Association FReeLUG
Boite postale 3
91241 SAINT MICHEL SUR ORGE CEDEX

Initiation à la robotique LEGO© Mindstorms

Salon e-magiciens

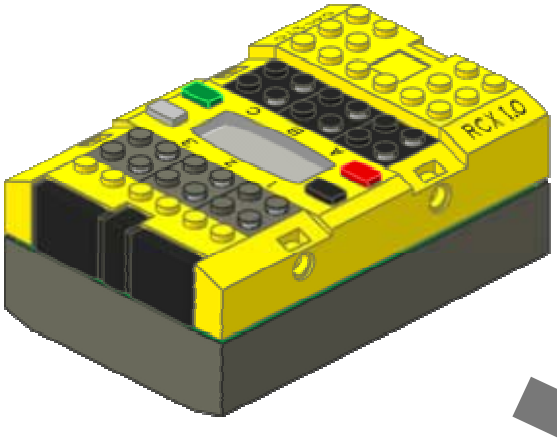
Valenciennes

24-26 novembre 2004

date : 24 novembre 2004
Version : 1.1

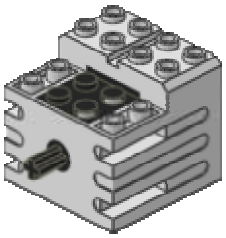
I) Présentation du matériel

Vous avez à votre disposition un kit LEGO Mindstorms référence 3804 comprenant :



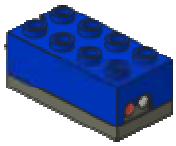
Une brique programmable (RCX)

1 borne Infrarouge (connectée au PC)



2 moteurs

2 capteurs de contact



1 capteur de lumière

Ainsi que de nombreuses pièces LEGO® afin de réaliser les montages.

Le cœur est donc le RCX qui contient un microprocesseur ainsi que de la mémoire.
Il peut stocker jusqu'à 5 programmes.

En haut, se situe les 3 entrées pour les capteurs (1, 2 et 3) et en bas les trois sorties pour les moteurs (A, B et C). Entre les deux séries de connecteurs, se situent les boutons de marche-arrêt, choix et exécution de programme.

Lors de l'exécution d'un programme, il peut exécuter 7+1 tâches en parallèle.

Nous vous proposons de séparer votre quadrinôme en 2 binômes dans un premier temps. Le premier binôme s'occupe de construire le modèle en suivant les instructions de montage et le second commence la programmation.

II) La programmation : NQC

Le kit Mindstorms fournit un outil de programmation LEGO. Cependant afin d'aller plus loin dans les possibilités, nous utiliserons le langage de Dave Baum, le NQC (Note Quite C) qui ressemble fortement au langage C tout en incluant des fonctions spécifiques pour la brique LEGO.

Attention : le langage s'inspirant du C signifie aussi que les majuscules et minuscules dans les mots clés sont importantes.

Nous vous fournissons une base de programme afin d'accélérer l'apprentissage (cf. page 8).

Le RCX peut exécuter 1+7 tâches, c'est à dire une tâche principale (main) et 7 tâches secondaires. Toutes les tâches s'exécutent en parallèle indépendamment les unes des autres.

Au début du programme, fournit page 8, nous avons fait les déclarations des moteurs et des capteurs :

Le moteur droit se brancher sur la sortie C et le moteur gauche sur la sortie A.

```
// Declaration des moteurs
#define moteur_gauche OUT_A
#define moteur_droit  OUT_C
```

Le capteur de contact (gris) sur l'entrée 1, et le capteur de lumière (bleu) sur l'entrée 2.

```
//Declaration des capteurs
#define  contact SENSOR_1
#define  lumiere SENSOR_2
```

La tâche main est écrite et déclare les 2 capteurs qui seront utilisés par la suite.

