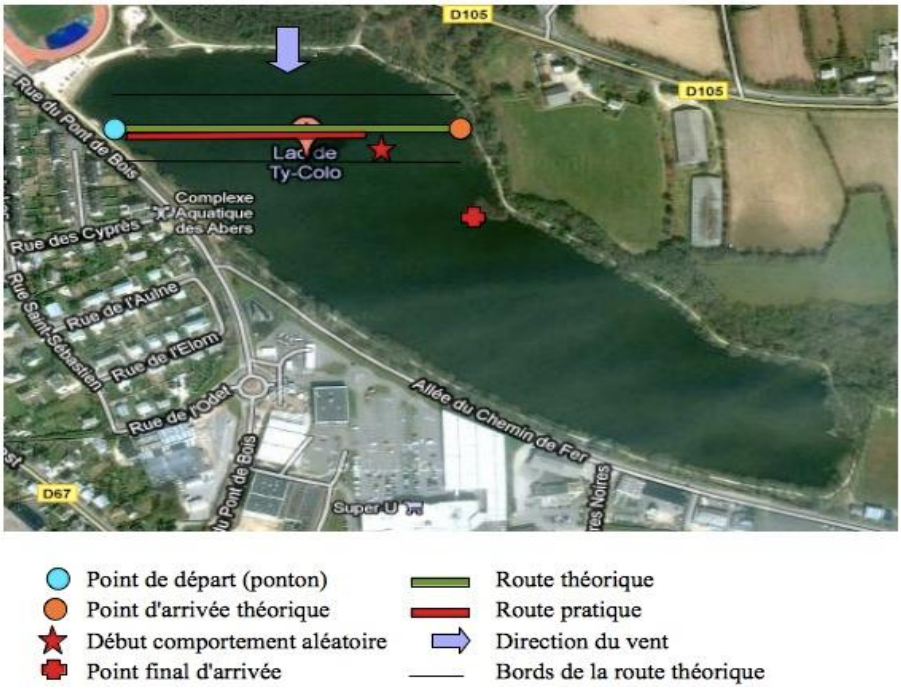


Image 10 : lac de Ty-Colo

Après réflexion, l'hypothèse la plus probable pour expliquer ce comportement insolite du navire est la suivante : en voulant le faire revenir vers nous, nous avons fait sortir le navire de sa route. Il ne se trouvait plus dans une position favorable par rapport au vent. Il essayait toujours d'atteindre le point GPS visé mais avec le vent de face cette fois, d'où les incessants cercles...

Résultat : fonction « SuiviRoute » partiellement validée.

Elec essai 3 : deuxième essai de la plate-forme automatisée

Date : mardi 24 mai 2011

Objectif : validation de la fonction « initialisation »

Conditions : vent moyen du nord, 6 nœuds avec courtes rafales à 10 nœuds. Houle : nulle. Mer du vent : quelques centimètres.

Mise en œuvre : la mise en œuvre est identique à l'essai précédent à ceci près que le navire est suivi en permanence par un opérateur sur un matelas pneumatique. Cela permet d'observer en direct les mouvements du safran, invisibles depuis le ponton, et de pouvoir ramener rapidement le navire s'il tend à trop s'éloigner, d'où un gain de temps.

On charge la fonction « initialisation » dans la carte. On s'attend donc à ce que le navire tourne sur bâbord, détecte le premier cap de mise à la cape, puis tourne sur tribord et détecte le second cap. On lance le navire du ponton avec une légère vitesse initiale.

Observations : La fonction initialisation ne semble pas du tout fonctionner pour plusieurs raisons :

Observation 1 : le navire est instable lors la mise à la cape : il a en effet tendance à virer de bord trop facilement, malgré un angle de barre relativement faible (5°) et un vent peu important.
Observation 2 : le navire est stable en vent arrière : étant donné que le safran est faiblement incliné et que lors de l'empannage (passage vent arrière) le navire subit un contre-couple dû à la voile, il ne

tourne pas comme on l'espérait mais maintient son cap. Le navire croit donc qu'il a détecté un cap de mise à la cape.

Observation 3 : ces situations de stabilité et d'instabilité arrivent de façon aléatoire alors que nous sommes par vent calme sur un lac peu agité.

Etant donné ces considérations, nous considérons qu'avec notre navire il est impossible de déterminer la direction du vent par la fonction « initialisation ».

Observation 4 : une observation d'ordre électromécanique est à faire concernant le servomoteur : lors d'un des essais, le bras, le servo et le palonnier du safran se sont mis sur un même axe, entrainant le blocage du système. Le servo a dû forcer puis détecter l'anomalie et cesser de fonctionner. Cela est facilement réparable puisqu'il suffit de débrancher puis rebrancher le servo. En revanche ce cas de figure pourrait compromettre une traversée océanique car personne ne sera là pour stopper l'alimentation électrique et la remettre (Image 11).

Image 11 : blocage dû à l'alignement des axes bras/palonnier

Résultats :

Résultat 1 : abandon de la fonction « initialisation » et renoncement à trouver la direction du vent pour diriger le navire. On tend vers une version allégée du programme où seule « SuiviRoute » sera appelée. Si le navire n'avance pas (augmentation de la distance séparant le bateau du waypoint à atteindre), alors on se considère face au vent et on passe à la fonction « RemonterVent » (Schéma 9).

Résultat 2 : mise en place d'une butée pour éviter que le bras mécanique ne se retrouve en situation bloquante.

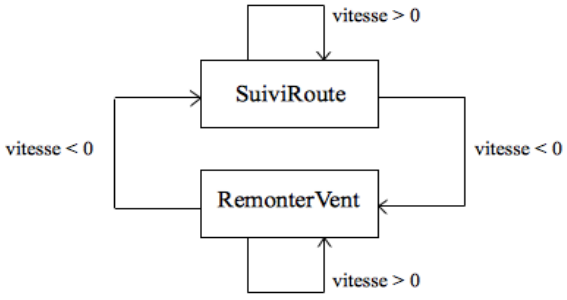


Schéma 9 : nouvelle stratégie

Elec essai 4 : validation finale

Date : mardi 31 mai 2010

Objectif : validation stratégie finale

Conditions : vent constant du Nord-Nord-Ouest de 10 nœuds, mer calme

Mise en œuvre : cette fois nous testons la nouvelle stratégie définie lors de l' « elec essai 2 » (dernier essai à Ty-Colo) en mer, près de la plage du Moulin Blanc à Brest. Par sécurité, le navire est suivi en kayak. Par crainte de problèmes supplémentaires nous n'avons pas enregistré la trace G.P.S, afin de simplifier le programme.

Le premier test consiste à longer automatiquement la plage sur toute sa largeur (700m) avec la fonction « SuiviRoute ». La plage étant orientée Ouest-Est, et le vent venant du Nord-Nord-Ouest, les conditions sont donc favorables.

Le second test consiste à relier trois points G.P.S entre eux selon un parcours Nord – Sud puis Est – Ouest avec donc un passage face au vent pendant lequel le navire devra s'adapter. On fera appel à la fonction « stratégie ».

Observations :

Observation 1 : la fonction « SuiviRoute » qui avait déjà été validée à Ty-Colo refait ses preuves; l'asservissement est précis et le waypoint d'arrivée est validé. Même lorsque l'on pousse le bateau jusqu'à le faire virer de bord, il revire et reprend sa route rapidement en corrigeant l'écart engendré (Image 12).

Observation 2 : La fonction « stratégie » fonctionne comme attendu. Après une première route vers le sud, le navire valide le waypoint et prend sa route vers le Nord-Ouest. Il se retrouve alors face au vent. Plutôt que de faire marche arrière, il tend alors à virer de bord tout seul. Or pour qu'il passe à sa stratégie de remontée au vent, il faut que le navire recule. Mais il continue d'avancer et tire des bords naturellement. Il va alors continuer ainsi quelque temps ensuite il se met à dériver jusqu'au port du moulin blanc sans jamais réussir à atteindre le troisième waypoint (Image 13).

