16/06/09

CHALLENGE *MICROTRANSAT*

RAPPORT DE PROJET INDUSTRIEL

ENSI 2010

Pierre-Henri REILHAC

Table des matières :

Introduction

En plusieurs siècles, la traversée de l'atlantique est devenue courante. Mais elle n'en est pas pour autant plus évidente et constitue encore un véritable défi pour les petites embarcations. Dans le but de faire évoluer les techniques en robotique maritime, le challenge Microtransat a été mis en place. Il a pour objectif une traversée de l'atlantique par un robot voilier. Il combine à la fois les domaines de la robotique et de la navigation à la voile. Le robot voilier doit être autonome et capable d'affronter les nombreux dangers de l'océan.

Ce challenge n'a pas encore été réalisé, ce qui constitue un véritable défi et laisse la porte ouverte à toutes les approches envisageables. Depuis plusieurs années des concours sont organisés dans cet optique. Leur but est de rassembler les équipes qui travaillent sur ce challenge. A travers diverses épreuves, ces concours permettent de confronter les différentes approches technologiques choisies par les concurrents, que ce soit au niveau de l'architecture du bateau, des capteurs, de l'architecture de l'électronique ou de l'intelligence embarquée.

Cette année une équipe Microtransat a vue le jour à l'ENSIETA. Nous souhaitons participer à un de ces concours : le « World Robotic Sailing Championship » qui se déroulera cet été à Porto, au Portugal. C'est dans cet objectif que ce projet industriel a été réalisé.

Nous aborderons ainsi dans un premier temps la mise en place de ce projet. Dans un deuxième temps, nous expliquerons les travaux de recherche réalisés ayant abouti à la mise en place d'un programme adapté. Nous terminerons par la réalisation de l'électronique et des tests.

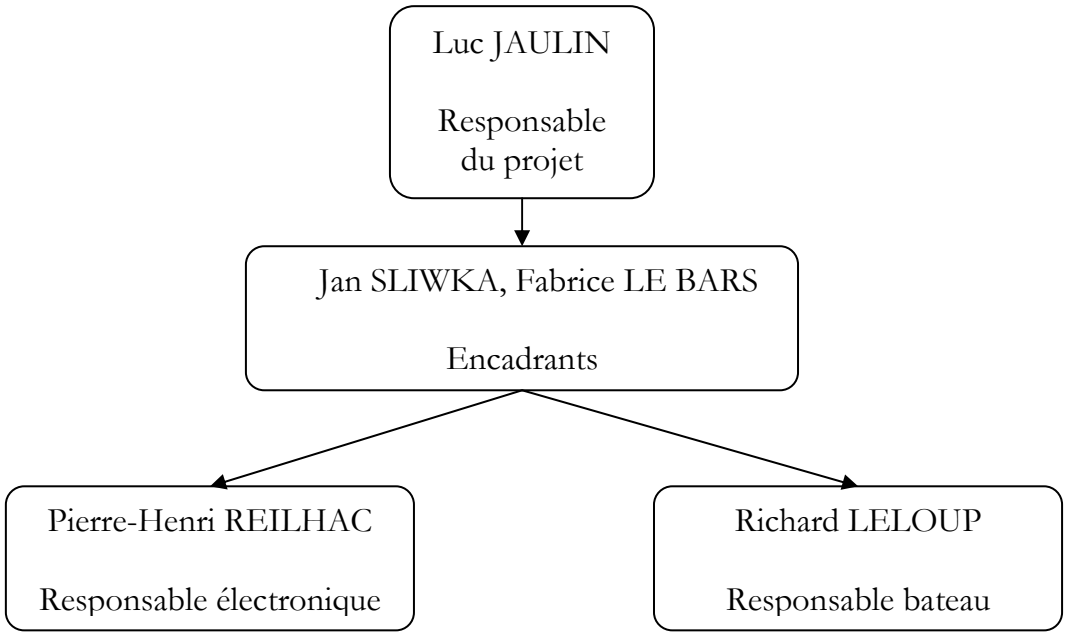
1. Mise en place du projet

1.1. Définition des objectifs et cahier des charges

1.1.1. Mise en place de l'équipe

L'équipe Microtransat est constituée de plusieurs membres, chacun ayant des connaissances différentes. La participation au concours impose de posséder un voilier robot. Le projet étant nouveau, nous n'avions aucune structure existante, ni aucun a priori sur l'architecture de l'électronique. L'objectif général est donc de réaliser un voilier et une électronique adaptée. La tache à réaliser étant relativement vaste et nécessitant beaucoup de travail, de temps et de connaissances spécifiques, nous avons du déterminer dès le début le rôle de chacun. Au vu de mes connaissances et de mes goûts, j'ai été désigné comme responsable de la partie électronique. La construction du bateau a été confiée à Richard LELOUP, en première année, passionné de voile. La supervision des travaux ainsi que les démarches administratives ont été prises en charge par les encadrants.

Organigramme du projet :

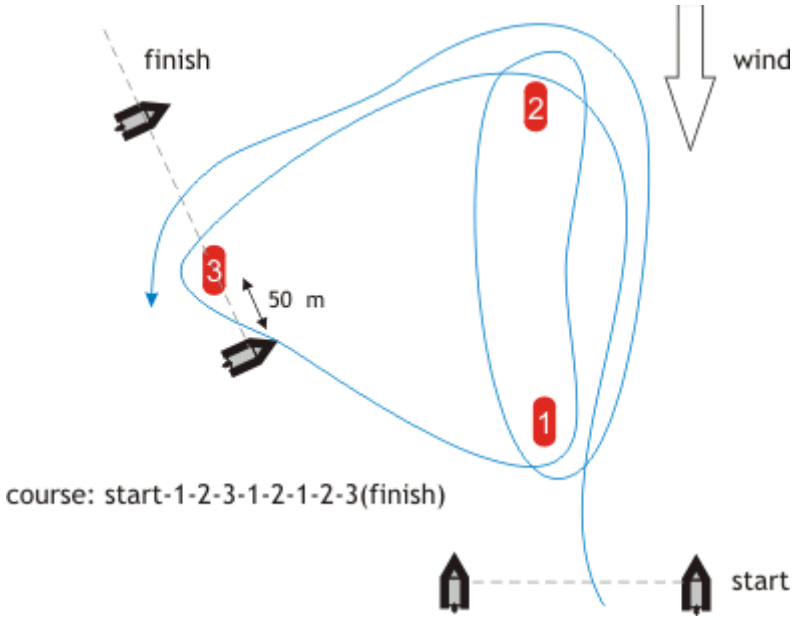


1.1.2. Définition des objectifs

Le projet étant nouveau, tout restait à faire. Nous avons donc décidé d'identifier et de fixer les objectifs à moyen et long termes. L'objectif à long terme était clair dès le départ : mettre en place une équipe Microtransat à l'ENSIETA et participer, quand cela sera possible, à une traversée de l'atlantique par un voilier en autonome. Cela demande beaucoup de travail, de main d'œuvre, de tests et d'expérience, ce qui ne le rendra pas atteignable avant plusieurs années. Nous avons donc décidé de nous fixer des objectifs plus modestes, réalisables à moyen terme. L'objectif est d'acquérir de l'expérience dans le domaine. Nous avons alors choisi de participer au « World Robotic Sailing Championship ». Une fois cette décision prise, nous avons du mettre en place des réunions afin de spécifier les différents éléments à prendre en compte et réaliser un cahier des charges à respecter pour notre projet.

1.1.3. Réalisation du cahier des charges

Une fois les rôles attribués, nous avons organisé plusieurs réunions afin de définir un cahier des charges pour l'électronique et pour le bateau. Pour cela, nous nous sommes basés sur les épreuves du concours. D'après les informations fournies par le site officiel, les épreuves des années précédentes avaient pour objectif de faire suivre un parcours de bouées au bateau :



Cahier des charges du bateau :

- le bateau doit être commandable par un système électronique
- le système de commande du bateau doit être fiable
- le bateau doit être transportable dans une voiture
- l'architecture doit être maîtrisée
- le bateau doit être facilement et rapidement réparable
- un emplacement doit être prévu pour le boîtier étanche de l'électronique
- l'accès à l'électronique et aux branchements doit être facile
- le bateau doit être robuste et résister à une mer agitée et des vents importants
- le bateau doit pouvoir fonctionner en mode dégradé : perte d'un safran, détachement d'un gréement.
- l'emplacement de l'électronique doit être protégé de l'eau
- le bateau, s'il se retourne, de doit pas prendre l'eau
- les systèmes de commandes doivent êtres simples, faciles à utiliser, faciles à changer en cas de problèmes et protégés de l'eau
- Le bateau doit pouvoir navigué à la fois en eau douce et en mer

Pour l'électronique, le débat fut plus long : différentes architectures ont été proposées, chacune ayant ses partisans. Nous avons alors réaliser un cahier des charges et confronté chaque solution à celui-ci.

Cahier des charges de l'électronique :

- le système doit connaître sa position
- le système doit connaître son orientation
- le système doit connaître la direction du vent
- le système doit consommer le moins possible
- le système doit être robuste aux coupures de courant
- le système doit être facilement réparable
- le programme ne doit pas rester bloqué en attente
- le système doit être capable de redémarrer tout seul
- le système doit être solide et résister aux chocs et vibrations
- le système doit être suffisamment petit pour loger dans un boîtier étanche
- le système doit sauvegarder ses positions ou les transmettre en temps réel
- le système doit être facilement reprogrammable
- le système doit pouvoir facilement commander le bateau
- un opérateur doit pouvoir prendre la main sur la commande du bateau en cas de problème (condition imposée par le règlement du concours).

1.2. Recherche des différentes solutions techniques et choix de l'électronique

1.2.1. Différentes solutions envisagées :

Afin de trouver l'architecture la mieux adaptée à notre projet, nous avons organisé des réunions du type « brainstorming ». Au cours de celles-ci, chacun apportait son avis sur les différentes possibilités envisagées. Grâce à l'expérience de chacun nous avons pu avancer rapidement. Personnellement, c'est mon expérience en programmation sur microcontrôleur ainsi que la réalisation de systèmes fiables dans le temps qui m'ont permis de proposer une solution à base de microcontrôleurs. Mes connaissances générales acquises au cours de ma formation et au cours de réalisations personnelles m'ont rendu légitimes mes choix quand aux différentes solutions proposées. Dès la première réunion, trois possibilités proposées sont sorties du lot : un mini PC, un PDA et une solution à base de microcontrôleur.

Nous avons alors décidé d'énumérer les avantages et inconvénients de chacune, ce qui nous a conduit à la réalisation du tableau suivant :

Solution envisagée	Avantages	Inconvénients
Mini PC embarqué	- très grande puissance de calcul, - portabilité du code - architecture connue et déjà utilisée - contrôle simple des actionneurs (Labjack)	- abstraction trop importante - consommation électrique très importante (autonomie entre 2 et 3 heures) - volumineux - fragile, sensible aux chocs - plantage possible
PDA	- grande puissance de calcul, - petit, loge facilement dans le bateau - contrôle simple des actionneurs (Labjack) - intègre déjà le GPS et le téléphone	- consommation électrique importante (autonomie entre 6 et 12 heures) - architecture pas bien maîtrisée - très sensible au plantage - ne peut pas redémarrer automatiquement
Microcontrôleurs	- consommation faible - conception maîtrisée - contrôle simple des actionneurs - liaison i2c et RS232 - puissance de calcul faible mais suffisante - l'ensemble peut être très robuste au plantage et aux coupures de courants - solide et facile à réparer - Petite, loge facilement dans le bateau	- faible puissance de calcul, impossible d'avoir des logiques trop évoluées - une fois réalisée, difficile de changer - pas de port USB maître

1.2.2. Choix de la solution

Chacune de ces solutions, comme nous venons de le voir, présente des avantages et des inconvénients. Nous avons donc débattu sur la solution à choisir. Personnellement, j'étais tout à fait d'accord sur le fait que la solution du PC embarqué avait fait ses preuves, comme le montre la réussite du projet SAUCE. Cependant je reste perplexe sur la fiabilité dans le temps. De plus, ce système consomme beaucoup et est relativement fragile, deux points très négatifs par rapport au cahier des charges. La solution du PDA n'était à mon avis pas encore adaptée. En effet, utilisant quotidiennement ce type d'appareil, j'ai fait de nombreuses tentatives de stabilisation du système avec l'aide de développeurs spécialistes dans ce domaine, je ne peux que constater une très forte instabilité dans le temps. J'ai donc défendu ma proposition, à savoir celle à base de microcontrôleur. Elle est loin d'être parfaite, mais répond au mieux aux exigences du cahier des charges, en particulier la fiabilité dans le temps et la faible consommation. En revanche, le système étant fabriqué à l'école, il est à l'état de prototype. Son unicité en fait un point faible : il ne peut pas être trouvé tel quel dans le commerce pour le changer rapidement en cas de défaillance. De plus la puissance de calcul est relativement faible, mais reste suffisante pour notre objectif de concours. Néanmoins, pour la traversée de l'atlantique, le système aura sûrement besoin d'une d'intelligence artificielle, beaucoup trop complexe pour être implantée dans un microcontrôleur. Il faudra, alors, remettre en question ce choix pour l'architecture de la plateforme électronique.

Après plusieurs réunions et exposés visant à défendre l'une ou l'autre des propositions, il fut finalement décidé d'adopter la solution des microcontrôleurs, tout du moins pour le concours. En plus des avantages cités précédemment, elle reste celle que je maîtrise le mieux.

