Programmation d'un robot Lego

Thomas Forgione Korantin Auguste Martin Carton
Maxime Arthaud Thomas Wagner Maxence Ahlouche

December 5, 2013

0.1 Méthodes de programmation du robot

0.1.1 Programmation graphique

Au début du projet, nous avons commencé à programmer le robot comme indiqué dans le sujet, i.e. en utilisant le logiciel de programmation graphique disponible sur Windows. Seulement, ce logiciel, bien qu'accessible à tout le monde, nous a à tous semblé très peu pratique à utiliser, si bien que nous n'avons pas été très productifs lors des premières séances.

En effet, dès que nous souhaitions faire autre chose que des instructions de base (par exemple, faire tourner le moteur à une vitesse dépendant de la valeur d'un capteur), nous passions beaucoup trop de temps à essayer différentes manières d'implémenter ces instructions sans que notre schéma devienne trop lourd.

De plus, nous avons rencontré un problème que nous n'avons pas réussi à résoudre: lorsque nos voulions faire avancer notre robot, il avançait de manière saccadée. Bien que nous ayons quelques hypothèses sur l'origine de ce problème, nous n'avons pas réussi à le résoudre de manière simple via le logiciel fourni.

À cause de ces désagréments, nous avons décidé d'essayer d'autres méthodes pour programmer le robot.

0.1.2 Librairie Python

Un des membres de notre équipe a déniché sur Internet une librairie en Python, qui permet de contrôler un robot branché en USB. Cette librairie (qui s'appelle `python-nxt`) nous a permis de régler notre problème d'avancement saccadé.

Elle nous a également permis de programmer notre robot de manière très simple et très propre. Un programme que nous avions mis plusieurs heures à implémenter via le logiciel graphique a été réalisé en moins d'une demi-heure via cette librairie.

Elle présente toutefois un inconvénient majeur: elle ne permet pas (ou alors, nous n'avons pas trouvé comment faire) de télécharger le programme sur le robot; par conséquent, le robot devait rester branché au PC lors de l'exécution du programme, et nous devions le suivre avec le PC. Ce qui, vous en conviendrez, n'est pas le summum de la commodité.

0.1.3 NXC

Finalement, nous avons décidé de programmer le robot en utilisant le langage NXC (*Not Exactly C*), développé spécifiquement pour les robots NXT.

Bien qu'il ne soit pas trivial de trouver de la documentation à jour pour ce langage, nous avons finalement réussi à l'utiliser proprement.

Le compilateur que nous avons trouvé nous permet également de télécharger le programme sur le robot, donc nous avons résolu le problème posé par la librairie en Python, ainsi que ceux posés par le logiciel de programmation graphique.

Ayant découvert ce langage tardivement dans les séances de GRO, nous n'avons réalisé qu'un seul programme l'utilisant: un programme qui longe les murs.

0.2 Longeage de murs

Comme dit précédemment, nous avons réalisé ce programme en utilisant NXC.

Nous avons fait le choix, dans un but d'économie de capteurs, de ne longer que les murs à gauche du robot.

0.2.1 Longeage d'un mur

Afin de longer un mur, nous avons posé un capteur à ultrasons sur le côté gauche de notre robot. Lorsque nous détectons que nous sommes trop éloignés du mur, nous tournons légèrement à gauche.

Cette méthode donnant des résultats peu probants si la direction initiale du robot n'est pas exactement parallèle au mur, nous avons décidé d'ajouter un capteur à ultrasons à l'avant du robot. Ainsi, nous tournons désormais à gauche, tout en avançant légèrement, dès qu'on détecte que le mur est trop loin; et si on se retrouve face à notre mur, alors on tourne vers la droite jusqu'à ne plus l'être.

Ainsi, nous pouvons suivre un mur droit de manière assez régulière.

0.2.2 Passage de l'extérieur d'un angle

Le passage de l'extérieur d'un angle (i.e. quand un mur tourne à gauche) se fait naturellement à partir du code de longeage d'un mur: il suffit de tourner à gauche (tout en avançant, afin d'éviter les collisions avec les murs) jusqu'à ce qu'on soit à nouveau à côté d'un mur. Naturellement, il a fallu adapter les coefficients de tournage, afin d'éviter que le robot ne prenne un angle trop large, et se retrouve finalement face au mur qu'il est censé longer.

0.2.3 Passage de l'intérieur d'un angle

Pour le passage de l'intérieur d'un angle (i.e. quand un mur tourne à gauche), le code découle aussi du premier algorithme. En effet, quand on se retrouve face à un mur, il suffit de tourner à droite jusqu'à ne plus détecter de mur face à soir, puis d'avancer.

0.2.4 Lissage du mouvement du robot

Afin d'éviter d'avoir un mouvement erratique lorsqu'on n'est pas exactement parallèle à la paroi que l'on doit longer (mouvements de "rebondissement" sur le mur, par exemple), nous avons décidé d'améliorer notre algo via une astuce très simple: le changement de direction dépend de la distance du mur. Ainsi, si nous nous éloignons légèrement du mur (à cause d'une trajectoire non parallèle

au mur, par exemple), alors nous tournerons très légèrement à gauche, sans ralentir. Ainsi, nous nous retrouverons presque parallèle au mur.

Dans le cas d'un virage à gauche du mur, le capteur détecte un mur à l'infini; dans ce cas-là, nous tournons autant que possible, jusqu'à retrouver un mur. Dès que le capteur détecte un mur, on repasse dans le cas précédent; par conséquent, notre robot ne se retrouve quasiment jamais face au mur qu'il devait longer. De plus, ceci présente l'avantage que notre robot est désormais capable de faire demi-tour autour d'un mur qui s'arrête d'un coup, sans virages (comme dans le cas des planches fournies par notre professeur pour tester le robot).

Dans le cas d'un virage à droite du mur, notre algorithme ne change pas.

Ainsi, nous avons un robot capable de longer les murs de manière très fiable, peu importe l'angle des murs.