Travaux Pratiques d'Automatique

ENSIMAG - Filière SLE

2009-2010

Mode de fonctionnement des TP et présentation du robot LEGO

1 Fonctionnement des TP

Les travaux pratiques se déroulent sur 4 heures. A l'issu de ces quatre heures, un compte rendu est à rendre aux enseignants vous encadrant. Ce compte rendu fera l'objet d'une note. Les enseignants sont là pour vous aider, vous conseiller, vous expliquer alors il serait dommage de ne pas faire appel à eux en cas de besoin...

2 Documentation de base

Pour plus de détails concernant le langage de programmation des robots LEGO mindstorms en NXC (Not eXactly C), nous vous conseillons de vous reporter à l'ouvrage de référence :

Not eXactly C (NXC) Programmer's Guide, John Hansen

qu'il est possible de télécharger à l'adresse :

`http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf`

Il comporte une liste des fonctions NXC disponible ainsi qu'une explication de la syntaxe de ce langage très proche du C.

3 Présentation du robot LEGO mindstorms NXT

3.1 Présentation générale

La gamme LEGO mindstorms NXT permet la construction modulaire de robots divers. Un robot est l'association d'une structure mécanique avec un ensemble de capteurs, d'actionneurs et d'un microprocesseur permettant de traiter les informations fournies par les capteurs afin d'en déduire les actions à effectuer afin de réaliser une mission donnée à priori.

3.1.1 Structure mécanique

Lors de cette série de travaux pratiques, nous nous intéresseront à une structure de robot roulant à trois roues. Deux de ces roues sont motorisées de manière indépendante, la troisième étant non motorisée. Les deux roues motorisées sont fixes tandis que la troisième est libre permettant ainsi au robot de tourner sur lui-même. Dans la littérature roboticienne, ce type de robot est souvent associé à l'unicycle (vélo à une roue largement utilisé dans les arts du cirque). Il est possible d'agir sur ce robot en faisant varier la vitesse de rotation des deux roues motorisées. Les effets produits seront :

- un effet de translation lié à la somme entre les deux vitesses de rotation des deux roues motorisées
- un effet de rotation du robot sur lui même lié à la différence entre les deux vitesses de rotation des deux roues motorisées

En schématisant le robot de la manière suivante :

3.1.2 Ordinateur central NXT

Les robots lego Mindstorms sont commandés au moyen d'un microcontrôleur central de type ARM doté d'un OS propre.

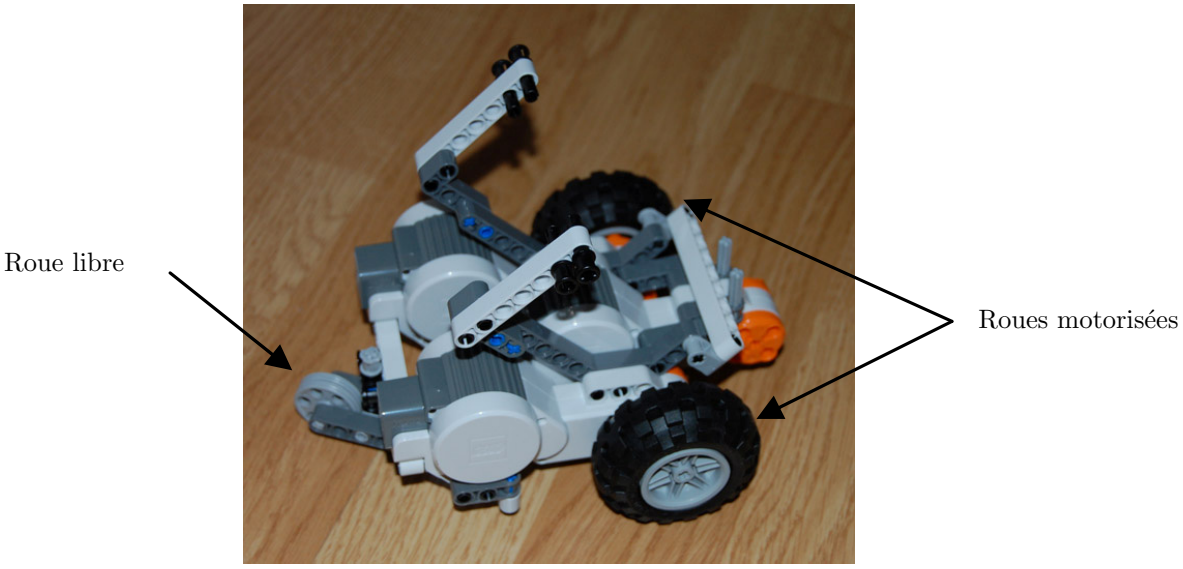

Figure 1 – Structure mécanique du robot

3.1.3 Les capteurs

a. Les différents type de capteurs

Sur cette structure, on peut monter différents types de capteurs tels que :

