Premiers programmes en C# "étape par étape" avec une carte Panda 3

[Mise à jour le : 16/7/2018]

1. Préambule

Pour mener à bien ce tutoriel vous devez disposer d'une carte Panda 3. Les outils logiciels nécessaires à sa programmation doivent être installés sur le PC. Le firmware de la carte doit être à jour. Si ce n'est pas le cas : suivez le « **Guide d'installation des logiciels** » ici.

Le cours sur les fondamentaux du langage C#, accessible sur LEARN est un excellent préalable et un complément à ce tutoriel.

On trouvera également des tutoriels en anglais sur le site de la société GHI Electronics.

2. Premier programme : Blink

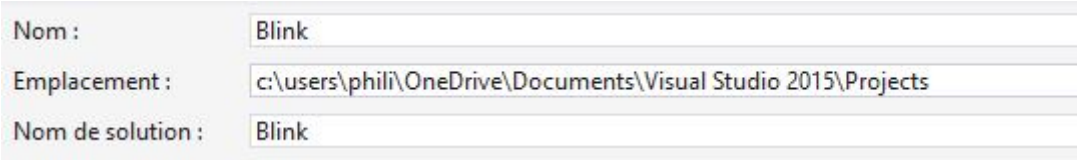
Cahier des charges du programme 1

Faire clignoter la LED1 de la carte Panda 3 !

Etape 1 : Créer un projet

- Ouvrez l'**IDE Visual Studio** en cliquant sur l'icône suivante : puis sélectionnez : **Fichier (File) → Nouveau projet (New Project)** ou **[Ctrl+Maj+N]**,
- Dans la boîte de dialogue "Nouveau projet" (New Project) sélectionnez: **Modèles(Templates) → Visual C# → Micro Framework et Console Application**.

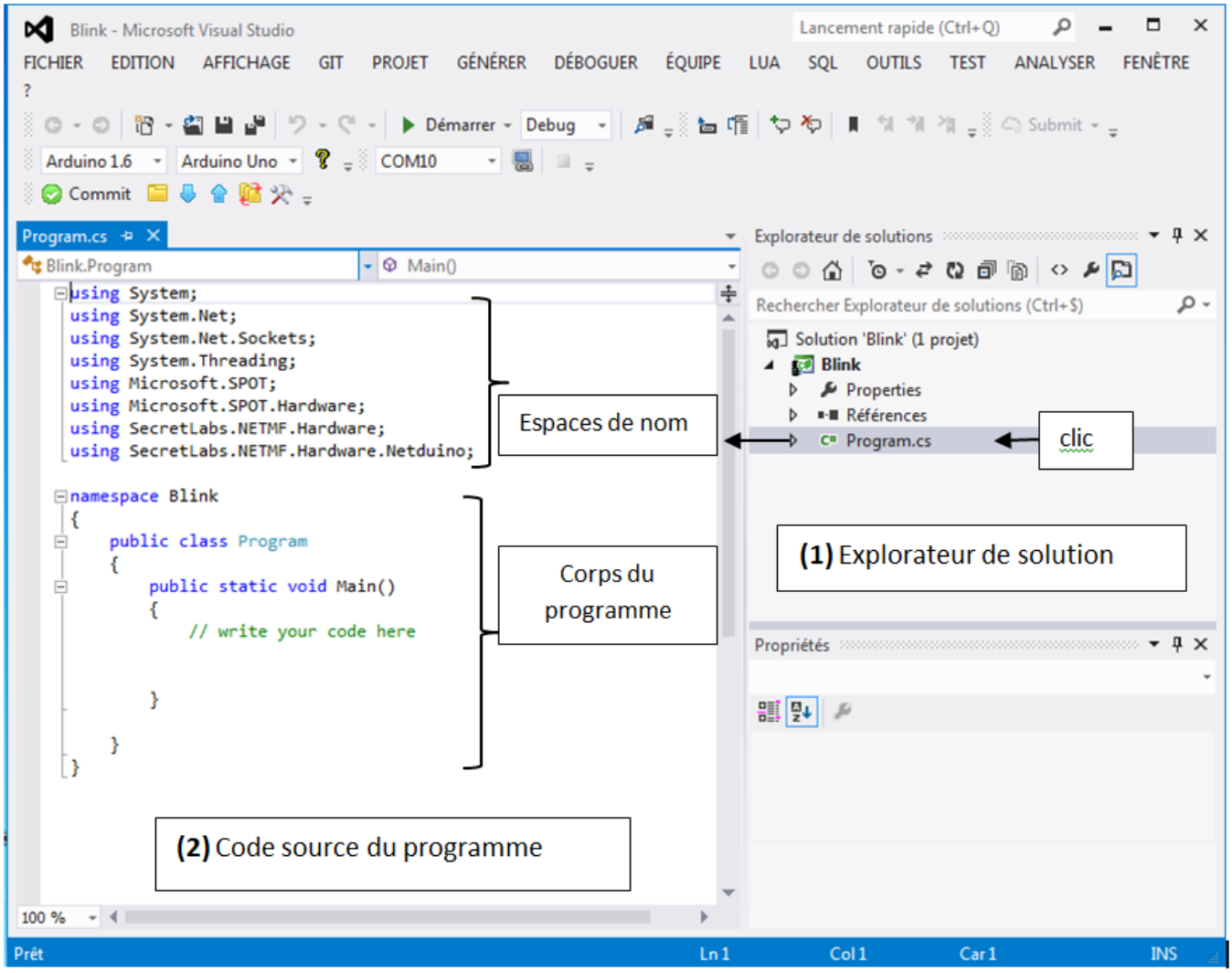
- Donnez le nom "**Blink**" à l'application puis cliquez sur **Ok**.



