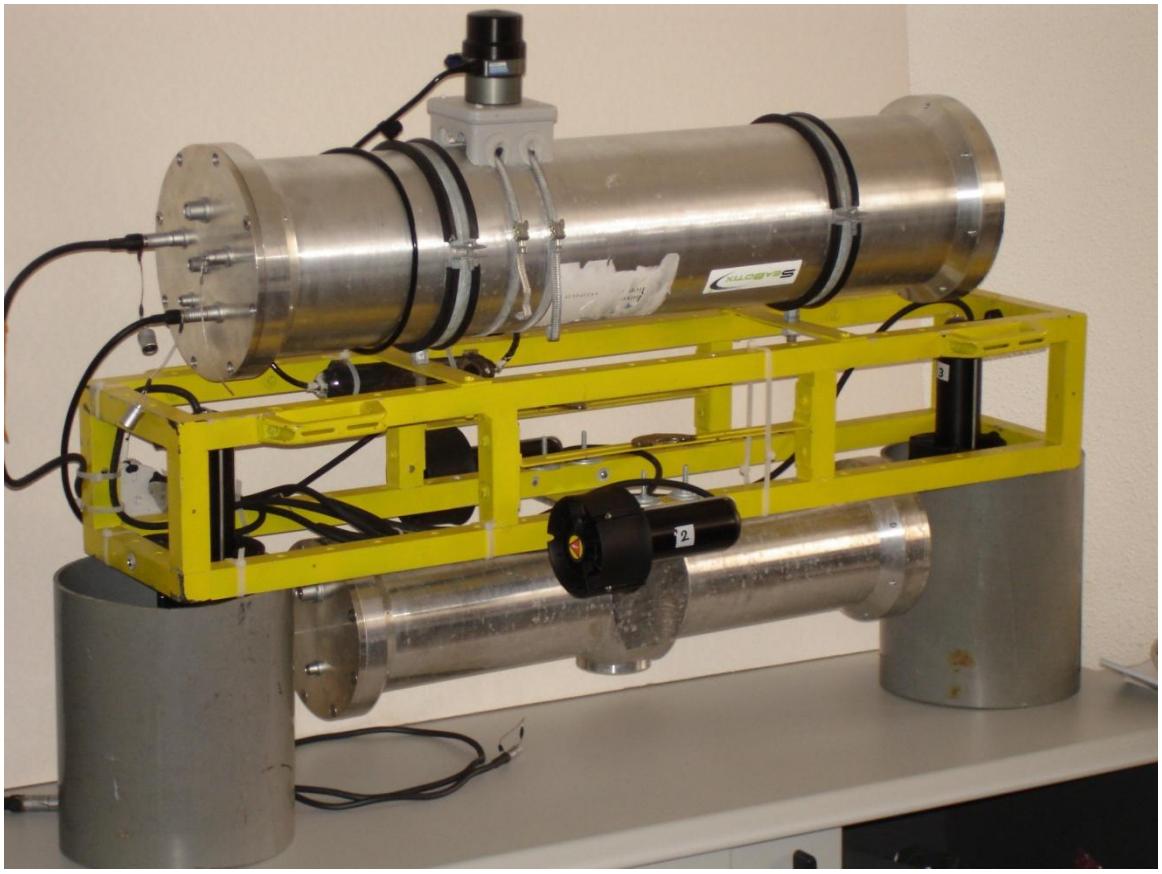
01/06/2009



I3S

TETARD

Student Autonomous Underwater Challenge

Alexandre Emanuelli / Cyril Tisserand

SOMMAIRE

INTRODUCTION

Le TER que nous avons choisi s'inscrit dans le cadre du Projet Têtard, ce projet ayant pour objectif la réalisation d'un sous-marin autonome. Celui-ci participera au concours Européen SAUC-E (Student Autonomous Underwater Challenge – Europe) auquel participent de nombreuses équipes issues de différentes universités. La principale caractéristique du sous-marin étant d'effectuer une série d'objectifs sans aucune intervention extérieure.