Nous en déduisons que notre navire n'est décidément pas stable face au vent. Seule la fonction « SuiviRoute » est stable et fiable.

SuiviRoute :

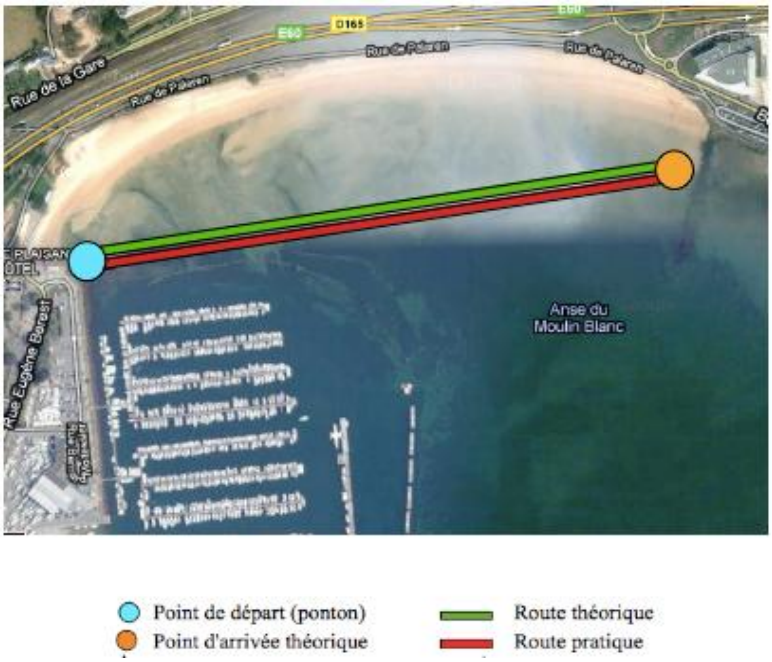


Image 12 : « SuiviRoute » au Moulin Blanc

Stratégie :

Image 13 : « strategie » au Moulin Blanc

<u>Résultat :</u> abandon de la stratégie et maintient de « SuiviRoute » pour seule et unique stratégie.

b) Améliorations et perspectives

Notre navire n'est encore qu'à l'état de prototype et de nombreuses améliorations sont à faire. Ce paragraphe peut s'avérer utile pour des étudiants qui reprendraient éventuellement le projet.

1- Améliorations mécaniques

– Strater la structure en mousse; cela permettra d'augmenter la solidité de la mousse qui épouse la forme de tous les chocs auxquels elle est soumise. Cependant, l'ensemble dérive-mousse ne sera plus démontable après l'opération, ce qui peut poser problème si une éventuelle manutention s'avère nécessaire.

– Empêcher le blocage du servomoteur : faute de temps, cette opération n'a pu être faite. Il est impératif de placer une butée limitant le débattement du bras reliant servo et palonnier, sous peine de blocage du système comme ce qui est arrivé lors de l' « Elec essai 2 » décrit plus haut.

– La commande de barre est également trop fragile (nous avons cassé le palonnier du servo), il faut revoir cette commande et trouver un servo qui ne se mette pas en sécurité automatiquement.

– Poser des panneaux solaires sur la face supérieure de la mousse laissée disponible à cet effet et les relier à l'électronique.

– Il est inutile d'augmenter la surface vélique. En effet lors des essais nous avons pu constater qu'un vent de 10 nœuds suffit à faire gîter le navire jusqu'à ce que l'eau recouvre sa surface supérieure.

2- Améliorations électroniques

– L'électronique fonctionne comme souhaité et il n'y a pas lieu de la changer, car elle n'a pas montré de signe particulier de défaillance lors des divers essais. Attention cependant lors des communications avec la carte SD qui tend à donner des résultats aléatoires. Le mieux est d'utiliser au minimum la carte SD, elle peut s'avérer cependant utile pour lire les différentes informations suite aux essais. En outre, il semble impossible d'utiliser un « SD.print » et un « Serial.print » de façon conditionnelle dans le code, il faut donc faire le choix entre la console ou la carte SD. La solution pour laquelle nous avons opté est de prévoir deux programmes, un spécifique « SD » pour les essais à l'eau et un spécifique « Serial » pour les essais à terre.

– Il serait également judicieux de revoir le système de prise de contrôle à distance. Le système mécanique d'interrupteur que nous avons conçu semble défaillant. Le mieux serait de modifier directement la carte Arduino pour y intégrer un module de prise de contrôle à distance sous format électronique. Cependant il est fort possible que nos problèmes proviennent d'une portée radio trop faible.

3- Amélioration de la programmation : Le choix entre le mode « remonter vent » et le suivi normal de la route s'effectue en comparant la distance séparant le bateau du waypoint à atteindre à 2 moments séparés dans le temps. Si cette distance augmente on change de mode, mais problème: on ne change jamais de mode car le bateau ne recule jamais, il remonte "naturellement" au vent. Voici les diverses solutions auxquelles nous avons pensé pour la suite du projet:

1) Conserver le système tel quel, supprimer la méthode de remontée au vent et commander uniquement en suivi de route, ce qui fonctionne par vent suffisant mais n'est pas performant.
Solution peu élégante...

2) L'instabilité observée quelquefois est probablement due à un asservissement un peu brutal. Diminuer la valeur des coefficients de l'asservissement notamment en ce qui concerne l'écart à la route pourrait régler le problème.

3) Le choix actuel du type de mode est inefficace. Voici des idées à développer:

-Le bateau est instable par vent de face et effectue de brusques virements de bord voire des demi tours complets: contrôler ceci via la boussole et détecter lorsqu'un changement de mode est nécessaire car le comportement du bateau est anormalement instable.

-Contrôler la vitesse GPS (donnée s'étant révélée fiable et assez précise), la moyenner, la projeter sur la direction à suivre et mettre un seuil minimum en dessous duquel on effectue un changement de mode. Lorsque le bateau sera instable cette vitesse diminuera grandement et les instabilités seront ainsi détectables. Problème: ceci ne fonctionne pas par faible vent... (Est-ce important? De toute façon on ne fera rien de bon sans vent...).

4) Ajouter un capteur permettant de savoir de quel côté est bordée la voile, ainsi il est aisé de savoir d'où vient le vent. Ce n'est pas vraiment un capteur de vent, cela peut être très fiable et peu coûteux (contact magnétique) et nous simplifierait grandement la tâche.

5) Il serait souhaitable de mettre les waypoints dans un fichier séparé par souci de clarté du code.

6) Remettre en fonction (dé-commenter et tester) les fonctions d'écriture dans la carte SD afin d'enregistrer la trace GPS et les décisions clés du bateau.

Conclusion

Concernant les points positifs, nous pouvons dans un premier temps souligner la réussite de la voile laissée libre, dans la mesure où nous n'avons eu aucun réglage à faire ni aucun incident à déplorer de ce côté là. Le navire avance correctement quelle que soit la direction du vent dès l'instant que l'on ait choisi un bon compromis de réglage d'écoute. En outre, nous sommes parfaitement capables de relier entre eux deux points G.P.S dans la mesure où le vent ne vient pas de face. On peut donc dire que l'objectif initial du projet consistant à relier deux points G.P.S à l'aide d'un navire à voile autonome a été atteint. Plus précisément, le programme fonctionne parfaitement par conditions calmes avec un vent favorable. Il est en revanche plus aléatoire lorsque le vent n'est pas favorable. Enfin, il n'a pas été testé dans des conditions météorologiques difficiles, comme une mer agitée ou un vent trop important.

Concernant les points négatifs, nous constatons que notre plate-forme, de par son instabilité face au vent, n'est pas compatible avec un algorithme de détection du vent de face. Pour résoudre ce problème il faudrait soit rajouter de nouveaux capteurs, soit se passer de cette information.

Enfin, il est à noter que ceci n'est qu'un début pour l' « Improbable » et que le principe de modularité ayant été respecté, on peut lui envisager un grand avenir : tant sur le plan mécanique, à commencer par de petits rajouts pour corriger ses quelques défauts, que sur le plan électronique avec l'ajout éventuel de nouveaux capteurs, et que sur le plan de la programmation en consolidant la stratégie globale. Pour qu'un jour peut-être notre petit bateau sillonne fièrement les flots tumultueux de l'Atlantique.