- **Cliquez** sur Program.cs dans la fenêtre **Explorateur de solution**

Remarque: L'emplacement (Location) identifié ci-dessous peut changer en fonction de la version du logiciel et de l'arborescence des répertoires du PC.

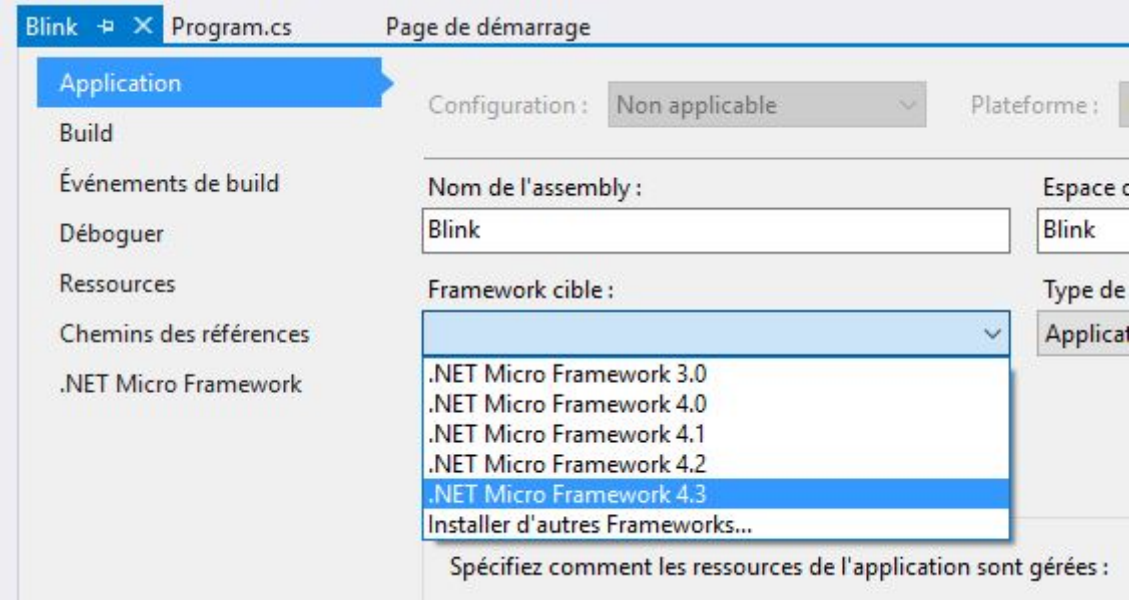
L'IDE est alors configuré comme sur la copie d'écran ci-dessous (ou un équivalent selon sa version) :



Remarque: Les espaces de nom sont des racourcis. Le projet en contient plus ou moins selon son type.

Le projet ***Blink*** est contenu dans la solution ***Blink***. Vous allez écrire le code dans le fichier **Program.cs**.

- Définissez la **version du framework** en sélectionnant **Projet → Propriétés de Blink → Framework cible** comme dans la copie d'écran ci-dessous.



