Programmation LEGO - Quick Start Guide

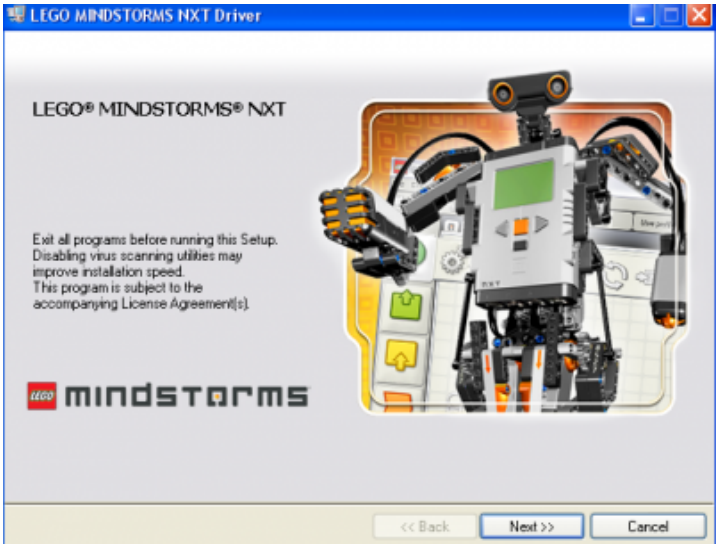
Ce document vous explique pas à pas l'installation de l'environnement de développement et du compilateur BricxCC. De plus, à travers des exemples de code nous vous guidons pour vos premiers développement pour le système LEGO NXT.

1. Installation du driver NXT

Avant de pouvoir développer votre premier programme vous devez installer le driver NXT pour Windows. Ce driver est nécessaire pour que Windows puisse détecter la brique NXT lorsqu'elle sera connectée à votre ordinateur à l'aide du cable USB.

- Commencez par télécharger l'archive PhantonDriver_Download1.zip. à l'aide du lien suivant : phantondriver_download1.zip
- Décompresser l'archive en double-cliquant dessus
- Vous obtenez alors les fichiers qui suivent

- Il suffit d'exécuter le programme **setup.exe** pour exécuter l'installation du driver
- Le programme d'installation doit se lancer



- Il suffit de cliquer sur **Next** pour effectuer l'installation
- A partir de maintenant votre brique NXT est reconnue par Windows

2. Installation de BricxCC et configuration de l'environnement de développement (IDE)

Le logiciel Bricx Command Center (BricxCC) permet d'écrire, compiler et télécharger/exécuter vos programmes en NXC (Not eXactly C) dans la brique LEGO (e.g. NXT ou l'ancienne RCX). Le logiciel inclut différents langages C/C++, Pascal et Java. Dans ce tutorial, nous allons uniquement nous intéresser au langage NXC (Not eXactly C). En effet, ce langage a plusieurs avantages. Tout d'abord, le C est un langage très utilisé et un des plus courants dans les systèmes embarqués (e.g. en robotique). Ensuite, le langage C est portable, c'est-à-dire que le code développé pour une plateforme peut être "facilement" réutilisé sur d'autres systèmes (Norme ANSI). Enfin, comme la plupart des compilateurs C, NXC est disponible gratuitement sous la licence publique Mozilla. De plus, le langage NXC permet de développer rapidement du code pour les systèmes basés sur NXT.

Etape 1 : Téléchargement

1. RDV sur la page http://sourceforge.net/projects/bricxcc/files/bricxcc/bricxcc%203.3.8.9/ [http://sourceforge.net/projects/bricxcc/files/bricxcc/bricxcc%203.3.8.9/]
2. Nous allons utiliser la version **3.3.8.9**
3. Téléchargement du fichier **bricxcc_setup_3389.exe**

Etape 2 :

Une fois l'archive .exe téléchargée, double-cliquer sur le fichier .exe et choisissez le répertoire d'installation (nous vous conseillons le répertoire E:\virtualisation)