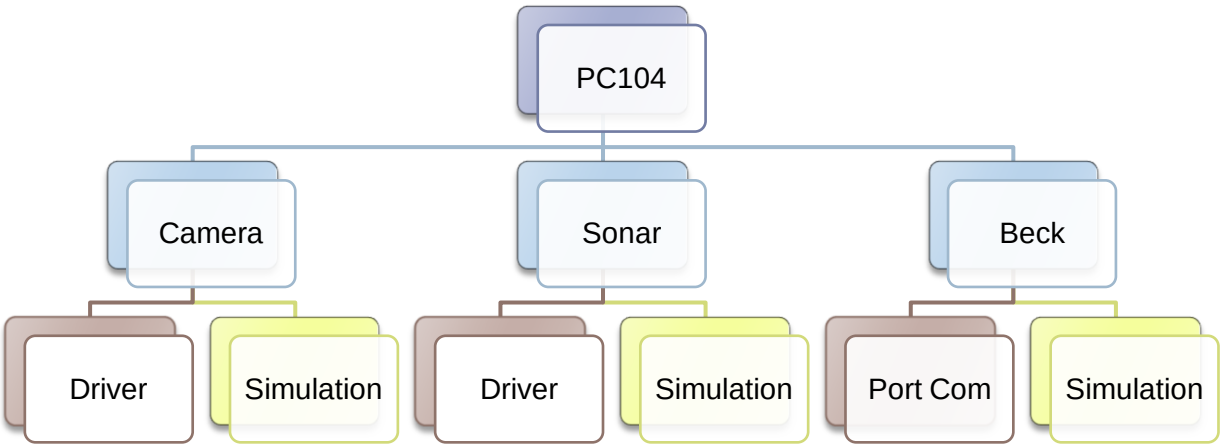
Au cours du développement de ce projet, nous allons développer tout l'ensemble du programme qui permettra la réalisation du parcours de façon automatique. Ce programme interagira avec la carte Beck ainsi qu'avec les différents capteurs externes.

Tout au long de ce rapport, nous détaillerons les objectifs que nous nous sommes fixés à notre arrivée, pour ensuite vous détailler l'aspect matériel et logiciel dans lequel nous avons du évoluer. Puis nous expliquerons quelles ont été nos tâches, comment celles-ci ont été réparties et les différents problèmes auxquels nous avons pu être confrontés.

LES OBJECTIFS

Au cours de ce TER, nous avons choisi de reprendre un maximum de chose qui avait été fait l'an dernier. Il était intéressant de repartir de l'architecture de l'année précédente et de réutiliser certains des éléments. Les tâches étaient donc les suivantes :

- Récupérer et porter le code sur la dernière version de Microsoft Robotics Studio.
- Créer un environnement de simulation qui servira de support pour le développement.
- Créer un sonar simulé pour l'intégrer dans l'environnement de simulation.
- Réagencement de l'architecture afin d'avoir une séparation nette entre les composants matériels et le logiciels.
- Modification du contrôle autonome afin de le remplacer par un système d'automate à pile.
- Ajout d'un système de log en XML qui servira de trace d'exécution du programme, et qui pourra être remis au jury du concours à la fin de l'évolution.



- PC104 : Regroupe tous les éléments dédiés au traitement des informations provenant des webcams et du sonar, ainsi qu'à l'exécution de l'automate.
- Camera, Sonar et Beck sont les périphériques externes avec lesquels communique la PC104 (demande de trame sonar, d'images, et de modification de comportement).
- Les drivers : Ceux-ci sont en charge de la communication entre les périphériques physiques et le programme sur la PC104.
- Les simulations correspondent aux périphériques dans l'environnement de simulation au travers d'un comportement émulé.

ENVIRONNEMENT DE DEVELOPPEMENT

Nous allons maintenant exposer l'environnement dans lequel nous avons modifié l'architecture logicielle en place.

LOGICIEL UTILISE

MICROSOFT ROBOTICS STUDIO 2

Microsoft Robotics Studio est une plateforme de développement dédié à la robotique. Il fonctionne selon le principe des Web services, c'est-à-dire que tous les services (module) transmettent des messages par protocole http et plus particulièrement par XML. Ce SDK (Software Development Kit) intègre aussi une bibliothèque assez vaste de services de simulation (camera, infrarouge …). Nous serons probablement amener à créer de nouveaux services dont nous pourrions avoir besoin lors de la réalisation de l'environnement de simulation.

Le runtime de Microsoft Robotics Studio est composé de deux modules :

- CCR pour *Concurrency and Coordinate Runtime*, Gere l'exécution asynchrone et parallèle des divers éléments de l'application Robotics
- DSS pour *Decentralized Software Services*, est un service léger qui combine les aspects traditionnelle de l'architecture et des Services Web, procurant une grande flexibilité tous en restant simple d'utilisation. Pour écrire des applications de robotique.

Microsoft Robotics Studio intègre un environnement de simulation générique basé sur le moteur de rendu 3D AGEIA PhysX.

De plus celui-ci intègre un écran de control pour visualiser l'état de la simulation ainsi que de démarrer et / ou arrêter des services

Microsoft Robotics Studio oblige les développeurs à signer les librairies et service utilisé. Ainsi les services tiers utilisant cette dll et/ou service sont sur de la validité et authenticité de celui-ci.

VISUAL STUDIO 2008

Microsoft Visual studio est un IDE permettant d'utilisé les fonctionnalités du Framework .NET. Microsoft Robotics Studio créant les services, Microsoft Visual Studio nous permet de modifier le code du service créé.