Industrial project: the Autonomous Ship

Nowadays, while technologies reach incredible records particularly in the field of robotics, not an uninhabited automatic boat has succeeded in crossing the Atlantic Ocean. This is a true challenge and we want to be part of it. Knowing that more a system is complicated, more it tends to fail, the guiding line of our project will be: simplicity.
Thus we have to design a wind powered, automatic boat, with only two sensors (G.P.S and compass) and one actuator (servomotor).

The project has two parts. The first one is the mechanical design of the boat which has to be:

- simple and resistant
- quickly designed and built

Thus the boat will be a half-reduced version of the optimist, a small sailing boat destined to the training of the youngest. In this way we just have to work on the plans of the builder, which saves a lot of time. Moreover this kind of boat has the reputation to be solid and robust.

Anyway we modified those plans a little to make the boat able to cross the Atlantic Ocean. The boat is indeed a four-part device:

- the main one is an aluminium keel with removable masses, which handles the three other parts, avoids the drift and adds rolling stability;
- the second one is the synthetic moss, designed according to the optimist plans, which provides buoyancy;
- the third one is the saffron, in aluminium too;
- the last one is the rigging, which mechanical clearance is limited by a taut rope.

To earn time, most of those parts are built by the ENSTA Bretagne's workshop. Once finished, the boat is equilibrated with the keel masses and tested on a lake. The tests show that it is well working, stable as expected and it is able to go about easily. Moreover during this test the optimal length of the taut rope is defined to have the rigging efficient in any wind direction, given that it is not commanded.

The second part of the project deals with electronics and programming. The electronic card is provided by “Arduino”, an Italian brand known for its reliability. G.P.S and compass are included in the pack with no interface programming to do. The language used is close from the C-language that we already know.

Our program is facilitated by the mathematical manipulation of vectors which is better than numbers. Moreover, among all the functions of our program, three are essential: first, the initialisation function finds the wind direction by testing several caps and detecting the one from which the boat does not evolve anymore. Second, the normal route-following function makes the boat follow a route defined by two G.P.S waypoints. Last, the front-wind route-following function makes the boat follow a route when the wind is not favourable, by going about every time the boat is close from the route's borders. The basic principle of our program is very simple and divided into four actions:

1- wind detection
2- comparison with the route to follow
3- choice between normal route-following function or front-wind route-following function
4- route-keeping check and restart of the entire process if exit from the route

Anyway during the first tests, the initialisation function was not efficient, so we failed to detect the wind direction. We thus decided to make our program even simpler with only two actions in a loop:

1- choose normal route-following function and check the boat has speed in the right direction

2- if no speed, switch to front-wind route-following function and check the boat has speed in the right direction, if not go back to normal route-following function

To conclude, this project is a true challenge within the extent that never a non-commanded sailing boat has been successful in crossing the ocean. Our boat is working well and is able to follow a route, even if we are still far from an ocean-crossing. Indeed we do not know how the boat will behave in real sea conditions. Let's hope that other students will take the challenge after us and lead all our work to a success.

Remerciements

à :

Bruno AIZIER
Julien LE GRANVALET
Luc JAULIN
Ivon GALLOU
Le personnel de l'atelier de l'ENSTA Bretagne
Le personnel de « Voiles Océan »

Bibliographie

Différents sites internet:

Sur les méthodes de calcul autour des coordonnées GPS:
http://www.movable-type.co.uk/scripts/latlong.html

Sur les cartes Arduino et la programmation:
http://www.arduino.cc/

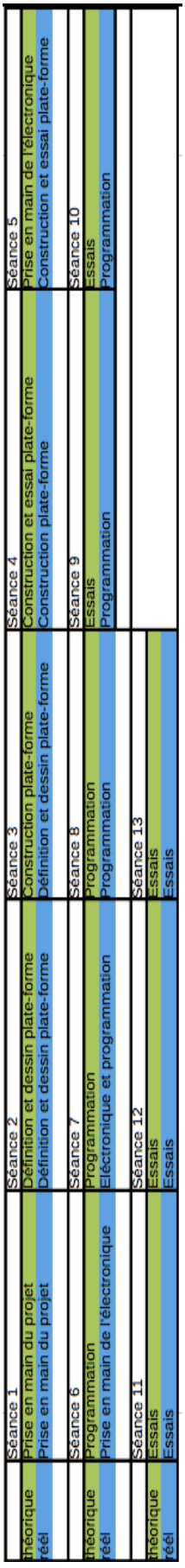
Les plans d'un optimist:
http://www.bateaubois.com:8080/articles.php?lng=fr&pg=12

Pour le choix des cartes électroniques périphériques de l'Arduino:
www.lextronic.fr

Annexes

Annexe 1 : Emploi du temps
Annexe 2 : Répartition des tâches
Annexe 3 : Détails mécaniques
Annexe 4 : Programme

Annexe 1 : Emploi du temps

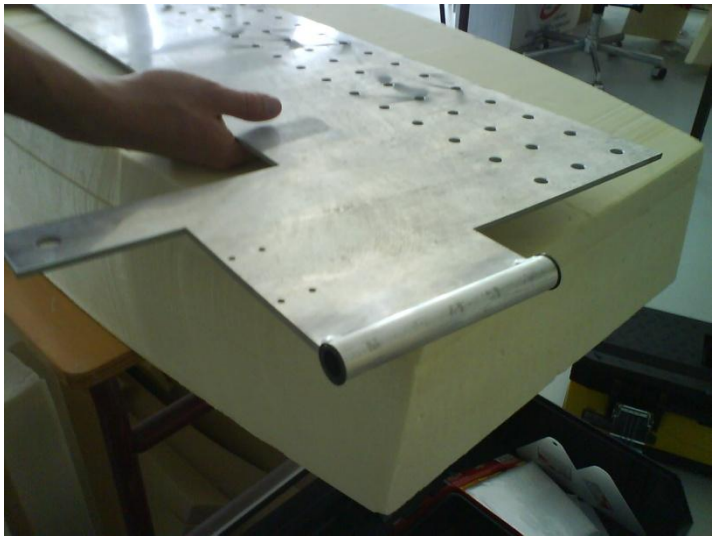


	Séance 1	Séance 2	Séance 3	Séance 4	Séance 5
théorique	Prise en main du projet	Définition et dessin plate-forme	Construction plate-forme	Construction et essai plate-forme	Prise en main de l'électronique
réél	Prise en main du projet	Définition et dessin plate-forme	Définition et dessin plate-forme	Construction plate-forme	Construction et essai plate-forme
	Séance 6	Séance 7	Séance 8	Séance 9	Séance 10
théorique	Programmation	Programmation	Programmation	Essais	Essais
réél	Prise en main de l'électronique	Elécronique et programmation	Programmation	Programmation	Programmation
	Séance 11	Séance 12	Séance 13		
théorique	Essais	Essais	Essais		
réél	Essais	Essais	Essais		

