POLYTECH
LILLE

2012
2013

COMPTE RENDU DE PROJET IMA4 ROBOT DE GRANDE TAILLE LE CENTAURE

DAVY RIBREAU (SA)

CEDRIC VANDERMEERSCH (SA)

POLYTECH LILLE – AVENUE PAUL LANGEVIN, 59655 VILLENEUVE D'ASCQ

Remerciements

Avant toute chose, nous tenons à remercier sincèrement nos tuteurs de projet, Messieurs LITWAK et REDON, qui nous ont fourni l'aide, l'assistance, et le matériel nécessaire à la bonne avancée de notre projet.

Leur disponibilité nous aura été précieuse tout au long de ce projet, tout comme leurs conseils avisés, et nous espérons leur donner entière satisfaction pour le rendu final de ce projet, dont le compte rendu fait partie.

Plus généralement, nous remercions le département IMA et l'école POLYTECH LILLE, qui nous enseigne et nous inculque les notions qu'un bon ingénieur se doit d'avoir.

Sommaire

Table des matières

Introduction

En notre qualité d'étudiants en quatrième année d'école d'ingénieur, à Polytech Lille, département Informatique-Microélectronique-Automatique (IMA), il nous a été proposé de réaliser un projet d'une importance conséquente quant à la qualité de notre diplôme, la validation de notre année, et la découverte de nouveaux horizons.

Le projet sur lequel nous avons choisi de travailler est le robot de grande taille, communément appelé Centaure. Ce dernier ayant été conçu, dans sa forme, par des étudiants du département Conception Mécanique (CM), afin d'être destiné aux étudiants IMA, il incarne la cohésion et la bonne entente qui règne entre les différents départements de l'école. Ainsi, plusieurs promotions d'IMA ont eu pour but d'améliorer les performances du Centaure, et c'est à nous de prendre le relais cette année - relais qui n'avait plus été transmis depuis quelques années, ce qui nous a particulièrement attiré quant au choix de ce projet.

Ainsi, l'objectif principal de ce projet est de faire revivre le Centaure ; les moteurs qui le composent doivent être commandés via une interface série. Il faudra dès lors réaliser un ordinateur embarqué de manière à rendre le robot parfaitement libre, pour recevoir les commandes de comportement. Ce robot doit également être commandé en coordination avec des informations 3D obtenues grâce à un système Kinect. Le but de cette manœuvre est de permettre au robot de naviguer dans le hall de l'école, en évitant les obstacles. Il pourra ainsi servir pour diverses applications faisant appel à de telles capacités (service lors de réception,...).

Vu la quantité assez conséquente de travail requise, le projet sera composé de deux binômes, mélangeant Systèmes Communicants et Systèmes Autonomes. L'un sera chargé de la partie contrôle, l'autre de la partie détection.

Dans ce rapport, nous préciserons les différents travaux qui auront été effectués, manière de constater l'avancement du projet. Ainsi, nous commencerons par présenter le robot Centaure de manière plus précise. Nous nous orienterons par la suite vers le côté travail, ce rapport traitant du côté commande, l'autre rapport présentant la partie KINECT. Nous évoquerons en conclusion le bilan que nous tirons de ce projet, l'état final du Centaure, et de possibles ouvertures pour nos successeurs.

Présentation du centaure

A - Description du centaure

Le robot Centaure est issu d'un partenariat entre les départements CM et IMA. Il est composé de deux moteurs + roues prélevés sur un fauteuil roulant, et une roue flottante assurant la direction. Les moteurs, alimentés par deux batteries 12V/25Ah, peuvent ainsi assurer la propulsion. Les batteries se trouvent à l'intérieur d'un châssis aluminium/plexiglas.

À l'intérieur du châssis se trouvent également les variateurs – reliés aux batteries et moteurs, la boîte à fusibles et conversion de tension, l'Arduino et l'ordinateur embarqués, ainsi que les divers câblages nécessaires au fonctionnement du Centaure. Le Centaure est également équipé d'un écran LCD suspendu en hauteur par deux barres d'aluminium, ce qui lui donne, avec un peu d'imagination, la forme à l'origine du surnom. Il est également équipé d'une boussole pour arduino, de quatre capteurs infrarouges (IR) avant et deux capteurs IR arrière, éléments essentiels de notre projet.