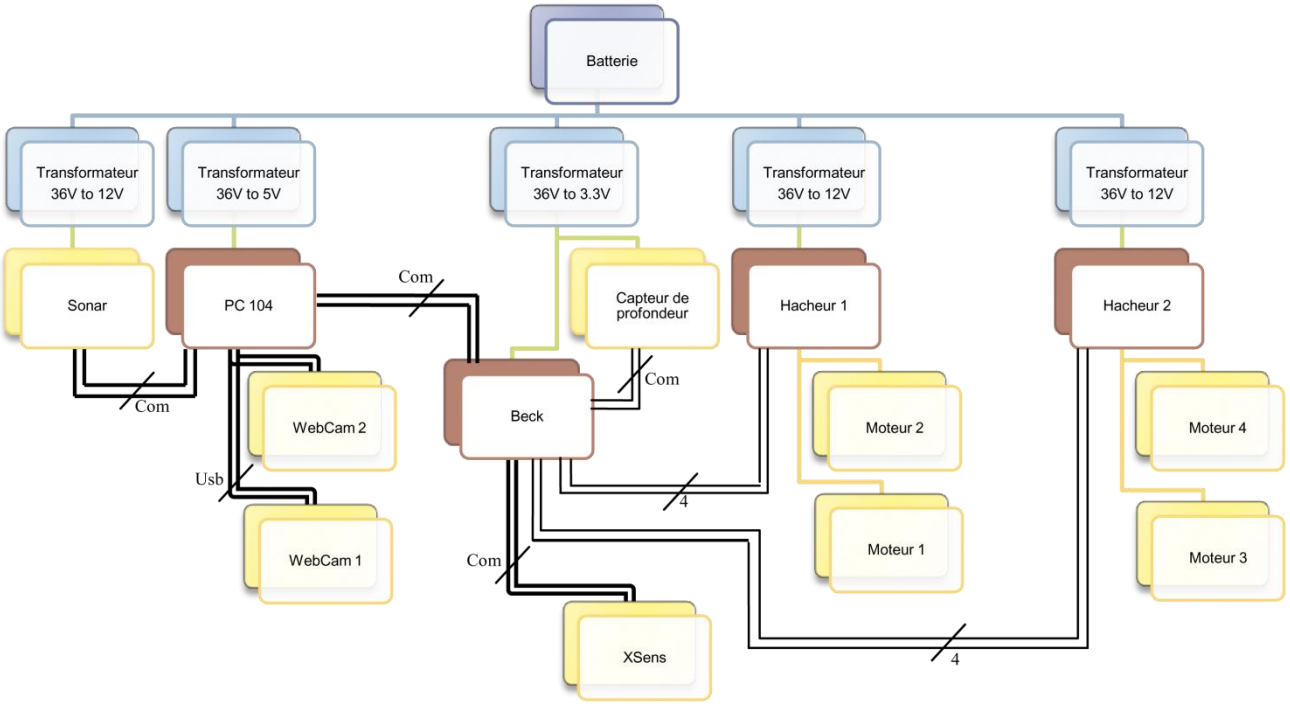
Nous utiliseront uniquement l'IDE de programmation en C# car le code des années précédente a été codé dans ce langage et que nous souhaitons réutilisé la majorité du code.

C#

C# est un langage de programmation impératif orienté objet à typage fort. Créer dans le but d'utiliser toute les possibilités du Framework .NET. Très proche de java de part la syntaxe, C# intègre en plus, comme C++, la surcharge d'opérateur.

ARCHITECTURE MATERIELLE

L'architecture existante du projet nécessitait la communication entre la PC104 et différents périphériques (webcam, sonar, Beck). L'on va donc détailler par la suite les différents éléments avec lesquels nous avons du interagir pendant le développement.



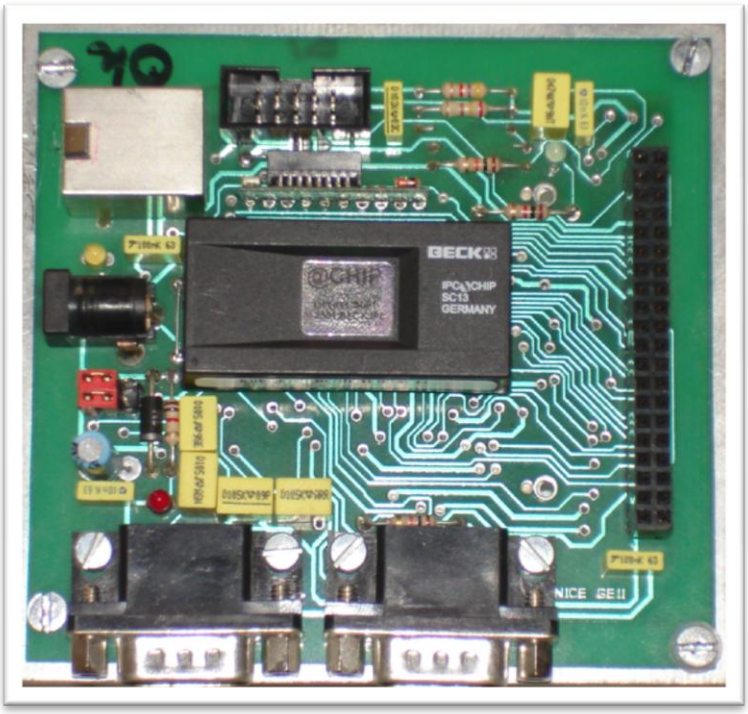
- Représente les différents périphériques du sous-marin, à la fois les capteurs et les éléments de déplacement.
- Tous les composants qui régissent le pilotage du sous-marin (du contrôle des moteurs, à la détection des objets grâce au sonar et aux webcams).
- Les éléments étant en charge de la découpe et de la répartition du courant au sein du sous-marin.
- La source principale d'alimentation du sous-marin. Situées en bas de celui-ci afin d'accroître sa stabilité avec un centre de gravité très bas.