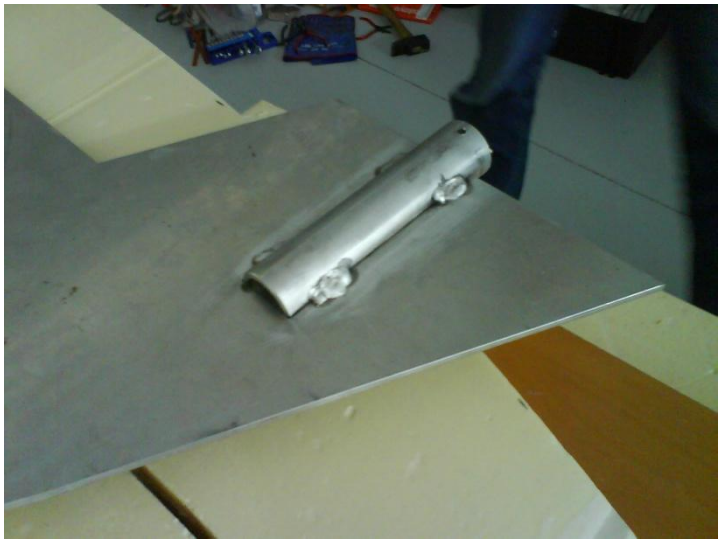
Annexe 2 : Répartition des tâches

Partie mécanique	Définition de l'ensemble mécanique Usinage de la mousse Usinage des différentes pièces en aluminium Finitions et montage Fabrication de la voile Essais	Rémi COQUELIN, François DE MEGILLE Rémi COQUELIN, Julien LEGRANVALET atelier de l'ENSTA BRETAGNE Rémi COQUELIN, François DE MEGILLE Bruno AISIER, François DE MEGILLE, Voiles Ocean (Brest) Rémi COQUELIN, François DE MEGILLE
Partie électronique	Choix et commande de l'électronique Conception du module de prise de contrôle à distance Assemblage des éléments électroniques	Rémi COQUELIN, François DE MEGILLE François DE MEGILLE Rémi COQUELIN
Programmation	Ecriture des fonctions de base Fonctions de suivi de route er remontée au vent Fonctions de lissage et d'initialisation Essais	Rémi COQUELIN, François DE MEGILLE Rémi COQUELIN François DE MEGILLE Rémi COQUELIN, François DE MEGILLE
Rapport	Ecriture du rapport	Rémi COQUELIN, François DE MEGILLE