Le langage NQC est extrêmement proche du langage C, notamment dans sa syntaxe et la façon d'écrire un programme. Ci-dessous un rappel rapide des principes de bases.

a) Bases

```
// ceci est un commentaire
int x ;        // déclaration d'une variable de type Integer
x = 2 ;        // x prend la valeur 2, ne pas oublier le ; après une instruction
x = 3 *3 ;     // x prend la valeur 9
```

Operateurs : `=` `*` `/` %(modulo) `+` `-` `+=` `-=` `*=` `/=`

b) Conditions

Une *condition* est écrite de la façon suivante :

expr 1 *operateur* expr 2 et rend pour résultat vrai ou faux (`true` / `false`)

exemples : `a == 5` `b >= 12`

Les *operateurs* disponibles sont les suivants :

expr 1 == expr 2	vrai si expr1 est égal à expr 2
expr 1 *!*= expr 2	vrai si expr 1 est diffèrent de expr 2

expr 1 < expr 2	vrai si expr 1 est plus petit à expr 2
expr 1 <= expr 2	vrai si expr 1 est plus petit ou égal à de expr 2
expr 1 > expr 2	vrai si expr 1 superieur à expr 2
! condition	négation logique de la condition

action : mise en fonctionnement / arrêt d'un moteur …

```
if ( condition ) {
   action(s)
} else {
   action(s)
}

while (condition) {
   action(s)
}
```

`until (condition) ;` attend que la condition soit vraie pour continuer

par exemple : `until (SENSOR_1 == 1) ; // attend un appuie sur le bouton`

`break;` Sort immédiatement d'une boucle `while` ou d'un `if`.

c) autres commandes utiles

`Wait(100);`	attend 1 seconde
`Random(10);`	rend un résultat aléatoire entre 0 et 10 inclus.
`Wait(Random(100));`	attend un temps aléatoire entre 0 et 1 seconde

`PlaySound(son)` Joue le « son »

`son` => SOUND_CLICK, SOUND_DOUBLE_BEEP, SOUND_DOWN, SOUND_UP, SOUND_LOW_BEEP, SOUND_FAST_UP.

Pour de plus amples informations sur NQC et ses possibilités, des guides et livres sont disponibles auprès de l'équipe de FreeLUG.

III) Initiation – les moteurs

Afin de mettre en route et d'agir sur le sens de rotation des moteurs, vous disposez des instructions ci dessous.

Par défaut, les 3 moteurs sont dans le mode « avant » et pleine puissance.

outputs = sorties (OUT_A, OUT_B, et OUT_C). Dans notre cas, puisqu'elles ont étés prédéfinies, il s'agit de `moteur_droit` et `moteur_gauche.` Il est possible d'additionner les sorties de la manière suivante : OUT_A + OUT_C

`On(outputs)`	Mise sous tension des sorties.
`Off(outputs)`	Eteint les sorties, moteurs bloqués.
`Float(outputs)`	Eteint les sorties, moteurs libres.
`Fwd(outputs)`	Positionne le sens de rotation du moteur en « avant »
`Rev(outputs)`	Positionne le sens de rotation du moteur en « arriere »
`Toggle(outputs)`	Inverse le sens de rotation
`OnFwd(outputs)`	Met sous tension dans le sens de rotation avant
`OnRev(outputs)`	Met sous tension dans le sens de rotation arriere
`OnFor(outputs, temps)`	Met sous tension pour un temps donné (temps)

`SetPower(outputs, power)` Puissance de rotation du moteur de 0 à 7 ou en utilisant les constantes OUT_LOW, OUT_HALF, et OUT_FULL. Attention, il ne s'agit pas de la vitesse, mais du couple.

a) Votre tâche est donc d'écrire le corps du programme « action » afin de mettre en route les deux moteurs afin que le robot avance.
Le fait de mettre sur `On()` les moteurs, lance la commande et passe à l'instruction suivante. Il faut donc explicitement arrêter les moteurs par la commande `Off()`.

Télécharger votre programme et appuyer sur le bouton « run » (en vert) sur le RCX.

b) Dans un second temps, séparez les commandes des moteurs droit et gauche afin de faire naviguer votre robot en zigzags. Faite le tourner, soit en laissant un moteur en roue-libre `Float()`, soit en les faisant tourner dans des sens différents.

IV) Initiation – capteur de contact

Dans cette partie, vous allez apprendre à utiliser le capteur de contact. Dans un premier temps, les moteurs seront en fonctionnement uniquement lorsque le bouton sera maintenu appuyé. Dans un second temps, une pression mettra le robot en mouvement et une seconde devra l'arrêter.

Le nom du capteur a été défini précédemment :

```
#define  contact SENSOR_1
```

Le capteur répondra maintenant sous le nom de « contact ».