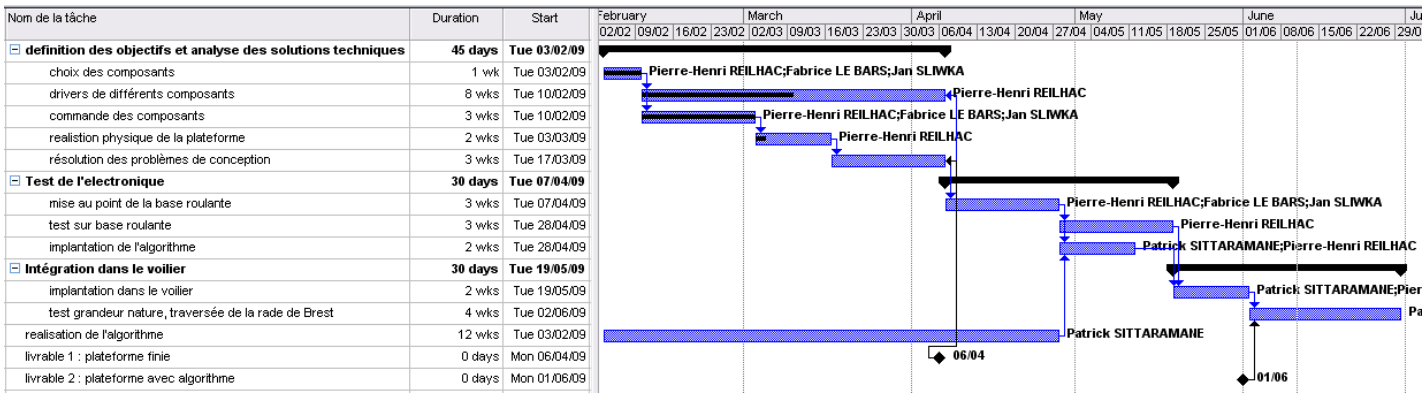
1.3. Gestion de projet

Une fois le projet mis en place, l'équipe constituée, les taches réparties, les objectifs définis, le cahier des charges réalisé et le choix de l'architecture validé, j'ai décidé de réaliser un emploi du temps de mon projet, en me fixant des objectifs intermédiaires à cours terme. Grâce à cela, j'ai pu surveiller mon avancement et éviter de prendre du retard.

Etant seul à travailler sur la réalisation de l'électronique, j'ai du choisir une organisation très linéaire comme nous allons le voir.

Voici le fichier Microsoft Project réalisé au début du projet. Les trois étapes citées ci-dessus correspondent aux trois grandes parties.

GANTT réalisé au début du projet :



Nom de la tâche	Duration	Start
definition des objectifs et analyse des solutions techniques	**45 days**	**Tue 03/02/09**
choix des composants	1 wk	Tue 03/02/09
drivers de différents composants	8 wks	Tue 10/02/09
commande des composants	3 wks	Tue 10/02/09
realistion physique de la plateforme	2 wks	Tue 03/03/09
résolution des problèmes de conception	3 wks	Tue 17/03/09
Test de l'electronique	**30 days**	**Tue 07/04/09**
mise au point de la base roulante	3 wks	Tue 07/04/09
test sur base roulante	3 wks	Tue 28/04/09
implantation de l'algorithme	2 wks	Tue 28/04/09
Intégration dans le voilier	**30 days**	**Tue 19/05/09**
implantation dans le voilier	2 wks	Tue 19/05/09
test grandeur nature, traversée de la rade de Brest	4 wks	Tue 02/06/09
realisation de l'algorithme	12 wks	Tue 03/02/09
livrable 1 : plateforme finie	0 days	Mon 06/04/09
livrable 2 : plateforme avec algorithme	0 days	Mon 01/06/09

1.3.1. Définition des objectifs et analyse des différentes solutions

La première partie est la partie la plus importante, car c'est elle qui va déterminer l'ensemble du projet. Elle consiste tout d'abord à définir les objectifs, à moyen et long termes, de sorte à orienter nos recherches. Dans un second temps nous avons organisé de nombreuses réunions afin de réfléchir sur les différentes approches possibles pour répondre au problème. Suite à cela nous avons fixé le choix de l'architecture. Vient alors le choix des composants et la réalisation de la plateforme, ainsi que le codage des drivers. A la fin de cette phase, nous avons clairement énoncé le problème, nous avons analysé et fixé les différents objectifs à atteindre, nous avons choisis et réaliser une structure pour le développement du projet.

1.3.2. Phase de test

Une fois la structure réalisée, il a fallu passer à la programmation et à la mise en place du programme. Au cours de cette phase j'ai analysé le comportement d'un bateau, j'ai étudié la navigation à voile. J'ai aussi dû tester les différents composants et vérifier leurs spécificités (ex : GPS, boussole). Suite à cela, j'ai cherché une architecture adaptée pour le programme. Le bateau n'étant encore opérationnel, il a fallu créer des plateformes pour tester les différents éléments du projet. Cette partie a donc eu pour objectif de valider les différentes solutions sélectionnées et de simuler les logiques de comportement du bateau (approche d'une bouée, logique vent).

1.3.3. Phase finale : implantation dans le voilier

Cette dernière phase est l'aboutissement du projet : la plateforme électronique est montée sur le bateau. Il faut l'adapter et tester les différentes fonctions. Il faut modifier certains paramètres et mettre au point les derniers réglages. L'objectif à la fin de cette partie est d'avoir un bateau prêt à participer au concours.

Pour les deux premières parties, le déroulement du projet a été tout à fait conforme à ce qui avait été prévu, chaque partie a été réalisée dans les temps. En revanche, à cause d'un retard au niveau de la fabrication du bateau, la dernière étape n'a pas pu commencer à temps. En attendant la livraison du bateau, d'autres tests ont été effectués sur les différentes fonctions et logiques, permettant ainsi de résoudre de nouveaux problèmes qui n'avaient pas encore pu être détectés.

2. Réalisation du programme

2.1. Du cahier des charges aux choix pour le programme

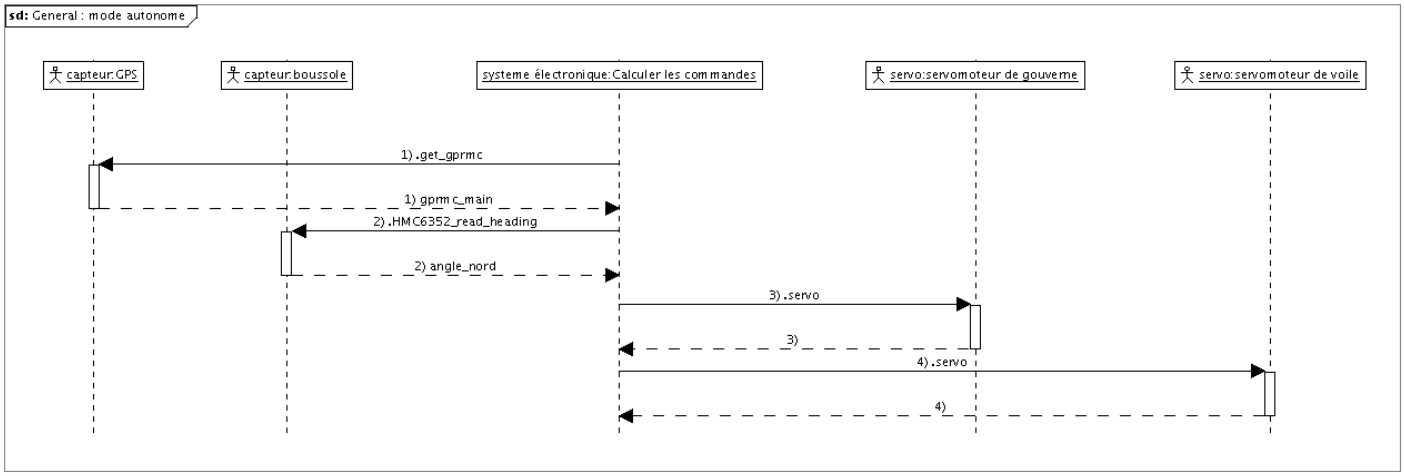
Une fois l'architecture choisie, je me suis occupé du programme embarqué. Du programme, dépendra l'efficacité de l'ensemble. C'est pourquoi l'étude et le choix de l'organisation du programme jouent un rôle important dans le projet.

Le cahier des charges général étant établi, il m'a semblé judicieux de reprendre ses spécificités et de les traduire en caractéristiques pour le programme. En effet, même si l'architecture électronique répond à un maximum de spécificités, un programme non adapté pourrait annihiler les avantages de la solution choisie. Inversement, certaines spécificités non réalisées par la partie électronique pourraient être corrigées par un programme adapté. La mise en place du programme est par conséquent une partie importante dans la réussite du projet.

Tout d'abord le choix du langage utilisé. Pour la programmation des microcontrôleurs type PIC, il existe 3 principaux langages : l'assembleur, le basic et le C. mon choix s'est porté sur le C, car il permet une programmation plus simple que l'assembleur. De plus, je maîtrise ce langage bien mieux que le basique et l'ENSIETA possède un compilateur en C pour les PICs très performant : le PICC de CCS.

La navigation à la voile nous impose de connaître certains paramètres, que l'on retrouve dans le cahier des charges : l'orientation du bateau, sa position et la direction du vent. Pour le concours, nous avons aussi besoin de connaître la position des bouées. Après avoir étudié la construction du bateau, j'ai remarqué que seuls deux actionneurs sont accessibles sur notre voilier : la commande des safrans (direction) et le réglage de la grand-voile (propulsion). Notre programme peut donc être vu comme le système global suivant :

Use Case global du système en mode autonome :



Le cahier des charges ne spécifie pas uniquement les entrées/sorties. D'autres caractéristiques concernent directement le programme :

a) Le système doit être capable de redémarrer tout seul

Le fait que le programme redémarre tout seul est ici dû au choix de la structure : dans un microcontrôleur, une fois alimenté, le programme s'exécute. En revanche, dans l'éventualité d'une autre structure, ce point est crucial pour un bon fonctionnement de l'ensemble.

b) Le système ne doit pas rester bloqué en attente

Comme nous venons de le voir, le système doit dialoguer avec des composants extérieurs comme les capteurs. Ce dialogue est parfois à double sens : le système doit interroger le capteur et attendre sa réponse. Si par erreur, le capteur venait à ne pas répondre ou que le microcontrôleur ne détecte pas la réponse (ce problème peut arriver de temps en temps avec la liaison rs232), il ne faut pas que le système reste bloqué. Il faut donc faire attention au temps consacré à l'attente pour les réponses. Il faut éviter la possibilité d'entrer dans une boucle infinie. Afin de parer à ce type de problème, j'ai choisi d'utiliser tout au long du programme un timer « chien de garde ». Ce timer présent dans les PICs fait redémarrer le système s'il n'est pas réinitialisé suffisamment souvent. Ainsi, si le système venait à rester en attente d'une réponse qui n'arrivera pas, le timer fera redémarrer le programme et le système pourra reprendre son fonctionnement normal.

c) Le système doit être robuste aux coupures de courant

Cette spécificité a été ajoutée car ce phénomène peut arriver souvent. Le programme ne doit pas être en train de modifier des données vitales ou enregistrer des informations pour les réutiliser plus tard quand la coupure de courant survient. Dans le cas contraire une fois le courant revenu et le programme remis en marche, les informations males enregistrées seront prise en compte et le programme utilisera sans le savoir des informations erronées et ne jouera plus correctement son rôle. J'ai donc décidé, afin de limiter le plus possible ce risque, de ne pas utiliser d'informations stockées en dehors de celles mises dans la mémoire externe. Le programme ne peut donc plus utiliser d'historique, seuls des informations temps réel sont acceptées. L'écriture dans la mémoire externe est restreinte au minimum : j'ai choisi de n'autoriser le programme à y écrire qu'à la validation d'une bouée. C'est-à-dire que lorsque le système estime avoir contourné la bouée et qu'il faut atteindre la bouée suivante, il change le numéro de la bouée à atteindre dans la mémoire. Cette opération ne prend que quelques millisecondes et n'intervient qu'une petite dizaine de fois au cours d'une épreuve. De cette façon, si une coupure de courant venait à perturber le bon déroulement du programme, il y a très peu de chance pour qu'elle détériore les données vitales au bon fonctionnement du programme. Cette solution a un inconvénient non négligeable : elle empêche d'utiliser un historique pour calculer la trajectoire du bateau, mais elle a à mon sens un avantage bien plus important : la fiabilité. Grâce à cette méthode, si le programme est écrit en conséquent, le bateau arrivera toujours à se remettre en course après une coupure de courant.

d) Un opérateur doit pouvoir prendre la main sur la commande du bateau en cas de problème

Cette partie pourrait être réalisée directement par le programme, mais pour des raisons de sécurité, j'ai choisi de réaliser une plaque électronique indépendante, très simple et très fiable. Cette solution évite de surcharger le programme et diminue le risque de conflits. De plus, en cas de problème avec l'électronique, le bateau peut toujours être ramené à bon port pour être réparé.

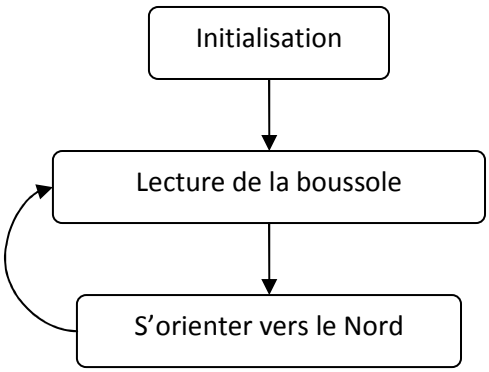
Conclusion :

Après une étude approfondie du cahier des charges, j'ai déterminé les spécificités qui pouvaient s'appliquer au programme. Après cela, je les ai interprétées et traduites en caractéristiques propres à la programmation. J'ai pu ainsi fixer certaines règles pour la suite du développement du code afin de respecter le cahier des charges et suivre une réalisation logique. Cette étude nous a conduit entre autre à chercher à éviter tout type d'historique.