Annexe 3 : Détails mécaniques

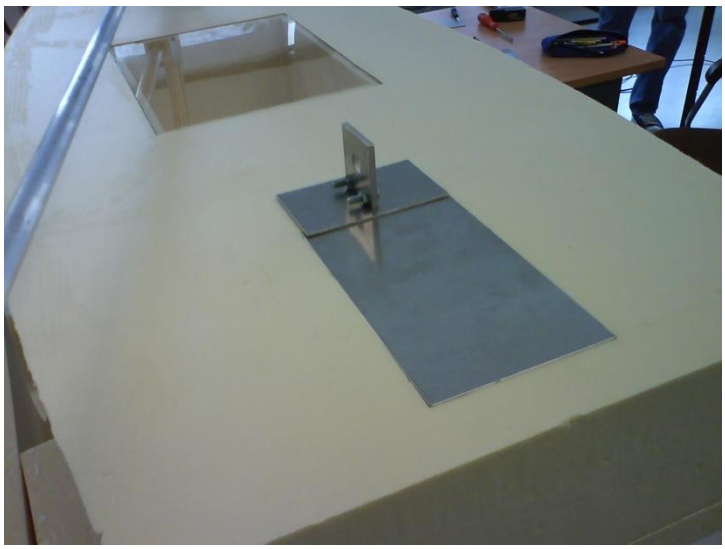


Liaison pivot entre la tôle principale et le safran

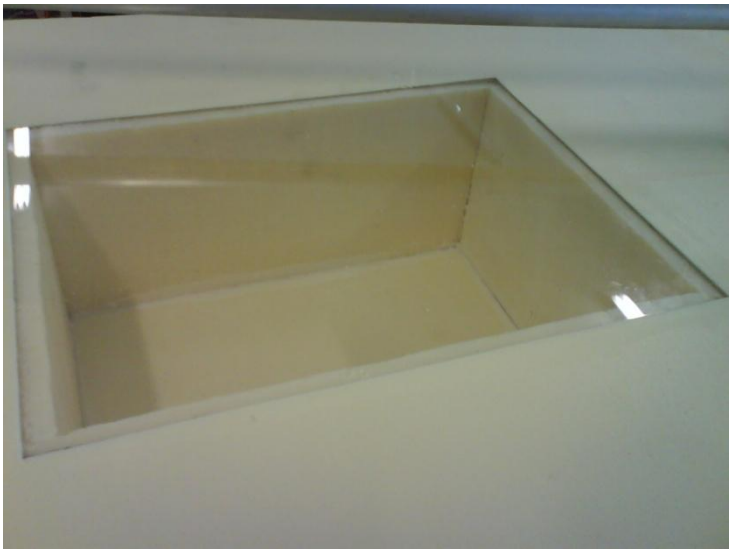


Support du mât soudé sur la tôle principale

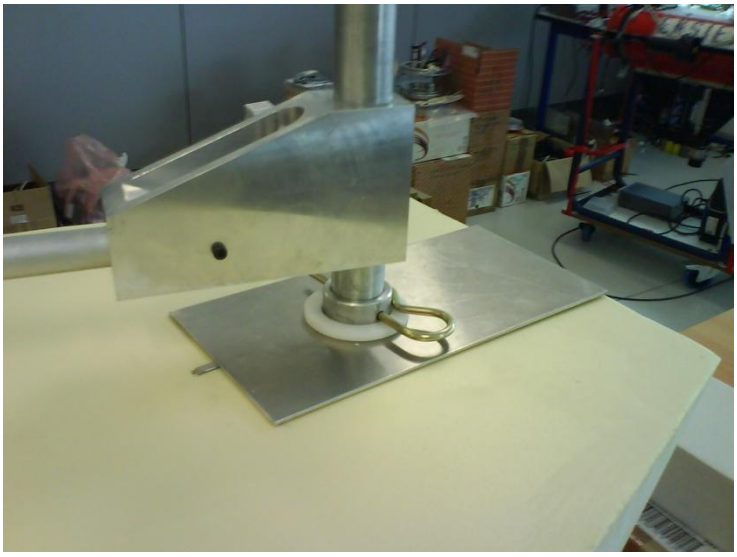
Le safran en place sur le bateau



Le compartiment contenant le servomoteur et l'attache pour l'écoute



Le compartiment qui reçoit la boite étanche contenant l'électronique



Les liaisons bôme-mât et mât-coque

Annexe 4 : Programme

Voici à titre indicatif le programme complet permettant au voilier de fonctionner:

```
#include <Servo.h>
#include <Wire.h>
#include "SDuFAT.h"

//adresses
int HMC6352Address = 0x42;
int slaveAddress;
#define GPM 0x68 // GPM I2C Address                    // Global variable
#define MEM_PW 8 //patte d'alimentation SD

//constantes de réglage
int neutreservo=83;
int courseservo=45;
int nbrectrlcap=10;
int tempslissage=100;
int coefflargroute=5;
long largroutemin=20;
long largroutemax=50000;
float pi=3.141592654;
long rmoyterre=6371030;
float VminGPS=1;
float coeffproportionnel=40;
float coeffecart=8;
float dt=1;
float detangleaupres=0.7; //équivalent à un angle de 45°
int reactivite=20;

//Variables globales
byte Address;                    // Global variable
float Data;
char chaine[10];                 // array to hold the data for the ftoa function
int posservo=83;
int WPlatdegres[10];
float WPlatminutes[10];
char WPlatletter[10];
int WPlongdegres[10];
float WPlongminutes[10];
char WPlongletter[10];
long WPvalidate[10];
float targetWPlat;
float targetWPlong;
float previousWPlat;
float previousWPlong;
int targetWP=1;
int selectcap=-1;
float capvent[2];
boolean cgtborbdone=false;
boolean verifstrat=false;
```

```
boolean suiviroute=true;
long previousdist;
long cmptcgt=0;

//Objets
Servo servo;  // create servo object to control a servo

void setup()
{
  servo.attach(2);  // attaches the servo on pin 2 to the servo object
  //alimentation boussole:
  pinMode(16, OUTPUT);
  digitalWrite(16, LOW);
  pinMode(17, OUTPUT);
  digitalWrite(17, HIGH);
  slaveAddress = HMC6352Address >> 1;   // This results in 0x21 as the address to pass to TWI
  Wire.begin();
  // on my MicroSD Module the power comes from a digital pin
  // I activate it at all times
  pinMode(MEM_PW, OUTPUT);
  digitalWrite(MEM_PW, HIGH);

  Serial.begin(9600);

   //Liste des waypoints:
  //stade
  //WP1:
//  WPlatdegres[0]=48;
//  WPlatminutes[0]=25.135;
//  WPlatletter[0]='N';
//  WPlongdegres[0]=4;
//  WPlongminutes[0]=28.441;
//  WPlongletter[0]='W';
//  WPvalidate[0]=15;
//  //WP2:
//  WPlatdegres[1]=48;
//  WPlatminutes[1]=25.082;
//  WPlatletter[1]='N';
//  WPlongdegres[1]=4;
//  WPlongminutes[1]=28.432;
//  WPlongletter[1]='W';
//  WPvalidate[1]=15;
//  //WP3:
//  WPlatdegres[2]=48;
//  WPlatminutes[2]=25.135;
//  WPlatletter[2]='N';
//  WPlongdegres[2]=4;
//  WPlongminutes[2]=28.441;
//  WPlongletter[2]='W';
//  WPvalidate[2]=15;

//Ty-Colo:
```

```
// WPlatdegres[0]=48;
//  WPlatminutes[0]=25.873;
//  WPlatletter[0]='N';
//  WPlongdegres[0]=4;
//  WPlongminutes[0]=36.966;
//  WPlongletter[0]='W';
//  WPvalidate[0]=15;
//  //WP2:
//  WPlatdegres[1]=48;
//  WPlatminutes[1]=25.880;
//  WPlatletter[1]='N';
//  WPlongdegres[1]=4;
//  WPlongminutes[1]=36.762;
//  WPlongletter[1]='W';
//  WPvalidate[1]=15;

//Moulin Blanc
  //WP1:
  WPlatdegres[0]=48;
  WPlatminutes[0]=23.739;
  WPlatletter[0]='N';
  WPlongdegres[0]=4;
  WPlongminutes[0]=25.990;
  WPlongletter[0]='W';
  WPvalidate[0]=40;
  //WP2:
  WPlatdegres[1]=48;
  WPlatminutes[1]=23.818;
  WPlatletter[1]='N';
  WPlongdegres[1]=4;
  WPlongminutes[1]=25.708;
  WPlongletter[1]='W';
  WPvalidate[1]=40;
  //WP6:
  WPlatdegres[2]=48;
  WPlatminutes[2]=23.734;
  WPlatletter[2]='N';
  WPlongdegres[2]=4;
  WPlongminutes[2]=26.054;
  WPlongletter[2]='W';
  WPvalidate[2]=40;
//   //WP3:
//  WPlatdegres[2]=48;
//  WPlatminutes[2]=23.797;
//  WPlatletter[2]='N';
//  WPlongdegres[2]=4;
//  WPlongminutes[2]=25.408;
//  WPlongletter[2]='W';
//  WPvalidate[2]=70;
//  //WP4:
//  WPlatdegres[2]=48;
//  WPlatminutes[2]=23.732;
```

```
//  WPlatletter[2]='N';
//  WPlongdegres[2]=4;
//  WPlongminutes[2]=25.423;
//  WPlongletter[2]='W';
//  WPvalidate[2]=70;
//  //WP5:
//  WPlatdegres[2]=48;
//  WPlatminutes[2]=23.738;
//  WPlatletter[2]='N';
//  WPlongdegres[2]=4;
//  WPlongminutes[2]=25.767;
//  WPlongletter[2]='W';
//  WPvalidate[2]=40;
}

int GetDouble(){                                // Get double register value from GPM

  int Value = 0;
  byte H_Byte = 0;
  byte L_Byte = 0;

  Wire.beginTransmission(GPM);
  Wire.send(Address);                           // Send register start address
  Wire.endTransmission();

  Wire.requestFrom(GPM, 2);                             // Request double register
  while(Wire.available() < 2);                  // Wait for double byte
  H_Byte = Wire.receive();                    // Store one
  L_Byte = Wire.receive();                   // Store two

  Value = (H_Byte * 10) + L_Byte;               // Adjust for one byte

  return(Value);
}


int GetSingle(){                                // Get single register value from GPM

  int Value = 0;

  Wire.beginTransmission(GPM);
  Wire.send(Address);                           // Send register start address
  Wire.endTransmission();

  Wire.requestFrom(GPM, 1);                             // Request register
  while(Wire.available() < 1);                  // Wait for single byte
  Value = Wire.receive();                     // and store

  return(Value);
}

void GPSHeading(float* cap){ //renvoie le cos et le sin du cap
```

```
  float heading;
  Address = 44;                         // Point to Heading hundreds and tens
  Data = GetDouble();                     // Read registers from GPM
  heading=10*Data;
  Address = 46;                         // Point to Heading units
  Data = GetSingle();                     // Read registers from GPM
  heading=heading+Data;
  Address = 47;                         // Point to Heading tenths
  Data = GetSingle();
  heading=heading+(Data/10);
//  Serial.print("cap GPS ");
//  Serial.println(heading);
  //  SD.print("log.txt","\nCap GPS: ");
    //SD.println("log.txt", ftoa(chaine, heading, 2));
  heading = radians(heading);
  cap[0]= cos(heading);
  cap[1]= sin(heading);
}

float GPSSpeed(){ //renvoie la vitesse sur un flottant
  float vitesse;
  Address = 52;                         // Point to Speed hundreds and tens
  Data = GetDouble();                     // Read registers from GPM
  vitesse=Data*10;
  Address = 54;                         // Point to Speed units
  Data = GetSingle();                     // Read registers from GPM
  vitesse=vitesse+Data;
  Address = 55;                         // Point to Speed units
  Data = GetSingle();                     // Read registers from GPM
  vitesse=vitesse+(Data/10);
  return vitesse;
}

float GPSLatitude(){ //renvoie la latitude en radians dans notre système (-pi/2 ; pi/2)
  float latitude;
  float minutes;
  char letter;
  Address = 14;                         // Point to Latitude tens
  Data = GetSingle();                     // Read registers from GPM
  latitude= Data*10;
  Address = 15;                         // Point to Latitude units
  Data = GetSingle();                     // Read registers from GPM
  latitude= latitude+Data;
  Address = 16;                         // Point to Latitude minutes tens
  Data = GetSingle();                     // Read registers from GPM
  minutes= Data*10;
  Address = 17;                         // Point to Latitude minutes units
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data;
  Address = 18;                         // Point to Latitude minutes tenths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/10;
```

```
  Address = 19;                         // Point to Latitude minutes hundredths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/100;
  Address = 20;                         // Point to Latitude minutes thousandths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/1000;