PC104

La carte PC104 est de la marque VersaLogic et de type Cheetah. Elle est équipée d'un processeur cadencé à 1.8Ghz qui a été diminué à 1.6Ghz pour éviter des problèmes d'alimentations. Elle est le maître d'orchestre du sous-marin, sont rôle est de diriger le sous-marin. Pour cela elle possède un automate à état ainsi que un driver pour communiquer avec toutes les pièces du sous-marin, et plus particulièrement avec le microcontrôleur Beck afin d'y envoyer les consignes de déplacement sur les 3 axes.

MICROCONTROLLER BECK

Le microcontrôleur SC13-LF de la marque Beck est un microcontrôleur temps réel. Il a pour but de contrôler et modifier si besoin la profondeur, l'inclinaison et la direction du sous-marin. Pour cela le microcontrôleur dispose d' un accès aux contrôleurs des moteurs (les hacheurs), ainsi qu'au capteur de profondeur et à la Xsens.



MOTEUR

Au nombre de 4, ces moteurs ont été créés par Seabotix. D'une puissance nominal de 80 watts, 2 sont alignés horizontalement et règlent la vitesse de déplacement ainsi que la rotation et 2 autres, disposés verticalement permettent de contrôler la profondeur. Les moteurs sont généralement manipulés par paire.

HACHEUR (MOTEUR)

De marque Lextronic, le hacheur MD22 adapte la tension à délivrer aux moteur en fonction de l'entrée numérique de 4 bits fournie par le microcontrôleur.

TRANSFORMATEUR (MOTEUR) 36VDC TO 12 VDC

Les transformateurs régulent la tension pour éviter de surcharger les hacheurs et les moteurs car la tension de sortie des batteries est de 36VDC, qui bien trop élevée pour ceux-ci.

TRANSFORMATEUR (BECK, PC104) 36VDC TO 3,3VDC, 5VDC, 12VDC

Sur le même principe que les transformateurs des moteurs, ceux-ci servent à réguler la tension pour quelle convienne aux différentes périphériques, 5V pour la PC104, 3.3V pour la carte Beck et 12V pour le sonar. Il sert aussi de bus de communication entre la carte Beck et l'ensemble des périphériques (capteur de profondeur, Xsens et les hacheurs des moteurs).

CAPTEUR XSENS

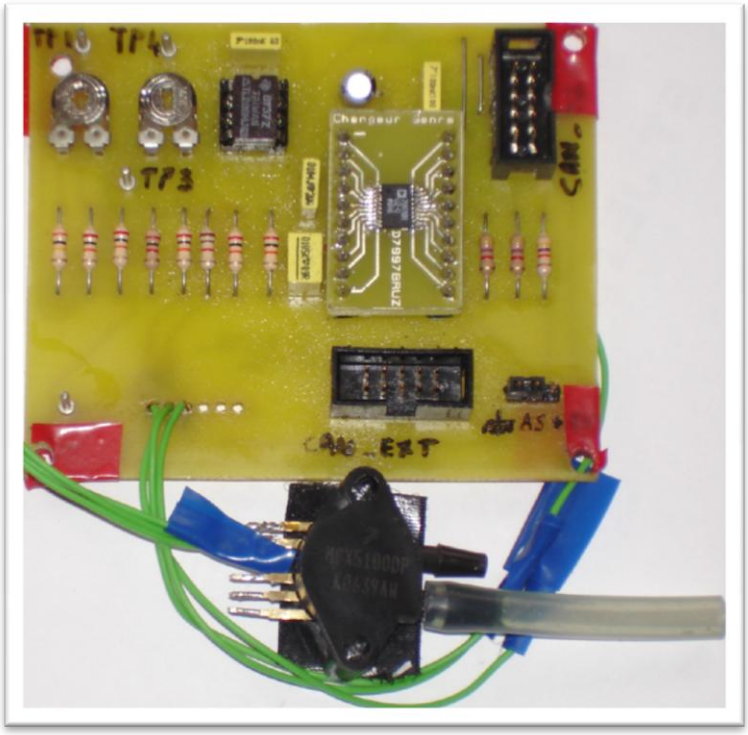
Achetée cette année la XSens dispose d'un compas de précision comprenant des capteurs inertiels de type MEMS (micro-électromécaniques) ainsi que d'une centrale d'altitude miniature. La XSens remplace désormais le gyroscope et l'accéléromètre poussif et vieillissant. Elle va aussi, à terme compléter le capteur de profondeur, peu précis.

