Fiche conseil pour robots MindStorms

Qu'est-ce que LEJOS ?

Les robots MindStorms sont à l'origine développés pour être programmés facilement grâce à une interface graphique. Après avoir flashé le firmware LEJOS sur le robot NXT (ce qui est déjà fait sur les robots de l'IRIT) et installé le logiciel adéquat, on peut utiliser le langage Java pour programmer avec un plus haut niveau d'abstraction.

Ce qu'il faut installer

Sous Windows, on peut par exemple utiliser Eclipse avec l'extension LEJOS. Pour d'autres configurations, se reporter au site de Lejos : www.lejos.org.
Les différentes étapes (dans l'ordre !) :

1) Installer Java en version 32-bits ! (seule version compatible Lejos)

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
-> aller dans « Java SE Development Kit » puis « Windows x86 »

2) Installer Lejos

http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/GettingStarted.htm
->en suivant ce tutoriel, installer Lejos sur son ordinateur et installer les drivers USB
Si Lejos n'est pas flashé sur la brique NXT, on peut le faire comme indiqué sur le site.

3) Installer Eclipse (en version 32-bits !)

Lien pour le téléchargement :
https://www.eclipse.org/downloads/download.php?file=/oomph/epp/neon/R2a/eclipse-inst-win32.exe

4) Installer l'extension Lejos pour Eclipse

http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm
-> aller dans la catégorie « Installing the Eclipse plugin »
Ensuite, il faut importer le projet dans Eclipse, faire un clic droit dessus et faire Lejos NXJ> Convert to Lejos NXJ project. On peut alors commencer à programmer !

Les déplacements

Pour permettre à votre robot de se déplacer, rien de plus simple, il y a une librairie pour ça !
Il suffit d'importer lejos.robotics.navigation.DifferentialPilot et de créer dans votre classe un nouvel objet comme suit :

```
static DifferentialPilot pilote = new DifferentialPilot(wheelDiameter, trackWidth, motorG, motorD);
//wheelDiameter = diamètre d'une roue en mm
```

//trackWidth = écart entre les 2 roues en mm
//motorG = moteur gauche (par exemple Motor.C)
//motorD = moteur droit
Cette librairie propose par exemple des déplacements simples, des rotations et des déplacements en suivant des arcs.
Lien vers la documentation de la librairie :
https://lejos.sourceforge.io/nxt/nxj/api/lejos/robotics/navigation/DifferentialPilot.html

Quelques librairies utiles

Quelques librairies assez utilisées sont lejos.nxt. : Button (gestion des boutons de la brique NXT), LCD (gestion de l'écran LCD), LightSensor (gestion des capteurs lumineux), Motor (gestion des ports utilisés pour les moteurs : Motor.A à .C), NXTRegulatedMotor (gestion des moteurs), SensorPort (gestion des ports utilisés pour les capteurs : SensorPort.S1 à .S4), UltrasonicSensor (gestion des capteurs Ultrason).
On rencontre également souvent lejos.robotics.RegulatedMotor et lejos.util.Delay.
On retrouve la documentation des différentes librairies sur le site :
https://lejos.sourceforge.io/nxt/nxj/api/index.html

Tester son code

Il est très important de tester son code régulièrement afin de vérifier la fiabilité de sa programmation en situation réelle.
Pour tester en filaire :
1) Brancher le câble sur son ordinateur et sur le robot NXT
2) Sélectionner un Main et cliquer sur Run as > Lejos NXT Project
3) Une fois l'upload complet, le robot exécute le code
On peut alors débrancher le câble si on le souhaite.
On peut également utiliser une connexion Bluetooth au lieu du câble : il suffit d'appairer son ordinateur et son robot NXT, et de Run le code comme ci-dessus. Cependant, cela ne fonctionne pas à l'IRIT.

Gagner en précision

A part si vous êtes très chanceux, vous remarquerez qu'au bout de quelques rotations, votre robot est décalé… C'est parce que les paramètres de DifferentialPilot sont assez difficiles à définir. Le diamètre de la roue doit être mesuré avec la roue légèrement appuyée (comme elle l'est avec le poids du robot) et l'écart entre les roues doit être mesuré non par en partant du centre de chaque roue mais par rapport à un point de la roue légèrement décalé vers le centre du robot.
En réalité, le meilleur moyen d'améliorer la précision des déplacements et des rotations reste de régler les paramètres empiriquement : on modifie ces derniers graduellement et on teste si la rotation ou le déplacement effectué par le robot se rapproche de ce qui est voulu, et ce jusqu'à obtention d'un résultat satisfaisant. Je conseille personnellement de travailler surtout sur la rotation car il y a rarement des problèmes avec la distance de déplacements.

Quelques conseils

Concernant la structure du code, une simple séparation entre un Main qui sera Run sur la brique du robot NXT, et des classes qui seront utilisées dans ce main, convient.
Ajouter un while(true) suivi d'un Button.waitForAnyPress() ; dans le Main afin de pouvoir réexécuter son code facilement par simple pression sur un bouton.

Attention, Motor.X.stop() ; a un délai d'exécution assez élevé, ce qui fait que si on l'exécute pour le moteur d'une roue et tout de suite après pour le moteur de l'autre roue il y aura un décalage entre l'arrêt des 2 moteurs et donc une rotation involontaire du robot. Utiliser des forward() ; et des setSpeed(0) et setSpeed(une valeur plus élevée) permet d'éviter ce problème. On peut également utiliser pilote.stop() ;
Utiliser les robots avec peu de batterie peut affecter leur performance.
Les robots sont assez sujets aux dérapages, limiter leur vitesse et leur accélération peut être utile, ainsi qu'ajouter des élastiques autour de leur roues.
Il faut supprimer des programmes de la brique NXT de temps en temps afin d'éviter un problème d'espace mémoire.

À vous de coder !