Cours ISC - 2A
Note sur la génération de code NXC pour LEGO

Thao Dang

CNRS-VERIMAG, 2, av. de Vignate, 38610 Gieres, France

1 Calcul avec des entiers

Puisque NXC ne permet pas de flottants, on peut donc utiliser des des entiers pour coder des nombres en virgule fixe. Pour ceci, on multiplie des nombres avec un facteur d'échelle fixe avant des calculs arithmétiques. A l'étape finale, il faut diviser le résultat par ce facteur.

Il faut alors prendre en compte les bornes inférieure et extérieure des entrées (les données fournies par des capteurs de lumière) et des sorties (les vitesses des deux roues) du modèle de contrôleur en SIMULINK. Voir l'annexe pour un petit manuel.

2 Vitesses du robot

Pour communiquer les vitesses des deux roues (calculées par le contrôleur) aux moteurs, en NXC on peut utiliser les commandes **OnFwd** and **OnRev** où les vitesses sont spécifiées par le pourcentage de la vitesse maximale, qui dépend de l'état de la batterie. Pour information, nos essais ont montré qu'à 75% de la vitesse maximale, la vraie vitesse du robot pourrait varier de $20cm/s$ à $28cm/s$. Notez que celle-ci dépend aussi du type de revêtement du sol.

Pour plus d'informations, voir le manuel de NXC disponible sur la page

http://bricxcc.sourceforge.net/nbc/

3 Structure du programme NXC - Exemple

Pour plus d'informations concernant des aspects temps-réel, voir le document

http://www-verimag.imag.fr/~raymond/edu/lego/lustre4lego/lustre4lego.pdf

```
/*
Lustre output procedures must be defined by the user
before including the Lustre code.
The names and profiles can be deduced from the Lustre file:
<node-name>-O-<var-name>(<var-type>)
*/
void controleur_O_vg(int V) {
 NumOut(0, LCD_LINE3, V);
 if (V>100) V=100;
 if (V<-100) V=-100;
 OnFwdRegEx(OUT_A, V, OUT_REGMODE_SPEED, RESET_NONE);
}
void controleur_O_vd(int V) {
 NumOut(0, LCD_LINE4, V);
 if (V>100) V=100;
 if (V<-100) V=-100;
 OnFwdRegEx(OUT_B, V, OUT_REGMODE_SPEED, RESET_NONE);
}


/*
We include the (compiled) Lustre code, which contains the function
for reading inputs and the step function.
*/
#include "controleur.ec2nxc"

task main () {
```

```
SetSensorLight(IN_4);
SetSensorLight(IN_3);

int i=0;
int j=0;
int k=0;

int cycles_counter=0;

while (cycles_counter < 3000) {

  // Prepare and launch a step...
  cycles_counter++;

  // Read the inputs.
  j=Sensor(IN_4);
  k=Sensor(IN_3);

  // Read the inputs. This function is defined in "controleur.ec2nxc" and must
  // before calling the step procedure.
  controleur_I_x(j, k );

  // The step function, which execute the operations of one cycle.
  // This function is defined in "controleur.ec2nxc"
  controleur_step();
 }
}
```