– **Capteurs à ultrasons :** Les capteurs à ultrasons émettent un signal sonore et en mesurant la réfection de celui-ci sur un obstacle permettent une mesurent de la distance séparant de l'obstacle. Ce capteur fournit un signal numérique en centimètre compris entre 0 cm et le rayon d'action du capteur. Le rayon d'action de ce type de capteur dépend de la puissance d'émission. En pratique, ce type de capteur voit difficilement des obstacles situés à plus d'1 à 1,5 mètre. Ce capteur est un capteur numérique à la norme I2C.

Figure 2 – Capteur à ultrasons

Programmation : Le capteur à ultrasons se programme de la manière suivante :

```
task main(){
      unsigned int distance;
      string msg;

      SetSensorLowspeed(IN_1);              // Initialise la communication
      while(true){                          // I2C sur l'entree 1
          distance = SensorUS(IN_1);        // Lecture de la mesure
          msg = NumToStr(distance);         // sur l'entree 1
```

```
        msg = StrCat("d = ",msg," cm");
        TextOut(0, LCD_LINE1, msg ,true); // Affichage sur la ligne 1
        Wait(100);                        // de l'ecran LCD
        }
    }
```

– **Capteurs lumineux :** Le capteur optique est un capteur qui mesure une intensité lumineuse (il ne s'agit pas d'une caméra). Il permet ainsi de différentier le blanc du noir et donc par exemple de faire suivre une ligne au sol au robot. Ceci se réalise en forçant le robot à maintenir une mesure d'intensité proche d'un seuil souhaité qui correspond à une mesure centrale de la ligne à suivre. S'il il s'en écarte, la mesure d'intensité variera. **Le capteur donne une mesure comprise entre 0 et 100**.

Figure 3 – Capteur optique

Programmation : Le capteur lumineux se programme de la manière suivante :

```
task main(){
    unsigned int intensite;
    string msg;

    SetSensorLight(IN_1);                // Initialise la communication
    while(true){                         // sur l'entree 1
        intensite = Sensor(IN_1);        // Lecture de la mesure
        msg = NumToStr(intensite);       // sur l'entree 1
        msg = StrCat("Intensite = ",msg); // Affichage sur l'ecran LCD
        TextOut(0, LCD_LINE1, msg ,true);
        Wait(100);
        }
    }
```

– **Capteurs sonores :** Le capteur sonore est l'analogue du capteur optique dans le domaine du son. Il renvoie une mesure d'intensité sonore et permet de programmer des fonctions réagissant à la voie. **Le capteur donne une mesure comprise entre 0 et 100**.
Programmation : Le capteur sonore se programme de la manière suivante :

```
task main(){
    string msg1,msg2;

    SetSensorSound(IN_1);
    while(true){
        until(SENSOR_1 > 60);
        msg1 = "Il y a trop";
        msg2 = "de bruit !";
        TextOut(0, LCD_LINE1, msg1 ,true);
        TextOut(0, LCD_LINE2, msg2 );
        Wait(1000);
```

Figure 4 – Capteur sonore

```
        msg1 = "Ah...c est mieux";
        TextOut(0, LCD_LINE1, msg1 ,true);
        }
    }
```

– **Capteur de contact :** Le capteur de contact est en fait un capteur de pression : il détecte une pression sur son extrémité orange. Il peut être utile dans les opération de préemption d'objet (fait d'attraper et de tenir un objet).

