De l'illustration du guidage à l'optimisation d'un plan par un robot Lego Mindstorm NXT

Elodie Chanthery, G Le Corre, Pierre-Emmanuel Hladik

▶ **To cite this version:**

Elodie Chanthery, G Le Corre, Pierre-Emmanuel Hladik. De l'illustration du guidage à l'optimisation d'un plan par un robot Lego Mindstorm NXT. Journal sur l'enseignement des sciences et technologies de l'information et des systèmes, 2016, 15, 10.1051/j3ea/2016006 . hal-01392601

HAL Id: hal-01392601
https://hal.science/hal-01392601

Submitted on 4 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

De l'illustration du guidage à l'optimisation d'un plan par un robot Lego Mindstorm NXT

E. Chanthery[1,2], G. Le Corre[2], P.-E. Hladik[1,2]
elodie.chanthery@laas.fr
[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2] INSA; 135 avenue de Rangueil, Toulouse, France

Résumé : Cet article présente un projet mené au Département Génie Électrique et Informatique de l'Institut National des Sciences Appliquées (INSA) de Toulouse dont le but est la commande d'un robot Lego NXT. L'objectif est d'illustrer les différents niveaux de commande, du guidage bas niveau effectué par une régulation de position jusqu'à l'optimisation d'un plan de mission.

1 Introduction

La création d'une manipulation de travaux pratiques doit répondre à des objectifs variés. Aux objectifs pédagogiques évidents, peut s'ajouter celui de disposer d'un démonstrateur qui puisse intéresser le public étudiant visé, mais également le public moins averti des journées portes ouvertes et autres visites de département, qui au travers de ce démonstrateur se fera sa propre idée d'un domaine scientifique voire même d'un futur métier.

Nous souhaitons ici illustrer la commande d'un système, pour laquelle on peut distinguer plusieurs niveaux qui correspondent à différents champs du domaine de l'automatique en général (voir figure 1).

C'est dans cette optique qu'a été développée au département Génie Électrique et Informatique (GEI) de l'INSA de Toulouse une manipulation sur la commande d'un robot Mindstrom NXT de Lego sur deux roues motrices et une roue libre pour l'équilibre semblable à un rover. Le robot doit suivre de manière efficace une ligne noire au sol pour guider sa trajectoire (commande

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article publié par EDP Sciences et disponible sur le site http://www.j3ea.org ou http://dx.doi.org/10.1051/j3ea/2016006

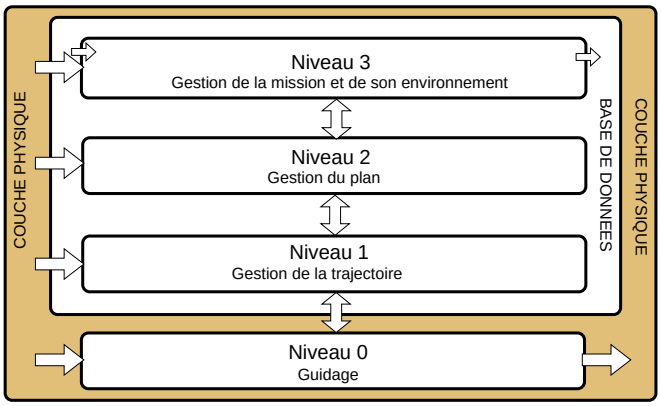

FIGURE 1 – Les différents niveaux de commande dans une architecture d'autonomie

de bas niveau de type commande continue : Niveau 0) ; choisir correctement un chemin en fonction d'un critère et de contraintes données (commande de haut niveau type Intelligence Artificielle : Niveau 2) ; suivre correctement le chemin voulu (commande type contrôleur d'exécution : Niveau 1).

Le public étudiant concerné par ce projet est constitué des étudiants de niveau $M1$ de la filière Automatique et Électronique du département GEI de l'INSA de Toulouse. Ce TP fait partie d'une série de six manipulations en automatique qui tournent sur un semestre au rythme d'un TP par semaine [3]. Le TP se déroule sur 2h45 et s'effectue en binôme. L'enseignement de « Graphes » est également dispensé pendant ce semestre de manière classique sous forme de cours magistraux et de travaux dirigés. Ainsi, en début de semestre, le TP est une introduction au cours, alors qu'en fin de semestre il en est l'illustration.