SONAR

De marque Tritech, le Micron est le plus petit radar de haute intensité à compression d'impulsion. Il nous permet de détecter les objets du bassin ainsi que les bords. Son utilisation doit être limitée au strict minimum, due a sont très long temps de scan. Lors d'un scan complet (360°) avec une qualité de sortie plutôt bonne, il faut alors compter une bonne minute. Le scan nécessite d'ailleurs l'arrêt du sous-marin afin d'éviter le bruit sur l'image qui pourrait alors apparaître lors d'un scan en déplacement.

CAPTEUR DE PROFONDEUR

Fait lors d'une année précédente par le département GEII de l'IUT de Nice, celui-ci a une précision de 10 bit. L'étalonnage ayant été fait manuellement sa marge d'erreur est tout de même importante, de l'ordre de 20 cm.



WEBCAM

2 Webcams sont embarquées sur Têtard, de marque Logitech et Labtec. La première est située à l'avant tandis que l'autre en dessous du sous-marin. Elles servent, grâce à un traitement du signal, à détecter les objets proches. Elles sont préférées au sonar pour les objets à courte distance ou aux objets lumineux. Elles sont généralement sollicitées lors des phases d'approche finales.

ORGANISATION

N'étant que 2 membres à travailler sur le programme de contrôle autonome du sous-marin, plus particulièrement sur la carte PC, nous avons dû diviser notre travail de manière à ne pas interférer l'un sur l'autre.

Chaque début de semaine consistait à faire le point sur ce qui allait être réalisé durant celle-ci. Cela permettait de mettre en avant les différents problèmes que l'on serait amené à rencontrer. Ensuite, chaque vendredi nous faisions à nouveau le point sur ce que chacun avait réalisé, et procédions à quelques tests afin de nous assurer de la validité du morceau de code avec le reste de l'application.

Afin de travailler plus efficacement, nous avons mis en place un SVN qui nous permettait alors de pouvoir à tout moment rapatrier le code nouvellement produit.

Pour nous aider dans certain transfère de donné volumineux (par exemple la récupération de Borland C++ 5) nous disposions d'un FTP.

Un Wiki a été mit en ligne pour pouvoir au fur et à mesure écrire les objectif atteint ainsi que les problèmes rencontré durant leur réalisation.

REPARTITION DES TACHES

Nous étions trois a travaillé sur Têtard, 2 de l'université de Nice / Sophia Antipolis (Alexandre Emanuelli et Cyril Tisserand) et un autre venu de Grenoble (Mathieu Bailet).

Chef du projet Têtard, Mathieu Bailet c'est occuper de la modélisation du sous-marin, mais nous ne détaillerons pas plus sa partie qui n'est pas lié avec la notre.

Nous nous somme répartie les taches au fur et à mesure des avancements et en fonction de la disponibilité de chacun

LISTE DES EVENEMENTS IMPORTANTS

Date	Evènement
23/03	Debut TER
2/04	Fin des cours
15/04	Arriver du chef de projet
18/04	Obtention du code
8/05	Compilation sans échec du projet
29/05	Fin du TER

TACHE ALLOUE A ALEXANDRE

Date	Tache
15/04	Prise en main de Robotics
22/04	Installation de la carte SanDisk
13/05	Simulation du sonar
22/05	Environnement de simulation
29/05	Ecriture du nouveau protocole de communication avec la carte Beck + interface de test.

TACHE ALLOUE A CYRIL

Date	Tache
15/04	Prise en main de Robotics
18/04	Log XML
6/05	Portage
8/05	Réorganisation des services
15/05	Automate
21/05	Liaison des services entre eux et intégration automate
22/05	Intégration sonar simulé
29/05	

TRAVAIL EFFECTUE

CHANGEMENT DU MATERIEL

Le sous-marin utilisait un disque dur de 2 pouces ½ convenant parfaitement en termes de performances, mais qui consommait trop de courant. Ayant une carte SanDisk achetée l'année précédente pour remplacer le disque dur, nous l'avons installée à la place du disque dur.

Pour l'installer nous avons du récupérer le nom et la marque de la PC104 pour pouvoir obtenir la documentation technique. Une fois ceci fait, l'installation nous a juste demandé du temps. Il a fallut aussi réinstaller tout l'environnement de travail de Microsoft Robotics Studio.

PORTAGE

Notre première mission a été de porter le code de Microsoft Robotics 1 à Microsoft Robotics 2. Pour les driver le portage a été facile, car ceux-ci n'utilisaient aucunes bibliothèques spéciales de Microsoft Robotics, et seuls les changements appliqués sur les propriétés ont du être faites.