Figure 5 – Contacteur

Programmation : Le contacteur se programme de la manière suivante :

```
task main(){
      string msg;

      SetSensorTouch(IN_1);
      while(true){
          if (SENSOR_1 == 1)
          {  msg = "Aieuuuuuh";
             TextOut(0, LCD_LINE1, msg ,true);
             Wait(1000);
          }
          msg = "Ah...\c{c}a va mieux";
          TextOut(0, LCD_LINE1, msg ,true);
          }
```

```
}
```

b. Mode de fonctionnement des capteurs

Il existe 8 différents mode de fonctionnement des capteurs. Le mode de fonctionnement d'un capteur se détermine par la fonction `SetSensorMode()`. Les plus utilisés sont :

- `SENSOR_MODE_RAW` : dans ce mode, le capteur renvoie un entier entre 0 et 1023. Le sens physique de cette valeur dépend du capteur. Par exemple, le capteur de contact renvoie une valeur proche de 50 lorsqu'il est enfoncé au maximum tandis qu'il renvoie 1023 lorsque rien ne le touche. Il est donc possible d'ajuster la force de la préemption. Le capteur d'intensité lumineuse renvoie des valeurs comprises entre 300 (très lumineux) et 800 (très foncé).
- `SENSOR_MODE_PERCENT` : ce mode est équivalent au précédent en faisant correspondre 0 à 0% et 1023 à 100%. C'est le mode de fonctionnement par défaut du capteur d'intensité lumineuse.
- `SENSOR_MODE_BOOL` : dans ce mode, le capteur renvoie 0 ou 1. Le capteur de pression fonctionne par défaut dans ce mode. Tous les capteurs peuvent y être mis, il renverra 1 si la sortie en mode raw est au dessus de 562. Sinon, le capteur renverra 0.

Il existe des mode de fonctionnement spécifiques aux capteurs de températures (non utilisés dans ces TP) et des modes détectant des fronts montants, descendants ou les deux.

3.1.4 Les actionneurs

Le robot lego ne dispose que d'un seul type d'actionneur : les servomoteurs. Il s'agit d'un moteur électrique intégrant un asservissement en son sein.

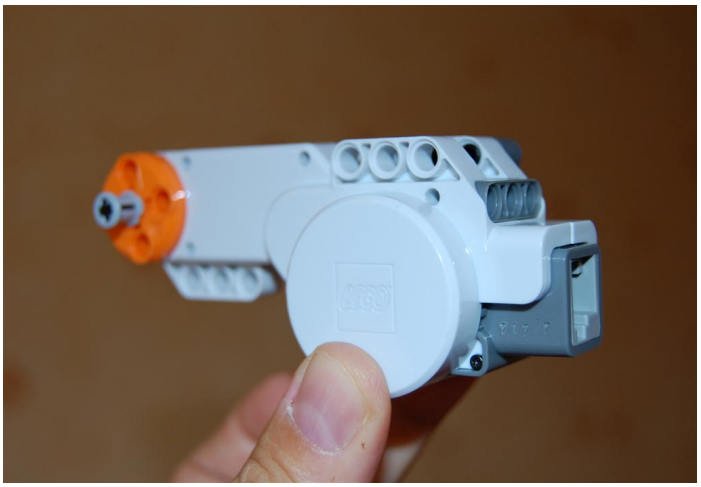

Figure 6 – Servomoteur

Programmation : Les servomoteurs sont programmables à l'aide des fonctions de base suivantes :

- Les 3 sorties du calculateurs NXT sont associées aux alias `OUT_A`, `OUT_B` et `OUT_C`. Il est ensuite possible d'envoyer des consignes simultanément à deux moteurs en utilisant les alias `OUT_AB` (pour les moteurs A et B) ou `OUT_AC` (pour les moteurs A et C) voire aux 3 en utilisant `OUT_ABC`.
- Les servomoteurs sont dotés de codeurs optiques qui permettent une mesure de la position angulaire des roues à 2π près. Pour obtenir ces informations, il faut utiliser les routines :
  ↪ `ResetTachoCount(`*alias_du_port*`)` pour ré-initialiser le compteur tachymétrique.
  ↪ `x = MotorTachoCount(`*alias_du_port*`)` pour récupérer la mesure tachymétrique.
  La mesure de la vitesse n'est pas disponible et doit être reconstruite à l'aide de la mesure de position. Ne pas oublier que le compteur tachimétrique a une limite et qu'il est nécessaire ou de le réinitialiser périodiquement.
- Une mesure de la commande est disponible en utilisant la fonction :
  ↪ `x = MotorActualSpeed(`*alias_du_port*`)`
- Une mesure de la consigne (identique à la commande en boucle ouverte) peut être obtenu grâce à la fonction :
  ↪ `x = MotorPower(`*alias_du_port*`)`

Les servo-moteurs disposent de plusieurs mode de fonctionnement :

- Le premier, en **boucle ouverte**, déconnecte le correcteur PID préprogrammé dans le firmware. On peut ainsi directement commander la puissance appliquée au moteur en utilisant la fonction :
  ↪ `OnFwdReg(`*alias_du_port*`,`*power*`,OUT_REGMODE_IDLE)`

qui applique la puissance *power* au moteur défini par l'alias *alias_du_port*. La fonction `OnRevReg` dont les paramètres sont les mêmes permet de faire tourner le moteur dans le sens opposé.