Le choix d'un robot Lego Mindstorm s'explique par son faible coût, la facilité de sa mise en œuvre, ainsi que par le caractère ludique de la manipulation qui devient abordable par un public non averti. Le projet a été amorcé avec deux projets tutorés de niveau $M1$ (environ 60h). Il a ainsi été possible de définir les briques de commande à illustrer et d'avoir un cadre logiciel approprié.

La suite de cet article est organisée de la manière suivante : la section 2 est dédiée au robot Lego et son environnement logiciel. L'illustration des différents niveaux de commande dans une manipulation de TP est décrite dans la section 3. La section 4 présente les retours d'expérience après une

année. Enfin, la section 5 conclut l'article.

2 La maquette du robot Lego et son environnement logiciel

2.1 Le robot et son environnement

Le support utilisé pour illustrer les différentes notions d'automatique est un robot construit avec les éléments Mindstrom NXT 2 de la marque Lego. De nombreuses activités pédagogiques ont été réalisées avec ce support à l'INSA de Toulouse pour illustrer des notions variées telles que l'algorithmique, l'automatique, le traitement d'images, le temps réel ou la conception de systèmes.

Dans le cadre de ce travail pratique, le robot fourni aux étudiants (voir figures 2 et 3) est un robot à roues différentielles avec deux roues indépendantes alignées sur le même axe et une roue libre pour l'équilibre. Les capteurs embarqués sur le robot sont :

- un capteur NXT LineLeader (figure 4a) composé de huit couples photodiodes/phototransistors permettant de mesurer la position de la ligne noire sur un fond blanc (voir figure 4b),
- un capteur ultrason Mindstorm renvoyant une valeur proportionnelle à la distance le séparant d'un obstacle. Ce capteur n'est pas utilisé dans l'activité actuelle, mais pourrait l'être dans l'avenir (voir partie 3.4),
- deux codeurs optiques sur les axes de rotation des moteurs permettant d'avoir la position de chacune des roues avec une précision de 1 °. Ces capteurs sont directement intégrés dans les moteurs Mindstorm.

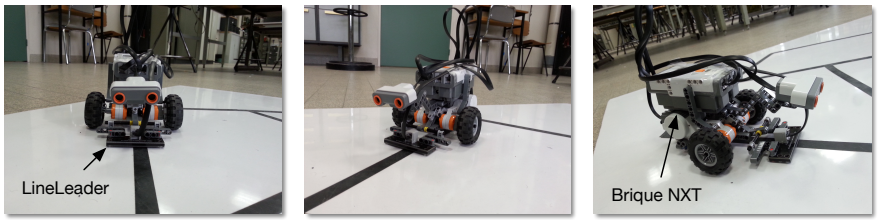

FIGURE 2 – Le robot Lego

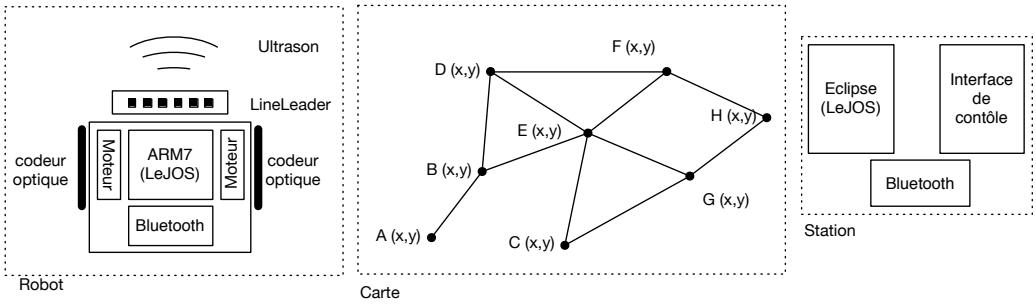

FIGURE 3 – Schéma de principe du robot, de la carte et de la station

(a) Capteur NXT LineLeader

(b) Schéma de principe du suivi de ligne

FIGURE 4 – Le capteur NXT LineLeader et son schéma de principe

Les actionneurs du robot sont uniquement constitués par les deux moteurs qui contrôlent les roues. Ce contrôle peut être réalisé en position ou en vitesse.

Le calculateur embarqué dans le robot est celui proposé par Lego et est principalement constitué d'un microprocesseur 32 bits ARM7 avec la possibilité de communiquer via Bluetooth ou en filaire par USB. Un ordinateur fixe avec Windows sert de station de base pour envoyer les ordres et les plans de la mission et pour programmer le robot (voir figure 3).