Lors de notre « prise de contact » avec le Centaure, ce dernier était également équipé de détecteurs suiveurs de lignes, et d'un détecteur à ultrasons, utilisés lors de précédents projets. Ces éléments n'ont pas été jugés utiles à la réalisation de notre projet.

B - Cahier des charges fonctionnel

Nous précisons dans ce CDCF les objectifs fixés par nos tuteurs, que sommes chargés d'atteindre lors du rendu final. Ainsi, pour ce projet sur le robot Centaure, nous devons :

→ Utiliser un module Kinect pour l'acquisition d'images 3D.
→ Utiliser un système de commande IR pour diriger manuellement le robot.
→ Diriger le robot via un site web adaptable pour des supports PC, Tablette, Smartphone.
→ Créer un mode d'arrêt d'urgence.
→ Mettre en place un capteur de charge des batteries.

Les contraintes suivantes devront être respectées :

→ Le rendu total du projet, à savoir un robot respectant le cahier des charges ci-dessus, ainsi qu'un Twiki faisant état de l'avancement du projet, un rapport global de projet, une vidéo de présentation à but promotionnel, ainsi qu'une présentation orale, devra être accompli avant le 7 Mai 2013, date de soutenance.

→ Le robot devra être capable de se mouvoir dans le hall de l'école Polytech Lille. Ce hall est de forme rectangulaire, avec des obstacles comme des rampes d'escalier ou des poteaux. Le sol, en lino, devrait empêcher tout patinage des roues.

→ La vitesse du robot ne doit pas excéder les 8km/h, en vue d'une utilisation publique.

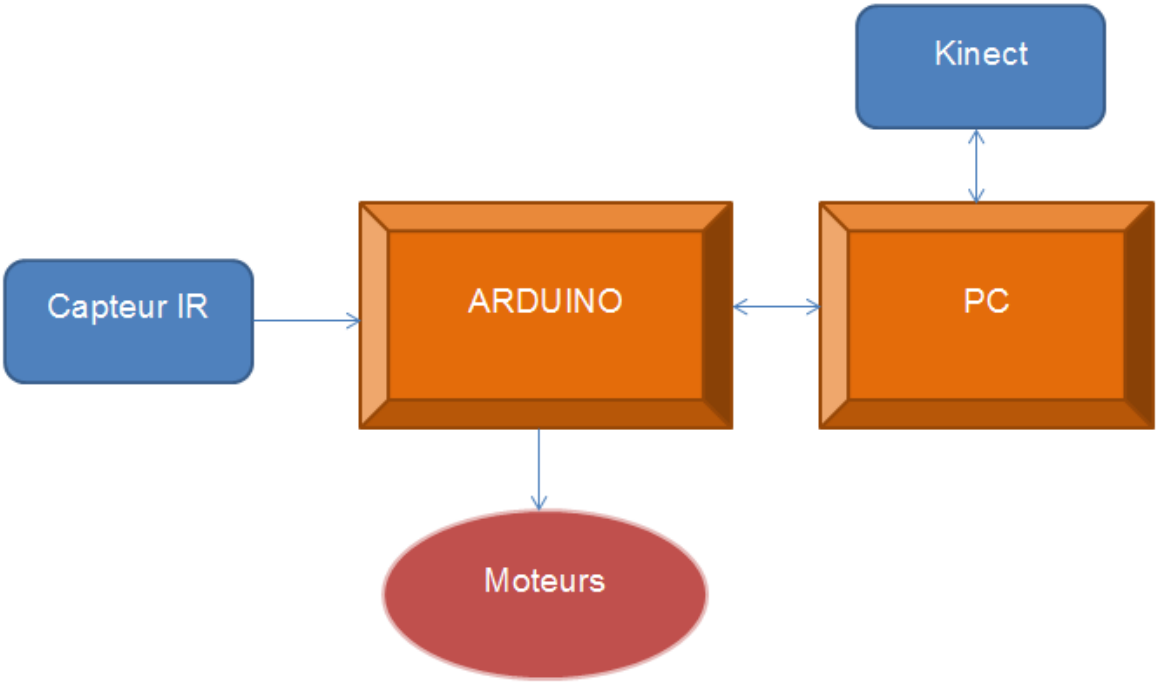
Dans le cadre des objectifs fixés par ce CDCF, nous avons carte blanche quant aux technologies à adopter pour la réalisation de ces objectifs.