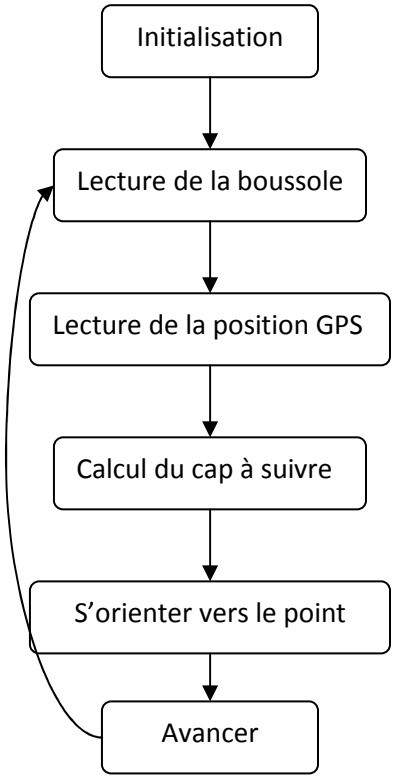
2.2. Programmation séquentielle

Afin de mettre en place un programme cohérent avec le projet, j'ai continué à étudier le cahier des charges et la navigation à la voile. Pour cela j'ai rencontré plusieurs spécialistes du domaine. Au travers de leurs témoignages, j'ai pu comprendre les bases de la navigation. Je me suis inspiré des méthodes qu'ils utilisent pour réaliser mon programme.

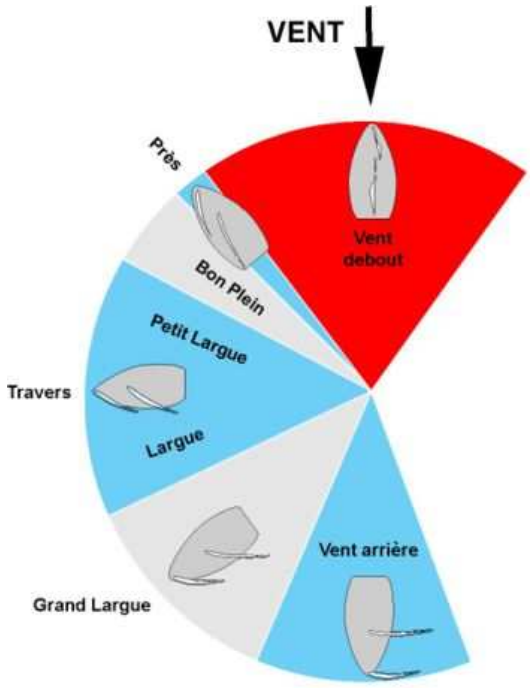
Tout d'abord, la première chose évidente mais essentiel : il faut savoir diriger le bateau. Comment contrôler un voilier quand on ne sait pas l'orienter dans une direction donnée ? C'est sur ce principe très simple que j'ai commencé mon programme :



Une fois cette fonction validée par plusieurs tests, j'ai continué à faire évolué le programme : le système peut maintenant rejoindre un point connu. Pour cela le système doit être capable de se situer dans l'espace et de calculer dans quelle direction est le point à viser, puis il doit être en mesure de s'orienter. Le programme ayant ainsi évolué et est devenu :



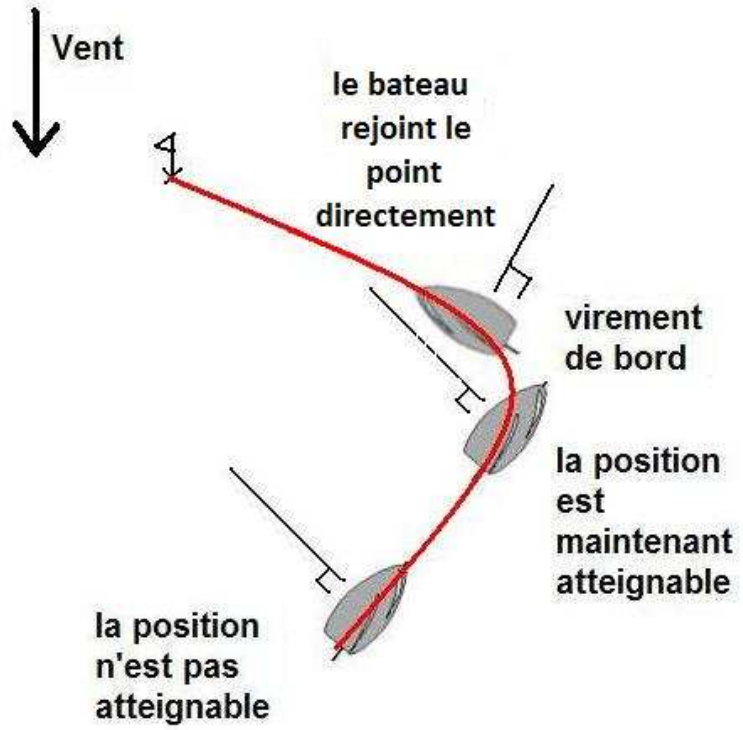
Après avoir validé cette partie par de nombreuses expériences, le système est maintenant capable de connaître sa position, de connaître son orientation et de se diriger dans la direction d'un point. Ce programme serait parfait pour un véhicule terrestre, pouvant se déplacer par ses propres moyens. Hors pour notre projet de voilier, le système de propulsion est le vent. J'ai donc continué à faire évoluer mon programme de sorte à pouvoir prendre en compte ce paramètre. Au cours de mes recherches sur la navigation à la voile, j'ai trouvé ce que les marins appellent la rose des vents :



Ce dessin récapitulatif donne entre autre une information importante : le réglage de la voile est fonction directe de la direction du vent et de l'orientation du bateau. Mais on remarque aussi sur ce schéma que toutes les directions ne sont pas accessibles : on ne peut pas remonter face au vent. En réalité, il existe un cône de 45° de part et d'autre de l'axe du vent qui n'est pas accessible directement. D'après l'expérience des marins que j'ai rencontrés, il est aussi préférable d'éviter le vent arrière, peu efficace et rendant le bateau instable. Afin d'éviter ces directions, les marins effectuent des bords : cela consiste à naviguer au près puis virer de bord pour rejoindre le point visé. L'art de la navigation réside dans le compromis entre la vitesse et distance à parcourir. Pour l'instant je ne cherche pas à optimiser le comportement du bateau, mais à le fiabiliser. Les navigateurs calculent leur trajet et cherchent à le suivre pour remonter au vent correctement. Mais nous avons vu précédemment que je ne souhaitais pas utiliser d'historique ou enregistrer des informations pour les réutiliser plus tard, ce qui signifie également que je ne calculerai pas d'itinéraire pour essayer de le suivre. J'ai donc du chercher une autre solution. Le système que j'ai mis en place est en réalité le plus simple possible :

Lorsque les directions sont directement atteignables, le système règle la voile en conséquence et suit le cap. En revanche lorsque la direction nécessite de faire des bords, le système commence un bord et vérifie en permanence si la direction n'est pas devenue directement accessible. Si ce n'est pas le cas, le bateau continu à tirer son bord.

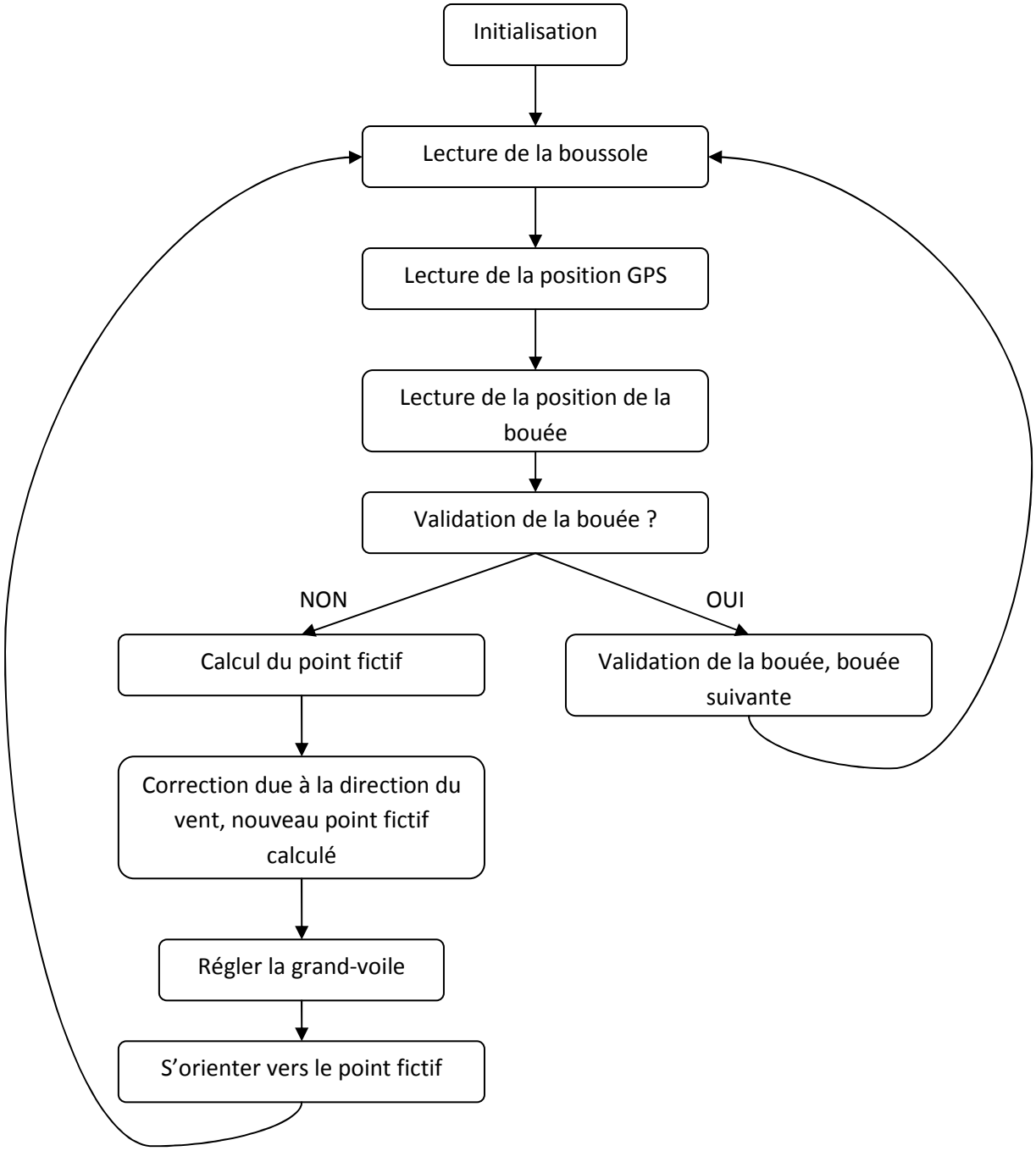
Logique « vent » utilisée par le bateau :



D'après les navigateurs, le virement de bord constitue le point le plus critique de la manœuvre. Afin d'éviter un nombre trop important de manœuvre, un contrôle par hystérésis a été mis en place. Comme aucun historique n'est enregistré, l'hystérésis se base sur l'orientation actuelle du bateau et l'angle qu'il devrait avoir. Grâce à ce principe, le système est maintenant capable de diriger le bateau vers un point donné, quelque soit la direction du vent. Le système pourrait donc être suffisant, mais pour les épreuves, il faut savoir rejoindre les bouées, mais également savoir la contourner du bon côté pour quelle soit validée. Pour parer à ce problème, il a fallu mettre en place une logique supplémentaire. Le système classique est la mise en place d'une logique d'approche de la bouée, mais ce système demande là encore l'existence d'un historique et la prévision du parcours, deux éléments que je cherche à éviter. Il a donc fallu que je mette en place une fonction qui soit valable quelque soit la position relative du bateau par rapport à la bouée. Pour l'instant mon système est capable de s'orienter ver un point. J'ai donc décidé de mettre au point une fonction qui calcule en permanence un point fictif à viser : le point dépende directement de la position du bateau par rapport à la bouée. Mais à l'approche de la bouée, les imprécisions GPS gênent la navigation et peuvent faire passer le bateau du mauvais coté de la bouée. Il faut donc trouver une fonction capable de palier à ce problème. Après de nombreux essais infructueux, j'ai fini par en trouver une : elle consiste à chercher en permanence à atteindre la tangente à un cercle ayant pour centre la bouée. Si le rayon est bien dimensionné pour compenser les erreurs GPS, le bateau contournera la bouée sans avoir à utiliser d'historique ni sans avoir calculé un itinéraire. En réalité je me suis basé sur le comportement des marins : même s'ils calculent leur trajet, à chaque instant ils visent un point fictif afin de garder le bon cap.

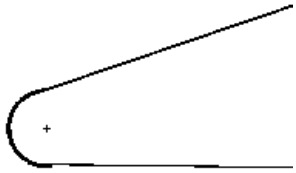
Pour valider la bouée, le système utilise un angle entre la position du bateau et le nord, de centre la bouée. Cet angle est enregistré dans la mémoire externe et est spécifique au parcours demandé. La validation de la bouée est le seul moment où des informations permanentes sont enregistrées. Une fois la bouée validée, le système cherche automatiquement à atteindre la bouée suivante.

Le programme est ainsi devenu :

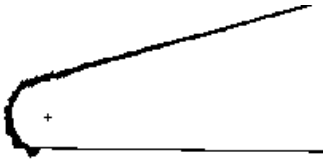


Pour valider cette technique de contournement, des tests en grandeur réelle auraient demandé trop de moyens et auraient été difficiles à mettre en place. J'ai donc décidé de simuler mes logiques grâce à un logiciel. Ainsi j'ai pu essayer rapidement de nombreuses logiques différentes avant d'en choisir une. La simulation nous donne le comportement théorique du bateau à l'approche de la bouée :

Sans erreurs GPS :



Avec erreurs GPS :



Malgré un parcours moins précis, le bateau contourne correctement la bouée, la logique a ainsi été validée.

Conclusion :

La programmation utilisée ici est une programmation séquentielle : on modifie les données reçues en les faisant passer successivement dans différentes logiques afin de calculer un point fictif. Ce point, une fois calculé, est vu par le système comme le point à viser et oriente le bateau dans sa direction. Grâce à ce type de programmation, il n'y a qu'une seule boucle : la boucle principale. Le système arrive ainsi à rejoindre une bouée, la contourner et la valider, et ce sans utiliser d'historique ni d'itinéraire calculé. Les différentes caractéristiques définies par le cahier des charges sont tout à fait respectées.