Le robot se déplace dans un espace constitué de routes droites avec des intersections. Cet espace est représenté par une carte entièrement connue et décrite par les coordonnées des intersections et les arcs qui les relient (voir figure 3). Physiquement, l'espace est constitué d'une surface glissante blanche (type tableau blanc), les routes sont réalisées à l'aide de ruban adhésif noir mat.

2.2 L'environnement logiciel LeJOS

La programmation de la station et du robot s'effectue à l'aide de l'environnement de développement LeJOS NXJ qui est fait pour programmer le robot NXT en Java [1]. Ce projet comprend principalement :

- un *firmware* pour la brique NXT incluant une machine virtuelle Java,
- un ensemble de librairies sur le robot qui implémentent une interface de programmation Java pour interagir avec les composants NXT (contrôle des moteurs, lecture des capteurs, etc.),
- une boîte à outils pour l'édition des liens, le chargement des programmes et le debugage,
- une interface de programmation Java pour PC afin de communiquer avec le robot via Bluetooth et USB.

À cela s'ajoute un pluging Eclipse [2] permettant d'importer facilement dans cet IDE les outils et les librairies du projet LeJOS. La mise au point des programmes, leur chargement et leur exécution, sur le robot ou sur le PC, peuvent se faire ainsi entièrement à l'aide d'Eclipse.

Nous tenons à souligner que l'utilisation du langage Java et de LeJOS pour réaliser ce travail pratique n'est pas un problème pour des étudiants ne sachant pas programmer en orienté objet. L'intervention des étudiants se fait uniquement sur certaines parties bien identifiées du code. Elles peuvent être aisément comprises par des étudiants ayant l'habitude d'une programmation impérative dans des langages de type C, Ada ou autre.

2.3 Bibliothèques spécifiques au projet

Les différents éléments de la commande sont distribués sur les deux unités de calcul :

- le niveau 0 (commande bas niveau) correspondant au suivi de ligne est directement implémenté sur le robot,
- le niveau 1 (gestion de la trajectoire) est répartie entre le robot et la station fixe et s'effectue par l'échange de messages synchrones,
- les niveaux 2 et 3 (gestion du plan et de la mission) sont implémentés sur la station fixe.

Le code embarqué sur le robot est constitué d'un ensemble de classes (voir figure 5) dont une partie est dédiée au contrôle bas niveau du robot (`PilotRoberto`). Celui-ci s'appuie sur la classe `DifferentialPilot` de LeJOS qui offre des fonctions pour faire tourner le robot d'un angle donné

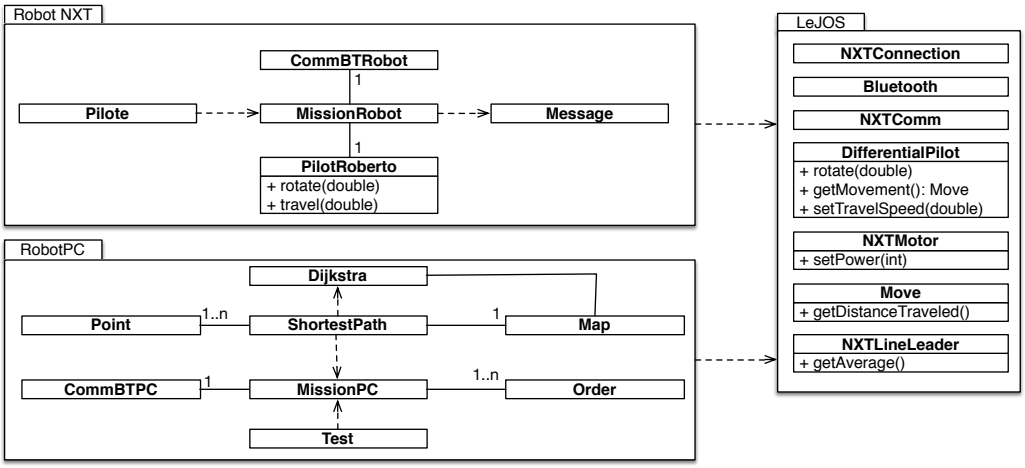

FIGURE 5 – le robot Lego

(`rotate`), pour fixer la vitesse du robot (`setTravelSpeed`) et pour récupérer les informations sur le mouvement en cours (`getMovement`) telle que la distance parcourue. Il est alors possible de concevoir une commande simple de suivi de ligne en combinant les informations de distance avec celles mesurées sur la barre de led (`NXTLineLeader.getAverage`) et le contrôle de la vitesse des moteurs (`NXTMotor.setPower`). Cette activité pédagogique sera détaillée dans la partie 3.1.

