Modèles de programmation : LUSTRE
TP 3 : programmation d'un robot Lego Mindstorms EV3

Le but de ce TP est de programmer en Lustre le contrôle d'un robot devant naviguer dans une pièce et manœuvrer en cas d'obstacles.

1 Description de la brique Lego Mindstorms EV3

Fig 1. Brique Lego Mindstorms EV3

Le robot que l'on souhaite programmer repose sur la brique Lego Mindstorms EV3. Cette brique est sortie en 2013 à la suite de la brique RCX (1998) et de la brique NXT (2006). Elle contient

- Un écran monochrome LCD 178x128 pixels.
- Un processeur Texas Instrument AM1808 mono-core reposant sur la technologie ARM9 (cadencé à 300MHz).
- Une mémoire principale de 64 MB, ainsi qu'une mémoire Flash de 16MB, et un emplacement pour une carte SD de 32 MB maximum.
- Deux ports USB, une connexion Wifi, une connexion Bluetooth.
- Quatre ports, numérotés de 1 à 4, pour brancher des capteurs Lego.
- Quatre ports, numérotés A à D, pour brancher des moteurs Lego.

Un des gros intérêts de cette brique est qu'elle peut tourner sous Linux (installé sur la carte SD) et que les bibliothèques pour lire les capteurs et piloter les moteurs sont disponibles en open-source à l'adresse https://github.com/in4lio/ev3dev-c (plusieurs exemples de programmes sont donnés en plus des drivers des moteurs et des capteurs). Toute la documentation et les tutoriels pour utiliser cette brique sous Linux est accessible sur le site https://www.ev3dev.org.

2 Description du robot

2.1 Architecture matérielle : capteurs et actionneurs

Le robot à piloter est présenté en figure 2. Il est composé de :

- Deux roues motrices : une roue droite et une roue gauche (l'équilibre arrière est assuré par une bille non motrice).
- Deux capteurs de touché, qui au moyen d'un dispositif de type « pare-chocs » (ou moustache de chat), permettent de détecter une collision avec un obstacle. Ces deux capteurs sont respectivement devant en bas à droite et devant en bas à gauche. Lorsqu'un capteur est enfoncé il envoie un signal « true » sur le port sur lequel il est branché. Lorsque le capteur n'est pas enfoncé, il envoie un signal « false ».

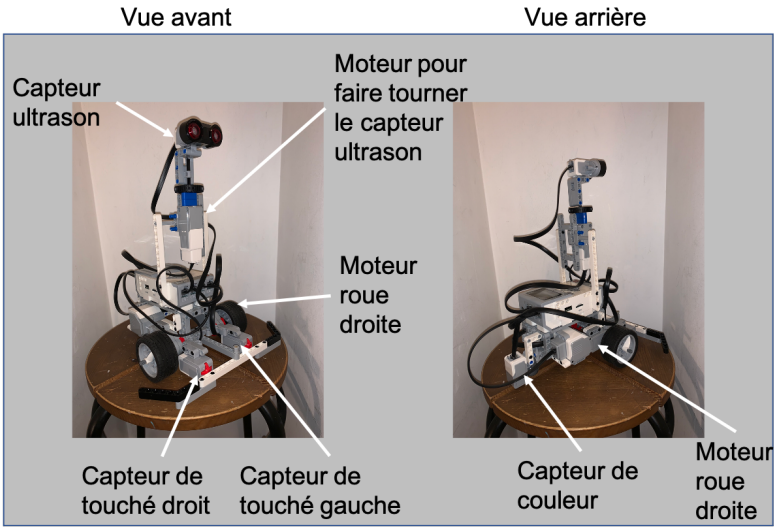

Fig 2. Vue avant et arrière du robot

Les capteurs et moteurs sont branchés comme suit :

- Le moteur de la roue droite est branché sur le port A.
- Le moteur de la roue gauche est branché sur le port B.
- Le capteur de touché droit est branché sur le port 1.
- Le capteur de touché gauche est branché sur le port 4.

Les moteurs des roues sont commandés en vitesse, vers l'avant ou vers l'arrière.

Le but du travail est de programmer le processeur du robot de manière à piloter les moteurs des roues en fonction des informations reçues sur les capteurs de touché (à droite et à gauche).

2.2 Architecture logicielle

Comme dit ci-dessus, la brique Lego EV3 intègre un processeur ARM9 sous Linux. Le programme de contrôle sera exécuté sur ce processeur. Ce programme est nommé `robot_isae.c`. Il est écrit en C. Son architecture est décrite ci-dessous :

1. Initialisation du robot, test de la présence des moteurs et des capteurs, identification des ports sur lesquels ils sont branchés. Initialisation du booléen « alive » à vrai.
2. Tant que alive=vrai
    a. Lecture des valeurs sur les capteurs (touché, ultrason, couleur).
    b. Exécution de la fonction de contrôle (calcul des ordres à envoyer sur les trois moteurs en fonction de la valeur des capteurs et de l'état courant du robot).
    c. Envoi des ordres sur les trois moteurs.
    d. Test de l'appui sur le bouton « back » : si appui alors alive=false, sinon rien.
    e. Attente de 10ms.
3. Fin tant que

Ce programme est organisé comme une boucle pseudo périodique d'au moins 10ms. L'intelligence du programme se situe à la ligne 2.b. **C'est cette partie que l'on demande de réaliser en Lustre**.

Toutes les autres parties du programme sont écrites en C (séquentiel), et sont déjà fournies. Vous n'aurez donc qu'à écrire le code Lustre réalisant la ligne 2.b. (Mais si vous

souhaitez aller voir et modifier le code C, le programme `robot_isae.c` est accessible et ouvert).

2.3 La mission à réaliser

Le but de ce robot est de naviguer au hasard dans une pièce en naviguant s'il touche un obstacle.

Pour ce faire, il roule droit devant lui :

- Jusqu'à ce qu'il rencontre un obstacle, dans ce cas il cherche à l'éviter.
- Ou jusqu'à ce qu'un timer se déclenche, lui disant qu'il a déjà beaucoup roulé vers l'avant sans rencontrer d'obstacle, dans ce cas il décide de tourner à droite (ou à gauche, c'est vous qui choisissez) puis et repart en réactivant le timer.

La détection d'obstacle est faite via les capteurs de touché.

- Si le capteur de touché droit a été enfoncé (un obstacle a été rencontré à droite), alors
    - le robot coupe les moteurs de ses roues, et attend que les roues soient effectivement arrêtées,
    - recule pendant un certain temps (durée à choisir),
    - coupe de nouveau les moteurs de ses roues, et attend de nouveau que les roues soient effectivement arrêtées,
    - tourne vers la gauche (puisque l'obstacle est à droite) pendant un certain temps (durée à choisir)
    - coupe de nouveau les moteurs de ses roues, et attend de nouveau que les roues soient effectivement arrêtées,
    - et repart droit devant lui.
- Idem si le capteur gauche a été touché, mais en tournant vers la droite.

Remarque : pour tourner à droite, il suffit de commander la roue droite vers l'arrière et la roue gauche vers l'avant. Idem pour tourner à gauche (roue droite vers l'avant et route gauche vers l'arrière).

Une des difficultés est de respecter l'inertie des moteurs : avant de changer un moteur de direction, il est nécessaire d'attendre qu'il soit arrêté. Chaque moteur contient un capteur de vitesse de rotation qu'il est possible de mesurer. Il faudra toujours attendre que ce capteur retourne zéro avant de modifier le sens de rotation du moteur.

2.4 L'architecture du code Lustre

Le nœud du robot doit être spécifié de la manière suivante :

```
--------------------------------------------------------------------------------
-- Main node
--------------------------------------------------------------------------------
-- Inputs:
--   sensors 1 to 4 (from ports IN1 to IN4)
--   speed of motors A to B (from ports OUTA to OUTD)
-- Assumption:
--   sensor_1 = right touch sensor
--   sensor_4 = left touch sensor
--   sensor_2 = color sensor
--   sensor_3 = utrasonic sensor
--   motor_speed_sensor_a = speed of motor A (right wheel)
--   motor_speed_sensor_b = speed of motor B (left wheel)
--   motor_speed_sensor_b = speed of motor C (engine to turn the ultrasonic sensor right and
left)
--   no motor on output port D
--------------------------------------------------------------------------------
node main_robot (
               sensor_1 : bool ;
               sensor_2 : int ;
               sensor_3 : int ;
               sensor_4 : bool ;
```

```
               motor_speed_sensor_a : int ;
               motor_speed_sensor_b : int ;
               motor_speed_sensor_c : int ;
               motor_speed_sensor_d : int ;
               )
--------------------------------------------------------------------------------
-- Outputs:
--   Required rotation direction for each motor (clockwise versus counterclockwise)
--   Required command mode for each motor (speed versus position)
--   Required value of the command for each motor
-- Assumption
--   Motor_A = right wheel
--   Motor_B = left wheel
--   Motor_C = engine to turn the ultrasonic sensor right and left
--   no motor on output port D
--------------------------------------------------------------------------------
returns (
       Clockwise_A : bool; Counterclockwise_A : bool;
       Cmd_Position_A : bool ; Cmd_Speed_A : bool; Value_A : int;
       Clockwise_B : bool; Counterclockwise_C : bool;
       Cmd_Position_B : bool ; Cmd_Speed_B : bool; Value_B : int;
       Clockwise_C : bool; Counterclockwise_B : bool;
       Cmd_Position_C : bool ; Cmd_Speed_C : bool; Value_C : int;
       Clockwise_D : bool; Counterclockwise_D : bool;
       Cmd_Position_D : bool ; Cmd_Speed_D : bool; Value_D : int;
       );
--------------------------------------------------------------------------------
```

Note : il est impératif que le nœud principal de votre programme se nomme « main_robot » (sinon, il vous faudra changer le programme C `robot_isae`).

Il est aussi impératif de respecter le nommage des flots d'entrée et de sortie ci-dessus. Ces flots sont écrits et lus par dans le programme `robot_isae`.

Remarque : le robot que l'on demande de programmer ne contient que deux moteurs. Les sorties C et D ne sont donc pas utilisées. Mais par souci d'évolutivité, votre programme retournera malgré tout des flots pour ces moteurs C et D (en les mettant à false (pour les flots booléens) et zéro (pour les flots entiers)).

3 Travail à faire

Votre travail consiste donc à concevoir et coder en Lustre le nœud « main_robot » qui code la fonction de contrôle du robot.

Conseil : pour programmer ce robot, vous aurez sans doute besoin de

- une minuterie qui décompte le temps (le nœud « stable » vu en TP1)
- un détecteur de fronts montants ou fronts descendants (vu en TP1)
- une bascule RS (le nœud RS_switch vu en TP1)

4 La chaîne de compilation / déploiement / exécution

Le cycle de développement que vous allez suivre comporte deux étapes :

- D'une part la partie Lustre proprement dite, réalisée sur le PC « host » (votre station de travail), et au cours de laquelle vous utiliserez les outils de la toolbox Lustre.
- Et d'autre part, le portage du code généré (le code C) sur le robot, sa compilation et son exécution.

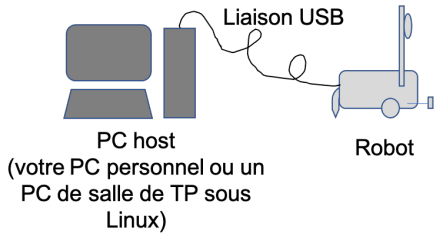

Fig 3. PC host + robot

4.1 Etape 1 : sur le PC host

L'étape 1 consiste à concevoir et développer le nœud « `main_robot` » (et les sous-nœuds appelés par celui-ci si vous le pensez nécessaire).

Consigne :
- appeler votre fichier Lustre « `tp3_robot.lus` » (fourni)
- indiquer au début du fichier en commentaire vos prénoms et noms
  => votre fichier doit commencer par quelque chose comme

```
--------------------------------------------------------------------------------
-- TP3 Robot Lego Mindstorms EV3
-- Archibald Haddock, Tryphon Tournesol
-- ENSEEIHT - 2022-2023
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
-- Main node
--------------------------------------------------------------------------------
-- Inputs:
--   sensors 1 to 4 (from ports IN1 to IN4)
--   speed of motors A to B (from ports OUTA to OUTD)
-- Assumption:
-- …
```

Au cours de cette étape, vous utiliserez les outils
- `lustre` : pour vérifier la syntaxe, le typage, la causalité et les horloges (si vous en utilisez) de votre programme.
- `luciole` : pour faire des simulations (si besoin).
- `lesar` : pour faire des vérifications (si besoin).

Lorsque vous voulez tester votre programme sur le robot, il est nécessaire de générer le code C correspondant. Pour ce faire, vous utiliserez
- `lus2c` : pour générer `main_robot.c` et `main_robot.h`

Si votre fichier lustre contient des objets externes (des constantes non initialisées dans le fichier lustre, ou des nœuds externes), il faut les définir dans un fichier additionnel, écrit en C, appelé :
- `main_robot_ext.h`

Sortie de l'étape 1 :
- le fichier `tp3_robot.lus` complété avec votre code Lustre.
- le code généré `main_robot.c` et `main_robot.h`
- le fichier additionnel `main_robot_ext.h` (si besoin)

4.2 Etape 2 : sur le robot

La seconde étape se passe sur le processeur du robot. On va s'y connecter via une session à distance ssh.

Pour booter le robot, appuyer sur le bouton central de la brique EV3 (cela peut prendre 1 à 2 minutes).

4.2.1 Connection du robot au PC host

Le robot se connecte au PC host via un port USB. Connecter le robot à un port USB du PC host.

Sur le PC Host, dans les paramètres réseaux, aller dans la connexion réseau Ethernet USB.
Aller dans les paramètres de la connexion filaire.
Aller dans IPV4 et sélectionner « réseau local uniquement », désactiver DNS et Routes.

Aller ensuite sur l'interface du robot. Repérer l'adresse IP du robot via l'écran du robot en navigant avec les boutons :
- Aller dans le menu « Wireless and networks »,
- puis dans « All network connections »,
- puis dans « Wired ».
- Si l'état est Disconnected, faites « Connect ».
- Lorsque le robot est « Connected », aller dans IPV4 et noter l'adresse IP.

Faites

```
ping adresse-ip-du-robot
```

pour vérifier que le robot est bien connecté et accessible depuis le PC host.
Si ça ne marche pas, déconnectez (sur le PC host ou sur le robot) et recommencez. Ça devrait finir par marcher.

4.2.2 Login sur le robot (via une connection ssh)

Pour rappel, l'OS tournant sur ce processeur est un Linux (Debian). Pour s'y connecter, il faut utiliser le compte :

- Login : robot
- Pwd : maker

Dans un terminal sur le PC host, se connecter au robot via ssh en faisant

```
ssh robot@adresse-ip-du-robot
```

Il est possible que votre PC vous demande :

```
Are you sure you want to continue connecting (yes/no)?
```

Vous répondez « yes », puis vous entrez le mot de passe (maker).
Vous devriez voir dans votre terminal le résultat suivant :

```
Linux ev3dev 4.14.117-ev3dev-2.3.5-ev3 #1 PREEMPT Sat Mar 7 12:54:39 CST 2020 armv5tejl
             _____     _
   _____   _|___ /  __| | _____   __
  / _ \ \ / / |_ \ / _` |/ _ \ \ / /
 |  __/\ V / ___) | (_| |  __/\ V /
  \___| \_/ |____/ \__,_|\___| \_/

Debian stretch on LEGO MINDSTORMS EV3!
Last login: Fri Sep 16 00:43:27 2022 from fe80::7831:c1ff:fe8b:c164%usb0
robot@ev3dev:~$
```

Votre session tourne maintenant sur le processeur du robot.

Aller alors dans le répertoire

```
/home/robot/ev3dev-c/eg/robot_isae
```

Ce sera votre répertoire de travail (copie, compilation et exécution) sur le robot.

Vous y copierez le code C généré par Lustre.

4.2.3 Copie et compilation des codes sur le robot

Il vous faut donc copier et compiler les codes (.c et .h) sur le robot.

La copie de ces codes se fait depuis le PC host par :

```
scp main_robot.c robot@adresse-ip-du-robot:ev3dev-c/eg/robot_isae/.
scp main_robot.h robot@adresse-ip-du-robot:ev3dev-c/eg/robot_isae/.
scp main_robot_ext.h robot@adresse-ip-du-robot:ev3dev-c/eg/robot_isae/.
```

Il devrait y avoir déjà dans ce répertoire un autre fichier appelé

```
robot_isae.c
```

Il s'agit du programme C global, déjà mentionné précédemment, englobant votre programme Lustre. Vous n'avez pas besoin a priori de modifier ce fichier. Par contre, il faut qu'il soit présent. Si ce n'est pas le cas, récupérez-le et recopiez le sur le robot par la même procédure que les fichiers précédents.

Pour compiler, il suffit ensuite d'utiliser le makefile et faire

```
make release
```

Cette commande appelle le compilateur gcc pour le processeur ARM9 du robot sur les différents fichiers. Vous pouvez avoir des warnings sur des variables non utilisées dans le code `robot_isae.c`. Ce n'est pas votre faute, donc ignorez ces warnings.

Cette opération génère (si tout se passe bien) un exécutable dans le répertoire Release.

4.2.4 Exécution du programme

Pour lancer le programme, le plus simple est de le faire dans le terminal en allant dans le répertoire

```
/home/robot/ev3dev-c/eg/robot_isae/Release
```

Exécuter le binaire

```
robot_isae
```

(attention, votre robot va sans doute démarrer. Pensez à le mettre sur des cales afin que ses roues ne touchent pas le sol).

Pour arrêter le robot, appuyer sur la touche « back » sur la brique EV3.

L'autre façon de lancer le robot est de le faire via l'interface du robot :

- Aller dans « File Browser »
- Aller dans « ev3dev-c »
- Aller dans « eg »
- Aller dans « robot_isae »
- Aller dans « Release »
- Et lancer « robot_isae »

Bon travail !