Le type du capteur a été défini dans le `main()` par la ligne suivante :

```
SetSensor(contact, SENSOR_TOUCH);
```

Ce capteur est au niveau logique 1 lorsque l'on appuie dessus et 0 sinon.

a) Moteurs en route si contact

Vous devez utiliser une boucle `while` infinie du style :

```
while (true) {
}
```

Elle a pour but de rendre la tâche infinie. A l'intérieur, vous devez écrire un test qui vérifie que le capteur est appuyé ou pas.
Si un moteur est déjà en route et que `On(moteurs)` est à nouveau lancé, la commande n'a pas d'effet.

b) Contact = bouton d'allumage et d'arrêt

Toujours à l'intérieur d'une boucle `while` infinie. Vous attendez d'avoir une pression sur le capteur de contact, qui a pour effet de lancer les moteurs. Une seconde pression sur le contact doit les arrêter.

Il est fortement conseiller de mettre un `Wait(10);` après le `On(moteurs)` et le `Off(moteurs)` afin d'éviter l'effet « collant » du contact.

Vous pouvez faire le test du capteur de contact, soit dans une boucle `while () {}` sans action, soit dans un `until();`.

V) Initiation – capteur de lumière

Le capteur de lumière permet de détecter des changements important dans la couleur du sol. Pour cette partie, nous utiliserons en plus la piste papier.
le but est d'avoir un robot qui :

- avance,
- s'arrête si le capteur de lumière constate que le sol change de couleur (blanc vers noir),
- recule,
- tourne d'un coté et avance à nouveau

Le nom du capteur a été défini précédemment :

```
#define  lumiere SENSOR_2
```

Le capteur répondra maintenant sous le nom de « lumiere ».
Le type du capteur a été défini dans le `main()` par la ligne suivante :

```
SetSensor(lumiere, SENSOR_LIGHT);
```

Le capteur de lumière renvoi au RCX une valeur (entre 0 et 100) qui correspond à la proportion de lumière réfléchie. Le capteur comporte une LED qui éclaire la surface et un deuxième composant qui lit l'indice de réflexion.

Afin de déterminer la valeur lue par le capteur, placez le robot sur une zone blanche de la feuille. Mettez le RCX en route et appuyez sur VIEW. La première pression fait apparaître un chevron sur l'écran sous le 1. Une deuxième pression place le chevron sous le 2 et sur l'affichage apparaît la valeur lue.

L'utilisation de la variable « lumiere » force une lecture. par exemple :

```
If (lumiere > 60) {…}
```

Etapes de programmation

a) Dans un premier temps, le robot avance en lisant la valeur. Lorsque la valeur passe sous un seuil (qui correspond au passage de papier blanc à noir) le robot s'arrête. Fin du programme.

b) Même comportement que a), mais cette fois le robot recule pendant un certain temps, puis tourne et avance à nouveau. Fin du programme.

c) similaire a b) mais cette fois le faire en une boucle infinie, le robot ne s'arrête plus, et reste dans la zone interne à la boucle noire imprimée sur la feuille.

d) utilisez la fonction `Random()` pour décider si le robot doit tourner à droite ou à gauche et de quel angle.

d) Au lieu de figer vos valeurs blanc et noir, le robot commence par lire la valeur de la ou il se trouve. Faites une moyenne entre le noir/blanc lu manuellement pour connaître la valeur approximative de seuil. L'idéal est de programmer sous forme d'hystérésis.

Bonus : modifiez votre programmation pour que le robot suive la ligne noire imprimée.

VI) Programme de base

```
// Initiation a la robotique
// (c) FreeLUG 2004
// http://www.freelug.org
//

// Declaration des moteurs
#define moteur_gauche OUT_A
#define moteur_droit  OUT_C

//Declaration des capteurs
#define  contact SENSOR_1
#define  lumiere SENSOR_2

task main() {
     SetSensor(lumiere, SENSOR_LIGHT);
     SetSensor(contact, SENSOR_TOUCH);

     start action;  // démarre la tâche action
}

task action() {

//  A vous de remplir cette section avec votre programmation
//  Vous n'avez pas a toucher au reste.



}
```

VII) Références

Site de NQC :

http://www.baumfamily.org/nqc_old/index.html (anciennes versions)
http://bricxcc.sourceforge.net/nqc/

Site de BrickCC : http://bricxcc.sourceforge.net/

[Auteurs]

Gaël FRAZIER gael.frazier@freelug.org
Jean Louis BERGAMO jlb@freelug.org