  Address = 21;                         // Point to Latitude minutes ten thousandths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/10000;
  latitude=latitude+(minutes/60);
  Address = 22;                         // Point to Latitude character register
  Data = GetSingle();                     // Read registers from GPM
  letter=Data;
  if (letter=='S'){             // Prise en conpte de la lettre et passage dans notre système
    latitude=-latitude;
  }
  return radians(latitude);
}

float GPSLongitude(){  //renvoie la longitude en radians dans notre système (-pi ; pi)
  float longitude;
  float minutes;
  char letter;
  Address = 23;                         // Point to Longitude hundreds
  Data = GetSingle();                     // Read registers from GPM
  longitude= Data*100;
  Address = 24;                         // Point to Longitude tens
  Data = GetSingle();                     // Read registers from GPM
  longitude= longitude+Data*10;
  Address = 25;                         // Point to Longitude units
  Data = GetSingle();                     // Read registers from GPM
  longitude= longitude+Data;
  Address = 26;                         // Point to Longitude minutes tens
  Data = GetSingle();                     // Read registers from GPM
  minutes= Data*10;
  Address = 27;                         // Point to Longitude minutes units
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data;
  Address = 28;                         // Point to Longitude minutes tenths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/10;
  Address = 29;                         // Point to Longitude minutes hundredths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/100;
  Address = 30;                         // Point to Longitude minutes thousandths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/1000;
  Address = 31;                         // Point to Longitude minutes ten thousandths
  Data = GetSingle();                     // Read registers from GPM
  minutes= minutes+Data/10000;
  longitude=longitude+(minutes/60);
  Address = 32;                         // Point to Longitude character register
```

```
  Data = GetSingle();                          // Read registers from GPM
  letter=Data;
  if (letter=='E'){                            // Prise en conpte de la lettre et passage dans notre système
    longitude=-longitude;
  }
  return radians(longitude);
}


void Boussole(float* heading)
{
  byte headingData[2];
  int i;
  float headingValue;
  Wire.beginTransmission(slaveAddress);
  Wire.send("A");              // The "Get Data" command
  Wire.endTransmission();
  delay(10);                   // The HMC6352 needs at least a 70us (microsecond) delay
  // after this command.  Using 10ms just makes it safe
  // Read the 2 heading bytes, MSB first
  // The resulting 16bit word is the compass heading in 10th's of a degree
  // For example: a heading of 1345 would be 134.5 degrees
  Wire.requestFrom(slaveAddress, 2);        // Request the 2 byte heading (MSB comes first)
  i = 0;
  while(Wire.available() && i < 2)
  {
    headingData[i] = Wire.receive();
    i++;
  }
  headingValue = headingData[0]*256 + headingData[1];  // Put the MSB and LSB together
  headingValue = headingValue/ 10; //cap sur 360°
  headingValue=360-headingValue;
//  Serial.print("cap boussole ");
//  Serial.println(headingValue);
  headingValue = radians(headingValue);
  heading[0]= cos(headingValue);
  heading[1]= sin(headingValue);
//Serial.println(heading[0]);
//Serial.println(heading[1]);
}


void BoussoleLissee(float* caplisse){
  float cap[2];
  float sinus[20];   //nbre pts lissage
  float cosinus[20];  //nbre pts lissage
  int nbrepointslissage=20; //nbre pts lissage DOIT ETRE EGAL AUX VALEURS DANS LES
TABLEAUX
  float sommecos=0;
  float sommesin=0;

  for (int i=0; i<nbrepointslissage; i++){//boucle pendant laquelle on prélève un cap tous les
```

```
tempslissage; on prélève nbrepointslissage valeurs.
    Boussole(cap);
    cosinus[i]=cap[0];
    sinus[i]=cap[1];
    delay(tempslissage);
   }
  for (int i=0; i<nbrepointslissage; i++){//somme de tous les caps prélevés ci-dessus
    sommecos=sommecos+cosinus[i];
    sommesin=sommesin+sinus[i];
   }
  caplisse[0] = sommecos/nbrepointslissage;
  caplisse[1] = sommesin/nbrepointslissage;
//Serial.println(caplisse[0]);
//Serial.println(caplisse[1]);
//    SD.print("log.txt","\nCap boussole lissee cos: ");
//    SD.println("log.txt", ftoa(chaine, caplisse[0], 2));
//        SD.print("log.txt","\nsin: ");
//    SD.println("log.txt", ftoa(chaine, caplisse[1], 2));
 }


char *ftoa(char *a, double f, int precision) {
  long p[] = {0,10,100,1000,10000,100000,1000000,10000000,100000000};

  char *ret = a;
  long heiltal = (long)f;
  itoa(heiltal, a, 10);
  while (*a != '\0') a++;
  *a++ = '.';
  long desimal = abs((long)((f - heiltal) * p[precision]));
  itoa(desimal, a, 10);
  return ret;
}

long dist(float latitudedep, float longitudedep, float latitudear, float longitudear){ // Retourne la
distance en mètres entre 2 points GPS avec une erreur maximale de 0.2%
  float latitude;
  float longitude;
  float a;
  float c;
  float d;
  float e;
  long distance;
  latitude= (abs(latitudedep-latitudear));
  longitude= min(min(abs(longitudedep-longitudear), abs((2*pi+longitudedep)-longitudear)),
abs(longitudedep-(longitudear+2*pi)));
  a = (square(sin(latitude/2.0))) + cos(latitudedep)*cos(latitudear)*(square(sin(longitude/2.0)));
  d=sqrt(a);
  e=sqrt(1-a);
  c = 2*atan2(d, e);
  distance= rmoyterre*c;
  return distance;
```

```
}

void GPSHeadingTh(float latitudedep, float longitudedep, float latitudear, float longitudear, float*
cap){  // Retourne le cos le sin du cap entre 2 points GPS
  float heading;
  float a;
  float b;
  float c;
  float d;
  float longitude;
  longitude= min(min(abs(longitudedep-longitudear), abs((2*pi+longitudedep)-longitudear)),
abs(longitudedep-(longitudear+2*pi)));
  a=sin(longitude)*cos(latitudear);
  c=cos(latitudedep)*sin(latitudear);
  d=sin(latitudedep)*cos(latitudear)*cos(longitude);
  b=c-d;
  heading=atan2(a,b);
  if (sin(longitudear-longitudedep)>0){                  //passage de [-180°; 180°] à [0°; 360°] pour
rejoindre la convention des caps d'un compas.
   heading=2*pi-abs(heading);
  }
  cap[0]=cos(heading);
  cap[1]=sin(heading);
}


float ConversionLatitude(int degres, float minutes, char letter){
  float latitude;
  latitude= degres+ minutes/60;
  if (letter=='S'){                                  // Prise en conpte de la lettre et passage dans notre système
    latitude=-latitude;
  }
  return radians(latitude);
}

float ConversionLongitude(int degres, float minutes, char letter){
  float longitude;
  longitude= degres+ minutes/60;
  if (letter=='E'){                                 // Prise en conpte de la lettre et passage dans notre système
    longitude=-longitude;
  }
  return radians(longitude);
}

void servocmd(int angle){                    //verifie que l'on ne depasse pas les butées de commande du
servo
  if (angle<=neutreservo+courseservo && angle>=neutreservo-courseservo){
    servo.write(neutreservo-(angle-neutreservo));
  }
  else{
    if(angle>neutreservo+courseservo){
      servo.write(neutreservo-courseservo);
```

```
    }
    if(angle<neutreservo-courseservo){
      servo.write(neutreservo+courseservo);
    }
  }
   Serial.print("Angle servo:");
   Serial.println(angle);
     //  SD.print("log.txt","\nangle servo: ");
//   SD.println("log.txt", ftoa(chaine, angle, 2));

}

float ecartroute(float latitudedep, float longitudedep, float latitudear, float longitudear){ //retourne
l'écart à la route définie par les 2 points GPS en m sur un long
float headingroute[2];
GPSHeadingTh( latitudedep, longitudedep, latitudear, longitudear, headingroute);
float proj[2];
proj[0]=-headingroute[1];
proj[1]=headingroute[0];
float headingtocurrentpoint[2];
GPSHeadingTh(latitudedep, longitudedep, GPSLatitude(), GPSLongitude(), headingtocurrentpoint);