Un point de la carte est décrit dans la classe `Point` par ses coordonnées physiques en abscisse et ordonnée par rapport au référentiel de la carte. Une carte (`Map`) est décrite par un ensemble de points et par les arcs existants entre ces points. La réalisation d'une mission consiste à envoyer au robot une suite d'ordres (`Order`) représentant un parcours qui suit des points connexes. Pour cela deux types d'ordre sont envoyés séquentiellement au robot : les ordres de rotation et les ordres de déplacement. Une rotation est décrite par un angle orienté et un déplacement par une distance positive. Afin de construire une séquence d'ordres, il faut calculer les angles et distances représentant un parcours sur une carte. Par exemple, pour le déplacement représenté par la figure 6 où le robot est initialement au point A et orienté dans le sens $\overrightarrow{AB}$, la séquence représentant la mission passant successivement par les points A, B et C est {R(0), M(56.5), R(-45), M(60)} avec R(x) une rotation d'un angle x et M(y) un déplacement linéaire de y.

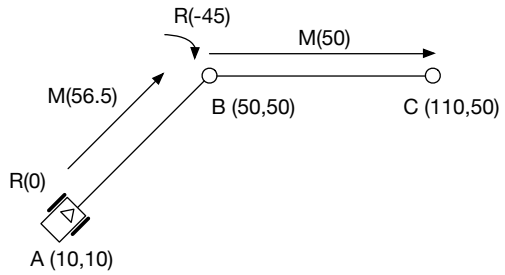

FIGURE 6 – Exemple de la décomposition séquentielle d'une mission en ordres

La communication entre le robot et la station est synchrone dans le sens où l'envoi d'un nouvel ordre par la station se fait uniquement après la réception d'un message envoyé par le robot indiquant qu'il a terminé l'ordre précédent.

Une rotation s'effectue simplement par un appel à la fonction `rotate` de `PilotRoberto` et un déplacement linéaire est réalisé par l'appel à `travel` de la même classe. Cette dernière fonction réalise un suivi de ligne et se termine lorsque la distance mesurée est égale à celle demandée. Le robot détermine la fin d'un ordre uniquement à partir des mesures de ses capteurs, il n'y a donc aucune vérification externe de la bonne réalisation de l'ordre.

La construction du plan de la mission et de la séquence d'ordre est réalisée par la classe `ShortestPath` sur la station fixe et utilise un algorithme de Dijkstra pour calculer le plus court chemin. Une classe spécifique a été implémentée pour cet algorithme uniquement pour en faciliter l'accès aux étudiants.

Pour le projet, une interface graphique a aussi été développée afin de saisir les ordres de la mission et pour en contrôler la bonne exécution (voir figure 7).

3 Illustration des différents niveaux de commande par une manipulation de TP

La manipulation proposée est destinée aux étudiants de 4ème année (équivalent $M1$) de la filière Automatique et Électronique du département GEI de l'INSA de Toulouse. Ces étudiants n'ont jamais fait de Java avant cette manipulation. Le cours de « Graphes » est également dispensé pendant ce

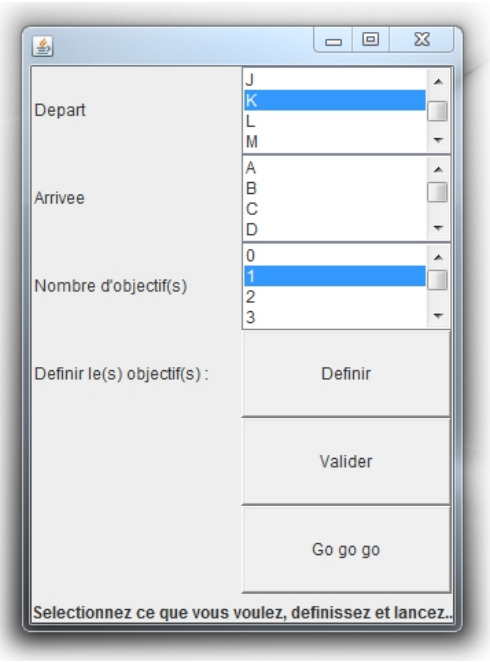

FIGURE 7 – Interface graphique de contrôle

semestre. Ainsi en début de semestre, le TP est une introduction au cours, alors qu'en fin de semestre il en est l'illustration. Les étudiants ont accès à un projet dans lequel toutes les classes sont déjà mises en place. Seules certaines parties du code sont à compléter pour faire fonctionner le projet.