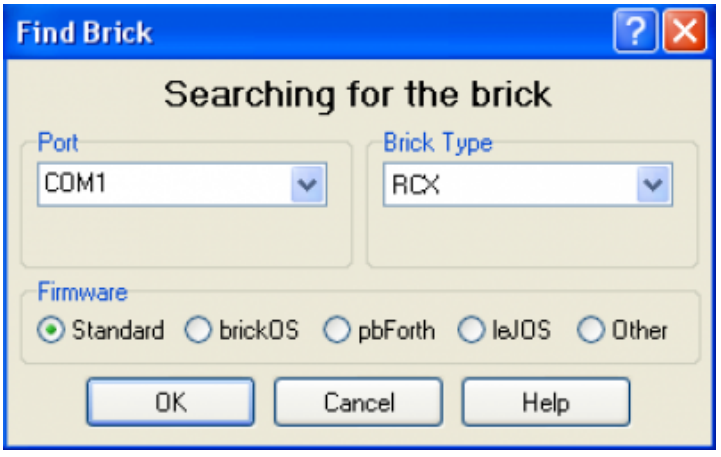
Etape 3 :

Connecté le cable USB entre votre PC et la brique NXT. Allumer la brique NXT.

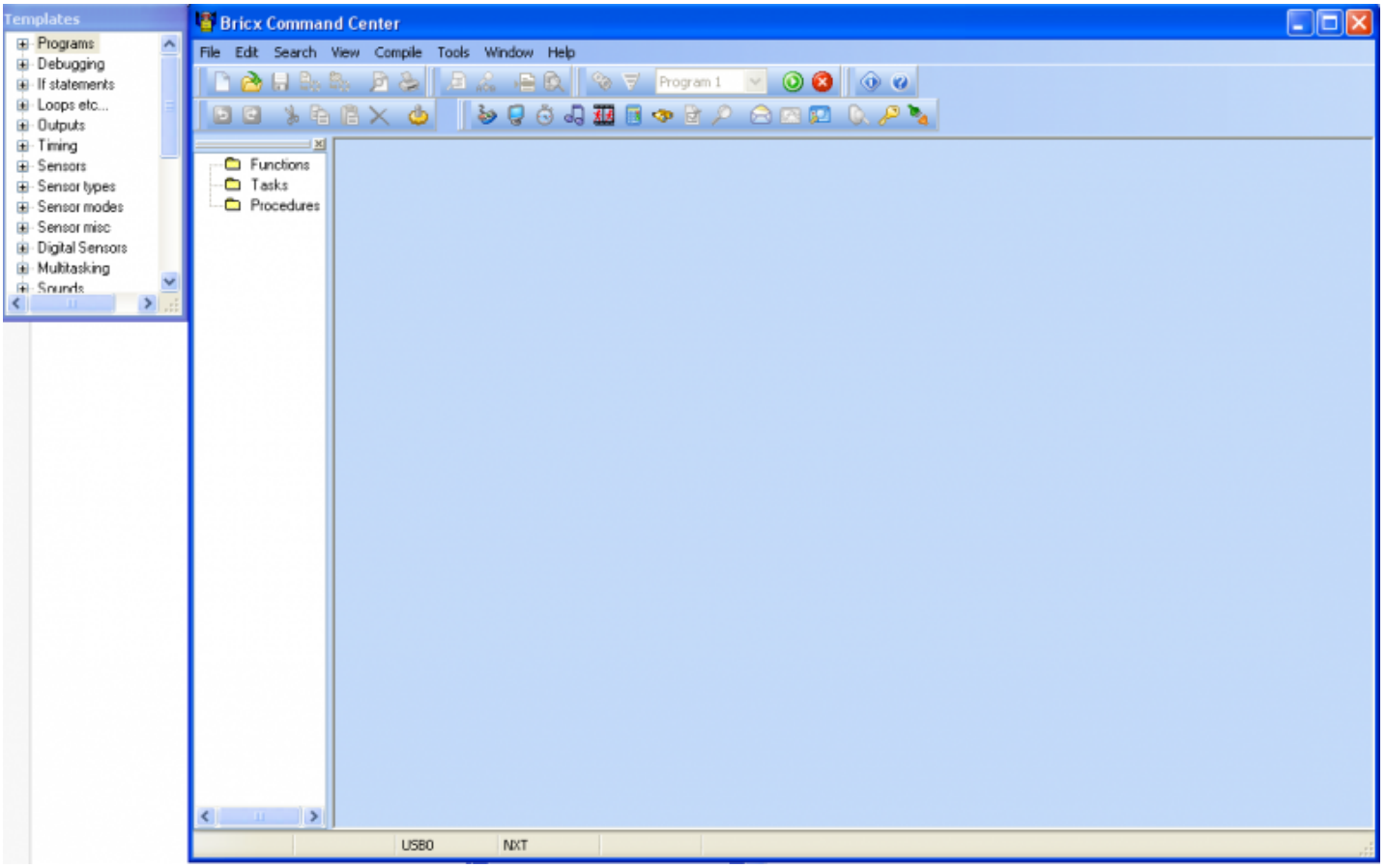
Etape 4 :