float distancedone=dist(latitudedep, longitudedep, GPSLatitude(), GPSLongitude());        //distance
parcourue
float ecart;
ecart= distancedone*(proj[0]*headingtocurrentpoint[0]+proj[1]*headingtocurrentpoint[1]);
   Serial.print("Ecart: ");
   Serial.println(ecart);
   //    SD.print("log.txt","\nEcart: ");
   // SD.println("log.txt", ftoa(chaine, ecart, 2));
return ecart;
}


boolean SuiviRoute(float latitudedep, float longitudedep, float latitudear, float longitudear){
  Serial.println("Suivi route");
     // SD.print("log.txt","\nSuivi route ");
  //calcul des variables nécessaires pour la fonction:
  long longroute= dist(latitudedep, longitudedep, latitudear, longitudear);
  long largroute= longroute/coefflargroute;
  float ecart;
  float theoricalheading[2];
  GPSHeadingTh( GPSLatitude(), GPSLongitude(), latitudear, longitudear, theoricalheading);
  float currentheading[2];
  float capboussole[2];
  float erreur;
 //Calcul de la largeur de la route
 if (largroute>largroutemax){
   largroute=largroutemax;
 }
  if (largroute<largroutemin){
   largroute=largroutemin;
 }
```

```
 //Vérification que le bateau est bien présent à l'intérieur du couloir de la route définie, calcul de
l'écart à la route:
 ecart=ecartroute(latitudedep, longitudedep, latitudear, longitudear);  //ecart à la route en mètres
 if(abs(ecart)<largroute/2){       //bateau présent dans le couloir
   ecart=ecart/largroute;          //ecart relatif prenant en compte la largeur de la route, varie de -1 à 1
sur la route.
   if (GPSSpeed()<=VminGPS){ // asservir sur cap boussole.
     BoussoleLissee(capboussole);
     if (capboussole[0]*theoricalheading[0]+capboussole[1]*theoricalheading[1]>0){
       erreur=capboussole[0]*theoricalheading[1]-capboussole[1]*theoricalheading[0];
//determinant donnant une image de l'erreur de cap, varie de -1 à 1 dans le sens de marche
       servocmd(neutreservo+(-coeffproportionnel*erreur+coeffecart*ecart));   //commande du servo
double propotionnelle et intégrale
       Serial.println("asservi sur boussole");
     // SD.print("log.txt","\nasservi sur boussole ");
     }
     else{                         //si on se trouve à l'envers du sens de marche
       if(capboussole[0]*theoricalheading[1]-capboussole[1]*theoricalheading[0]>0){
         servocmd(neutreservo-courseservo);       //virer sur tribord
         Serial.println("virer sur tribord boussole");
            //  SD.print("log.txt","\nvirer sur tribord boussole ");
       }
       else{
         servocmd(neutreservo+courseservo);       //virer sur babord
         Serial.println("virer sur babord boussole");
           //   SD.print("log.txt","\nvirer sur babord boussole ");
       }
     }
   }
   else{ // asservir sur cap GPS
     GPSHeading(currentheading);
     if (currentheading[0]*theoricalheading[0]+currentheading[1]*theoricalheading[1]>0){
       erreur=currentheading[0]*theoricalheading[1]-
currentheading[1]*theoricalheading[0];//determinant donnant une image de l'erreur de cap, varie de
-1 à 1 dans le sens de marche
       servocmd(neutreservo+(-coeffproportionnel*erreur+coeffecart*ecart));   //commande du servo
double propotionnelle et intégrale
       Serial.println("asservi sur gps");
          //  SD.print("log.txt","\nasservi sur gps ");
     }
     else{                   //si on se trouve à l'envers du sens de marche
       if(currentheading[0]*theoricalheading[1]-currentheading[1]*theoricalheading[0]>0){
         servocmd(neutreservo-courseservo);       //virer sur tribord
         Serial.println("virer sur tribord gps");
            // SD.print("log.txt","\nvirer sur tribord gps ");
       }
       else{
         servocmd(neutreservo+courseservo);       //virer sur babord
         Serial.println("virer sur babord gps");
          //     SD.print("log.txt","\virer sur babord gps ");
       }
     }
```

```
   }
  return true;
 }
 Serial.println("Impossible");
     // SD.print("log.txt","\nImpossible ");
 return false;
}

boolean remontervent(float latitudedep, float longitudedep, float latitudear, float longitudear){
  Serial.println("Remonter vent");
   //SD.print("log.txt","\nRemonter vent ");
  //calcul des variables nécessaires pour la fonction:
  long longroute= dist(latitudedep, longitudedep, latitudear, longitudear);
  long largroute= longroute/coefflargroute;
  float ecart;
  float currentheading[2];
  float capboussole[2];
  float erreur;
 //Calcul de la largeur de la route
 if (largroute>largroutemax){
   largroute=largroutemax;
 }
  if (largroute<largroutemin){
   largroute=largroutemin;
 }
 //Vérification que le bateau est bien présent à l'intérieur du couloir de la route définie, calcul de
l'écart à la route:
 ecart=ecartroute(latitudedep, longitudedep, latitudear, longitudear);  //ecart à la route en mètres
 if(abs(ecart)<largroute/2){       //bateau présent dans le couloir
   if (abs(ecart)<largroute/4){
     cgtborbdone=false;
   }
   if (abs(ecart)<largroute/3 || cgtborbdone==true){
     if (GPSSpeed()<=VminGPS){ // asservir sur cap boussole.
       BoussoleLissee(capboussole);
       if (capboussole[0]*capvent[0]+capboussole[1]*capvent[1]>0){
         erreur=-selectcap*detangleaupres+(capboussole[0]*capvent[1]+capboussole[1]*capvent[0]);
//determinant donnant une image de l'erreur de cap, varie de -1 à 1 dans le sens de marche
         Serial.print("erreur: ");
         Serial.println(erreur);
          //SD.print("log.txt","\nerreur:  ");
            //  SD.println("log.txt", ftoa(chaine, erreur, 2));
         servocmd(neutreservo+(coeffproportionnel*erreur));   //commande du servo double
propotionnelle et intégrale
         Serial.println("asservi sur boussole");
         // SD.print("log.txt","\nasservi sur boussole ");
       }
       else{                        //si on se trouve à l'envers du sens de marche
         if(selectcap<0){
           servocmd(neutreservo-courseservo);       //virer sur tribord
           Serial.println("virer sur tribord boussole");
           // SD.print("log.txt","\nvirer sur tribord boussole ");
```

```
        }
        else{
          servocmd(neutreservo+courseservo);       //virer sur babord
          Serial.println("virer sur babord boussole");
           //SD.print("log.txt","\nvirer sur babord boussole ");
        }
       }
     }
     else{ // asservir sur cap GPS
       GPSHeading(currentheading);
       if (currentheading[0]*capvent[0]+currentheading[1]*capvent[1]>0){
         erreur=-
selectcap*detangleaupres+(currentheading[0]*capvent[1]+currentheading[1]*capvent[0]);
//determinant donnant une image de l'erreur de cap, varie de -1 à 1 dans le sens de marche
         servocmd(neutreservo+(coeffproportionnel*erreur));
         Serial.println("asservi sur gps");
          //SD.print("log.txt","\nasservi sur gps ");
       }
       else{                  //si on se trouve à l'envers du sens de marche
         if(selectcap<0){
           servocmd(neutreservo-courseservo);       //virer sur tribord
           Serial.println("virer sur tribord gps");
            //SD.print("log.txt","\nvirer sur tribord gps ");
         }
         else{
           servocmd(neutreservo+courseservo);       //virer sur babord
           Serial.println("virer sur babord gps");
           //SD.print("log.txt","\nvirer sur babord gps ");
         }
       }
     }
   }
   else{
     cgtborbdone=true;
     Boussole(capboussole);
     Serial.println("Changement de bord pour remonter au vent");
     //SD.print("log.txt","\nChangement de bord pour remonter au vent ");
     if (selectcap<0){
       selectcap=1;
       Serial.println("virer sur tribord");
       // SD.print("log.txt","\nvirer sur tribord ");
       servocmd(neutreservo-courseservo);       //virer sur tribord
       for (int i=0; i<100; i++){
         int timing=100;
         if(capboussole[0]*capvent[1]+capboussole[1]*capvent[0]>detangleaupres){
           timing=0;
         }
         delay(timing);
       }
     }
     else{
       selectcap=-1;
```

```
      Serial.println("virer sur babord");