2.3. Le code

2.3.1. Organisation des fichiers

Afin de faciliter l'utilisation du code, j'ai décidé de le répartir dans différents fichiers :

Chaque composant utilisé a son propre dossier comportant ses drivers. Au début de ces fichiers, les éléments à définir pour utiliser le composant (ex : affectation des pins, constantes…) sont mis en commentaire. Ces éléments dépendent du projet et doivent obligatoirement être définis dans celui-ci.

Afin de pouvoir modifier facilement ces paramètres et les utiliser dans différents tests, je les ai rassemblé dans un fichier « configuration.h ». Ce fichier joint l'ensemble des liens vers les drivers et leurs paramètres. Il contient également la configuration des fusibles du PIC et les variables globales. Grâce à ce fichier, il est facilement possible de changer de plateforme de test, du moment que la carte électronique reste la même.

Pour chaque test réalisé, un fichier .c a été créé. Il comporte la fonction principale et un appel au fichier commun « configuration.h ».

2.3.2. L'encodage des données

La programmation sur microcontrôleur n'impose pas les mêmes contraintes que sur un ordinateur : les capacités de calcul ne sont pas les mêmes. Pour éviter des conflits lors de leur utilisation, j'ai décidé d'encoder les positions GPS. Une position GPS reçue dans une trame GPRMC est constituée de plusieurs éléments :

- latitude
  - Degrés
  - Minutes
  - Millièmes de minute
  - Sens
- Longitude
  - Degrés
  - Minutes
  - Millièmes de minute
  - Sens

Cette forme complexifie énormément le traitement de ces données. C'est pourquoi j'ai choisi de les convertir dans un autre format :

La latitude et la longitude sont converties en microdegrés, puis additionnées ou soustraites à 2^31 en fonction de leur sens (Nord : +, Sud : -, Est : +, Ouest : -). Les deux valeurs obtenues sont enregistrées dans des variables de type **int32**. Elles peuvent ainsi être directement utilisées dans des calculs sans que le code ait à traiter les différents cas de cardinalité.

Exemple :

48°25,795' N

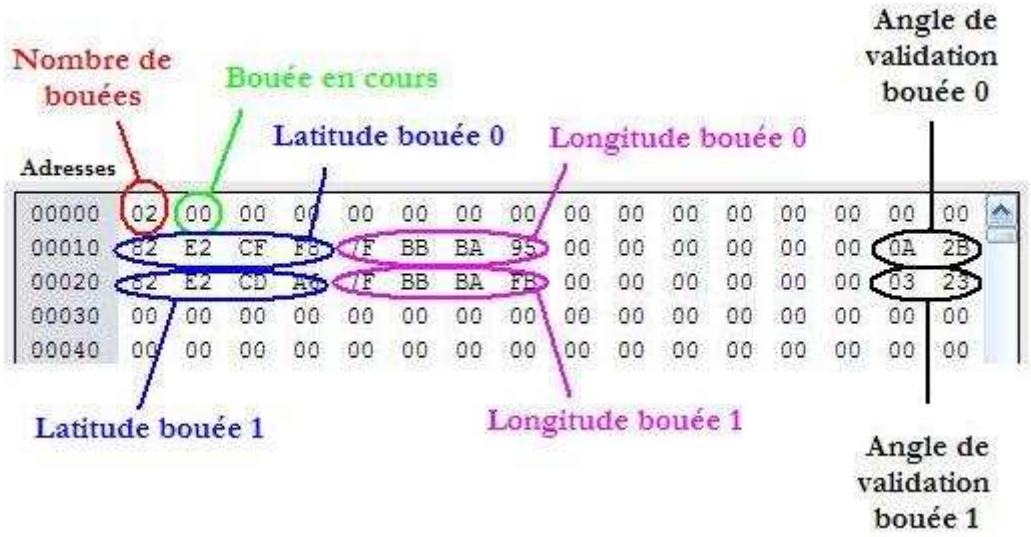
➔ 48 x 10^6 + (25.795 x 10^6)/60 **N**

➔ 2^31 **+** 48429916

➔ **2195913564**

Ce codage des positions GPS a permis de faciliter également l'enregistrement des positions dans la mémoire I2C.

Organisation de la mémoire :



L'adresse **0x00** contient le nombre total de bouées
L'adresse **0x01** contient le numéro de la bouée à viser
Les adresses de 0x02 à 0x09 ne sont pas utilisées pour le moment

A partir de **0x10** sont enregistrées les bouées.
Elles sont codées sur 16 bits :

Les 4 premiers sont la latitude	(MSB - - LSB)
Les 4 suivants sont la longitude	(MSB - - LSB)
Les 2 derniers sont l'angle de validation de la bouée	(MSB LSB)
Les autres bits ne sont pas encore utilisés.	

Cette organisation est personnelle, mais elle s'avère très pratique à utiliser car elle facilite l'écriture des fichiers contenants les positions des bouées : à partir de la deuxième ligne, chaque ligne correspond une bouée. Afin de pouvoir les enregistrer dans la mémoire, il faut soit utiliser le logiciel PICKit2 et modifier à chaque fois à la main les valeurs, soit modifier le fichier « Base 24AA1024.hex » avec un éditeur de texte puis le charger dans la mémoire[1].

2.3.3. Les erreurs commises

La programmation est une activité fastidieuse et personne n'est à l'abri d'une erreur de frappe ou de syntaxe. Mais les erreurs de code de sont pas les seules erreurs que j'ai commises lors de la réalisation de ce projet.

Un des points les plus vulnérables de la programmation en C est la gestion de la mémoire. Ce point est bien évidemment présent dans la programmation en C des PICs. Mais il est d'autant plus difficile à gérer que le compilateur PICC de CCS ne signale pas les erreurs de gestion de la mémoire. Cette spécificité m'a conduite à plusieurs erreurs, parfois difficiles à diagnostiquer. La détection de ces erreurs et leur résolution m'ont demandé beaucoup de recherche et de temps.

Le premier problème rencontré était assez difficile à identifier car il n'apparaissait pas tout les temps : le PIC fonctionnait correctement un certain temps, en travaillant sur une boucle de test et d'un seul coup son comportement devenait inattendu. Ce problème remettait en cause la fiabilité dans le temps du système, chose que je souhaitais éviter impérativement. Le code semblait pourtant bon puisqu'il fonctionnait normalement au début. Après de nombreuses heures passées à chercher cette erreur, j'ai fini par me rendre compte qu'un ***return*** était placé dans un cas particulier avant la libération de la mémoire (***free()***). Au cours des différentes boucles, il y avait une fuite de mémoire et le programme finissait par ne plus se dérouler correctement. La solution et donc de faire très attention à tous les cas possibles pour éviter les fuites de mémoire si l'on cherche à fiabiliser le système.

Les problèmes dus à la fuite de mémoire de sont pas les seuls problèmes qu'entraîne la gestion des données dans un PIC. En effet, un autre problème encore plus spécifique à la programmation sur PIC m'a longtemps laissé sans solution : une fois que le programme avait atteint 50% de la mémoire ROM du PIC, lorsque je rajoutais une ligne de commande très simple, la mémoire utilisée passait automatiquement à 70% puis à 98%. Ce phénomène n'était absolument pas logique par rapport à la taille du code et le programme ne s'exécutait plus. Ma première idée était que le problème venait de la version de mon compilateur, j'ai donc essayé avec la mise à jour, puis avec un autre type de PIC, mais une fois les 50% de la ROM dépassé, le problème survenait toujours.

[1] La programmation de la mémoire est expliquée dans l'annexe « mode d'emploi »

Après plusieurs recherches sur Internet et plusieurs discutions sur un forum dédié à ce compilateur, nous avons fini par comprendre l'origine du problème : la mémoire du PIC est organisée en banks. Pour s'exécuter correctement le programme doit avoir toutes les parties dépendantes les unes des autres dans la même bank. Or le compilateur de répartie pas tout seul le programme. Cela a pour conséquence que lorsqu'un programme est trop important et qu'il n'est pas réparti manuellement dans les banks, il ne peut pas dépasser 50% de la ROM sous peine de ne plus fonctionner. Une fois la cause identifiée, le problème fut rapidement résolu : dès que cela est possible, les fonctions sont détachées manuellement du programme afin de les répartir au mieux dans la mémoire. Pour cela il faut préciser avant la fonction : « #separate ». Le compilateur ne le fait pas automatiquement mais il empêche les séparations abusives. L'erreur a ainsi pu être réparée et le programme a pu continuer à évoluer et être complexifié.

Même si le PIC choisi a une mémoire importante, elle n'en reste pas moins limitée. La recherche d'optimisation de l'utilisation de cette mémoire m'a conduite à faire une autre erreur : oublier une caractéristique importante d'un PIC à savoir que la vitesse d'exécution n'est pas très grande. Cette erreur est apparue lors de la simplification de la fonction de récupération de la trame GPS du récepteur par le PIC. La fonction est très rébarbative : il faut transformer des chaînes de caractères ASCII représentant des nombres en valeurs. J'ai donc cherché à écrire une fonction simple, appelée plusieurs fois, qui reçois via la liaison série les caractères et qui renvoie la valeur. Mais le temps d'appel à cette fonction n'était pas négligeable et des données étaient perdues, rendant les coordonnées inutilisables. J'ai donc du revenir à une fonction longue et répétitive pour pouvoir récupérer l'ensemble des données. Contrairement à la programmation sur ordinateur où le temps d'exécution est généralement négligeable, il ne faut pas oublier qu'un PIC travaille beaucoup plus lentement.

Les erreurs que j'ai commises n'étaient pas forcément évidentes. Elles m'ont demandé beaucoup de travail de recherche et de test pour réussir à les reproduire pour ensuite les identifier pleinement et les corriger. Elles m'ont néanmoins permis d'approfondir mes connaissances dans la programmation des PICs et je saurai dorénavant les éviter.

3. Réalisation de l'architecture électronique, implantation et tests

3.1. Choix des composants

Une fois la solution de la plateforme à base de microcontrôleur choisie, j'ai dû commencer les recherches pour trouver les composants que je souhaitais utiliser.

Le premier élément à choisir est le microcontrôleur car c'est lui la pièce maîtresse du système : c'est lui qui déterminera les liaisons qui peuvent être utilisées pour communiquer avec les autres composants. Ayant l'habitude d'utiliser les microcontrôleurs PIC de la marque Microchip, mon choix a été dans cette direction. Au cours de nombreuses réalisations, j'ai acquis une certaine connaissance de ces produits et je peux affirmer qu'ils sont tout à fait adaptés au type d'application que nous cherchons à mettre ne place. Il m'a fallu tout de même vérifier qu'il existait un modèle correspondant exactement à nos besoins. Après l'étude de leur catalogue, j'ai fini par sélectionner le PIC18F2550. C'est un des plus performant, il est facile à trouver dans le commerce. Il possède une mémoire suffisamment importante pour contenir un programme conséquent ainsi que différentes liaisons pour communiquer avec les autres éléments dont : la liaison série, la liaison i2c et la possibilité d'utiliser un port USB esclave. De plus, ce composant consomme relativement peu d'énergie et peut travailler dans une grande fourchette de tension d'alimentation (2V à 5.5V). La plateforme aura donc comme base une plateforme de test pour le PIC18F2550.

Une fois le microcontrôleur choisi, il faut chercher les différents éléments pour répondre au cahier des charges, en vérifiant leur compatibilité avec le reste des éléments. Pour cela j'ai consulté plusieurs sites Internet décrivant la réalisation de robots. J'ai pu ainsi orienter mes recherches vers certains fabricants et revendeurs pour trouver les composants adaptés. Après avoir traduit le cahier des charges en capteurs, j'ai obtenue la liste des composants à chercher :

- un récepteur GPS
- une mémoire externe non volatile
- une boussole
- une girouette
- un émetteur/récepteur de données longue portée
- un système de commande pour les servomoteurs
- un régulateur de tension

Ces composants électroniques doivent consommer le moins possible et doivent pouvoir fonctionner dans une fourchette d'alimentation la plus grande possible (typiquement 3V à 5V). Il existe beaucoup de systèmes complets grand public. Mais il est difficile de trouver uniquement les composants électroniques. Il n'existe en réalité que très peu de fabricants de capteurs. Il s'avère par exemple qu'il n'existe sur le commerce que deux composants boussoles. Les appareils vendus dans le commerce intègrent forcément un deux ces deux modèles, tout du moins pour les appareils à prix raisonnable.

Après de nombreuses recherches et études de documents techniques, la liste des composants fut établie :

- Le microcontrôleur:		PIC18F2550 sur plateforme	(2V à 5.5V)
- le GPS	:	FV-M8 (RS232)	(3.3V à 5V)
- la mémoire	:	24FC1025 (i2c)	(2.5V à 5.5V)
- la boussole	:	HMC6352 (i2c)	(2.7V à 5.2V), typique 3V
- l'émetteur HF	:	Adeunis (RS232)	(4.5V à 36V)
- régulation de la tension :		TEN25-1211	(entrée DC 9V/18V, sortie DC 5V)

Pour la commande des servomoteurs, des systèmes existent mais j'ai choisi de réaliser personnellement à partir de deux PIC16F628 de sorte à pouvoir rajouter des fonctions telles que la détection automatique de la prise de commande manuelle.

En revanche aucune girouette simple n'a été trouvée. Après avoir contacté différents centres, un laboratoire nous a proposé de nous en prêter une. Mais après plusieurs demandes pour récupérer la girouette promise, nous avons appris que le laboratoire demandait en échange une étude complète de celle-ci, ce qui n'était pas réalisable par les membres de l'équipe (manque de temps et de connaissance en la matière). Nous nous retrouvons donc sans girouette. La technique alors envisagée est la suivante : nous partons du principe que le vent ne changera pas de direction le temps des épreuves. Juste avant la course, nous enregistrerons la direction du vent dans la mémoire. Nous nous gardons aussi la possibilité de transférer cette donnée, et cette donnée uniquement par l'émetteur HF. Ce point négatif nous vaudra peut-être une pénalité au concours, mais cela ne devrait pas nous empêcher de participer.