3.1 Commande bas niveau

Dans un premier temps, l'objectif est de mettre en place la commande de bas niveau sur le robot, c'est-à-dire de faire suivre la ligne noire au robot lors de son déplacement. Pour cela, les étudiants utilisent la donnée (`lineposition`) issue du capteur NXT LineLeader. Lorsque la ligne noire est située au centre de ce capteur, `lineposition` retourne la valeur 45. La boucle de régulation va donc suivre le schéma de la Figure 8 sachant que le correcteur agira à la fois sur les deux moteurs des roues, appelés moteur D (droit) et moteur G (gauche) qui seront contrôlés en vitesse. L'action sur les deux moteurs sera différente sachant qu'il faut augmenter la vitesse d'un moteur en réduisant celle de l'autre pour faire tourner le robot. Les étudiants

implantent donc un simple correcteur proportionnel. Ils étudient ensuite le réglage de la vitesse de rotation moyenne des deux moteurs et du gain du proportionnel afin d'obtenir une trajectoire satisfaisante. Cette commande n'est pas du niveau niveau $M1$, mais nous l'avons jugée pertinente pour montrer tous les niveaux de commande aux étudiants. Elle pourrait évidemment être plus évoluée.

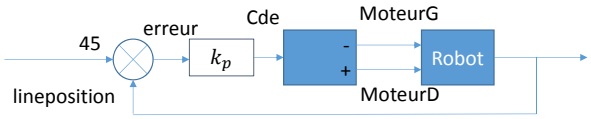

FIGURE 8 – Schéma de régulation

Cette régulation est testée en réel sur le robot, de manière à valider le niveau 0 de la commande.

3.2 Gestion de la trajectoire

Cette partie consiste à vérifier que la gestion de la trajectoire est bien effectuée par le robot. Pour cela, les étudiants doivent valider le bon séquencement d'une mission écrite « à la main ». Pour cela, ils créent une mission allant d'un point à un autre de la carte, à vitesse constante, en utilisant des commandes du type `listOfOrders.add(new Order(int angle, int distance, int speed))` ; avec les angles en degrés, la distance en millimètres et la vitesse en pourcentage. La mission est testée en réel sur le robot, de manière à valider le niveau 1 de la commande.

3.3 Gestion de plan

Cette partie consiste à élaborer une stratégie pour l'élaboration d'un plan de mission. Pour cela, il est nécessaire de définir un graphe dont les sommets sont les points visitables et les arcs les chemins entre ces points.

Dans une première partie, les étudiants doivent formaliser le problème d'optimisation à résoudre pour élaborer un chemin pour le robot en établissant un critère d'optimisation. Une annexe [3] aide les étudiants dans le cas où ils n'ont pas encore suivi le cours de « Graphes ». Le problème d'optimisation est défini comme la recherche d'un plus court chemin dans un graphe.

Les poids des arcs dépendent du critère choisi (par exemple la distance entre les points, ou bien la durée).

Les étudiants proposent tout d'abord sur le papier une méthode de recherche de chemin intuitive, par exemple un algorithme glouton [4] qui permettrait de trouver un chemin entre un point de départ et un point d'arrivée. Cette étape permet aux étudiants n'ayant pas suivi le cours de « Graphes » d'appréhender la complexité du problème.

Dans une seconde partie, les étudiants doivent étudier le code de l'algorithme de Dijkstra [5] en partie codé. Ils retrouvent ainsi les étapes du pseudo-algorithme donné en annexe, puis complètent le fichier en java de manière à coder la méthode `Trouvmin()` qui retourne le sommet le plus proche du sommet initial et la méthode `MajCcum(int noeud_retenu)` qui met à jour la valeur des noeuds dans une table. L'algorithme ainsi complété est testé en utilisant un critère de distance (plus court chemin). Dans un second temps, les étudiants doivent s'intéresser au chemin le plus rapide en terme de temps de trajet. Ils modifient la fonction de coût de manière à adapter l'algorithme et doivent trouver par eux-mêmes un test permettant de mettre en évidence ce changement de critère en changeant éventuellement les vitesses sur la carte.