Partie robot : présentation et évolution

A - Constatation et Définition des tâches

Le robot Centaure, tel que nous l'avons récupéré, était dans un état d'abandon. C'est-à-dire que l'on pouvait constater avec facilité qu'il n'avait pas été « touché » depuis quelques années, d'où l'attente de nos tuteurs à nous voir réussir notre projet. Les batteries (12V, 25Ah), une partie des capteurs avant (SHARP 2Y3A003 - portée de 40 à 300 cm) et des capteurs arrière (SHARP 2Y0A02 - portée de 20 à 150 cm), ainsi que divers instruments qui ne sont pas utiles au bon déroulement de notre projet, sont hors-service (capteurs de suivi de ligne, capteur sonar entre autres). Nos premiers pas consistent donc en un nettoyage de fond en comble, de manière à faire le bilan matériel et une projection plus aisée quant à la meilleure des solutions à adopter pour le projet. Nous décidons également de trouver la solution la plus esthétique possible, pour faciliter notre travail comme celui des personnes qui travailleront sur le Centaure à l'avenir (gaine thermorétractable, câbles nappe, gaine perforée).

Afin de faciliter la compréhension, nous établissons un schéma relationnel simplifié de l'ensemble de notre système :



En effet, nous avons envisagé la solution Arduino comme étant la plus adaptée à notre problème.

Le robot étant autonome et dialoguant avec un PC embarqué via liaison série, nous avons choisi

une architecture basée sur un Arduino MEGA + Shield. Ce système nous permet d'obtenir une liaison série ainsi qu'un contrôle direct des différents capteurs présents.

Le choix du MEGA s'est porté dans une vision de constante amélioration, sans être limité par l'environnement existant pour des projets futurs. Pour garantir un fonctionnement optimal, la connectique des capteurs sera également revue pour permettre la connexion avec le Shield.

Afin d'avoir le plus rapidement possible un support de travail fonctionnel, la restauration du système de propulsion du Centaure est notre priorité. Il faut donc prendre en compte le fonctionnement des variateurs (documentation technique + précédents rapports nécessaires), la façon de relier ces derniers aux moteurs ainsi qu'aux nouvelles batteries. En effet, il est également pressant de définir un moyen d'asservissement des roues, toujours dans l'optique de se reposer sur une méthode dont le fonctionnement est sûr, et nous permettre de bâtir notre stratégie de contrôle à partir de cette méthode.

Nous verrons par la suite que l'utilisation d'une boussole pour Arduino s'avèrera nécessaire au bon contrôle de direction…

B - Travail effectué sur le centaure

- ➔ La réparation des capteurs avant et arrière, ou plutôt la réalisation d'un nouveau câblage de ces derniers, adapté à la solution Arduino, se fait avec des prises type « JR ». Ces derniers doivent être prêts au plus tôt pour implanter les premiers codes de détection d'obstacle, et être directement reliés au 'mode d'arrêt d'urgence', comme prévu dans le CDCF.
- ➔ Le variateur est la partie la plus complexe qui a été traitée. On rappelle brièvement le fonctionnement d'un variateur. Le variateur est un hacheur abaisseur ou survolteur. On est donc en mesure de pouvoir influer sur la tension qui le traverse, soit en l'abaissant, soit en l'augmentant. Et influer sur la tension nous permet d'influer également sur la vitesse, car il y a proportionnalité entre les deux grandeurs.