      // SD.print("log.txt","\nvirer sur babord ");
      servocmd(neutreservo+courseservo);       //virer sur babord
      for (int i=0; i<100; i++){
        int timing=100;
        if(capboussole[0]*capvent[1]+capboussole[1]*capvent[0]<-detangleaupres){
          timing=0;
        }
        delay(timing);
      }
    }
    servocmd(neutreservo);  //remise en ligne droite
  }
 return true;
}
Serial.println("Impossible");
 //SD.print("log.txt","\nImpossible ");
return false;
}



void cgtpreviousWP(){
 Serial.print("Waypoint precedent devient la position actuelle");
 int latitude;
 float minutes;
 float longitude;
 Address = 14;                    // Point to Latitude tens
 Data = GetSingle();                 // Read registers from GPM
 latitude= Data*10;
 Address = 15;                    // Point to Latitude units
 Data = GetSingle();                 // Read registers from GPM
 latitude= latitude+Data;
 Address = 16;                    // Point to Latitude minutes tens
 Data = GetSingle();                 // Read registers from GPM
 minutes= Data*10;
 Address = 17;                    // Point to Latitude minutes units
 Data = GetSingle();                 // Read registers from GPM
 minutes= minutes+Data;
 Address = 18;                    // Point to Latitude minutes tenths
 Data = GetSingle();                 // Read registers from GPM
 minutes= minutes+Data/10;
 Address = 19;                    // Point to Latitude minutes hundredths
 Data = GetSingle();                 // Read registers from GPM
 minutes= minutes+Data/100;
 Address = 20;                    // Point to Latitude minutes thousandths
 Data = GetSingle();                 // Read registers from GPM
 minutes= minutes+Data/1000;
 Address = 21;                    // Point to Latitude minutes ten thousandths
 Data = GetSingle();                 // Read registers from GPM
 minutes= minutes+Data/10000;
 Address = 22;                    // Point to Latitude character register
```

```
  Data = GetSingle();                    // Read registers from GPM
  WPlatletter[targetWP-1]=Data;
  WPlatdegres[targetWP-1]=latitude;
  WPlatminutes[targetWP-1]=minutes;

  Address = 23;                          // Point to Longitude hundreds
  Data = GetSingle();                    // Read registers from GPM
  longitude= Data*100;
  Address = 24;                          // Point to Longitude tens
  Data = GetSingle();                    // Read registers from GPM
  longitude= longitude+Data*10;
  Address = 25;                          // Point to Longitude units
  Data = GetSingle();                    // Read registers from GPM
  longitude= longitude+Data;
  Address = 26;                          // Point to Longitude minutes tens
  Data = GetSingle();                    // Read registers from GPM
  minutes= Data*10;
  Address = 27;                          // Point to Longitude minutes units
  Data = GetSingle();                    // Read registers from GPM
  minutes= minutes+Data;
  Address = 28;                          // Point to Longitude minutes tenths
  Data = GetSingle();                    // Read registers from GPM
  minutes= minutes+Data/10;
  Address = 29;                          // Point to Longitude minutes hundredths
  Data = GetSingle();                    // Read registers from GPM
  minutes= minutes+Data/100;
  Address = 30;                          // Point to Longitude minutes thousandths
  Data = GetSingle();                    // Read registers from GPM
  minutes= minutes+Data/1000;
  Address = 31;                          // Point to Longitude minutes ten thousandths
  Data = GetSingle();                    // Read registers from GPM
  minutes= minutes+Data/10000;
  Address = 32;                          // Point to Longitude character register
  Data = GetSingle();                    // Read registers from GPM
  WPlongletter[targetWP-1]=Data;
  WPlongdegres[targetWP-1]=longitude;
  WPlongminutes[targetWP-1]=minutes;
}

void navigation(){
  previousWPlat=ConversionLatitude(WPlatdegres[targetWP-1], WPlatminutes[targetWP-1],
WPlatletter[targetWP-1]);
  previousWPlong=ConversionLongitude(WPlongdegres[targetWP-1], WPlongminutes[targetWP-
1], WPlongletter[targetWP-1]);
  targetWPlat=ConversionLatitude(WPlatdegres[targetWP], WPlatminutes[targetWP],
WPlatletter[targetWP]);
  targetWPlong=ConversionLongitude(WPlongdegres[targetWP], WPlongminutes[targetWP],
WPlongletter[targetWP]);
  if (dist(GPSLatitude(), GPSLongitude(), targetWPlat,
targetWPlong)<WPvalidate[targetWP]){    //distance de validation waypoint atteinte
    Serial.print("Changement de waypoint");
    targetWP++;
```

```
   previousWPlat=ConversionLatitude(WPlatdegres[targetWP-1], WPlatminutes[targetWP-1],
WPlatletter[targetWP-1]);
   previousWPlong=ConversionLongitude(WPlongdegres[targetWP-1], WPlongminutes[targetWP-
1], WPlongletter[targetWP-1]);
   targetWPlat=ConversionLatitude(WPlatdegres[targetWP], WPlatminutes[targetWP],
WPlatletter[targetWP]);
   targetWPlong=ConversionLongitude(WPlongdegres[targetWP], WPlongminutes[targetWP],
WPlongletter[targetWP]);
  }
}

void strategie(){
  if (verifstrat==false){
    cgtpreviousWP();
    verifstrat=true;
    suiviroute=true;
  }
  navigation();
  GPSHeadingTh(previousWPlat, previousWPlong, targetWPlat, targetWPlong, capvent);
  previousdist=dist(GPSLatitude(), GPSLongitude(), targetWPlat, targetWPlong);
  if (suiviroute==true){
    verifstrat=SuiviRoute(previousWPlat, previousWPlong, targetWPlat, targetWPlong);
  }
  else{
    verifstrat=remontervent(previousWPlat, previousWPlong, targetWPlat, targetWPlong);
  }
  delay(1000);
  cmptcgt++;
  if (previousdist<dist(GPSLatitude(), GPSLongitude(), targetWPlat, targetWPlong) &&
cmptcgt>reactivite){
    cmptcgt=0;
    if(suiviroute==true){
      suiviroute=false;
    }
    else{
      suiviroute=true;
    }
  }
}

void loop(){
//  Serial.print("Speed: ");                         //test vitesse GPS
//  Serial.println(GPSSpeed());
//  Serials.print("100000*longitude: ");                 // test longitude GPS
//  Serial.println(100000*GPSLongitude());
//  Serial.print("100000*latitude: ");                  // test latitude GPS
//  Serial.println(100000*GPSLatitude());
//  Serial.println(1000*ConversionLongitude(45, 25.2356, 'E')); //Test conversion coordonnées GPS
//  Serial.println(1000*ConversionLatitude(45, 25.2356, 'N'));
//  Serial.println(dist(ConversionLatitude(48, 22.666, 'N'), ConversionLongitude(4, 29.628, 'O'),
ConversionLatitude(48, 18.701, 'N'), ConversionLongitude(4, 29.947, 'O'))); //test distance
// float a[2];
```

```
// GPSHeadingTh(ConversionLatitude(WPlatdegres[0], WPlatminutes[0], WPlatletter[0]),
ConversionLongitude(WPlongdegres[0], WPlongminutes[0], WPlongletter[0]),
ConversionLatitude(WPlatdegres[1], WPlatminutes[1], WPlatletter[1]),  ConversionLongitude(8,
WPlongminutes[1], WPlongletter[1]),a);
//SuiviRoute(ConversionLatitude(WPlatdegres[0], WPlatminutes[0], WPlatletter[0]),
ConversionLongitude(WPlongdegres[0], WPlongminutes[0], WPlongletter[0]),
ConversionLatitude(WPlatdegres[1], WPlatminutes[1], WPlatletter[1]),
ConversionLongitude(WPlongdegres[1], WPlongminutes[1], WPlongletter[1]));
//delay(3000);
//float a[2];
//GPSHeading(a);
//test cgtprevwaypoint
//targetWP=1;
//cgtpreviousWP();
//   Serial.println(10*WPlatdegres[0]);
//   Serial.println(10*WPlatminutes[0]);
//   Serial.println(WPlatletter[0]);
//   Serial.println(10*WPlongdegres[0]);
//   Serial.println(10*WPlongminutes[0]);
//   Serial.println(WPlongletter[0]);
//Remonter au vent

//capvent[0]=-1;
//capvent[1]=0;
//Serial.println("selectcap = ");
//Serial.println(selectcap);
//remontervent(ConversionLatitude(WPlatdegres[0], WPlatminutes[0], WPlatletter[0]),
ConversionLongitude(WPlongdegres[0], WPlongminutes[0], WPlongletter[0]),
ConversionLatitude(WPlatdegres[1], WPlatminutes[1], WPlatletter[1]),
ConversionLongitude(WPlongdegres[1], WPlongminutes[1], WPlongletter[1]));
//targetWP=1;
//navigation();
Serial.println(targetWP);
strategie();
}
```