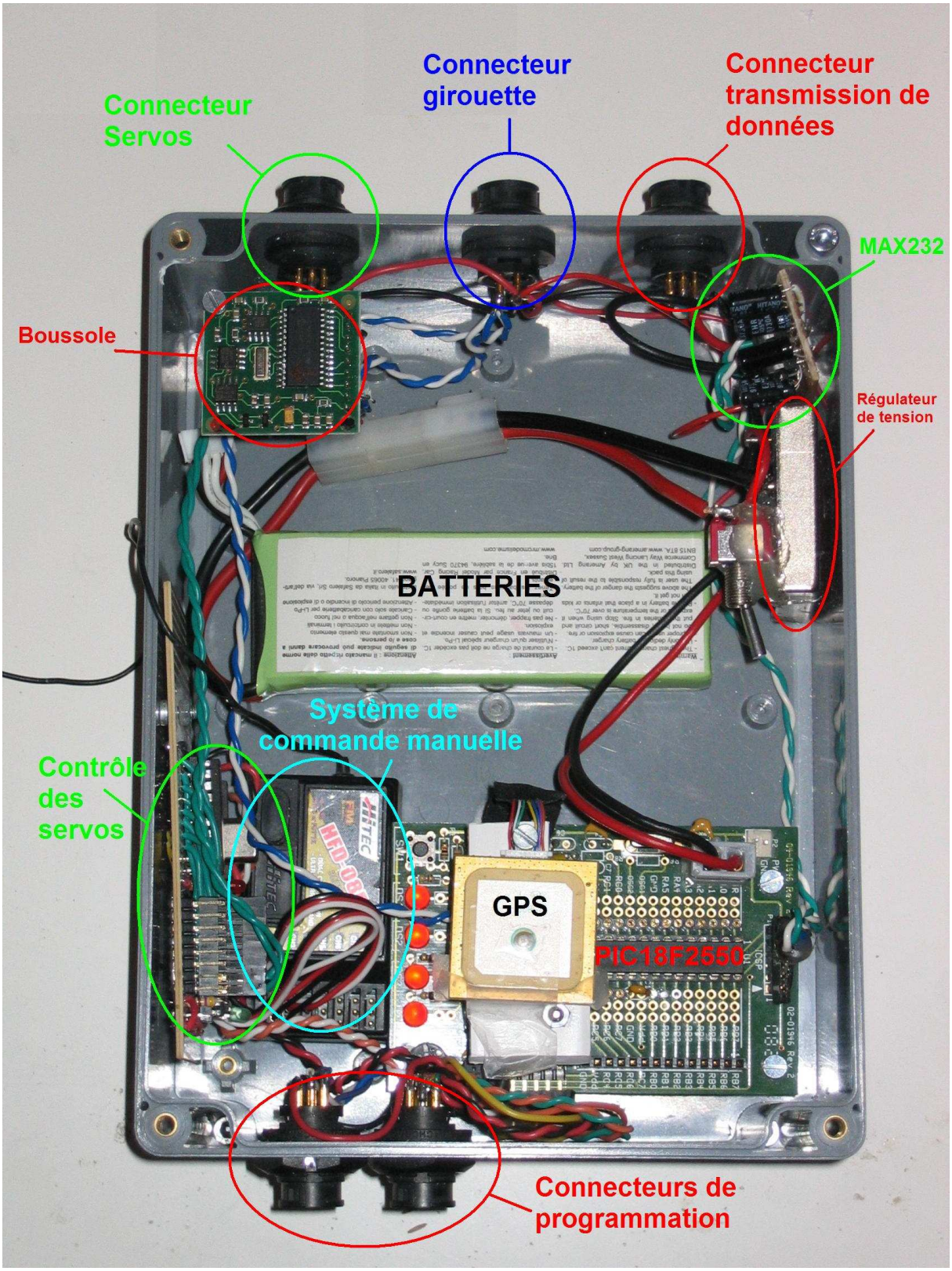
3.2. Présentation de la plateforme

Afin de les protéger de l'eau, les composants ont été intégrés dans une boite étanche. Les communications avec l'extérieur se font par l'intermédiaire de connecteurs étanches. Pour assurer le maintien des différents éléments dans la boite, j'ai utilisé des bandes autocollantes scratch. Cette technique, très souvent utilisée dans ce type de réalisation, a comme avantage de faciliter le montage et le démontage des éléments. Elle permet aussi d'absorber les vibrations et elle ne perd pas de son adhérence avec l'humidité. Ces solutions ont déjà été utilisées dans d'autres projets comme le projet SAUCE, et ont fait leurs preuves, c'est pourquoi elles ont été choisies.

La boite peut contenir jusqu'à quatre batteries, lui donnant une autonomie de plusieurs dizaines d'heures.

L'ensemble est directement reconfigurable en passant par les connecteurs étanches. Ainsi au début de la course les coordonnées des bouées ainsi que les informations nécessaires au parcours pourront être modifiées jusqu'au dernier moment.

Implantations des différents éléments dans le boîtier étanche :



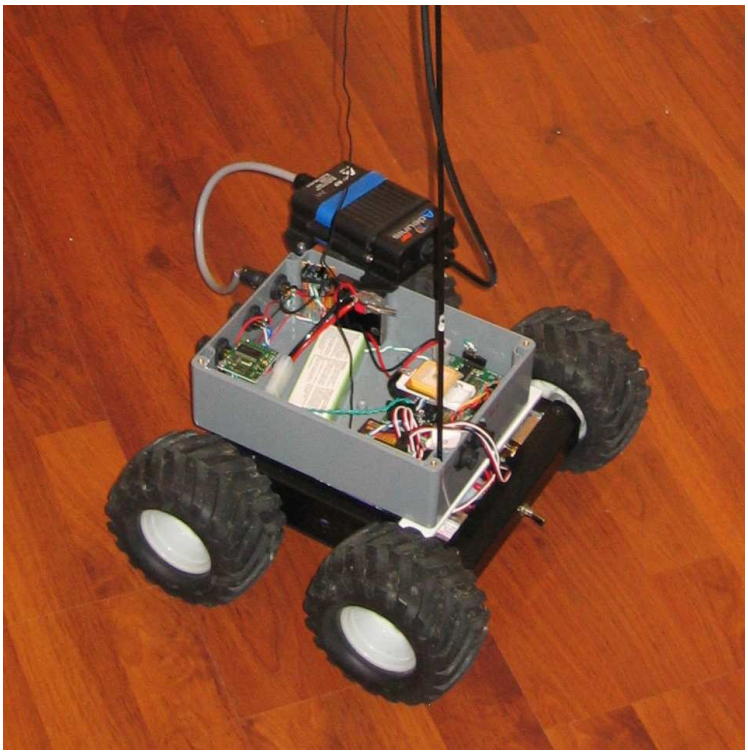
3.3. Mise en place des tests et résultats

Une fois la réalisation de la plateforme de développement finie, j'ai mis en place une série de tests afin de pouvoir développer correctement le programme. Pour cela j'ai dû rédiger des scénarios afin de pouvoir tester exactement ce que je souhaitais, sans être gêné par d'autres problèmes qui n'avaient pas encore été résolus.

Le voilier étant en cours de construction, il ne pouvait pas servir directement de plateforme de test. De plus, les tests sur le bateau nécessitent un plan d'eau et une connaissance suffisamment avancée du système pour éviter tout problème, ce que nous ne possédons pas encore.

Les premiers tests ont tout d'abord été réalisés uniquement avec la platine de développement : ces tests avaient pour objectif de vérifier le bon fonctionnement des différents drivers. J'ai placé par exemple la plateforme à un point GPS connu et j'ai vérifiée qu'elle m'indiquait la bonne position, à une précision de 5m près (précision de notre antenne GPS). Ces tests ont été réalisés pour tous les capteurs, afin de s'assurer que si un problème survenait, qu'il ne vienne pas des drivers.

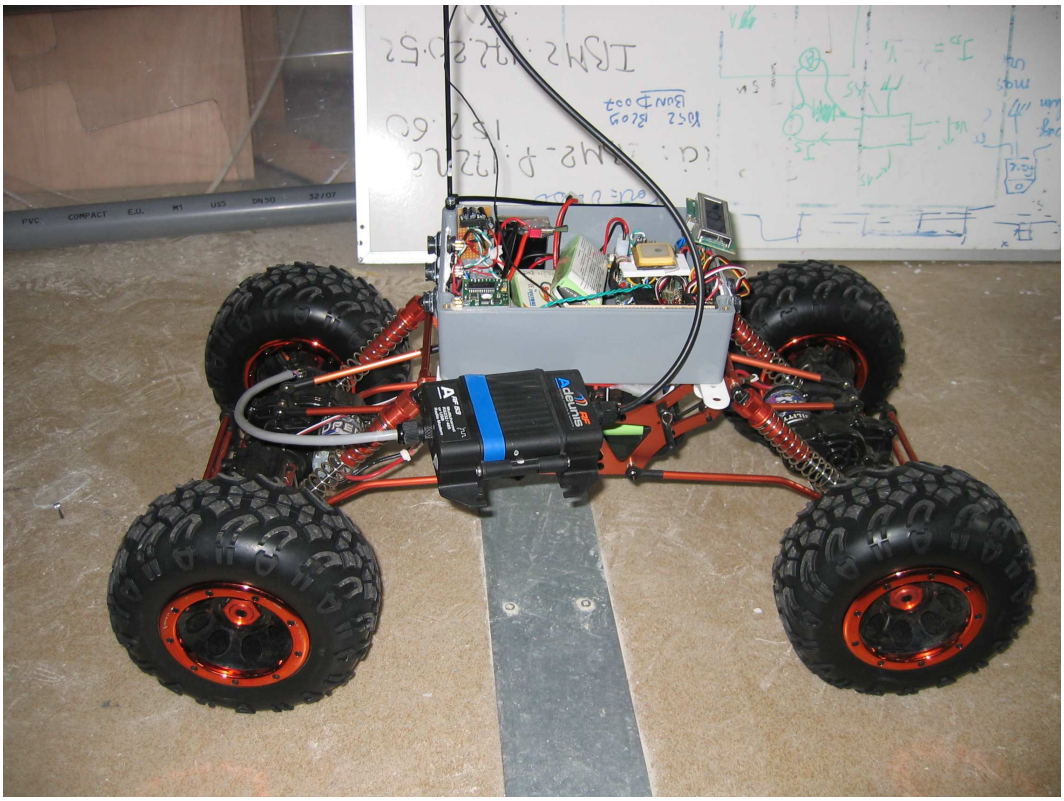
Je suis ensuite passé aux essais des différents programmes. Pour cela j'ai utilisé une base roulante déjà existante à l'école, que j'ai adapté pour qu'elle puisse recevoir la plateforme de développement et qu'elle soit commandable par le système.



Grâce à cette plateforme j'ai pu tester le bon fonctionnement de l'orientation du système vers le nord, puis le programme pour rejoindre un point. Les tests ont permis de valider ces fonctions tant que le robot est à une distance supérieur à 20m du point. En revanche, lorsque la distance est trop courte, le robot n'arrive plus à s'orienter correctement. Ce problème n'est pas directement dû au programme, mais à la façon dont la plateforme se déplace : elle est constituée de quatre roues motrices couplées deux par deux, de par et d'autre de la plateforme. Pour s'orienter, le système s'arrête, essaye de s'orienter dans la bonne direction et repart. A l'école, la seule surface suffisamment grande pour pouvoir faire ces tests est le terrain de foot. Mais il s'avère être très irrégulier pour une plateforme aussi petite. De plus, à cause de l'imprécision GPS, à l'approche du point, la plateforme a du mal à s'orienter, ce qui empêche de poursuivre correctement les tests.

Pour pouvoir continuer les tests, en particulier ceux sur la logique de contournement de la bouée, j'ai demandé à changer de plateforme de test. Afin que celle-ci puisse me permettre de continuer mes tests, j'ai du spécifier les caractéristiques qu'elle devait avoir et prouver par une démonstration les problèmes de la plateforme actuelle.

La nouvelle plateforme est plus haute, avec quatre grosses roues motrices et l'ensemble est entièrement sur suspension. Ces spécificités permettent à cette plateforme de ne plus être gênée par les irrégularités du terrain. De plus j'ai demandé à ce que l'orientation se fasse par des roues directrices et non par une différence de vitesse entre le coté droit et le coté gauche comme c'était le cas sur l'ancienne. Ainsi le déplacement sera beaucoup plus conforme à celui d'un bateau : pour s'orienter la plateforme devra avancer. Grâce à ce principe, elle constitue naturellement un filtre passe-bas mécanique qui filtre les oscillations de la position GPS.



La réalisation du bateau n'étant pas terminée, je n'ai pas pu faire les tests finaux. Mais normalement, l'ensemble des logiques ayant été validé par les tests précédents, l'intégration dans le bateau devrait être rapide et ne devrait nécessiter que quelques réglages pour être opérationnel.

Bilan

L'équipe Microtransat a vue le jour cette année à l'ENSIETA. Participer à cette traversée de l'atlantique par un robot voilier autonome est devenu notre objectif à long terme. Pour y parvenir, nous avons réalisé, en amont, une étude déterminant la faisabilité de ce projet. Une fois cette faisabilité avérée, nous nous sommes fixés des objectifs à moyen terme réalisable dans l'année. Nous avons donc continué notre étude et avons fini par rédiger un cahier des charges pour l'électronique et le voilier. Suite à cela nous avons réfléchi sur différentes solutions envisageables pour l'architecture électronique et nous avons décidé, aux vues de ses avantages, de développer pour le moment une plateforme à base de microcontrôleurs. J'ai alors commencé le développement de cette plateforme : d'après les critères donnés par le cahier des charges, j'ai déterminé les composants dont j'avais besoin, je les ai identifiés puis testés pour finalement les implanter dans la plateforme. J'ai réalisé totalement la plateforme de développement pour qu'elle puisse être incorporée dans le voilier. En parallèle, j'ai cherché une structure pour le programme afin qu'il corresponde au mieux à nos besoins. J'ai ainsi décidé d'appliquer une programmation séquentielle et de n'utiliser aucun historique. De nombreux tests et simulations m'ont permis de valider cette approche.