  Ce variateur dispose de multiples sécurités, dont toutes ne sont pas précisées dans la documentation technique. Notre variateur s'appuie sur des interrupteurs, qui basculent à des états complémentaires afin d'assurer la marche avant ou la marche arrière. Le « démarrage » du variateur nécessite une tension d'excitation, qui s'élève à 60mV. Il est à noter que le fonctionnement des variateurs ne tolère aucun retour à une tension nulle, auquel cas la sécurité de ce dernier sera réactivée. La documentation technique nous informe sur la manière de brancher les variateurs aux batteries et aux moteurs.
- ➔ Code Arduino pour la partie robot : (contrôle de direction, boussole, charge de batteries…)

C - Réalisation technique

1 – Partie matériel

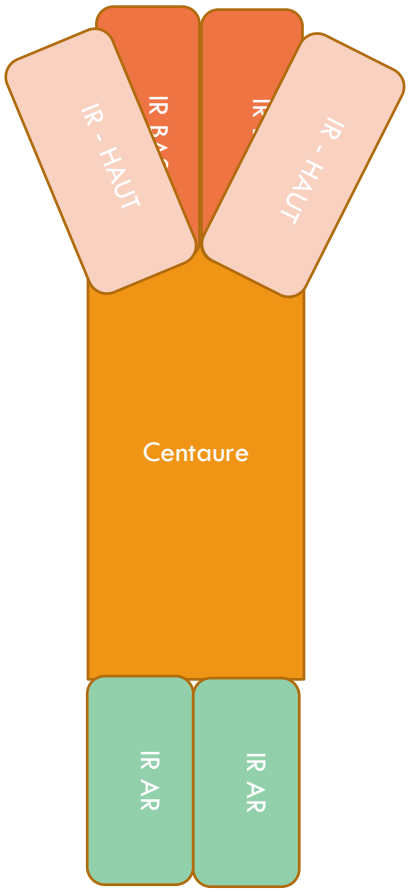
a) Positionnement des capteurs

Comme le Centaure ne sera commandé que par la Kinect, et que les détecteurs infra-rouge ne serviront qu'a un 'mode arrêt d'urgence' que si nous le définissons, ils ne servent qu'à détecter la présence d'obstacle proche du robot dans le but d'éviter la collision.

Nous expliquerons son comportement dans l'analyse du Software.

Pour cette raison, nous avons mis en place 4 capteurs à l'avant et 2 capteurs à l'arrière, les 2 capteurs arrière n'étant activés que lors d'une marche arrière.

A l'avant les 2 capteurs haut sont orientés vers l'extérieur pour prévenir le mieux possible les 'angles morts'. Les 2 capteurs bas sont quand à eux en parfait alignement avec le robot pour prévenir des obstacles bas.



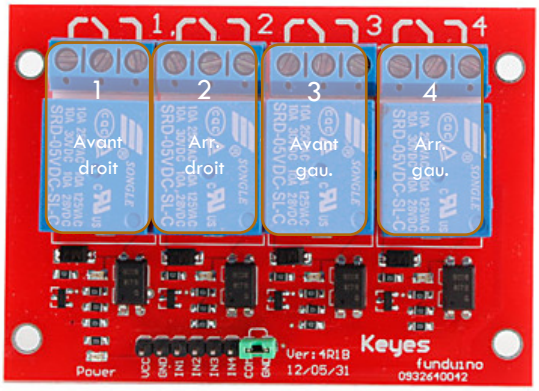
b) Contrôle des variateurs via carte relais

Les variateurs sont commandables en tension pour ce qui est de la vitesse de rotation des moteurs. Le sens de direction des moteurs est quant à lui commandable via un système d'interrupteurs ici réalisé grâce à des relais.

	Relais 1	Relais 2	Relais 3	Relais 4
Marche avant	X	-	X	-
Marche arrière	-	X	-	X
Rotation 1	X	-	-	X
Rotation 2	-	X	X	-

Fonction réalisable grâces aux interrupteurs

Les rotations permettent soit une rotation lié au différentiel des deux roues à droite, soit à gauche, ce qui permet de rendre le robot très manœuvrable.