Etape 2 : Editer le code source du programme

1. Déclaration des “Espaces de noms” et sélection des “bibliothèques” correspondantes

Une partie du code a été renseigné par Visual Studio. C’est le cas de la liste des espaces de noms installés par défaut et de la structure minimum du corps de programme (**repère 2 de la copie d'écran ci-dessus**). Nous allons commencer par compléter la liste des espaces de nom et installer les bibliothèques correspondantes.

Pour le moment seuls les espaces de nom suivants sont présents :

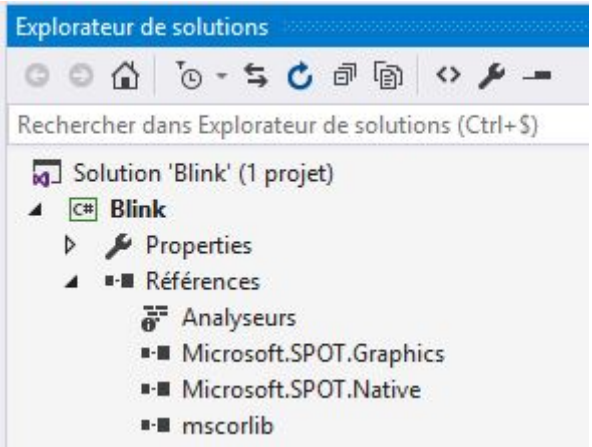
```
using System;
using Microsoft.SPOT;
```

Modifiez cette liste pour qu'elle corresponde à la copie d'écran ci-dessous.

```
using System;
using Microsoft.SPOT;
using System.Threading;
using Microsoft.SPOT.Hardware;
using GHI.Pins;
```

Visual Studio souligne **using GHI.pins** en rouge spécifiant qu’il y a un problème. Cela est dû au fait qu’il ne trouve pas la bibliothèque correspondant à ce nouvel espace de noms. **Une autre manipulation est nécessaire** !

Dépliez le répertoire “**Références**” (References) dans l’**explorateur de solution** pour obtenir la copie d'écran ci-dessous.



Pour ajouter la nouvelle bibliothèque: effectuez un clic droit sur “Références” (Reference) puis “Ajouter une référence” (Add Reference) et dans la liste, sélectionnez : **GHI.pins**. La nouvelle référence apparaît dans le répertoire “Références” et l’erreur disparaît dans le code.

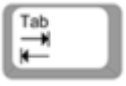
2. Partie déclarative du programme : construction d'un objet virtuel “Led” pour contrôler la LED de la carte

Entrez la ligne ci-dessous à la place du commentaire : *write your code here*

led.cs

```
var Led = new OutputPort(Pins.ONBOARD_LED, false); //(ligne 1)
```

Remarque : Vous pouvez constater l’effet de l’**autocompletion (intellisense)** au fur et à mesure de la construction de la ligne. Les propositions faites par le logiciel sont une aide précieuse lors de l’écriture du code.

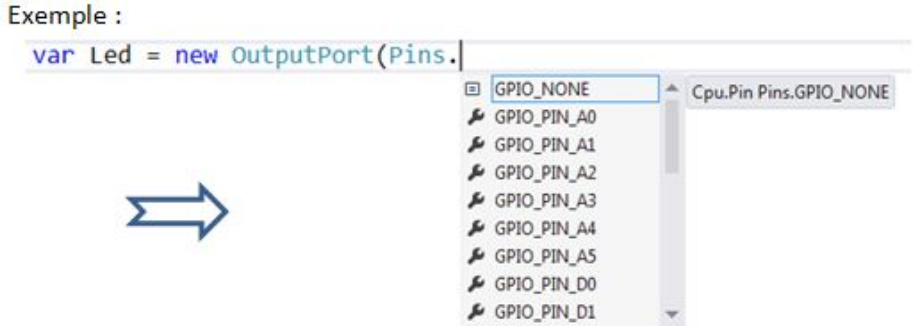
Il n’est pas nécessaire d’écrire complètement un mot : un **appui sur la touche tabulation** 

permet de l’insérer lorsqu'il est sélectionné.

Signification de quelques icônes

	Classe
	Constante
	Enumération
	Méthode ou fonction
	Mot clé du langage
	Propriété

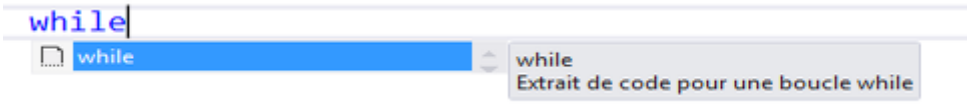
Exemple :



3. Partie exécutive du programme (le corps du programme) : placée dans une boucle

infinie

Ecrivez le mot ***while*** à la suite de la **ligne 1** dans le corps du programme. Le logiciel vous propose un gabarit (template) de code.