La réalisation du bateau n'étant pas totalement terminée, je n'ai pas encore pu le prendre en main ni implanter l'électronique. Cette partie sera réalisée cet été : nous ferons des tests avec le voilier afin de déterminer son comportement puis nous effectuerons les réglage sur l'électronique de sorte à être prêt pour le concours « World Robotic Sailing Championship ».

Le travail important a été réalisé par l'ensemble de l'équipe. Il pourra servir de base pour les prochains concours : les différentes plateformes de test qui ont été réalisées permettrons en particulier de gagner beaucoup de temps dans le développement. En revanche, le programme embarqué sera à modifier et à améliorer en fonction de nos résultats et de nos retours d'expérience du concours. Le voilier sera lui aussi en permanente évolution afin de palier au mieux aux différentes difficultés que nous rencontrerons.

Conclusion

Le challenge Microtransat était nouveau pour l'ENSIETA. Comme nous venons de le voir au cours de ce rapport, cette nouveauté m'a permis de participer pleinement à la mise en place de ce projet, comme cela se ferait dans le monde professionnel. J'ai pu ainsi adopter le comportement qu'aurait un ingénieur dans cette situation : j'ai commencé par analyser le problème proposé, puis j'en ai extrait des problématiques afin de rédiger un cahier des charge. J'ai ensuite envisagé plusieurs possibilités envisageables, je les ai confrontés et j'ai fini par choisir la mieux adaptée à nos besoins. Enfin je suis passé à la réalisation concrète de l'ensemble de l'électronique. Grâce à ce projet, j'ai pu participer à l'ensemble de la chaîne de développement, en partant d'un concept, en passant par l'analyse des problèmes et la recherche de solution et en finissant par la réalisation du prototype et ses tests.

Ce projet a tout à fait répondu à mes attentes et ce sur plusieurs points. Tout d'abord parce qu'il rassemblait mes deux passions : le modélisme et l'électronique. De plus ce projet étant nouveau, il m'a permis de concevoir un nouveau système, une activité que je recherche tous les jours dans différentes réalisations personnelles. Enfin il m'a permis d'en apprendre davantage sur le travail en équipe. En effet jusqu'à présent dans beaucoup de travaux en groupe, les différents membres avaient le même niveau de responsabilité. Ici ce n'était pas le cas : je faisais parti d'une équipe avec une hiérarchie. Bien qu'autonome sur la réalisation de l'électronique, j'ai tout de même respecté cette hiérarchie en présentant les grands axes de mes développements à mes encadrant et en les justifiants afin qu'ils soient approuvés. Cette expérience est à mon sens très importante pour un futur ingénieur. Nous devons savoir faire preuve d'initiative et de prise de position, savoir défendre notre travail mais aussi savoir écouter et tenir compte des remarques et des arguments des autres pour parvenir à un résultat optimum.

Bibliographie

Livres :

- **Programmation en C des PIC :** Auteur : Christian TAVERNIER
  Éditeur : DUNOD

  Ce livre m'a servi de référence pour la programmation des PIC.

- **PIC CMU C Compiler :** Auteur : CCS

  Manuel de référence du compilateur PICC CCS, il m'a permis de résoudre les problèmes liés au compilateur.

Sites Internet :

- **http://forum.xda-developers.com/index.php:**
  Forum dédié au compilateur de CCS. C'est sur ce forum que j'ai pu trouver les explications aux problèmes rencontrés avec le compilateur.
- **http://forum.xda-developers.com/index.php :**
  Forum de développeurs pour téléphones portables et PDA. C'est au travers de ce forum que j'ai acquis mes connaissances en PDA.
- **http://www.abcelectronique.fr :**
  Ce site met à disposition de ses membres un très grand nombre de documents techniques sur les composants électronique.
- **http://www.microtransat.org :**
  Site officiel du challenge Microtransat. Toutes les informations concernant ce challenge y sont données (règlement, participants…).
- **http://paginas.fe.up.pt/~jca/wrsc/:**
  Site officiel du « World Robotic Sailing Championship ». Toutes les informations concernant ce concours y sont données. Ce site présente les différents participants et explique le type d'épreuves qui sera donné.
- **http://www.gpss.force9.co.uk/autop.htm :**
  Site réalisé par un passionné. Il a déjà réalisé plusieurs bateaux et nous fait part de son expérience et des différentes techniques qu'il utilise. Ce site est une référence pour l'ensemble des participants.

Mailing list :

https://lists.sourceforge.net/lists/listinfo/microtransat-general

Cette diffusion générale nous permet de discuter avec l'ensemble des participants sur les différents problèmes rencontrés et faire par de nos expériences. Les discutions se font en Anglais.

English presentation

Microtransat challenge

The challenge:

The "Microtransat challenge" is an international challenge which consists of a transatlantic race of fully autonomous sailing boats. Nobody has already done it. That is why it is a real challenge.

The aim of this challenge is to stimulate the development of autonomous boats. In the future, such automatic sailing boats could be able to collect an unlimited number of data of measures from world's lakes and seas without other energies than solar energy, what could allows them to survey or to map zones alone for a long times.

Microtransat in ENSIETA:

A Microtransat team was created this year in ENSIETA by Luc JAULIN. As it is a new project, we are not able to take part of this challenge yet: we have no boat and we have any experience. That is why we chose to take part of another competition: the « World Robotic Sailing Championship ». The aim of this competition is to meet other Microtransat challengers. In this way, we could share ideas and results of our experiments. During this competition we will test our boat in less harsh environments. Normally we will be ready for this summer.

My industrial project's objectives:

My industrial project was created in this context, and presented several objectives:

- The first was to help to start this project, what means to create the team and to write specifications.
- Then, I was personally in charge of the realisation of the electronic platform.

The electronic part:

We looked for several solutions such as computer or PDA. But we finally chose a solution of micro-controllers' base. Further to this choice, I chose the several components to realize the electronic platform for our autonomous sailing boat.

According to our specifications, we needed:

- a micro-controller such as PIC
- a GPS receiver, in order to know the position on earth
- an electronic compass to know the angle of the boat, which is not necessary the same as the speed vector)
- servomotors with servo controllers to command the sail and the direction
- A HF receiver, which could control the boat manually and if necessary

For the moment we do not have Weather vane: we only save the direction of the wind before the tests. We hope that the wind will not turn during the tests.

All these components are installed in a waterproof tin. Thanks to this system, even if the boat overturns, the electronic is protected.

The software part:

To control the system in autonomous mode, I wrote in C the software part which is loaded in the microcontroller. This software must be very reliable in time. That is why I chose to use no history and calculate no route. The basic software can control the boat and position it in the right way. Besides I added several logic circuits: there is a logic to control the route according to the angle of wind (we can not go directly back up the wind). There is also a logic to bypass the buoys: we have to pass by the right side of them for the competition. Both logics change the direction of the boat and send the information concerning this new direction to the basic software.

Tests:

The tests with the boat are very difficult to put into practise. First of all because we need a pool, but also because if there is a problem during the tests, we can not directly stop it, and stop de boat. That is the reasons why I chose to do most of the tests with a rolling platform. Such a platform allowed me to do my tests in ENSIETA and when there were problems, I could recover the robot and correct them.

Conclusion:

At the present time the electronic part is tested and works on the rolling platform. We do not have already tested it with the boat but it is planned. In the future we will work on the logics to find a better intelligence.

ANNEXES

MANUEL D'UTILISATION

I) Programmation du PIC
II) Programmation de l'EEPROM

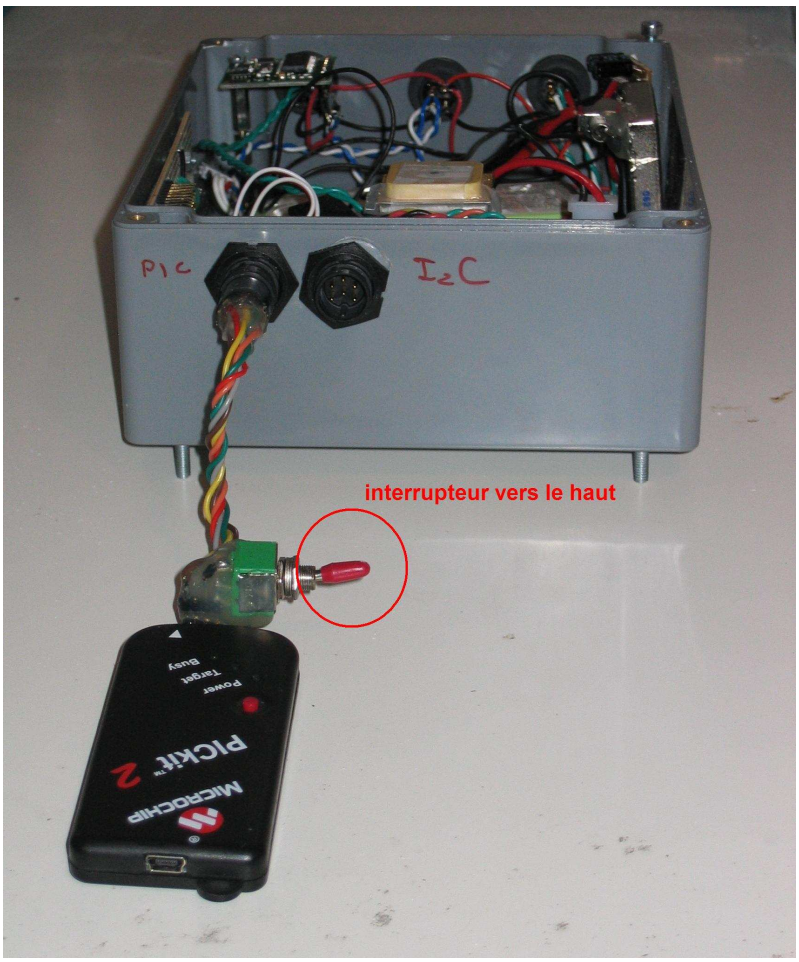
Matériel nécessaire :

- un PICkit2 et un câble USB
- le logiciel PICkit 2 v2.60 avec les drivers
  (Nécessite l'installation de « microsoft net framework » V2)

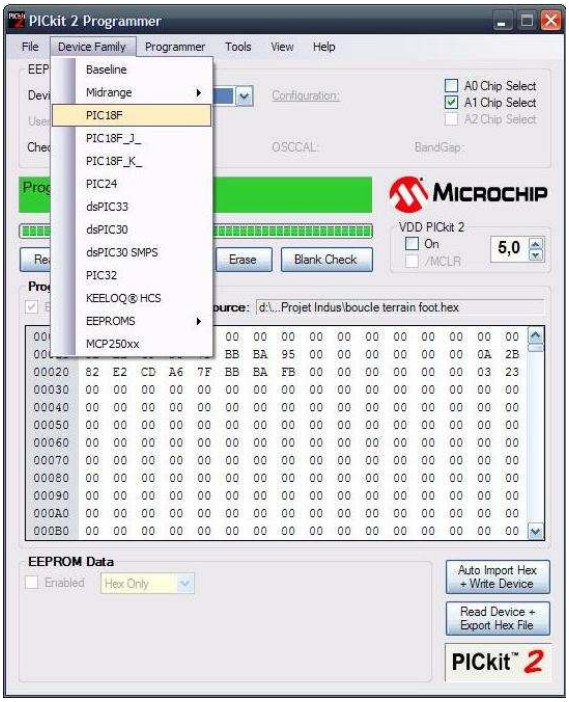
- le câble de liaison pour le boîtier étanche
- le boîtier étanche
- un fichier .HEX contenant le programme compilé

I) Programmation du PIC

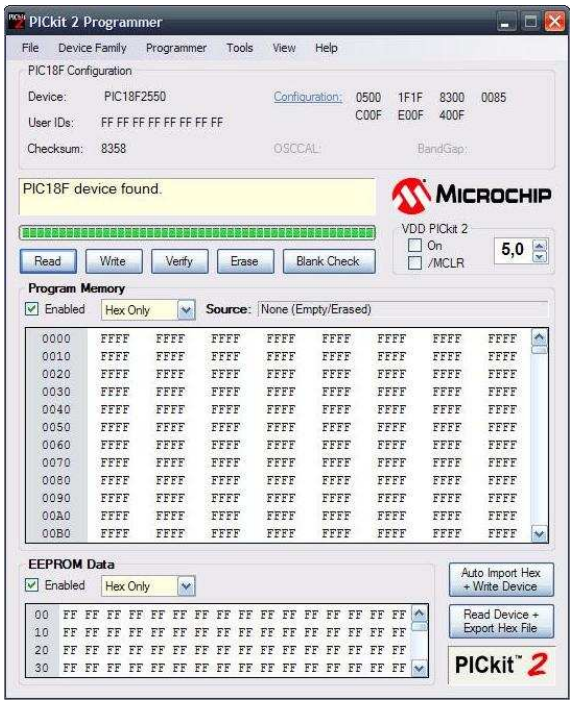
1) Connecter le PICkit2 à l'ordinateur et au boîtier étanche
2) Basculer l'interrupteur du câble vers le **HAUT** :