Nous avions 2 choix, soit concevoir la carte, ce qui a été fait (les schémas sont disponibles sur le Twiki), soit acheter une carte. Nous avons choisi d'acheter une carte vis-à-vis du rapport qualité-prix.

c) Carte de contrôle de la tension

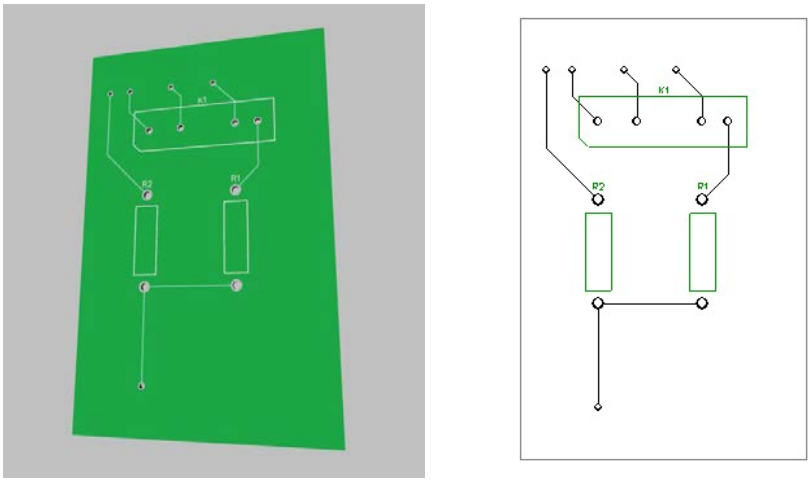
Nous devions réaliser un indicateur de niveau batterie, pour cela la solution du pont diviseur de tension a été retenue. Nous avons décidé de choisir des résistances avec une précision de 1% car vis-à-vis des différences de tension à observer un léger écart peut avoir de grosses répercussions.

De même, un pont diviseur de tension consomme beaucoup. Ce qui, dans un système autonome, n'est pas forcément adapté. Pour cela, nous avons intégré à notre carte un relais 5V qui permet de fermer le circuit du pont diviseur et de faire le relevé de valeur juste à ce moment. Ce qui permet d'éviter la dépense d'énergie inutile.

Pour définir les différents seuils de batterie et établir un pourcentage, nous avons relevé la tension des batteries en pleine charge et recherché le seuil de tension critique.

Seuil Batterie:

CAN Arduino (10 bits) :	Equivalent Tension Sortie Pont (V) :	Tension batterie (V) :	(%) :
1020	4,987	25,60	100
996	4,86	24,95	75
970	4,734	24,30	50
958	4,675	24,00	25
880	4,286	22,00	0



PCB de la carte

```
R1 = 62Kohm (à droite sur le schéma)
R2 = 15Kohm
```

2 – Partie logiciel

a) Asservissement en vitesse

Les moteurs sont asservis en vitesse, nous leur envoyons une référence en tension via une sortie PWM de l'Arduino. Pour cela nous utilisons la commande suivante :

analogWrite(PIN,val) // entre 0 et 255

, val étant comprise entre 0 et 255, 255 correspondant à 5V, PIN étant le numéro de pin choisi sur l'Arduino. Nous renvoyons une commande en équivalent tension comprise entre 5 et 55, 55 étant le maximum qui puisse être pris en charge par les variateurs.

Afin d'éviter tout pic de courant, nous avons instauré une rampe de tension. Ainsi chaque déplacement commence en douceur.

b) Présentation générale

Programmation via l'IDE Arduino. C'est une programmation en C. L'utilisation de l'Arduino nous permet de faire appel à de nombreuses librairies. Nous utiliserons la librairie MSTimer fournie par la communauté Arduino, permettant de réaliser des interruptions temps réel.
Nous utiliserons aussi la librairie Wire, qui crée un bus de données de type I^2C, la boussole étant connectée à celui-ci.

Les variateurs du robot sont utilisables qu'après une certaine configuration. Le programme utilise une sortie PWM simulant 60mV, ce qui a pour but de désactiver les sécurités. Les relais doivent être ouverts durant l'initialisation. Le robot ne commence à se mouvoir que lorsqu'il reçoit un ordre extérieur.