3.4 Extensions envisagées

La dernière partie (facultative) du TP vise à étendre l'algorithme pour prendre en compte d'autres critères ou d'autres façons de modéliser le problème. Les étudiants sont ainsi sensibilisés à différents problèmes plus complexes qui découlent de ce TP. On leur propose plusieurs extensions possibles :

- Proposer une solution permettant au robot de partir d'un point initial, d'arriver à un point d'arrivée, en passant par des points particuliers précisés et non ordonnés, de manière optimale (typique d'une tournée de facteur par exemple).
- Proposer une solution si les ressources du robot sont limitées, par exemple, s'il ne peut effectuer qu'une distance limitée à cause du carburant.
- Imaginer un rover sur mars qui doit effectuer des points de mesures de plus ou moins grande importance, avec une quantité de ressource limitée. Réaliser un algorithme permettant de résoudre ce problème en maximisant les récompenses sur les points de mesures tout en garantissant que les ressources seront suffisantes.

— Détecter avec le capteur ultrason des obstacles qui pourraient apparaître en cours de mission et déclencher une replanification de la mission. Le robot doit alors faire demi-tour et trouver un chemin qui prenne en compte ces nouvelles informations.

4 Retours d'expériences

La manipulation a été mise en place et testée durant l'année 2014-2015 par 3 groupes de 6 binômes, soit 18 binômes. Pour rappel, la manipulation entière dure 2h45. Les étudiants doivent préparer leur manipulation en prenant connaissance du code existant et des annexes techniques concernant le robot. On estime le temps de préparation de cette manipulation à environ 30 minutes. À l'issue de la manipulation, la majorité des étudiants a codé et testé un plus court chemin à distance minimale. Après la manipulation, les étudiants disposent d'un accès libre à la salle et aux manipulation pour terminer leur travail et éventuellement approfondir les extensions. Ils doivent rendre un compte-rendu écrit au bout d'une semaine. Ce travail autonome est estimé entre 2h et 3h.

Les étudiants ont apprécié de pouvoir faire le lien entre le cours de « Graphes » qu'ils allaient suivre ou avaient suivi et ont vu dans le TP une application concrète d'algorithmes souvent difficiles à appréhender.

Le fait de travailler en Java n'a pas été une difficulté car le langage est en beaucoup de points similaire au C++ que les étudiants étudient.

Les étudiants ont apprécié la liberté sur les tests à effectuer. Ils ont ainsi pû s'approprier le problème.

Les étudiants ont jugé que la plus grande difficulté du TP résultait dans la compréhension de l'algorithme de Dijkstra. Ils ont trouvé que le fait de laisser des parties de code à compléter était intéressant et que cela les obligeait à vraiment comprendre l'algorithme.

Enfin, la mise en place progressive des commandes à chaque niveau a été appréciée. Les étudiants ont trouvé intéressant de passer de la commande bas niveau de guidage jusqu'à la commande de haut niveau. Le lien entre l'automatique « classique » des correcteurs et la commande à base de graphes était un plus.

La manipulation a été présentée lors des journées portes ouvertes de l'établissement. Le public a été très intéressé. Le côté ludique de la manipulation et le choix possible du trajet par l'assistance a été apprécié. Le parallèle avec

les planificateurs d'itinéraires de style « mappy » que tout le monde connaît permet de faire le lien entre les thématiques du département et des problèmes de la vie courante.

5 Conclusion

Au travers de cet article, nous avons souhaité présenter un support applicatif simple, attractif et relativement facile à mettre en place, permettant d'illustrer la commande d'un système distinguant différents niveaux de commande correspondant à plusieurs champs du domaine de l'automatique en général. Ce projet d'optimisation de plan à l'aide d'un robot Lego est bien entendu perfectible, néanmoins il offre d'ores et déjà de nombreuses possibilités pédagogiques. L'ensemble de la manipulation (code et aide au montage du robot) est disponible sur demande aux auteurs.

6 Remerciements

Nous tenons à remercier les différentes étudiants qui ont participé au développement de ce projet, en particulier : Pauline Combes, Simon Hippert, He Huang, Pauline Martinez, Guillaume Morisot, Petar Petrov, Jean-Marc Rodrigues et Thibaut Sardan.

Références

[1] LeJos, *http ://www.lejos.org*, consultation novembre 2015.

[2] Eclipse, *http ://www.eclipse.org*, consultation novembre 2015.

[3] Polycopié de Travaux pratique d'automatique, 4ème année AE, année 2015-2016, Département Génie Electrique et Informatique, INSA de TOULOUSE, disponible sur demande aux auteurs.

[4] Thomas H. Cormen, *et al.*, *Introduction à l'algorithmique*, chap.16 "Algorithmes gloutons", Dunod, 2002, p. 361.

[5] Steven Skiena. Dijkstra's algorithm. *Implementing Discrete Mathematics : Combinatorics and Graph Theory with Mathematica*, Reading, MA : Addison-Wesley, 1990, p. 225-227.