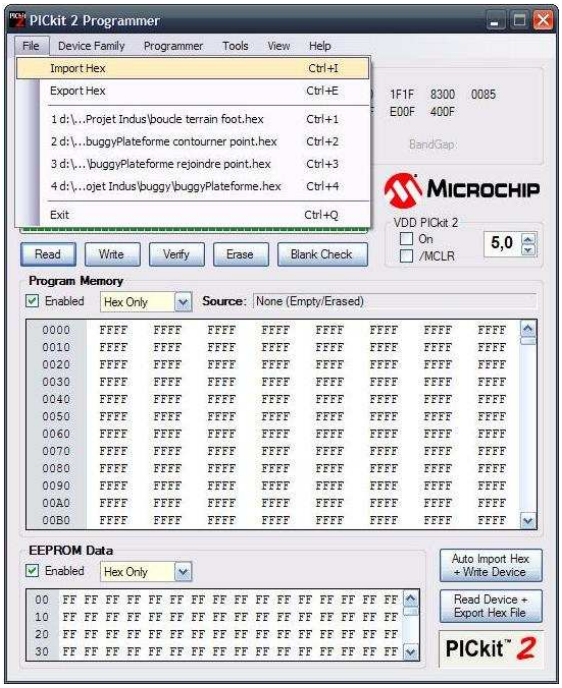
3) Lancer le programme PICkit 2 v2.60
4) Aller dans **Device Family** et sélectionner **PIC18F**



Une fois le PIC détecté, le programme indique :



5) pour charger un programme, aller dans **File** et sélectionner **import .Hex** :

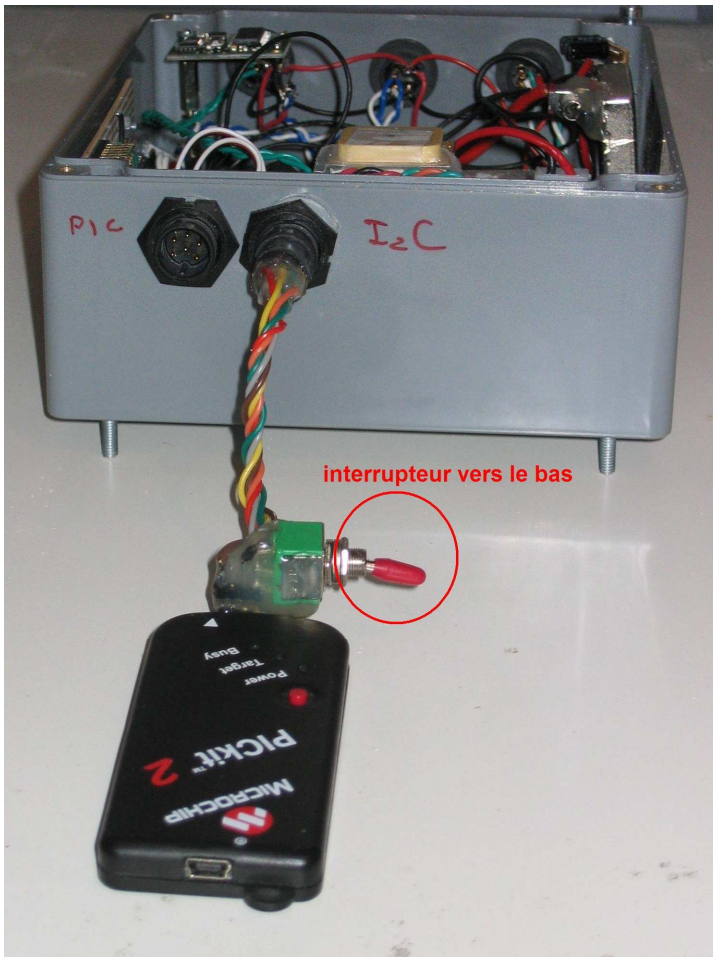


Après avoir chargé le fichier, cliquer sur **Write** pour le charger dans le PIC.

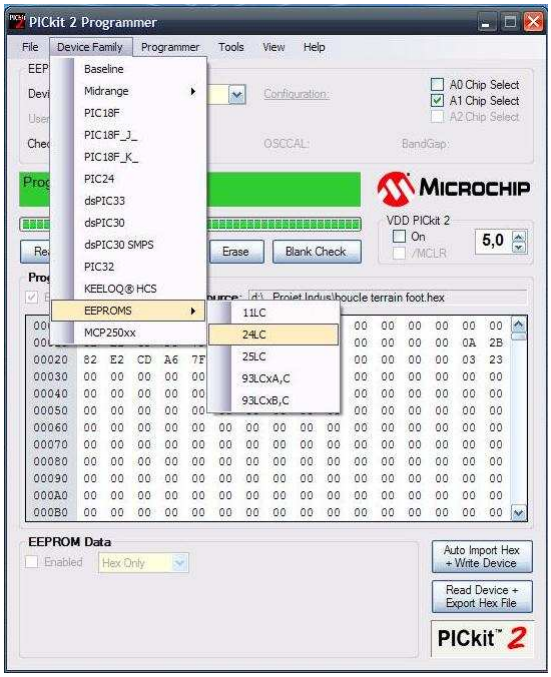
Si l'opération a réussi, le message **programming successful** apparaît sur font vert.

II) Programmation de l'EEPROM

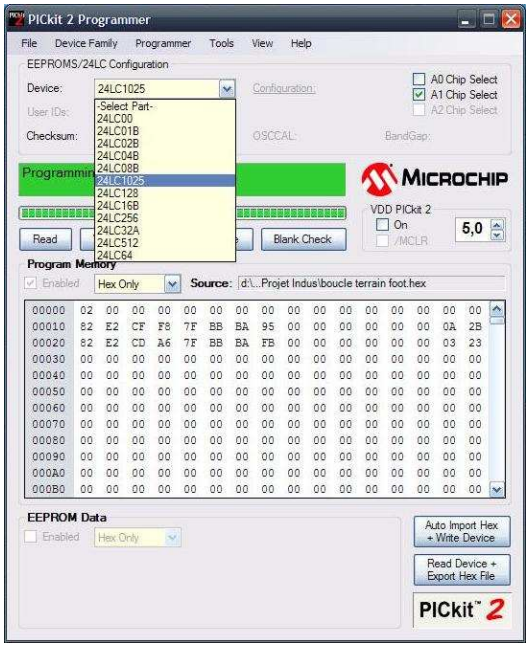
1) Connecter le PICkit2 à l'ordinateur et au boîtier étanche
2) Basculer l'interrupteur du câble vers le **BAS** :



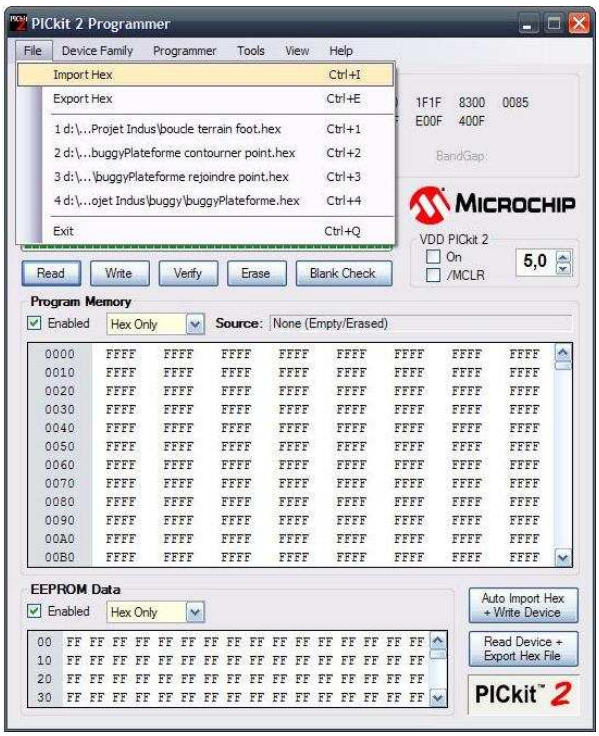
3) Lancer le programme PICkit 2 v2.60
4) Aller dans **Device Family** , **EEPROMS** et sélectionner **24LC**



Dans **Device**, sélectionner **24LC1025,**
Cocher **A1 Chip Select,**
Décocher **A0 Chip Select**



5) pour charger un programme, aller dans **File** et sélectionner **import .Hex** :



Après avoir chargé le fichier, cliquer sur **Write** pour le charger dans la mémoire.

Si l'opération a réussi, le message **programming successful** apparaît sur font vert.

Schéma de la carte de développement :

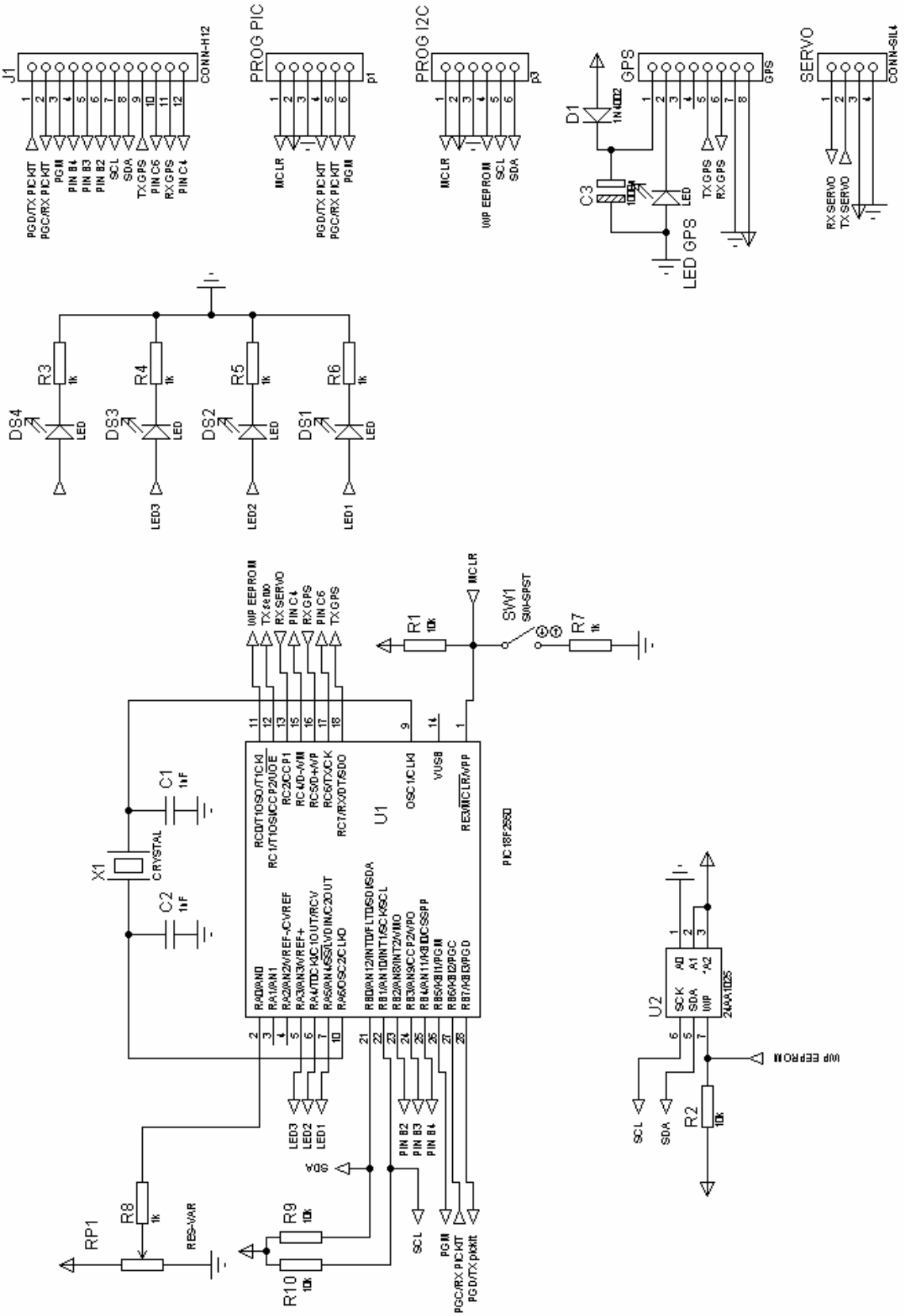
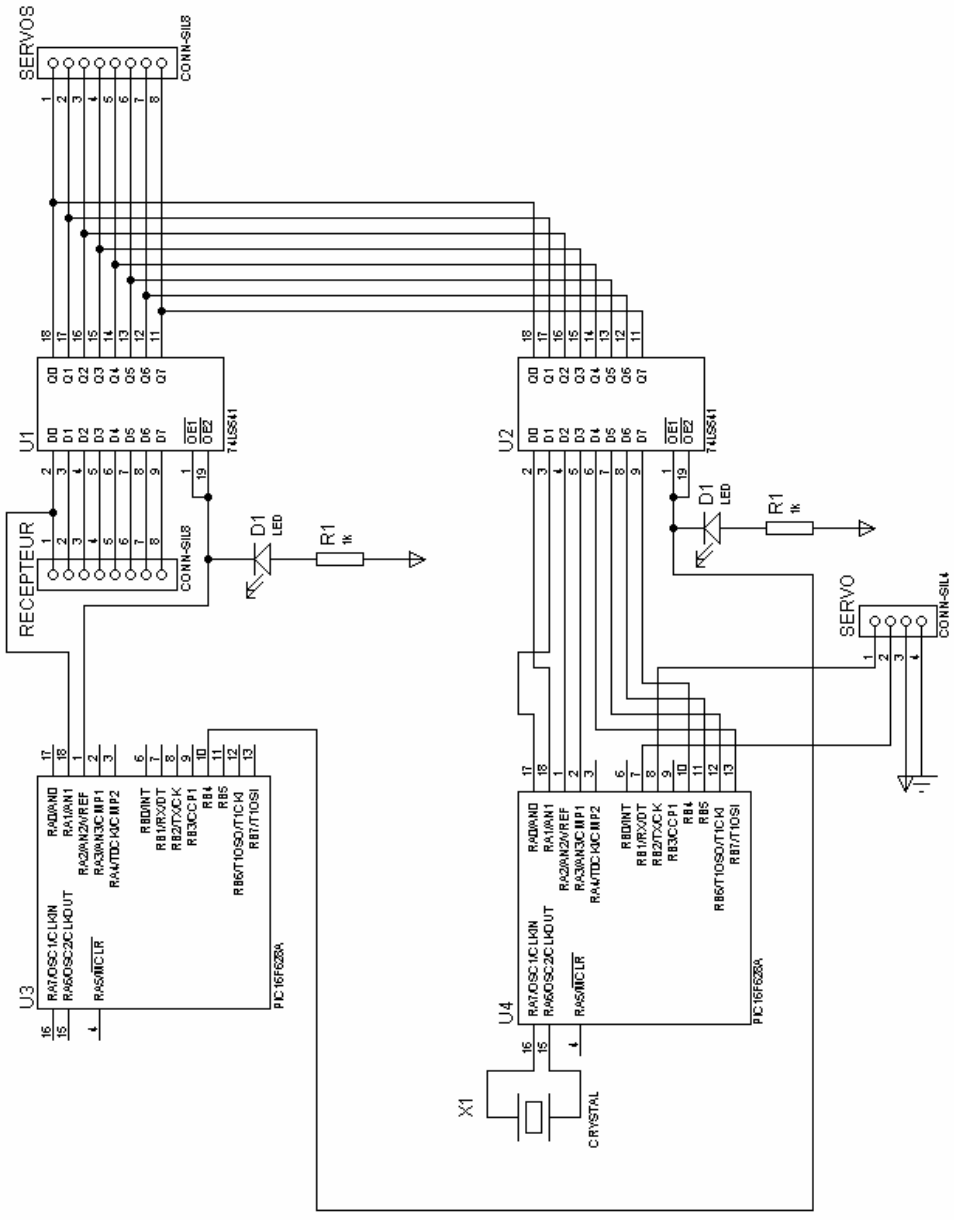
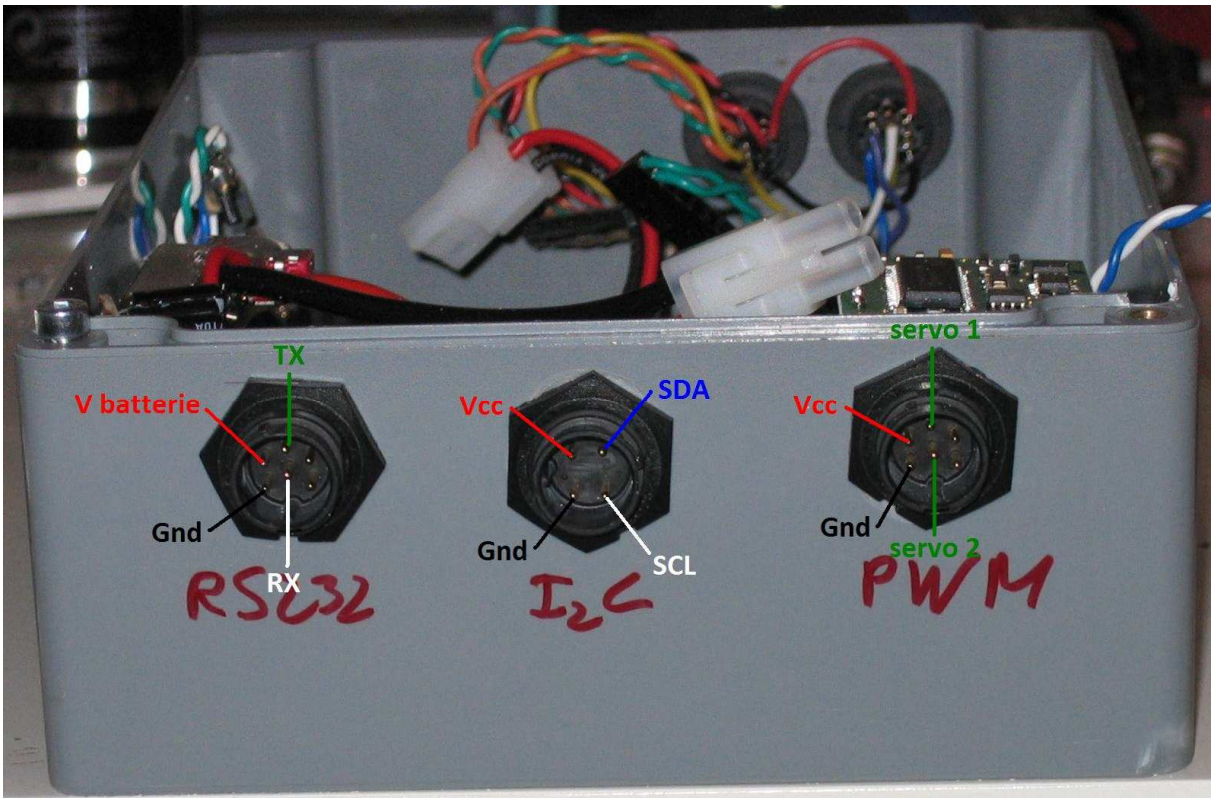