Pour que le robot soit opérationnel, la séquence suivante doit être réalisée lors du démarrage :

c) Système de secours

Le système de secours nous permet de préserver l'intégrité du Centaure en ignorant les ordres extérieurs via l'analyse des capteurs Infra-Rouge. Nous fixons dans des variables d'en-tête des seuils, qui ne doivent pas être dépassés. Ceux-ci sont disponibles en annexe. Les seuils fixés correspondent à environ 30cm. Le robot ne fait fonctionner en marche avant que ses détecteurs IR avant. Lors de la détection d'un dépassement d'un seuil, l'Arduino désactive les moteurs en ouvrant les relais. Après une phase d'attente (réglable par un délai) nous activons la marche arrière. Comme durant chaque marche arrière les détecteurs IR arrière sont activés. Le robot, tant qu'il n'y a pas de dépassement de seuil, continuera sa marche arrière durant un laps de temps défini par une boucle. Les capteurs IR sont branchés sur les CAN de l'Arduino.

d) Communication avec le PC

L'Arduino est en liaison directe avec le PC embarqué via une interface série fonctionnant à 9600 bauds. Celui-ci scrute son buffer-série et lit chaque octet reçu. La difficulté étant que le buffer stocke chaque réception avant sa lecture. Il faut donc que le PC embarqué n'envoie pas d'ordre superflu afin de ne pas parasiter le système. En respectant ces règles, nous obtenons un système temps réel. Pour réaliser une liaison entre 2 équipements nous avons dû mettre en place un protocole (voir ci-dessous). Celui-ci permet de faire correspondre un ordre (ici un caractère) envoyé par le PC avec une commande physique traduisant un mouvement pour le robot. Nous scrutons le buffer grâce à la commande **Serial.read()**. L'envoi de données se déroule de manière unique (une information à transmettre équivalent à une information transmise), il n'y a pas de réémission, ni de vérification de la réception. Pour cela nous utilisons la commande **Serial.print()**.

Lors d'un arrêt d'urgence détecté par les capteurs IR, les ordres reçus par l'Arduino sont ignorés mais vidés du buffer.

Protocole liaison Série	
Ordre PC :	**Signification Arduino :**
s	stop
a	avance rapide
r	recule rapide
q	avance lente
d	recule lente
x	rotation droite 90°
w	rotation gauche 90°
v	rotation droite 45°
c	rotation gauche 45°
f	courbe gauche
g	courbe droite

Ordre Arduino :	**Signification PC :**
s	arrêt urgence
p	batterie 100%
o	batterie 75%
i	batterie 50%
u	batterie 25%
y	batterie vide
t	attention dommage batterie

e) Carte du niveau de batterie (mini-carte)

Nous avons réalisé une carte permettant de capter le niveau de batterie. Le but de cette carte est d'être économe en énergie. Pour cela nous avons utilisé la librairie MSTimer, grâce à cette commande la génération d'interruption temps réel est rendu possible.

Toutes les 60 secondes l'interruption est déclenchée via la commande :

MsTimer2::set(60000, interrupTimer2);

interrupTimer2 étant le nom de la fonction appelée contenant le code activant le relais, ainsi que la partie du code permettant de faire un relevé de la valeur contenu dans le CAN. Le

tableau d'équivalence est défini dans la partie matériel. Grâce à ces valeurs nous avons défini des paliers qui transmettent l'information via le port série.

f) Système de boussole

Nous utilisons également un Compass Arduino, la boussole HMC6352, dans le cadre de ce projet. Pour rappel, le robot Centaure est équipé d'une roue flottante avant, qui perturbe la trajectoire indiquée au robot. La boussole est donc là pour corriger la trajectoire. Le HMC6352 utilise le bus I2C, qui permet une connexion plus aisée entre un microcontrôleur et ses périphériques (la boussole est ici considérée comme un équipement esclave). Attention à ne pas mettre la boussole à côté d'un élément pouvant entraîner de fortes perturbations électromagnétiques (valeurs aberrantes). Le module se connecte sur l'arduino via 4 ports : SDA et SCL pour l'utilisation du protocole I2C, VCC sur du 3.3V, et la masse. Nous utilisons également la librairie wire.h qui permet de communiquer avec des périphériques I2C.

Nous acquérons l'angle de la trajectoire du robot dans la fonction **lecture_boussole()**, et la stockons dans une variable appelée **myres**. La boussole va fournir une donnée comprise entre 0 et 3599, soit une valeur pour chaque dixième de degré. Afin de pouvoir corriger plus efficacement la trajectoire, nous convertissons ces valeurs de manière à avoir une valeur en degrés « lisible » (de 0 à 360°).