Démarrage de l'application

- Menu Démarrer → Tous les programmes → Bricx Command Center → Bricx Command Center
- Normalement, vous devriez voir la fenêtre suivante



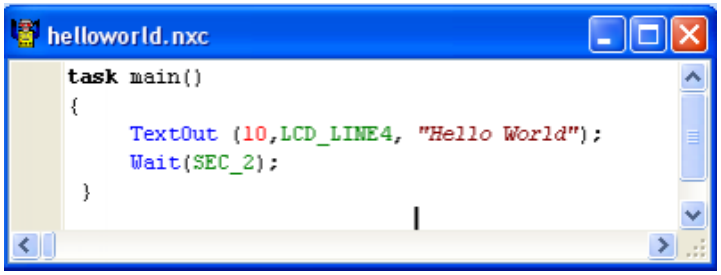
- Sélectionner **USB** et **NXT**, respectivement, pour le **Port** et **Brick Type**, puis cliquer sur **OK**
- La fenêtre principale de l'IDE s'affiche



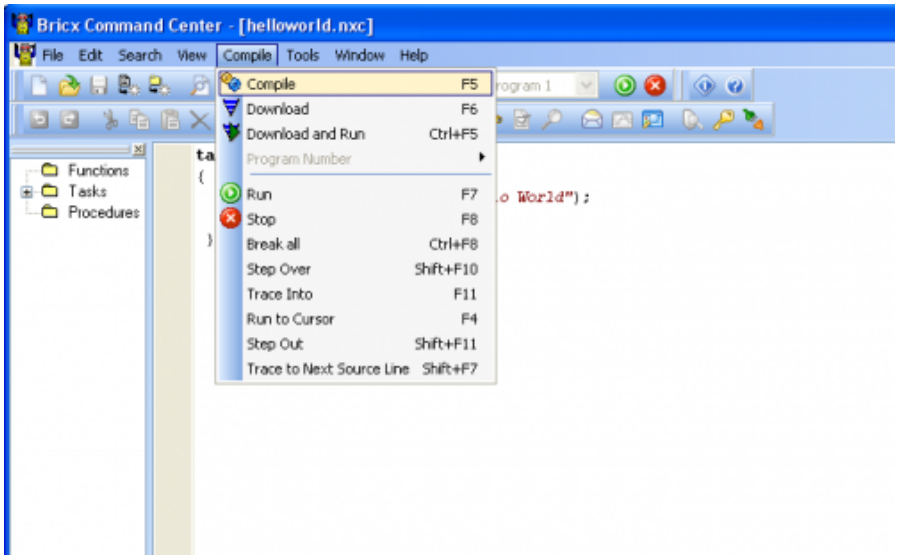
3. Mon premier programme NXC - Hello World

- Choisir **File → New**, puis **File → Save As** et sauvegarder le fichier dans un répertoire (e.g *E:\monprogramme*) avec, comme nom, par exemple **helloworld**
- Entrer le texte qui suit

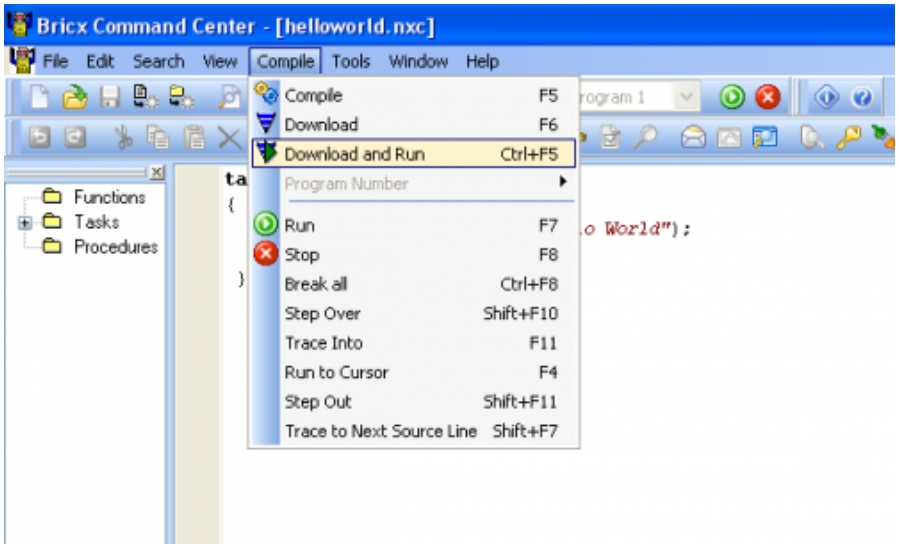
```
task main()
{
     //Guide NXT (cf. liens bas de page), page 373, sec. 6.44.2.47
     //TextOut (int x, int y, string str, long options)

     TextOut (10,LCD_LINE4, "Hello World");
     Wait(SEC_2);
 }
```



- Choisir **File → Save All**
- Il faut à présent compiler le programme



- Pour exécuter votre programme, nous allons le télécharger dans la brique NXT et l'exécuter



- Votre brique NXT doit afficher le résultat de votre premier super programme . Félicitations !!!

4. Et on fait tourner les moteurs...

- Commencer par créer un nouveau fichier (open a new file) et le sauvegarder avec le nom “helloMotor.nxc”. Il faut ensuite saisir le code qui suit.

```
// Fichier: HelloMoteur1_0.nxc
// Auteur: Mehdi
// Date: 13/09/2012
// Desc : Les moteurs doivent être connectés aux Ports A et C.

task main() {

  OnFwd(OUT_AC, 75);
  // Dans le guide NXC (cf. liens bas de page), Page 329 - 6.42.2.27 : OnFwd(byte outputs, char pwr)
  // Les sorties sont OUT_X avec X = {A,B,C, AB, AC,BC,ABC)
  // pwr est dans l'intervalle [0,100]
  Wait(500);  //en millisecondes
  OnRev(OUT_AC, 25);
  // Dans le guide NXC (Page 447 - 6.52.2.69)
  Wait(2000);

  Off(OUT_AC); // Arrêter et terminer le programme proprement
  StopAllTasks();

}
```

- Brancher deux moteurs (Ports A et C) à la brique NXT
- Sauvegarder tout, Compiler, Télécharger le code dans la brique et lancer votre programme (Download and Run)

5. On va détecter...Le fil rouge sur le bouton rouge...

L'objectif de cette section est d'apprendre à utiliser le capteur de contact LEGO

- Commencer par créer un nouveau fichier (open a new file) et le sauvegarder avec le nom “helloTouch.nxc”. Il faut ensuite saisir le code qui suit.

```
//FILE: helloTouch.nxc
//AUTH: Mehdi
//DATE: 14/09/2012
//DESC: Compter et afficher le nombre de fois que le capteur de contact est touché

task main()
{

  int touchCounter;  //variable permettant de stocker le nombre de fois que le capteur est touché

  SetSensor(IN_1, SENSOR_TOUCH);
  // SetSensor( byte &port, int config)
  // port : IN_X avec X = [1,2,3,4]
  // config : voir Guide NXC (page 304, sec. 6.3.9.2.34)

  touchCounter = 0;

  while(touchCounter <= 3)
  {

    if(SENSOR_1 == 1)
    {
      TextOut(10,LCD_LINE4, "Appuyé");
      touchCounter = touchCounter + 1;
      Wait(1000);
    }
    else
    {
      TextOut(10,LCD_LINE4, "Pas appuyé");
      Wait(1000);
    }
  } //fin du while

  TextOut(10, LCD_LINE4, "Fin");
  StopAllTasks();

} //fin du main
```

- Brancher le capteur de contact (Port 1) à la brique NXT
- Sauvegarder tout, Compiler, Télécharger le code dans la brique et lancer votre programme (Download and Run)
- Quand vous appuyez sur le capteur de contact, l'affichage sur la brique NXT doit changer

6. On tourne, on détecte et on fait de la musique - Les tâches

La brique NXT supporte le multitâches, avec NXC une tâche (task) correspond à un thread.

Quelques caractéristiques des tâches

- Toujours une tâche principale appelée main
  - task main()
- Exécutée au début d'application
- Une autre tâche s'exécute seulement si une tâche en cours l'appelle ou si elle est prévue dans le main
- Nombre de tâches limité par la plateforme (256)

- Définition d'une tâche
  - Pas de paramètres
  - Pas de retour

```
task LeNomDeMaTache()
{
  /* séquence de commandes */
}
```

- Lancement de tâches
  - start taskName;
- Ci-dessous un exemple de trois tâches

```
task music() {

  while (true) {
    PlayTone(TONE_A4, MS_500);
    Wait(MS_600);
  }
}


task controlRobot() {
 OnFwd(OUT_A,75);
}

task readSensor() {
  while(true)
  {
   if(SENSOR_1 == 1)
   {
     start  controlRobot;
     start music;
   }
  }
}

task main() {

  SetSensor(S1, SENSOR_TOUCH);
  start readSensor;
}
```

Plus loin avec les tâches

Essayons de réaliser un programme plus réaliste. On suppose que l'on souhaite réaliser un programme dans lequel le robot avance de façon circulaire. Mais lorsqu'il rencontre un obstacle, il doit réagir. Il est difficile de faire cela en une seule tâche, parce que le robot doit faire deux choses en même temps : avancer en rond et de surveiller les capteurs. Il est donc préférable d'utiliser deux tâches :

- une tâche pour qu'il se déplace
- une tâche pour réagir avec les capteurs.

Voici le programme :

```
mutex moveMutex;

task move_square()
{
  while (true)
  {
    Acquire(moveMutex);
    OnFwd(OUT_AC, 75); Wait(1000);
    OnRev(OUT_C, 75); Wait(500);
    Release(moveMutex);
```

```
  }
}

task check_sensors()
{
  while (true)
  {
    if (SENSOR_1 == 1)
    {
      Acquire(moveMutex);
      OnRev(OUT_AC, 75); Wait(500);
      OnFwd(OUT_A, 75); Wait(500);
      Release(moveMutex);
    }
  }
}

task main()
{
  Precedes(move_square, check_sensors);
  SetSensorTouch(IN_1);
}
```

La tâche principale définit simplement les types de capteur et planifie le démarrage des deux autres tâches en les ajoutant à la file d'attente du planificateur; après cela, la tâche principale se termine. La tâche `move_square` fait se déplacer le robot dans un carré indéfiniment. La tâche `check_sensors` vérifie si le capteur tactile est poussée et, le cas échéant, éloigne robot de l'obstacle. Il est très important de se rappeler que les deux tâches s'exécutent au même moment et cela peut conduire à des résultats inattendus, si les deux tâches essayent d'utiliser les moteurs comme ils sont censés le faire.

Pour éviter ces problèmes, nous avons déclaré un type étrange de variable, `mutex` (mutual exclusion) : nous ne pouvons agir sur ce type de variables qu'avec les fonction `Acquire` et `Release`. Puis nous écrivons des morceaux de code critique entre ces fonctions, assurant que seuls une tâche à la fois peut avoir un contrôle total sur les moteurs. Ces variables de type `mutex` sont appelés sémaphores et cette technique de programmation est nommé programmation concurrente.

Il faudra procéder de la même façon pour une variable partagée entre plusieurs tâches, voir l'exemple ci-dessous.

```
int x ; // variable partagée entre T1 et T2
mutex prot_x ; // mutex de protection de x

task T1()
{
 ...
 Acquire(prot_x );
 x = a - b; // opération protégée
 Release(prot_x );
 ...
}

task T2()
{
 int y ;
 ...
 Acquire(prot_x );
 y = x ;    // la valeur de x est garantie
 Release(prot_x );
 ....
}
```

7. Pour aller plus loin

Quelques ressources (A consulter dans cet ordre...)

Un excellent guide sur NXC avec plein d'exemples de code

nxc_tutorial.pdf

Encore des exemples de code (suivi de ligne, ...)

mindstorms.pdf

Des transparents de cours avec des exemples de code

robotnxtubo.pdf

Le guide du programmeur NXC

http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/index.html [http://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/index.html]

Le guide de référence (2447 pages...)

nxc_guide.pdf

8. Monsieur ça ne fonctionne pas...

RTFM !

— *mehdi.lhommeau@univ-angers.fr [mailto:mehdi.lhommeau@univ-angers.fr] 2013/09/19 15:58*