Pour les services plus complexes, le portage a été impossible, car la structure même de Microsoft Robotics a changé. Des bibliothèques étaient manquantes, et celles que l'on arrivait à changer n'étaient plus compatibles avec le code existant. Pour palier à ces problèmes, nous avons porté tous les drivers et les simulations des périphériques, mais tous les services de contrôle ont du être refait.

Nous en avons profité pour ajouter un automate à état afin de gérer les objectifs de mission ainsi qu'un system de log par XML.

LOG XML

Pour répondre aux exigences du concours ainsi que pour faciliter l'analyse des erreurs dans le cœur du programme, nous avons implémenté un system de log en XML.

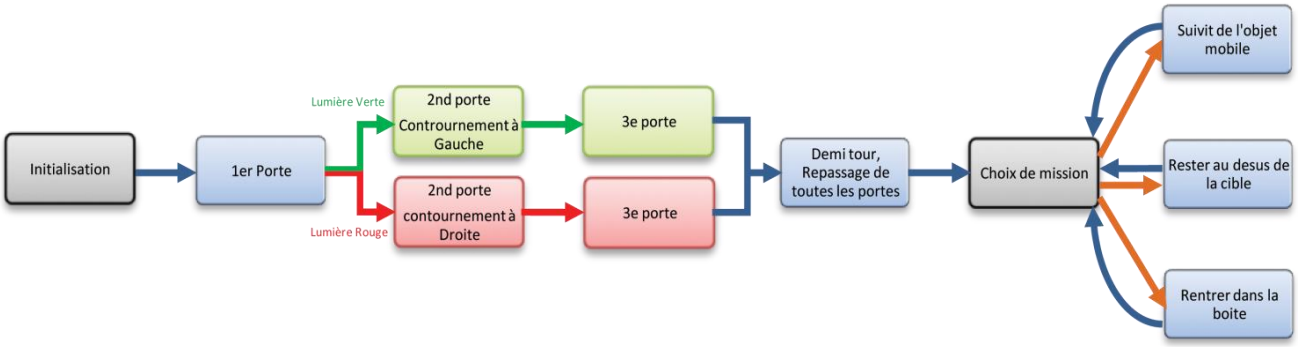
Le choix de la technologie a été porté à de grand débat, mais finalement les avantages qu'apporte le XML nous ont convaincu. Parmi les plus importants, la possibilité de traiter les données facilement pour un affichage et une clarté des informations écrites (grâce aux tags XML). De plus le format XML permet une réutilisabilité plus simple, car il contient à la fois la structure et les données. Il est donc très facile de concevoir un programme capable d'interpréter ce fichier afin de recréer l'évolution du sous-marin durant le parcours. En effet, chaque action du sous-marin (balayage sonar, les déplacements, les prises d'images, les erreurs, ...) sont mémorisées et enregistrées dans un fichier spécifié au moment de l'initialisation.

Ce système de log, offre en autre, plusieurs niveaux d'alerte, allant de la simple information d'action jusqu'à l'erreur d'exécution, en passant par les warnings.

L'AUTOMATE A ETAT

Puisque nous devions refaire les services de contrôle du sous-marin, nous en avons profité pour créer et intégrer un automate à états pour le piloter.

Les états correspondent aux différentes missions que le sous-marin doit suivre.



- Etat initial de l'automate ou état final. Le sous-marin est immobile en surface. Il configure l'ensemble de ses périphériques.

  Dans le cas de l'état final, l'automate se termine lorsque celui-ci a effectué l'ensemble des missions. Il entraine donc la remontée du sous-marin puis son arrêt progressif.

- Ensemble des objectifs uniques à atteindre.

- Succession d'objectifs à réaliser lorsque la lumière située sur la seconde porte est allumée en vert. Elle implique alors un virage à droite afin de se retrouver dans l'axe de la troisième porte.

- Succession d'objectifs à réaliser lorsque la lumière située sur la seconde porte est allumée en rouge. Elle implique alors un virage à gauche afin de se retrouver dans l'axe de la troisième porte.

- Désigne les missions qui ne peuvent être effectuée qu'une seule et unique fois mais dont l'ordre est défini aléatoirement.

Ne pouvant pas tester le sous-marin en conditions réelles à chaque modification, Microsoft Robotics Studio intègre un moteur de simulation, mais pour pouvoir simuler le sous-marin complet, il faut simuler tous les capteurs de ce sous-marin. Seule la simulation du sonar manquait des années précédentes, nous nous somme donc aussi attelé à la faire pour pouvoir tester le fonctionnement du portage. Ne disposant plus de l'environnement de simulation de SimplySim, nous l'avons remplacé par le notre.

LA SIMULATION DU SONAR

Un des points important dans la création de l'environnement de simulation du sous-marin est la réalisation d'un sonar simulé. Lors des années précédentes, ce point n'avait pas été traité, ou trop brièvement et ne retournait aucun résultat.