– Le second mode de fonctionnement est un mode en **boucle fermée**. La vitesse du moteur est alors rebouclée par un correcteur de type PID (Proportionnel, Intégral, Dérivé). On l'utilise de la manière suivante :
↪ `OnFwdReg(`*alias_du_port,speed,*`OUT_REGMODE_SPEED)`
La consigne *speed* de vitesse est alors automatiquement suivie par le moteur en compensant frottements, perturbations et autres phénomènes (dans la mesure du possible). Une autre fonction similaire permet d'éviter la réinitialisation du compteur tachimétrique : `OnFwdReg(`*alias_du_port,speed,*`OUT_REGMODE_SPEED,RESET_NONE)`

Les paramètres du contrôleur PID peuvent obtenus au moyen des fonctions :
↪ `MotorRegPValue(`*alias_du_port*`)`, `MotorRegIValue(`*alias_du_port*`)` et `MotorRegDValue(`*alias_du_port*`)`
qui fournissent respectivement les paramètres proportionnel, intégral et dérivée du régulateur du moteur correspondant à l'alias. Une manière alternative consiste à utiliser la fonction `GetOutput` (cf. Programmer's guide).
Ces paramètres peuvent être fixés au moyen de la fonction :
↪ `SetOutput(`*alias_des_ports,constante*$_1$*,val*$_1$*,...,constante*$_n$*,val*$_n$`)`
où *constante*$_i$ est `RegPValue`, `RegIValue` ou `RegDValue` pour régler respectivement les paramètres proportionnel, intégral et dérivée du régulateur du moteur correspondant à l'alias.

– Un troisième mode de fonctionnement existe afin de synchroniser les moteurs entre eux.

3.1.5 Quelques éléments autour de la notion de temps

L'OS propre au NXT n'est pas un OS temps réel. Il ne dispose donc pas de possibilité d'exécution périodique précise d'un programme. Cette fonction doit donc être réalisée "à la main" en utilisant les fonctions :

↪ `x=CurrentTick()` qui retourne la valeur en millisecondes de l'horloge interne (32-bit non signé)

↪ `x=FirstTick()` qui retourne la valeur en millisecondes de l'horloge interne au lancement du programme (32-bit non signé)

↪ `Wait(`*temps_en_millisecondes*`)`

Enoncé du TP

4 Identification du moteur LEGO

Les moteurs à courants continus sont des systèmes du second ordre, c'est à dire dotés de deux pôles. Le premier correspond à la partie mécanique du système tandis que le second correspond à la partie électrique du système. En pratique, la partie électrique est très rapide devant la partie mécanique et l'on peut approximer le système à un premier ordre :

$$\frac{\omega}{u} = \frac{G}{1+\tau p}$$

où ω est la vitesse angulaire du moteur et u sa tension d'alimentation (ou le ratio de PWM appliqué pour l'alimenter).

L'objectif de ces séance de TP est d'écrire le programme permettant d'identifier le système. Cette étape est utile (mais pas forcément nécessaire...) pour le réglage des correcteurs. Il permettra de mettre en place la structure générale du programme qui ensuite sera développé pour le suivi d'un mur ou d'une ligne.

5 Commande des moteurs

Implanter le correcteur PID pré-programmé sur le firmware afin d'asservir la position. Tester différents paramètres de réglage du PID. Que se passe-t-il pour

- $K_P = 40$, $K_I = 40$ et $K_D = 0$
- $K_P = 40$, $K_I = 40$ et $K_D = 80$
- $K_P = 40$, $K_I = 40$ et $K_D = 200$

Essayer de conclure sur l'influence des différents paramètres.

Trouver un réglage performant du PID en asservissement de vitesse et l'implanter. Quel est l'intérêt d'une boucle locale sur les moteurs dans le contexte d'un robot roulant ?