À chaque nouvelle trajectoire, nous attribuons la valeur myres à angle_traj_actuel, puis nous lançons une nouvelle acquisition de myres. Ainsi, la fonction **b_compare** peut comparer ces deux valeurs, et le programme exécutera la trajectoire recommandée selon la valeur **b_compare**.

Conclusion

En conclusion de ce projet qui nous aura tenus en haleine pendant de longues heures, nous pouvons affirmer que le Centaure a repris vie. La restauration complète du système de propulsion, la direction du Centaure, la correction de trajectoire, l'arrêt d'urgence, la détection d'obstacles, capteur de charge des batteries, toutes les applications stipulées dans le CDCF et réalisables dans cette partie ont été effectués.

De l'imagination, des compétences, de la curiosité, tels étaient les éléments qui permettaient d'avancer dans ce projet. Certes, ce n'est encore qu'un début, mais il est déjà possible d'imaginer de multiples améliorations à apporter à ce robot, tout comme les applications dans lesquels il pourrait avoir son utilité.

À une époque où la robotique est en plein essor, et que les robots commencent à se démocratiser dans nos vies, il était important, dans le cadre de nos études comme celui de notre avenir professionnel, de toucher à ce sujet. Et ce projet en a été une belle occasion.

Nous avons en tout cas hâte de voir ce que les futurs IMA pourront apporter à ce Centaure, et pourquoi pas le voir bientôt comme un élément de promotion de l'école, au même titre que NAO…

Annexes

CARTE RELAIS

Encoche Relais	Couleur
1G	***
1M	Noir (Pont 1) et Marron (variateur D)
1D	Blanc (variateur D)
2G	***
2M	Noir (Pont 1)
2D	Vert (variateur D)
3G	***
3M	Noir (pont 2) et Marron (variateur G)
3D	Blanc (variateur G)
4G	***
4M	Noir (Pont 2)
4D	Vert (variateur G)

'G' signifie port de gauche (bouclé avec le port M) pour 1 relais
'M' signifie port milieu pour 1 relais
'D' signifie port de droite pour 1 relais

Seuil Batterie:

CAN Arduino (10 bits) :	Equivalent Tension Sortie Pont (V) :	Tension batterie (V) :	Pourçentage (%) :
1020	4,987	25,60	100
996	4,86	24,95	75
970	4,734	24,30	50
958	4,675	24,00	25
880	4,286	22,00	0

Protocole liaison Série :

Ordre PC :	**Signification Arduino :**
s	stop
a	avance rapide
r	recule rapide
q	avance lante
d	recule lente
x	rotation droite 90°
w	rotation gauche 90°
v	rotation droite 45°
c	rotation gauche 45°
f	courbe gauche
g	courbe droite

Ordre Arduino :	**Signification PC :**
s	arret urgnece
p	batterie 100%
o	batterie 75%
i	batterie 50%
u	batterie 25%
y	batterie vide
t	attention dommage batterie

PIN Arduino :	Correspondance :	Couleur Fils Variateur :	PIN carte relais :	Couleur Fils Carte relais :
	CAPTEUR Infra-Rouge			
A1	IR-avant			
A2	IR-avant			
A3	IR-avant			
A4	IR-avant			
A5	IR-arrière			
A6	IR-arrière			
	VARIATEUR MOTEUR			
10	Alimentation Variateur droit et gauche	Jaune		
7	Entrée référence vitesse variateur droit	Gris		
12	Entrée référence vitesse variateur gauche	Gris		
8	Relais moteur droit avant		IN1	Orange
9	Relais moteur droit arrière		IN2	Noir
11	Relais moteur gauche avant		IN3	Blanc
13	Relais moteur gauche arrière		IN4	Gris
+5V			VCC	Rouge
GND			GND	Marron
	Cléf sécurité, à connecter à la batterie	Rouge		
	Masse	Rose		
	MINI-CARTE (capteur niveau batterie)			
45	Relais, (+5V fils rouge mini carte PIN Coudé)			
A8	Capteur batterie (Sortie fils vert mini-carte Prise FUTABA)			
	BOUSSOLE			
20	SDA - i2c Protocole			
21	SCL - i2c Protocole			

Seuil Robot :

Infra-rouge Avant :	420
Infra-rouge Arrière :	320