Schéma de la carte de commande des servomoteurs :

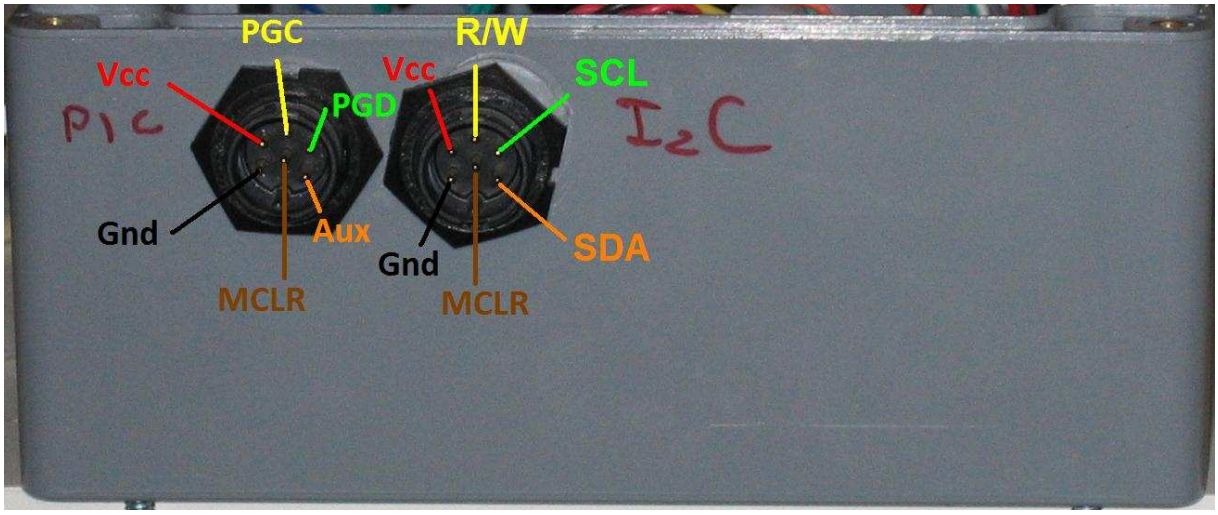


Câblage des connecteurs étanches :

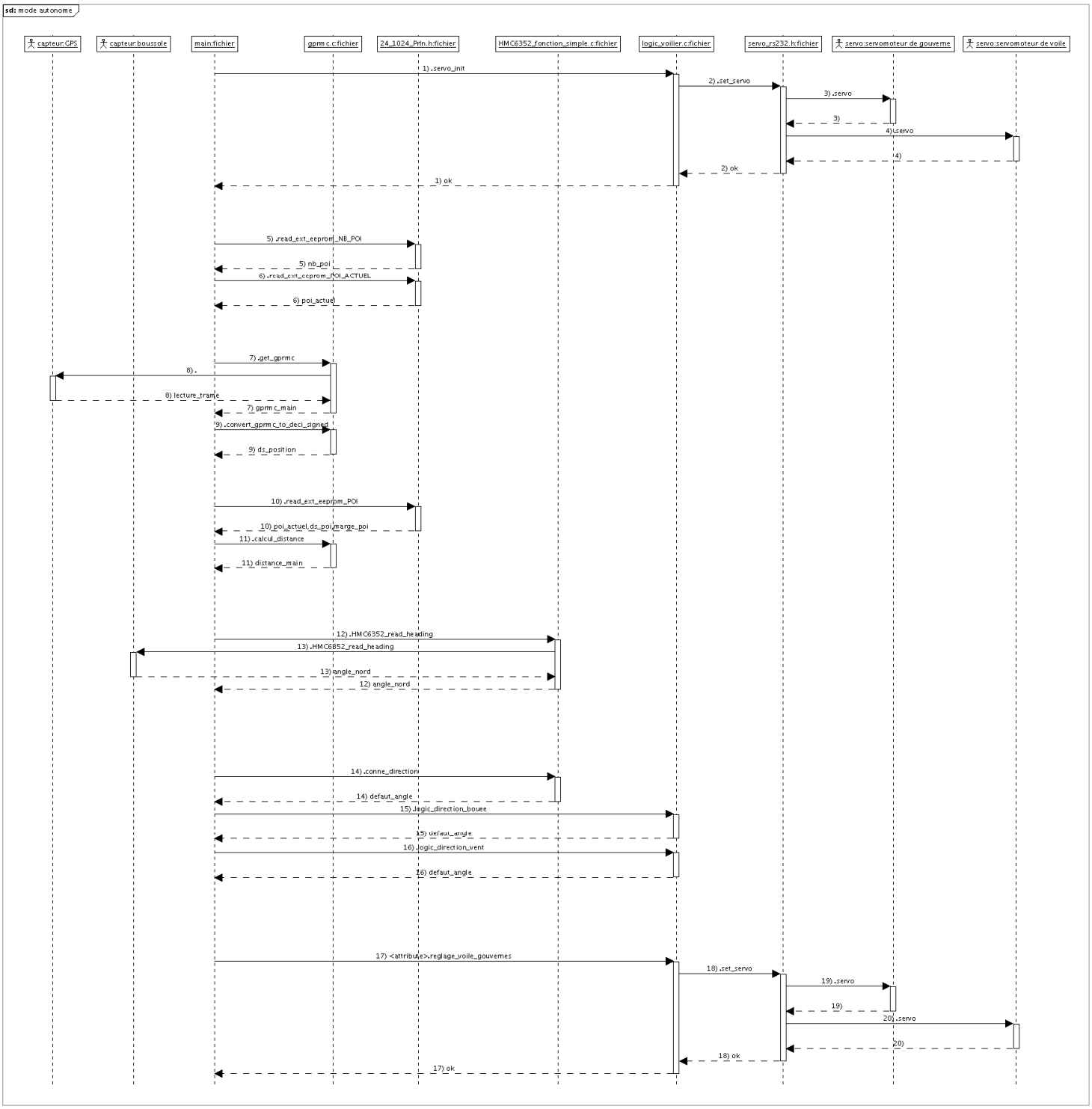
Face avant :



Face arrière :



Déroulement de la boucle principale du programme :



Algorithmes des logiques :

```
//#define rayon 20
//#define angle_mort 450
//#define angle_de_bord 500
```

Logique de contournement de la bouée :

```
#separate
int16 logic_direction_bouee(int32 distance_bouee,int16 angle_direction_bouee,int1 sens_direct){
  float32 angle_diff,rapport;
  int16 angle;

  rapport=distance_bouee;            //transtipage int32->float32

  if(distance_bouee!=0)              //calcul de la direction de la tangente
    if(rapport>rayon)
      rapport=rayon/rapport;
    else
      rapport=1;

  angle_diff=asin(rapport)*1800;     //conversion rad->0.1deg
  angle_diff/=PI;

  if(sens_direct)                    //choix du sens de passage
    angle = 3600+ angle_direction_bouee + angle_diff;
  else
    angle = 3600+ angle_direction_bouee - angle_diff;

  angle %=3600;
  return angle;
}
```

Logique de gestion du vent :

```
#separate
int16 logic_direction_vent(int16 direction_a_suivre,int16 direction_du_vent,int16 direction_du_bateau,int1 sens_direct){
  int1 vent_a_babor, vent_a_gauche;           // vent à gauche
  int16 angle_diff_dir, angle_diff_bateau, angle_temp;

    //valeur absolue de l'angle entre le vent et la direction
    if(direction_a_suivre > direction_du_vent){
      angle_diff_dir = direction_a_suivre - direction_du_vent;
      vent_a_gauche = 1;
    }
    else{
      angle_diff_dir = direction_du_vent - direction_a_suivre;
      vent_a_gauche = 0;
    }

    if(angle_diff_dir > 1800){
      angle_diff_dir =3600 - angle_diff_dir;
      vent_a_gauche++;
    }

  //valeur absolue de l'angle entre le vent et le bateau
    if(direction_du_bateau > direction_du_vent){
      angle_diff_bateau = direction_du_bateau - direction_du_vent;
      vent_a_babor = 1;
    }
    else{
      angle_diff_bateau = direction_du_vent - direction_du_bateau;
      vent_a_babor = 0;
    }

    if(angle_diff_bateau > 1800){
      angle_diff_bateau =3600 - angle_diff_bateau;
      vent_a_babor++;
    }

  if(angle_diff_dir>angle_de_bord)     // pas de bord a faire
    return direction_a_suivre;

  if(sens_direct){                          // bord en partant vent a droite
    if((angle_diff_dir>angle_mort-50)&&!(vent_a_babor)){      // si le bateau est vers la bouée et dans un petit angle
      return direction_a_suivre;
  }
    angle_temp = 3600+direction_du_vent+angle_mort;
    return angle_temp %= 3600;
  }
  else {                                    // bord en partant vent a gauche
    if((angle_diff_dir>angle_mort)&&(vent_a_babor))           // si le bateau est vers la bouée et dans un petit angle
      return direction_a_suivre;

    angle_temp = 3600+direction_du_vent-angle_mort;
    return angle_temp %= 3600;
  }
}
```

Les différentes liaisons entre les éléments: