Introduction rapide et incomplète à Simulink Real time

Génération de code automatique et noyau temps réel

Vincent MAHOUT

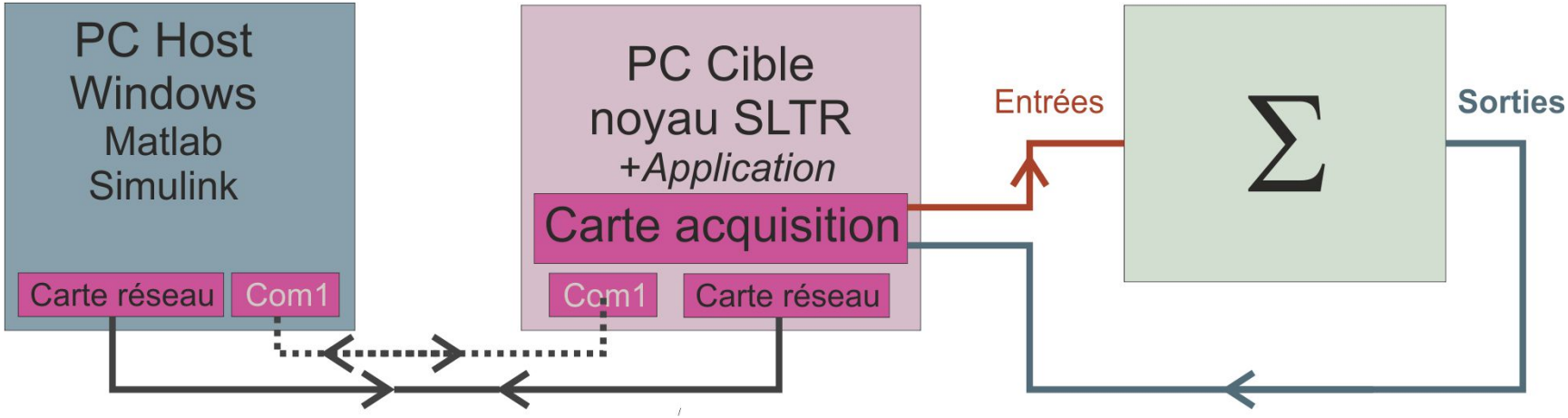
March 13, 2017

Idée de base

- A partir du correcteur conçu sous Simulink : générer directement le code correspondant et le charger dans une cible autonome
- Faire tourner la cible sous un noyau temps réel
- Avoir les outils nécessaires pour dialoguer entre l'unité de développement et la cible
  - Récupération de données expérimentales
  - Modification de paramètres (correcteur, consigne,..)
  - Création d'une interface utilisateur au besoin

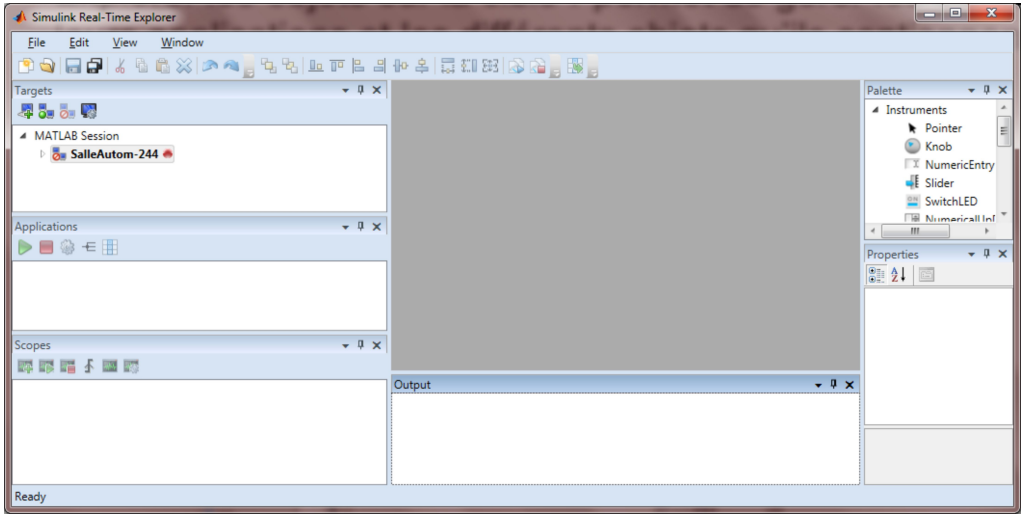
La configuration

- Trois entités différentes :
  - Le PC de développement
  - Le PC cible (ou microcontrôleur,...)
  - Le système à contrôler

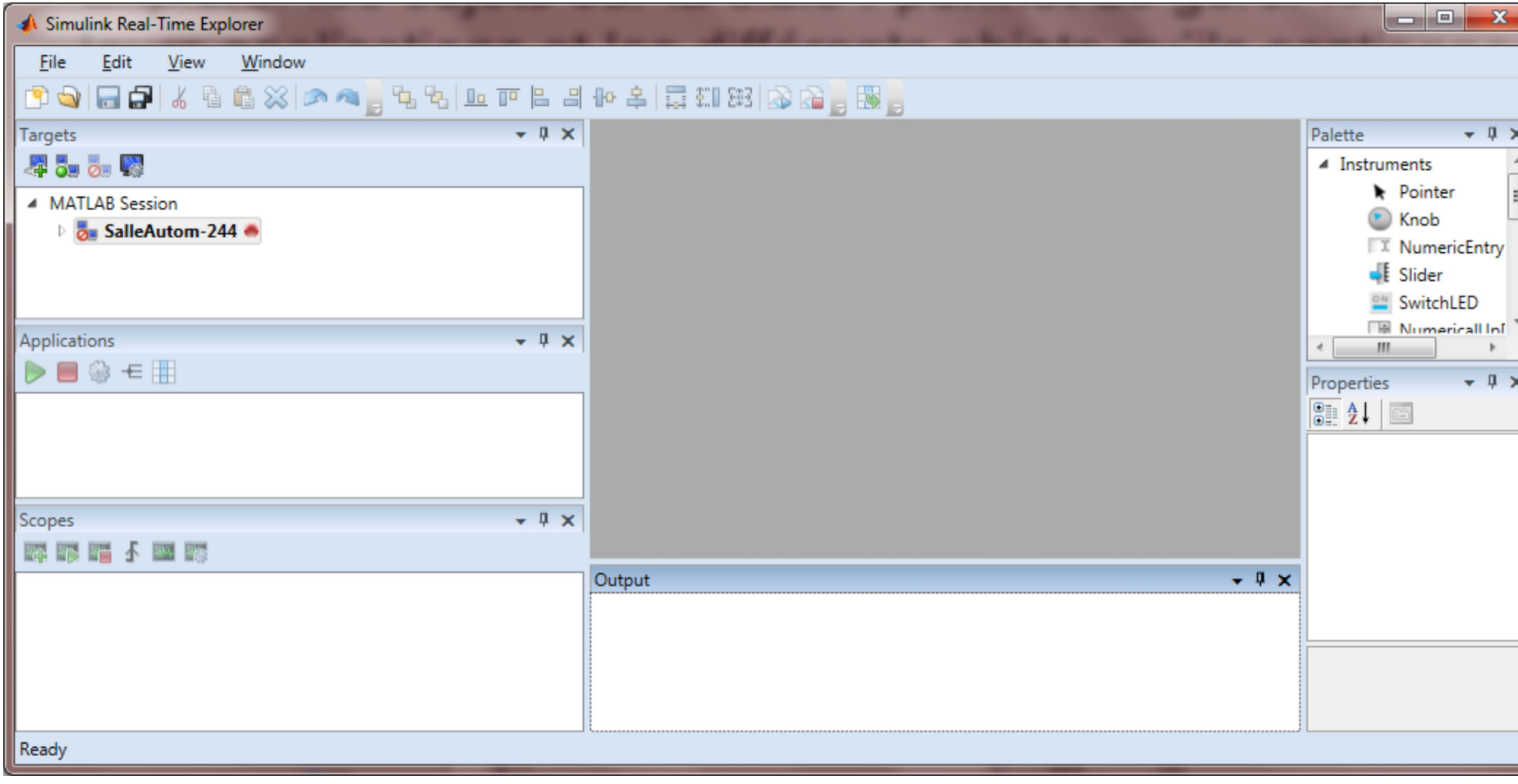


Configuration

- Pour "jouer" avec SLRT, il existe un *navigateur* qui permet (entre autre) :
  - gérer les connexions aux cibles ;
  - explorer les objets sur la cible comme les applications et les différents "objets" qui leur sont associés .



Configuration

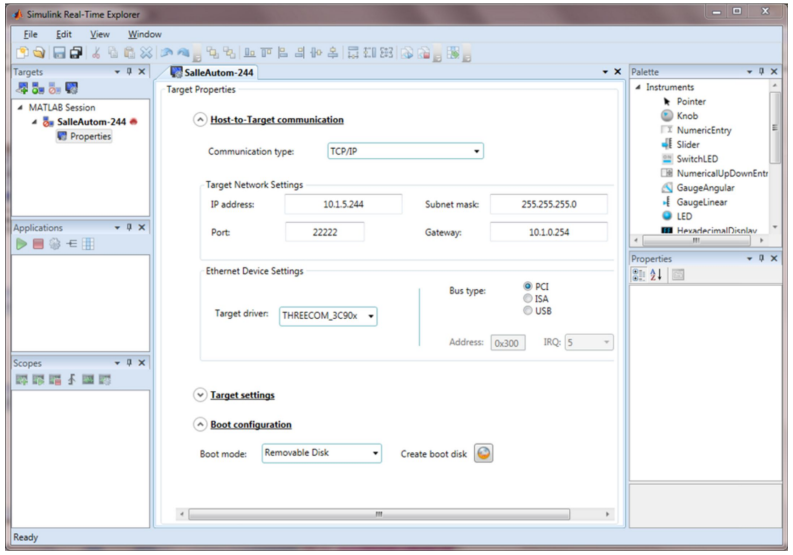


Gestion de la configuration

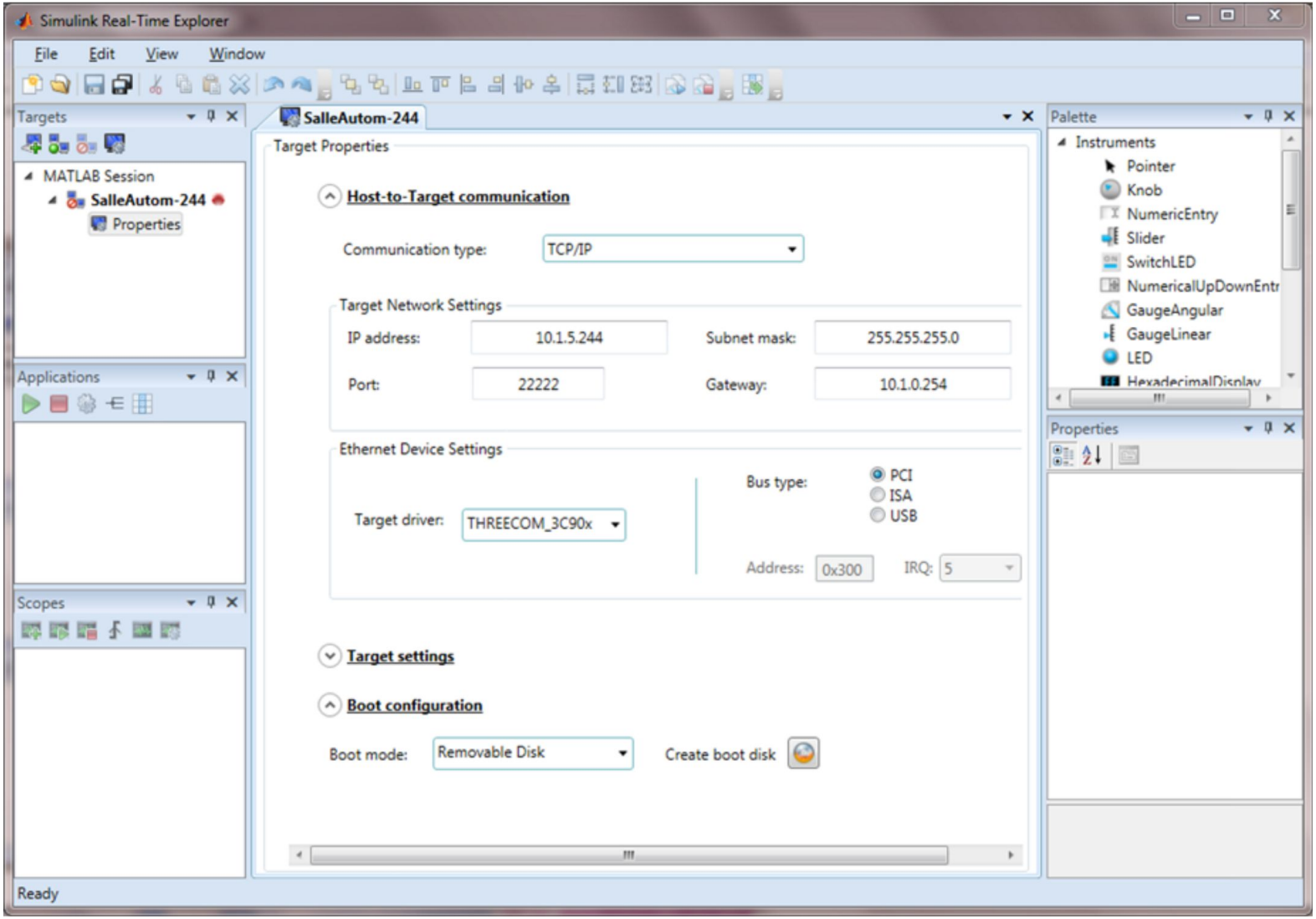
- Il est nécessaire d'avoir un compilateur C pour générer le code exécutable
- la commande **slrtgetCC** permet de voir si quel compilateur est déclaré
- La commande **slrtsetCC** permet de configurer celui-ci. Par exemple
  slrtsetCC('VisualC','C: \ Program Files (x86)\Microsoft Visual Studio 10.0')
- Seul les compilateurs *Microsoft* sont compatibles avec cette toolbox.

Gestion du noyau + configuration réseau

- Le PC cible demarre à partir d'un CD-ROM qui contient le système d'exloitation (SLRT)
- le navigateur **slrtexplr** permet aussi de fabriquer ce disque en spécifiant les caractéristique de la cible => nécessite de définir l'adressage IP de la cible.
- Chaque cible possède sa propre adresse.

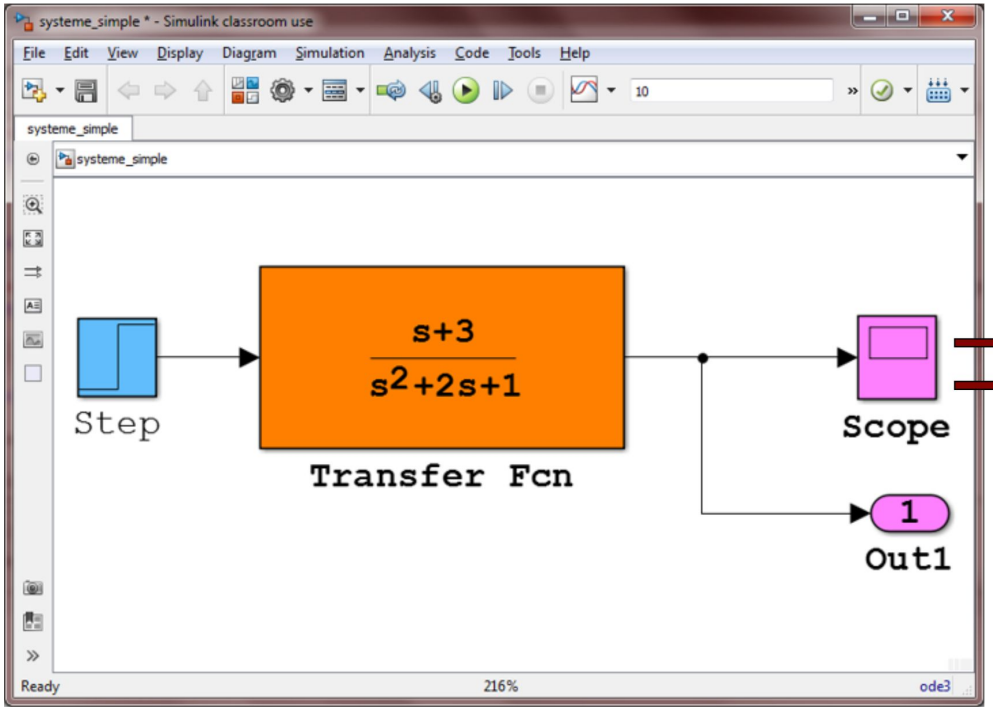


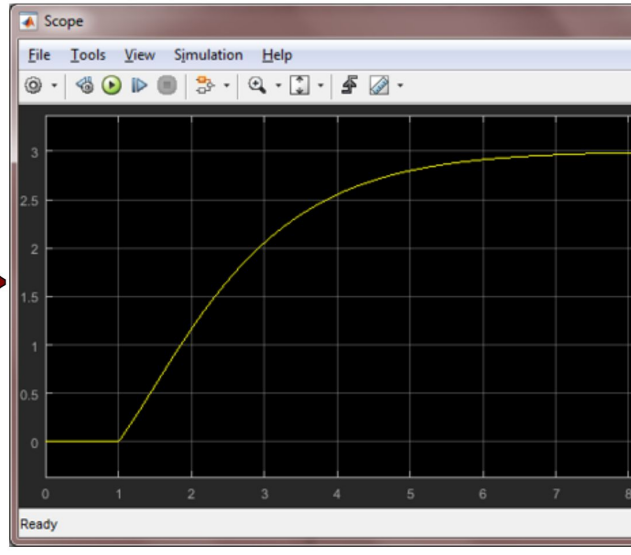
Gestion du noyau (zoom)



Faire tourner une application en local

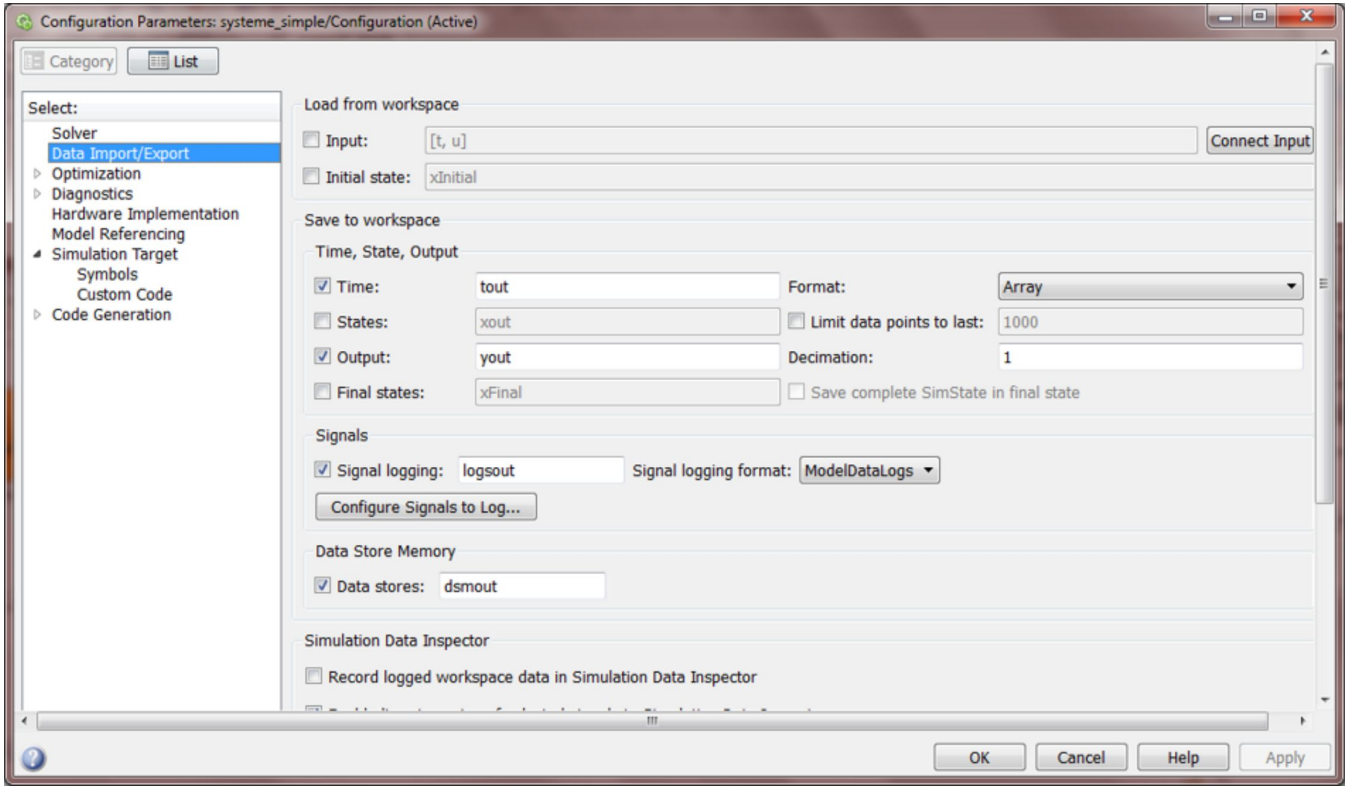
- Création de l'application simulink "classique".
- La simulation sur le PC de développement entraîne la visualisation sur le scope





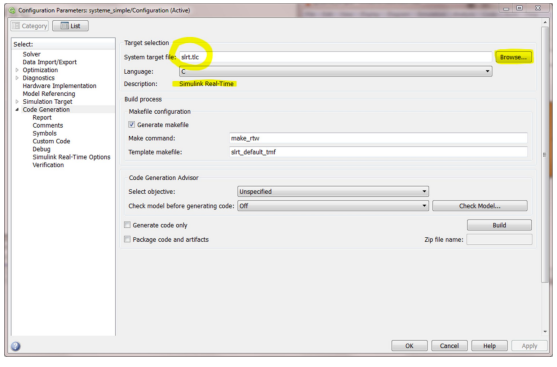
Données en local

- Le temps (relatif au système simulé), les sorties (bloc **output**) et les *états* peuvent être enregistrés dans le **workspace**

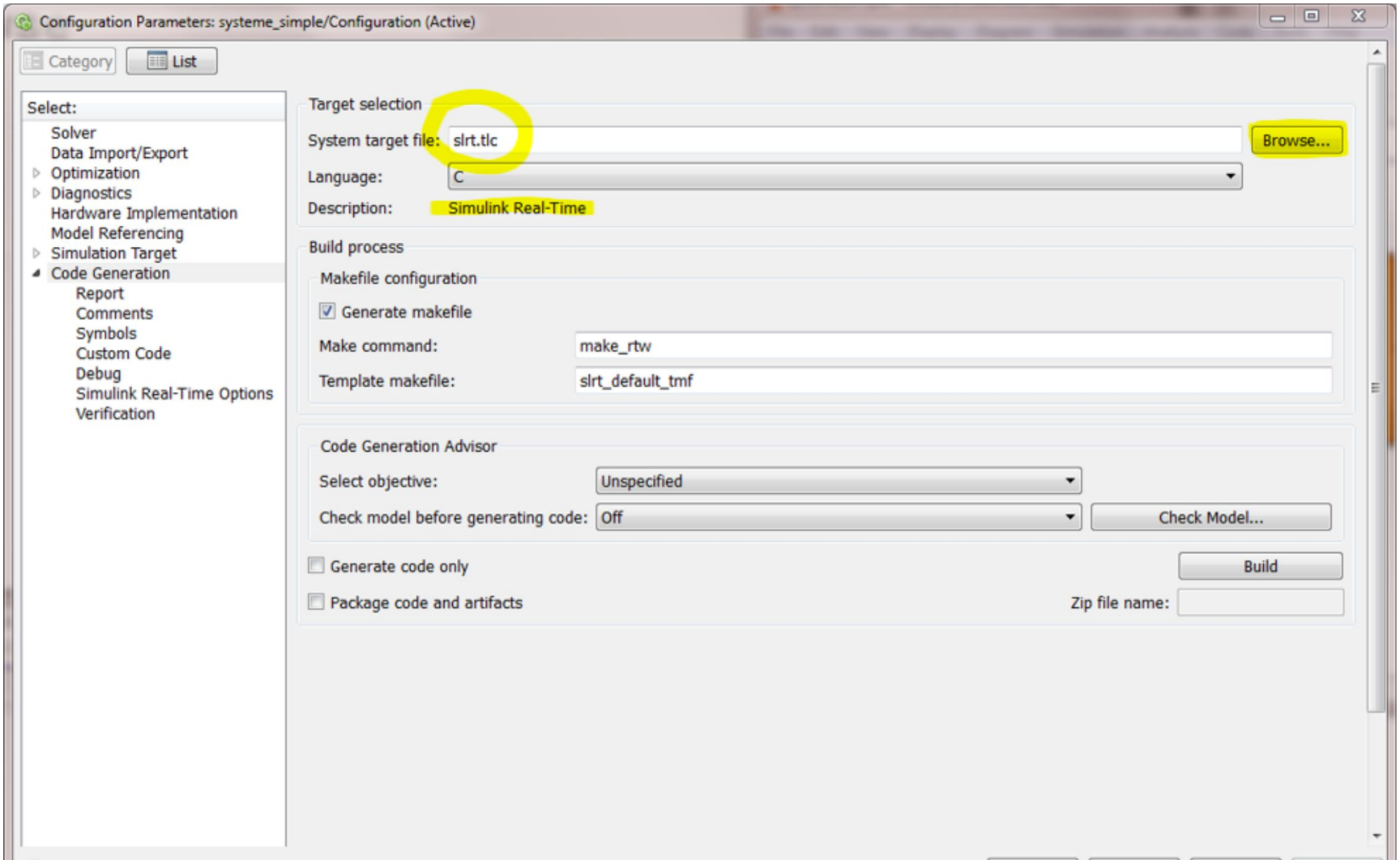


Création d'une application SLRT

- Pour faire tourner cette application sur le PC cible (sous "SLRT"), il faut modifer certains paramètres.
- Dans le configurateur de paramètres (menu simulation) sous l'onglet **Code Generation** , il faudra choisir la cible **slrt** (**S**imu**L**ink **R**eal **T**ime)
- La toolbox possède un *compilateur* qui permet de traduire des planches **Simulink** en langage C. Seconde étape : appel au compilateur C de Microsoft pour en faire du code exécutable.
- Le code C généré est lisible et exploitable.

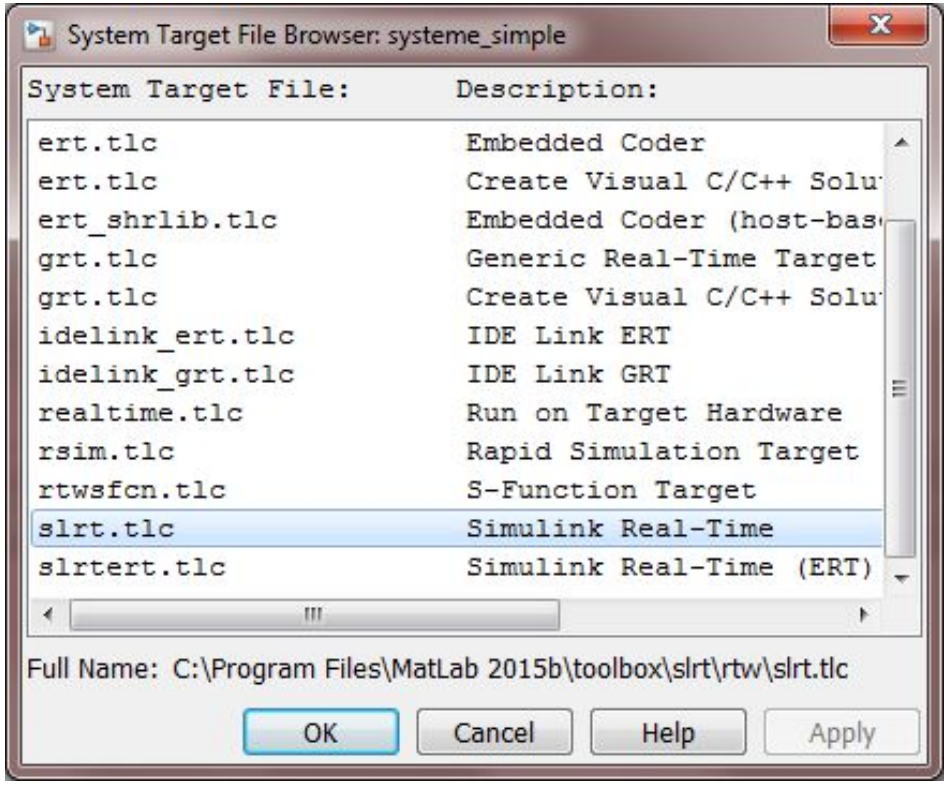


Création d'une application SLRT



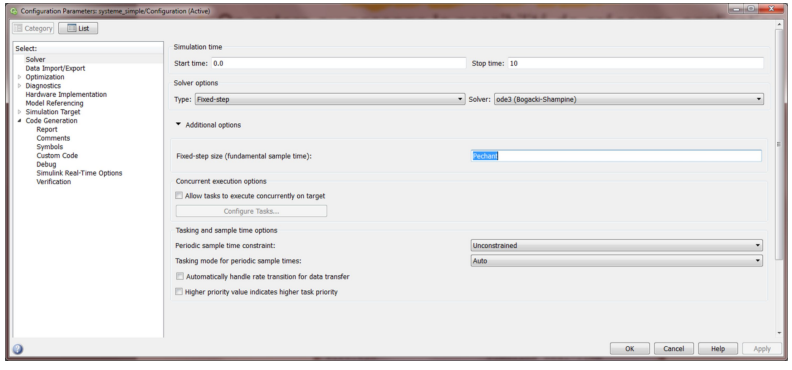
Création d'une application SLRT (2)

- On notera au passage la possibilité de créer une application pour des supports autres que "SLRT".

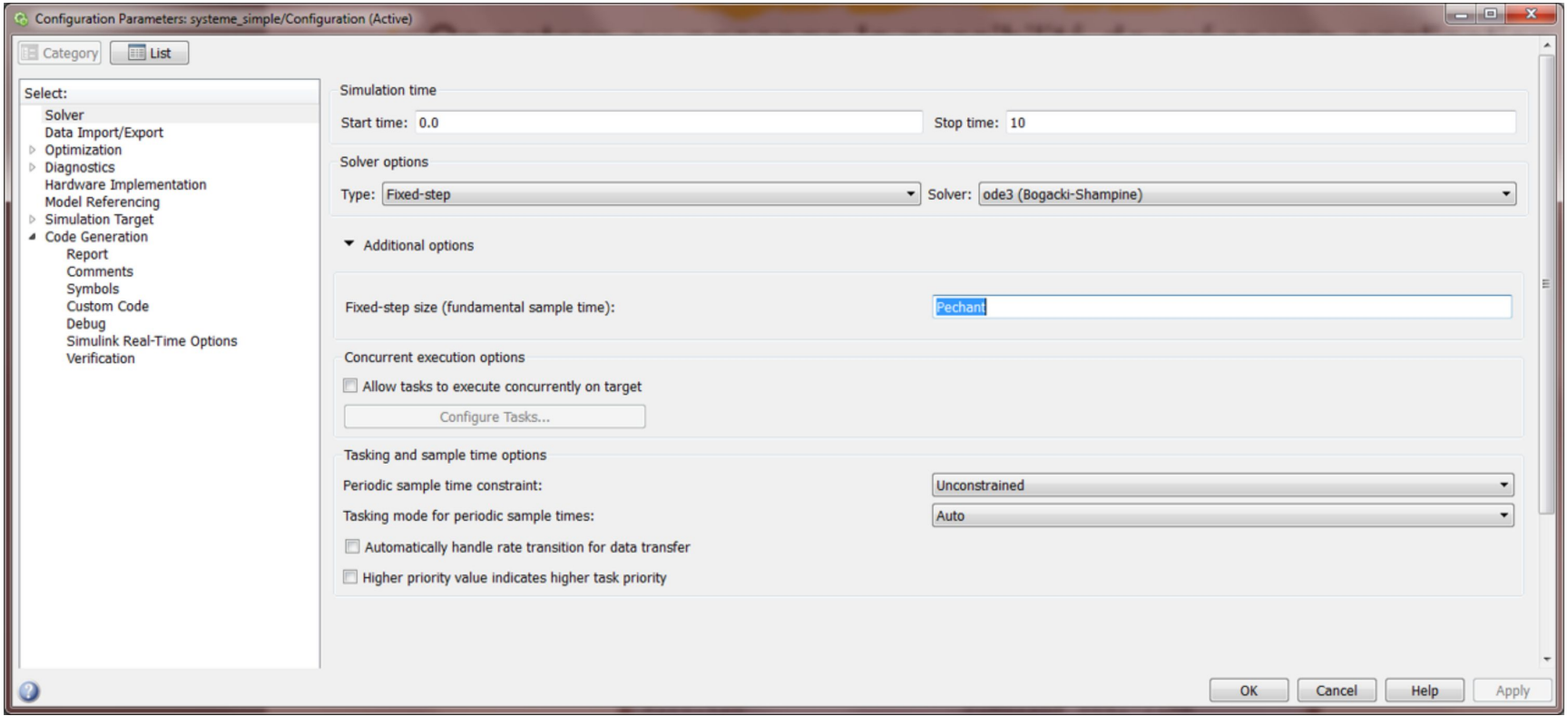


Création d'une application SLRT (3)

- Il faut ensuite définir le solveur. A ce stade il est encore possible d'utiliser un solveur à pas variable mais comme par la suite on va travailler en pas fixe (échantillonnage) il faut mieux partir sur un solveur à pas fixe.
- Les 3 variables à positionner sont donc : le solveur à pas fixe, la période d'échantillonnage (variable du workspace de préférence) et la durée de l'expérimentation (**inf** pour l'infini)

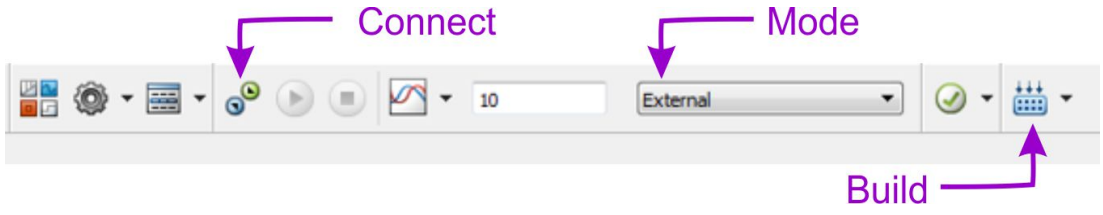


Création d'une application SLRT (3)



Création d'une application SLRT (4)

- A partir de là tout est simple, il suffit de cliquer sur le menu **build** dans la menu **Code->Code C/C++->Build** de l'application
  ou l'icône :



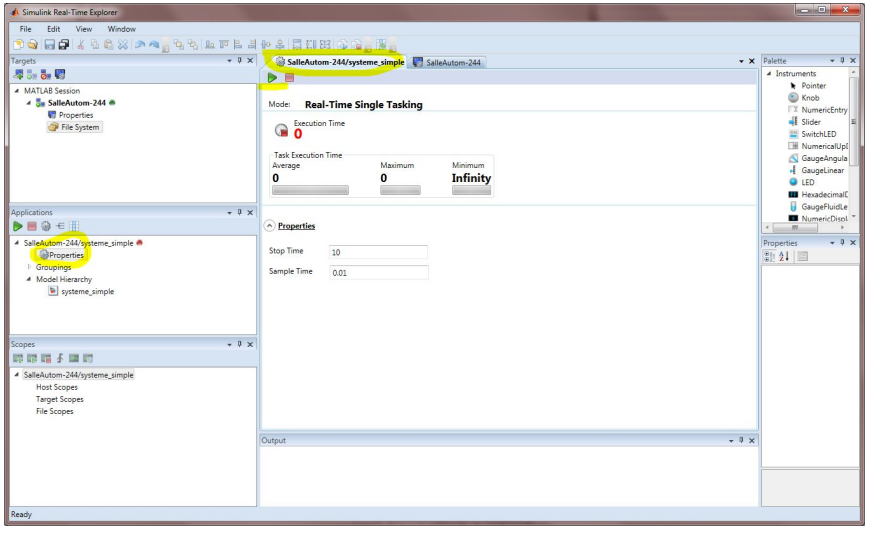
  - Génération du code C
  - Appel au compilateur C
  - Construction de l'applicatif **SLRT**
  - Chargement de l'exécutable sur la cible
  - Pilotage de l'exécutable depuis le PC de développement via **slrtexplr** ou le mode **external** sous **Simulink**

Execution sur la cible : à la main

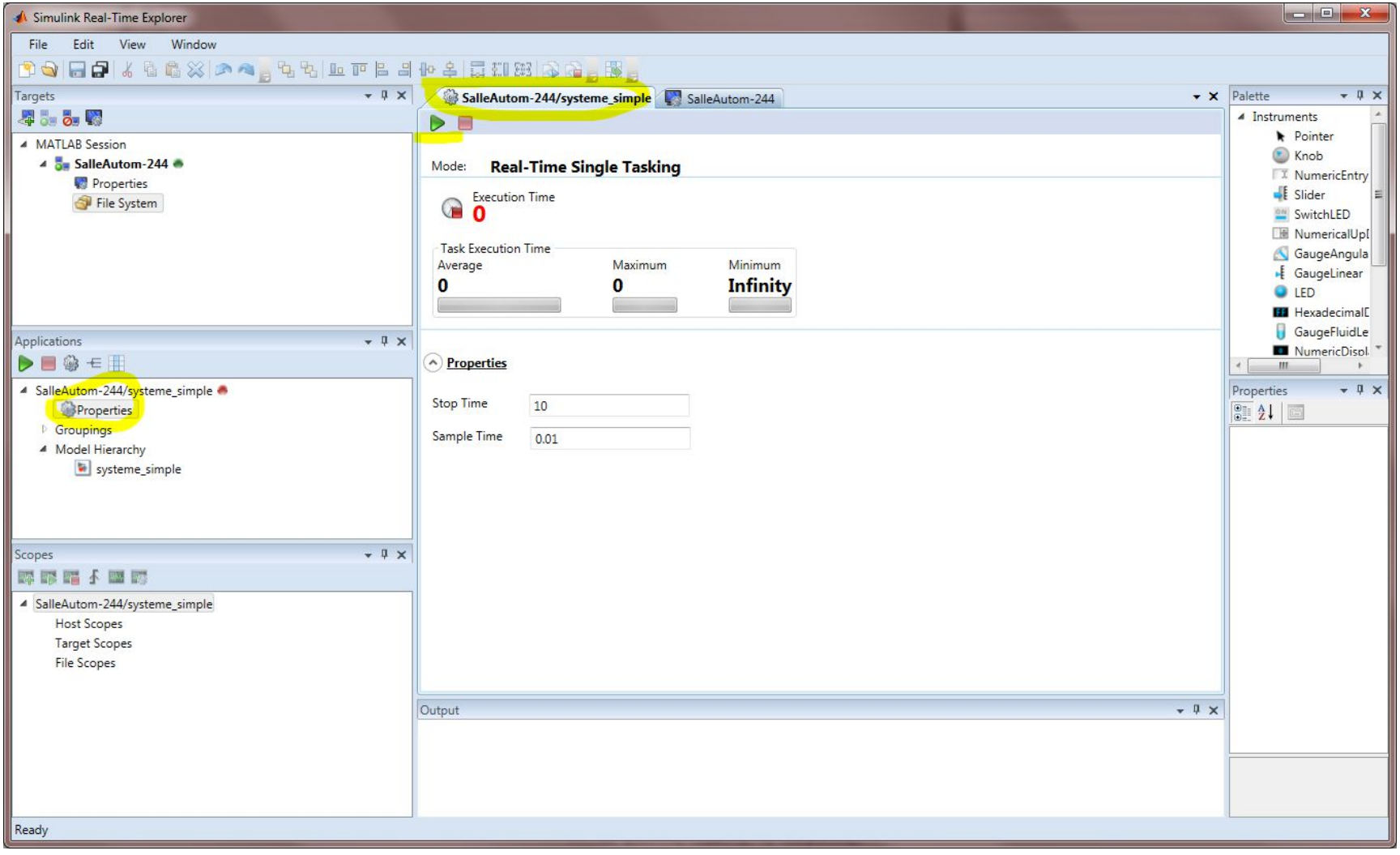
- Dans le workspace vous avez un objet (**tg**) qui est le point d'entrée de l'application sous "Matlab"
- Les infos contenues dans cet objet s'obtiennent par la commande **get(tg)**
- Pour passer en `Status "running"` il suffit de taper +**tg** (ou **start(tg)**
- Pour passer en `Status "stopped"` il suffit de taper -**tg** (ou **stop(tg)**)
- Sur la fenêtre du PC cible, on dispose de quelques infos : le nom de l'application, la taille mémoire utilisée, la période d'échantillonnage et l'exécution (**stopped** ou le temps courant)
- Voir le temps courant s'écouler sur la cible est une preuve que l'application est *vivante*

Execution sur la cible : le navigateur

- On peut préférer utiliser le navigateur **slrtexplr** pour agir sur la cible
- Ce dernier permet de gérer les cibles et les applications embarquées
- Une fois la cible connectée (elle n'y est pas par défaut même si l'application est chargée) on a la fenêtre suivante :



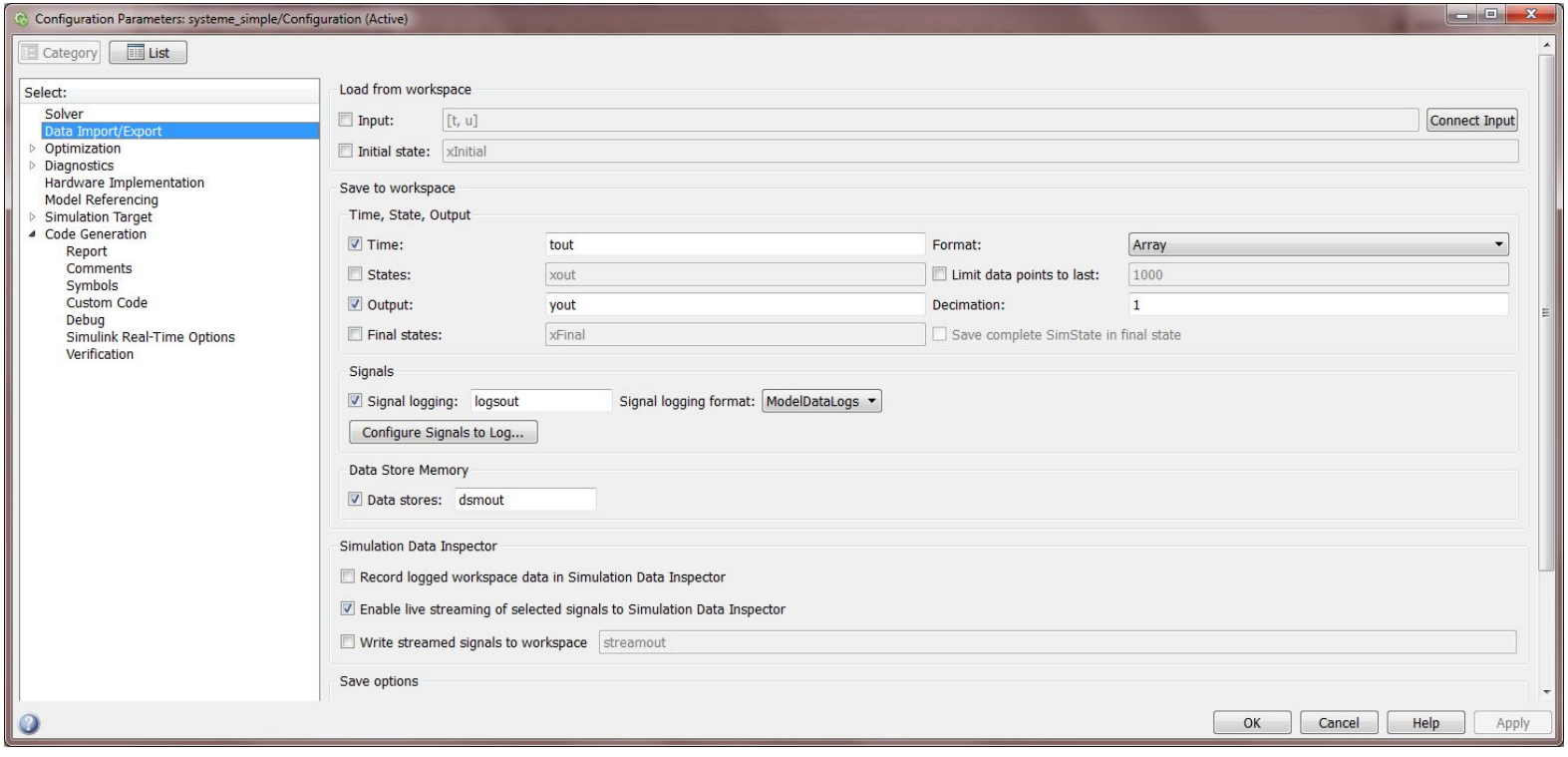
Execution sur la cible : le navigateur



Récupération des données : à la main

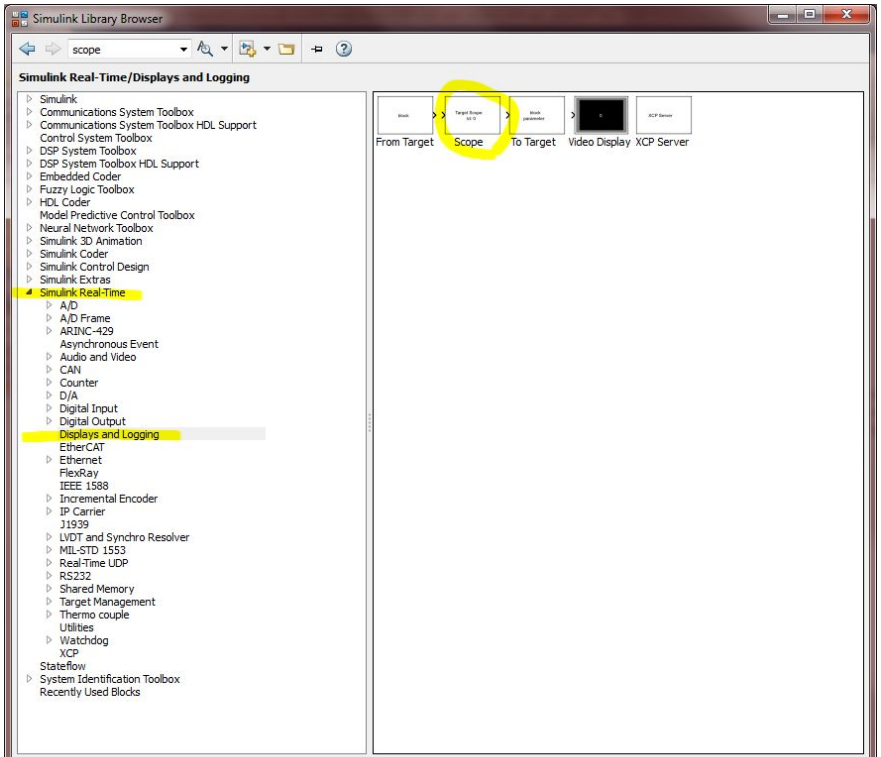
- Si l'on a placé des blocs **output** et que l'on a coché les options **Time** et **Output** dans l'onglet **Data Import Export** du configurateur de paramètres de l'application. Attention les noms données aux varaibles n'ont plus de sens car elles sont enregistrées directement sur la cible.
- Les commandes **temps = tg.timelog** et **y = tg.outputlog** permettent de récupérer le vecteur temps et celui du (ou des blocs) sortie dans l'espace de travail
- Il n'y a alors plus qu'à faire un **plot(temps,y)** pour avoir les résultats (après exécution)

Récupération des données : à la main

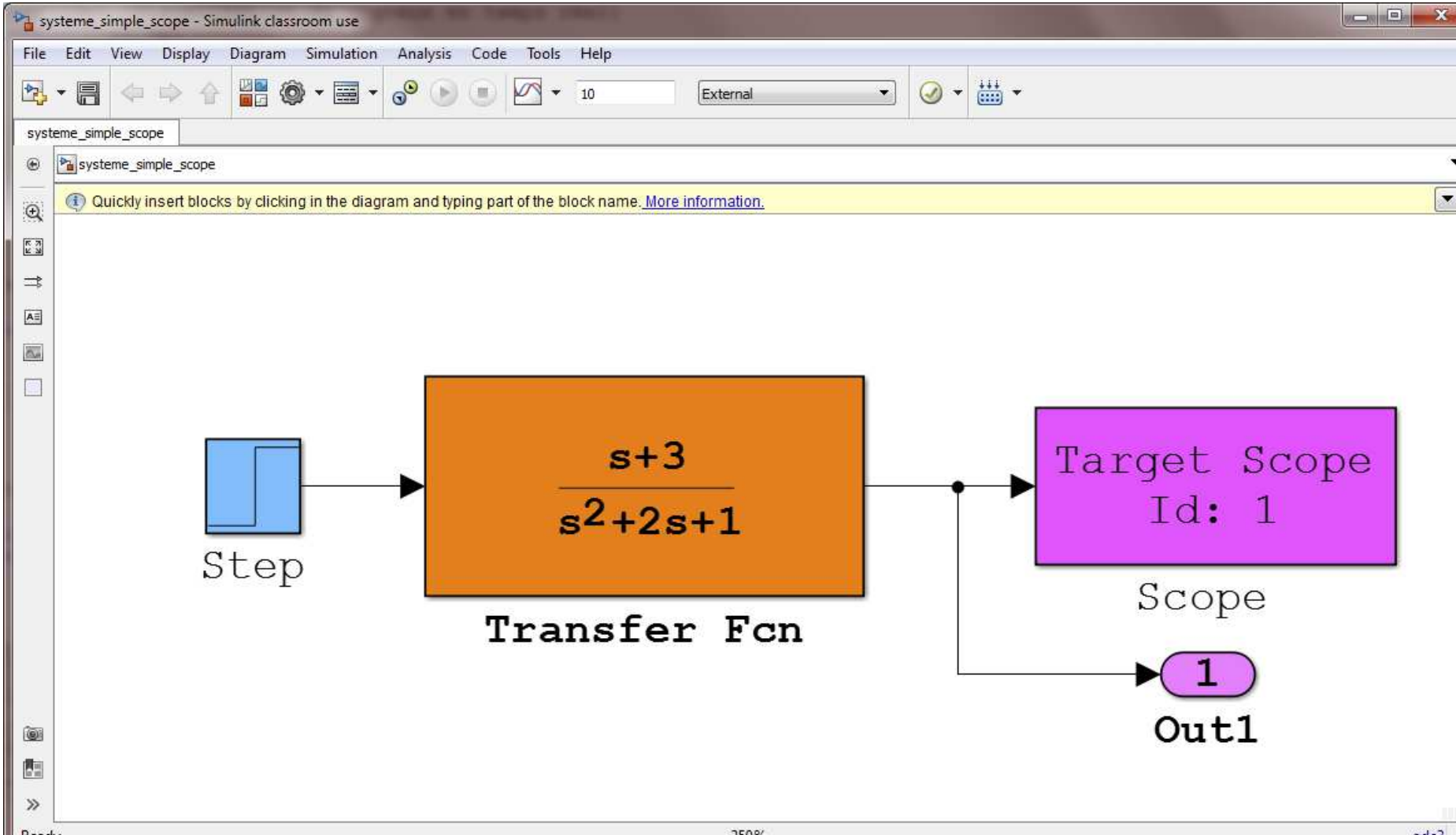


Visualisation de signaux en temps réel

- Il est possible de visualiser en temps réel l'évolution d'un ou de plusieurs signaux
- Cela suppose de créer un **scope** sur l'application
- Ce bloc se trouve dans l'onglet **Displays and Logging** de la bibliothèque **Simulink Real-Time**



Visualisation de signaux en temps réel



Remarques sur les scopes

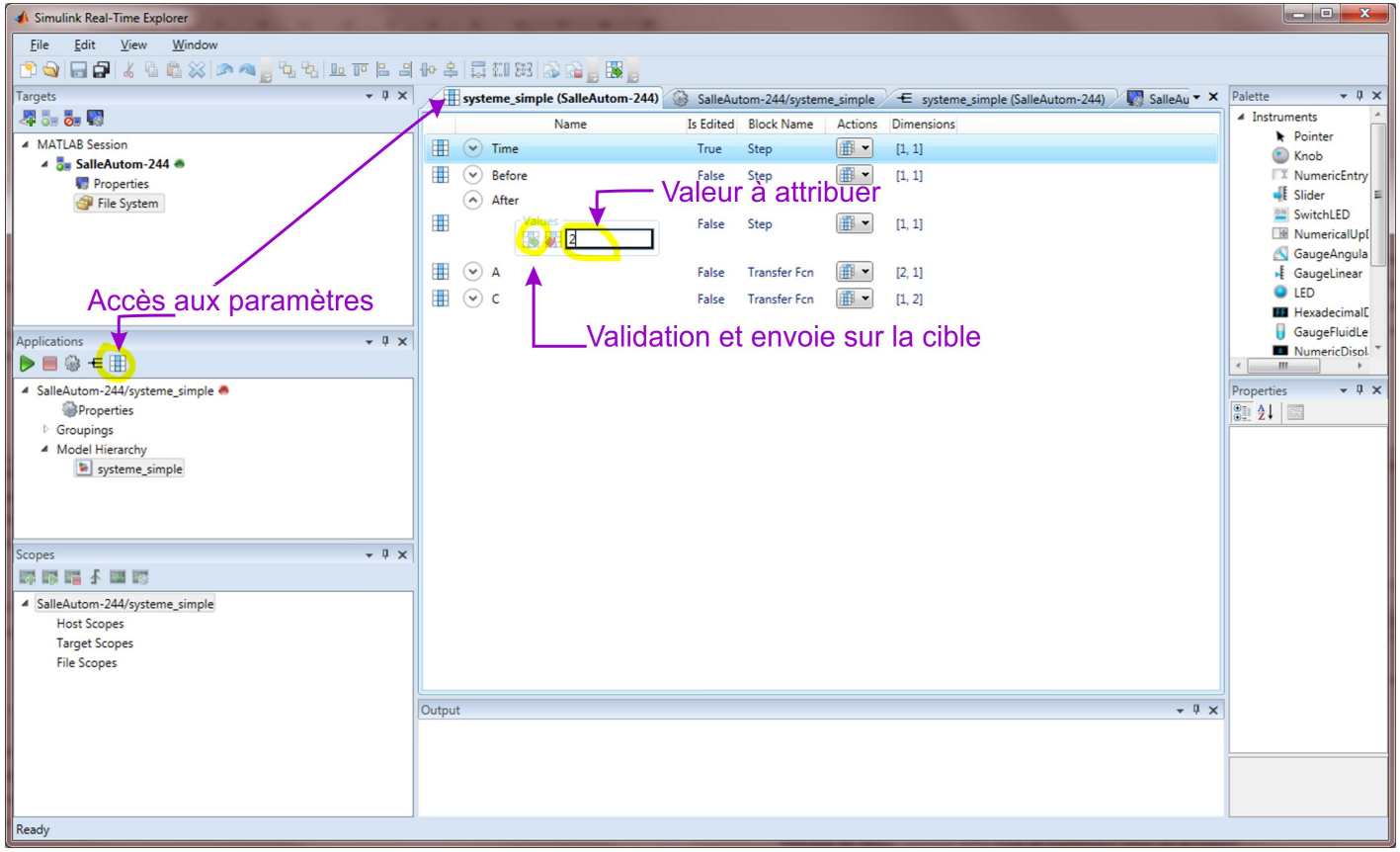
- Les scopes peuvent être créés à partir du navigateur (et aussi par des commandes matlab)
- Ils sont de 3 types :
  - **Target** : la visualisation se fait sur le cible (pas terrible mais utile)
  - **Host** : Fenêtre de visualisation sur le "PC host" (=> transfert en temps réel)
  - **File** : enregistrement dans un fichier de données sur le "PC Host(.mat)"
- Il peut en avoir plusieurs
- Différents modes possibles (*roll*, *redraw*,...) et autres paramètrages (*suréchantillonnage*, *trigger*,..) - A découvrir

Modifications de paramètres - commande Matalb

- La grande majorité des paramètres de l'application est modifiable en ligne et pendant que l'application tourne
- On peut récupérer la liste des paramètres de l'application par **tg.Showparameters = 'on'**
- On peut récupérer la valeur d'un paramètre par **tg.getparam(i)** , où i est le numéro du paramètre concerné. (Attention l'ordre des paramètres peut évoluer lorsque l'on modifie l'application)
- On peut modifier la valeur d'un paramètre par **Stat = tg.setparam(i,newval)** , où le structure renvoyée **Stat** contient le numéro du paramètre modifié ainsi que l'ancienne et la nouvelle valeur.
- Ces commandes sont très pratiques lorsque l'on veut créer une interface utilisateur à l'application (emploi de "GUI Matlab")

Modifications de paramètres - navigateur

- L'exporateur permet la visualisation et le modification des paramètres (plus direct)

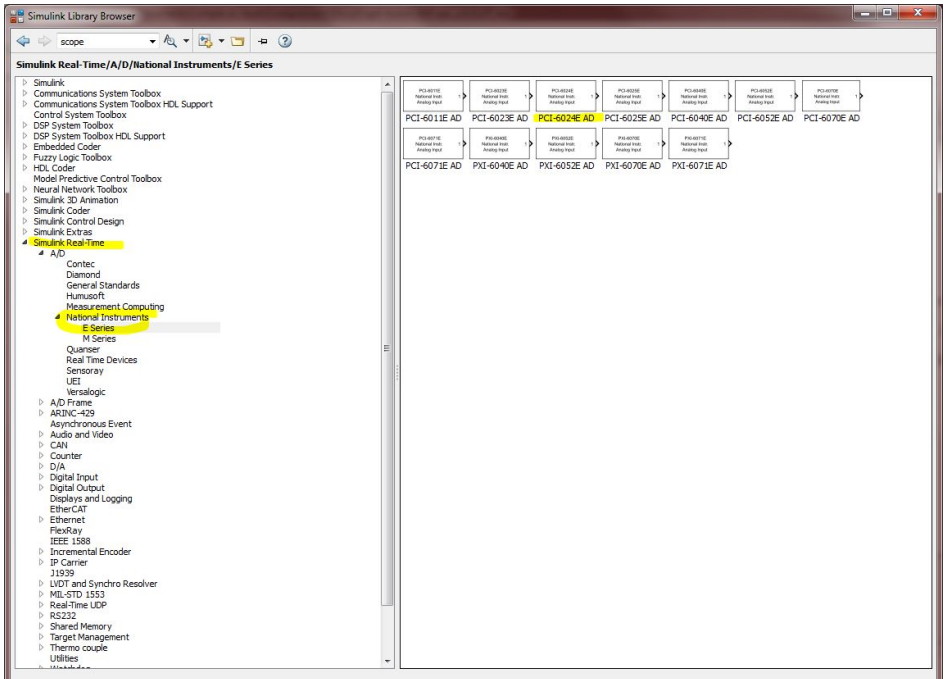


Modifications de paramètres - Simulink

- Si la planche Simulink est en mode **external** , la modification d'un paramètre d'un bloc affecte directement le paramètre correspondant sur la cible.
- Cette modification ne doit pas modifier la strcuture du modèle. Exemple : changer l'odre de la fonciton de transfert.

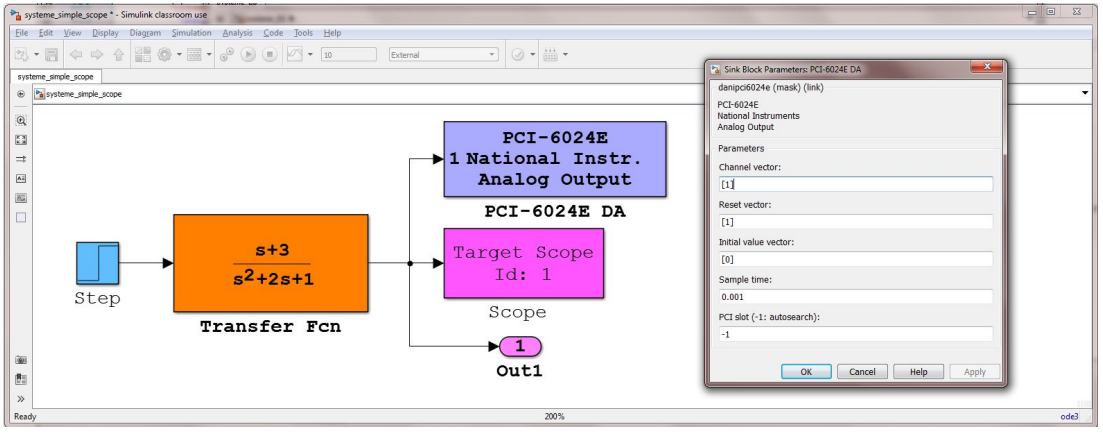
Interface avec l'extérieur

- Le PC cible est doté d'une carte d'acquisition "NI 6024E" : 16 entrées A/D + 2 sorties D/A
- Pour connecter votre application avec le process vous devrez utiliser les blocs spécifiques à cette carte
- La bibliothèque **Simulink Real-Time** est classée par type d'E/S puis par constructeur

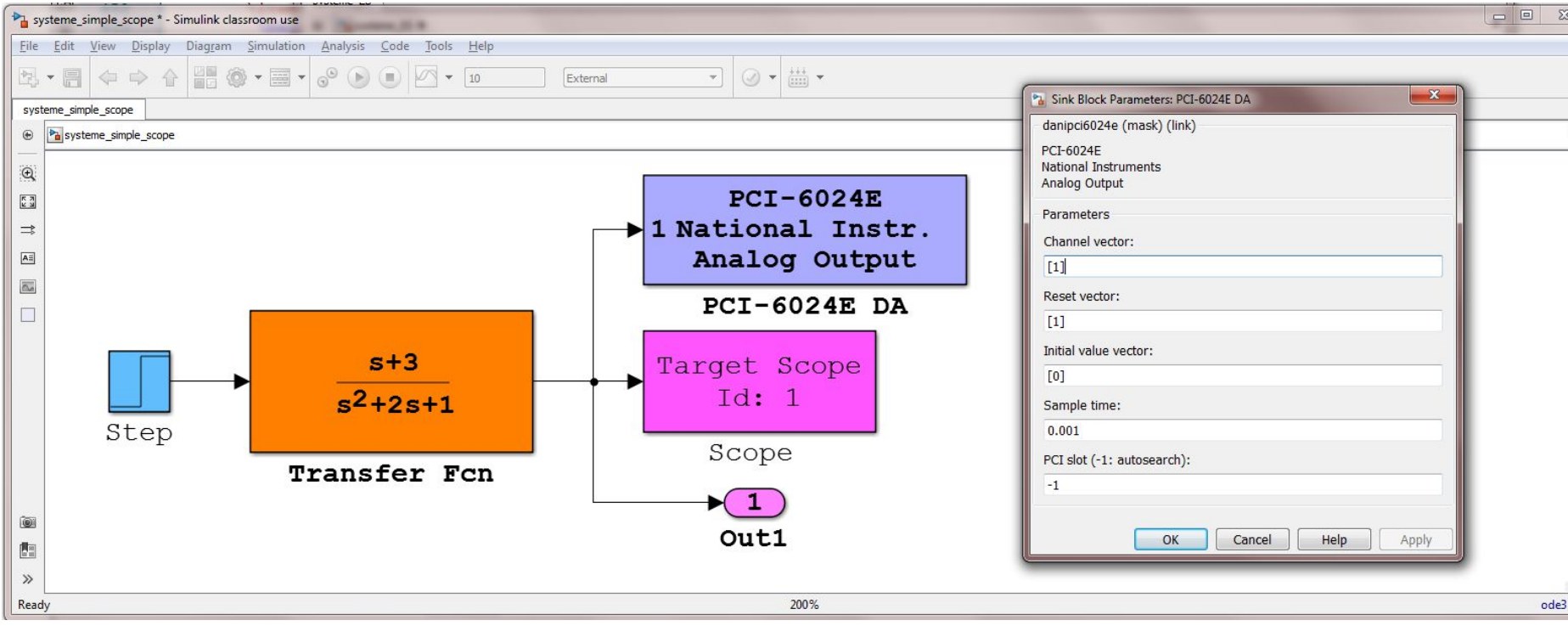


Insertion d'un bloc A/D

- Le bloc A/D (resp. D/A) permet de sélectionner le (ou les si on spécifie un vecteur) canal à utiliser
- Dans les paramètres on spécifie aussi la valeur à mettre sur le port ("Initial value" si "Reset value "vaut 1, dernière valeur écrite si 0) lorsque la cible ne tourne pas (plus)

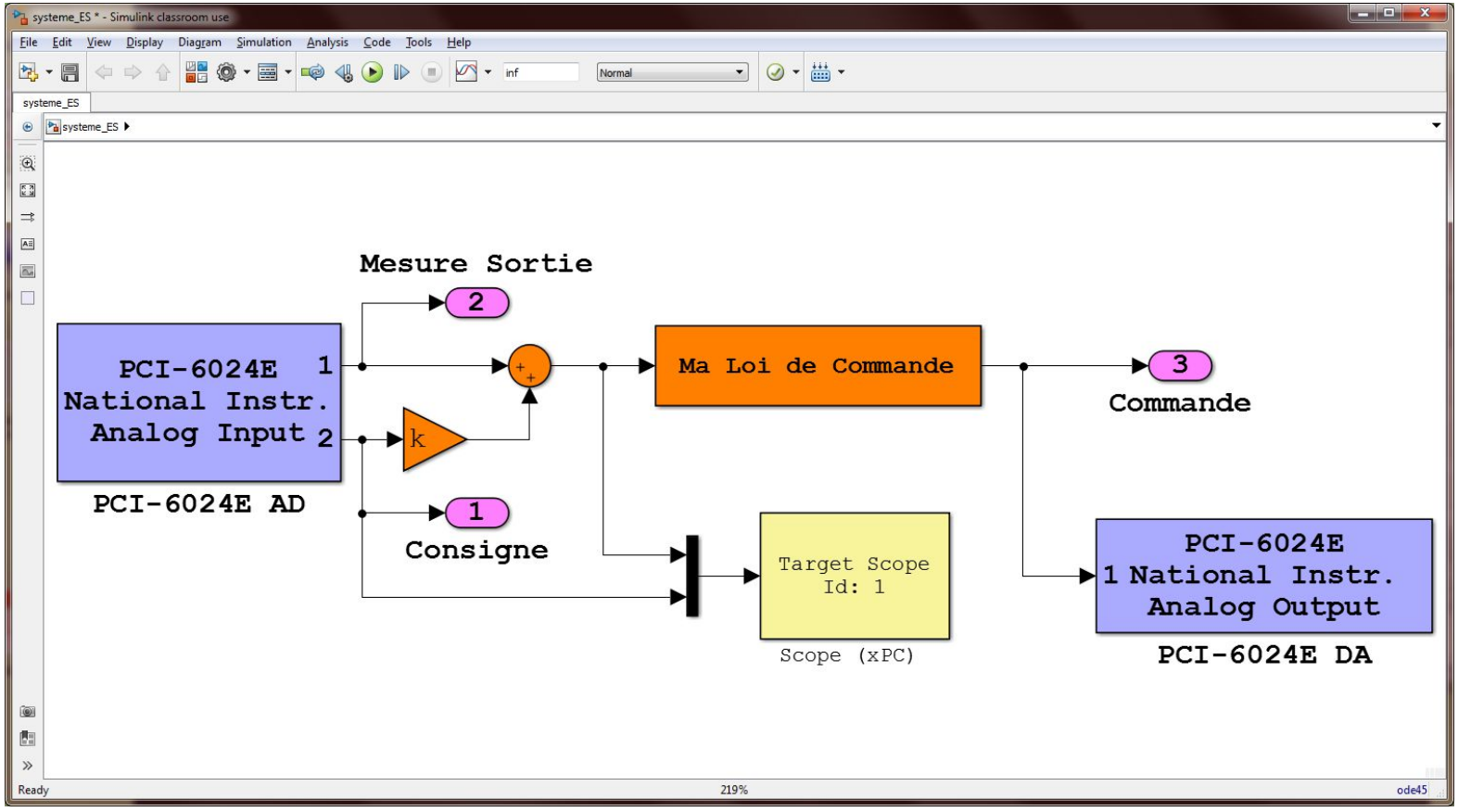


Insertion d'un bloc A/D



Application type

- Une application type avec consigne externe (potentiomètre ou GBF) donnera le schéma suivant :



Ce que je ne vous ai pas dit

- Pas d'explication sur la gestion du temps réel sous **slrt** (**interrupt mode** ou **polling mode**)
- Les possibilités de créer une application "bootable"
- Créer des layout sous le navigateur pour avoir une interface utilisateur
- Les possibilités de jouer avec un **Web browser**
- L'écriture directement en C pour créer une application (le code C généré par "RTW" est accessible...allez y jeter un petit coup d'oeil)
- ...