Afin de réaliser ce sonar simulé, il a fallut utiliser ce que l'environnement par défaut de Microsoft Robotics Studio offrait comme technologie. Le moteur graphique de l'environnement nous a permis d'utiliser le système de *ray tracing*. Cette technique est basée sur les rayons lumineux qui sont projetés sur les objets de la scène.

Dans notre cas, l'on a utilisé cette technique afin de détecter les objets de l'environnement. Chaque impact du rayon est alors enregistré (l'on mémorise la distance depuis la source jusqu'à l'objet). Si aucun objet ne se trouvait alors entre la source et la distance maximale définie, alors, c'est celle-ci qui faisait office d'objet distance. L'on peut alors constater un mur circulaire éloigné sur la trame simulée.

Cependant, le défaut majeur de cette trame est l'absence de bruit, car dans le cas de la simulation, la trame est réalisée à partir d'un faisceau lumineux, donc très précis, tandis que dans l'eau, la trame est issue du retour des ondes sonores sur les objets. Ce modèle est donc très satisfaisant dans le sens où il nous génère une trame sonar exploitable mais qui reste cependant très éloigné d'une trame sonar en milieu réel. Le traitement sur cette image reste nettement plus simple à traiter. Il se peut que lors de l'évolution en bassin, les résultats du traitement sonar soient défaillants.

ENVIRONNEMENT DE SIMULATION

L'environnement de simulation pour notre sous-marin est basé sur celui fournit par Microsoft Robotics Studio. Pour commencer il a fallu intégrer le nouveau sonar simulé et le configurer afin que ses angles de balayages soient les plus proches du sonar matériel. Le sonar simulé est donc paramétré afin de scanner les objets sur 360° avec un pas de 0,5°, ce qui nous permet d'obtenir 721 valeurs (l'on passe 2 fois sur l'origine). L'un des problèmes majeur à été de trouver une configuration adéquat afin d'obtenir un échantillon suffisant pour le traitement, mais aussi pas trop volumineux afin que le transit de celui-ci entre les services ne soit pas trop long. En effet lors des premiers essais, la trame sonar refusait de s'afficher.

Le sous-marin dispose aussi de 2 caméras distinctes qu'il a fallu à leur tour intégrer et paramétrer. Les 2 caméras retournaient au départ la même image malgré leurs orientations pourtant différentes. En fait chaque caméra préexistante dans Robotics Studio contenait des fonctions d'identification. Il a donc fallut reprendre une version minimale de se service et y ajouter les différentes fonctionnalités de manière incrémentale.

L'autre point fondamental était de modéliser l'ensemble des objets du bassin afin de pouvoir les intégrer. Les facteurs de taille ont été un souci. Chaque élément n'était pas proportionnel à la taille du sous-marin et les structures de gestion des collisions ne coïncidaient pas avec les objets, ce qui provoquait parfois des collisions sur des murs invisibles.

PROTOCOLE DE COMMUNICATION BECK

Durant cette dernière semaine, nous nous sommes attelés à la réalisation du nouveau protocole Beck. L'arrivée de la Xsens a nécessité l'ajout de nouvelles fonctionnalités. Le protocole de l'année précédente était incomplet et ne couvrait pas l'ensemble des fonctions requises pour le fonctionnement opératoire du sous-marin.

Pour cela, nous avons préféré nous inspirer du protocole de communication défini dans la Xsens, afin de préserver une cohérence dans les échanges de messages entre les différents composants (PC104, Beck et Xsens). Nous avons découpé les commandes en plusieurs niveaux allant des commandes de bas niveau (commande des moteurs individuellement), aux commandes de très haut niveau (comme avancer de 40mètres, tourner de 10°, plonger de 2mètres). Ces commandes de plus haut niveau sont intégralement gérées par la Beck qui grâce à ses boucles de contrôle permet de quantifier les déplacements. L'utilisation des commandes de bas niveau ne se fera donc jamais, malgré que nous disposions des informations que nous transmet la Beck.

Nous sommes donc en cours de conception d'une interface graphique afin de tester le contrôle dynamique et d'étalonner le modèle mathématique du sous-marin en piscine. Et par la suite de pouvoir tester le sous-marin en condition réelle en ayant le plus de chance de succès dans la réalisation des objectifs du concours.

PROBLEMES RENCONTRES

Nous détaillerons ici les divers problèmes que nous avons pu rencontrer tout au long de nos travaux sur le projet Têtard, ainsi que les solutions auxquelles nous avons eu recours lorsque cela était possible.

ABSENCE D'UN MEMBRE

Au début du TER, nous devions être 3 étudiants à travailler sur le projet, cependant un des autres membres de l'équipe étant difficilement joignable (n'a jamais répondu à un seul mail de notre part malgré les nombreux messages envoyés), nous sommes partis sur le principe que nous serions seulement deux étudiants à travailler sur le projet. Nous avons alors décidé de nous répartir les tâches en fonction des affinités et des dépendances afin de ne pas nous ralentir mutuellement.

LE CODE SOURCE

LA RECUPERATION

Notre coordinatrice de TER ne disposant pas de du code source écrit les années précédente, il a fallut reprendre contact avec les personne travaillant l'année dernière dessus pour pouvoir récupérer celui-ci. Nous avons malheureusement du attendre un certain temps pour que-ci nous le transmette.

INCOMPATIBILITE DU CODE SOURCE

Le code avait été compilé sous Microsoft Robotics Studio 1.1, et nous ne disposions plus que de la dernière version, c'est-à-dire la 2.0. Entre ces deux versions de nombreuses librairies ont été modifiées, déplacées ou ne fonctionnaient plus de la même façon (par exemple les http Post). L'outil de migration de Microsoft n'apportant pas les résultats escomptés, il a fallu l'effectuer à la main.

Pour garder le plus de code possible, les drivers ont été conservés mais les services de control ont été réécris. D'autres services ont été fusionnés et/ou réarrangés.

SIMPLYSIM

SimplySim est l'ancien environnement de simulation créé par l'entreprise du même nom. Malgré plusieurs demandes, il nous a été impossible de récupérer la totalité de cette librairie. Seulement quelques portions ont été récupérées et nous avons alors décidé de ne plus utiliser celle-ci. Suite à ça le code a été nettoyé pour ne plus faire appel à cette librairie et nous avons rajouté à nos objectifs la création d'un environnement de simulation.

L'ELECTRONIQUE

Entre temps, nous avons remis le sous-marin en route, et avons remarqué que le microcontrôleur SC13 de la carte Beck ne fonctionnait plus. Nous avons repris contact avec l'IUT pour récupérer un nouvel exemplaire de ce microcontrôleur. Heureusement, nous en avons retrouvé un dans nos réserve car notre contact à l'IUT n'avais aucune envie de nous aider bien que ceux-ci disposent aussi de fonds spécialement dédiés au projet têtard.

Lors de la réinitialisation du robot, il nous a clairement manqué un électronicien, afin de régler les problèmes électriques du robot (existant déjà l'année dernière mais non résolus pour les mêmes raisons). Car la PC104 redémarre de temps en temps sans aucune raison et ce n'est malheureusement pas la documentation manquante qui pourra nous aider à résoudre le problème. Ce problème existe toujours car nous n'avons pas les connaissances et les compétences nécessaires pour les résoudre. Il semblerait que néanmoins que la source du problème proviennent des transformateurs sur la carte de bus qui prennent une tension supérieure à celle délivrée par la batterie. N'ayant pas la tension minimale requise, celle-ci n'alimenterait donc pas suffisamment la carte PC.

CONCLUSION

Notre participation au projet Têtard nous à permis de découvrir et nous familiariser avec le monde de la robotique. Nous avons pu ainsi faire interagir nos programmes avec différents périphériques mais aussi d'autres programmes embarqués. Il a fallut au départ nous adapter et comprendre cette nouvelle architecture à base de services, bien que nous connaissions déjà C# et son environnement de développement. L'étude et la réalisation des tutoriaux de Robotics Studio ont été un point essentiel de notre apprentissage de ce nouvel outil. De plus, nous avons du faire preuve d'une grande autonomie tout au long du projet.

Durant toute la durée du TER, nous avons du faire face à de nombreux problèmes, qui en partie, n'étaient pas tous dans notre domaine de compétence. Mais nous avons su nous adapter et effectuer les recherches nécessaires afin de les corriger du mieux possible. Nous avons essayé au mieux de respecter le planning et la méthodologie que nous avions définie au départ pour faire progresser au mieux le projet. Les soucis majeurs rencontrés ne sont pas considérés comme pénalisant, dans le sens, où comme ceux-ci ont été résolus, ils ne poseront plus de problèmes les années suivantes.

Nous avons pu nous rendre des comptes des différences assez notables entre un environnement réel et environnement simulé. En effet, l'environnement simulé, offre un aspect complet mais très épuré de ce que l'on peut avoir dans la réalité. Cela s'est très bien affirmé dans la simulation du sonar, où l'image obtenue est d'une netteté qui ne pourra jamais être obtenue en situation réelle. Les données issues du traitement seront alors plus facilement exploitables. Une solution pourrait être d'intégrer un facteur de bruit, afin de détériorer volontairement les sources dans l'environnement de simulation afin d'avoisiner un environnement plus proche du cadre réel.

Cette année nous avions un étudiant qui s'est chargé de la programmation du composant de la carte Beck. Il a simplement suffit de mettre un protocole en commun afin de faire dialoguer la carte Beck et la carte PC104.

Finalement, nous avons pu atteindre l'ensemble des objectifs que nous nous étions fixés dans le temps imparti. Nous sommes même parvenus à mettre en place une interface de contrôle du sous-marin afin de réaliser le test du modèle dynamique du sous-marin. Il reste cependant encore du travail à fournir afin de parvenir à réaliser l'ensemble du parcours du concours.