Appuyez **deux fois** sur la touche de tabulation 

. Le logiciel construit la structure ***while*** pour vous.

Complétez la structure ***while*** avec le code ci-dessous (les commentaires **peuvent être omis**) :

```
while (true) // Constitue une boucle infinie
{ // début du bloc de code : ici la boucle while
  Led.Write(true); // Eclairer la Led
  Thread.Sleep(500); // pendant 0,5s
  Led.Write(false); // Eteindre la Led
  Thread.Sleep(500); // pendant 0,5s
} // fin du bloc de code : ici la boucle while
```

Etape 3 : Générer la solution

Dans l'éditeur, sélectionner : **Générer (Build) → Générer la solution (Build Solution) ou [Ctrl+Maj+B]**. Si il n'y a pas d'erreurs dans le code le logiciel indique : La génération a réussi (en bas et à gauche de la fenêtre) . Vous allez maintenant pouvoir transférer le programme dans la carte Netduino et l'exécuter.

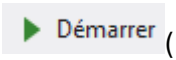
Etape 4 : Transférer le programme et l'exécuter

1. Affichage de la fenêtre de sortie Cette fenêtre donne des informations lors de l'exécution d'un programme. Elle est utile lors de sa mise au point ou pour remplacer un périphérique d'affichage.

Dans l'éditeur, sélectionner : **Affichage (view) → Sortie (Output) ou [Ctrl+Alt+O]**

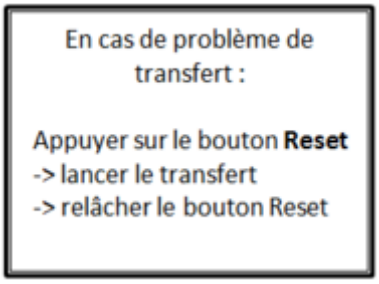
2. Transfert et exécution du programme dans la carte

Vérifiez que l'éditeur est en mode Debug . Pour transférer et exécuter le programme dans la carte,

cliquez sur 

(Start) ou appuyez sur la **touche F5**. Après une série d'actions, l'IDE doit afficher " Prêt " **(Ready) sur un fond rouge** en bas et à gauche de la fenêtre.

La LED de la carte doit clignoter !



Etape 5 : Tester le programme en mode pas à pas

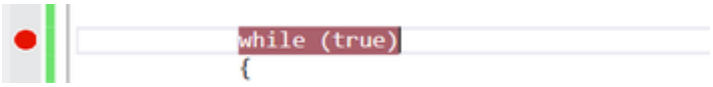
Pour mettre un programme "**au point**" (**déboguer**), vous pouvez l'arrêter, le redémarrer ou le mettre en pause.

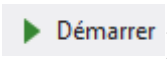
Pour bien comprendre les possibilités du débogueur, vous allez faire fonctionner le programme en **mode pas à pas.**

Cliquez sur l'icône "Arrêter" et modifiez votre code comme ci-dessous :

```
while (true)
{ // début du bloc de code : ici la boucle while
  Led.Write(true); /* Eclairer la Led */
  Debug.Print("Led éclairée"); // Affiche le texte entre guillemet dans la fenêtre de sortie
  Thread.Sleep(500); // Attente de 0,5s
  Led.Write(false); // Eteindre la Led
  Debug.Print("Led éteinte"); // Affiche le texte entre guillemet dans la fenêtre de sortie
  Thread.Sleep(500); // Attente de 0,5s
} // fin du bloc de code : ici la boucle while
```

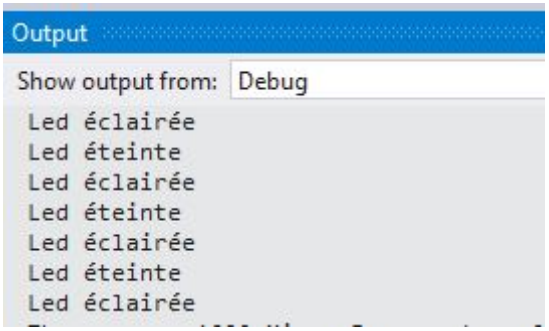
Placez un **point d'arrêt** (rond rouge) en cliquant à gauche du mot ***while*** comme ci-dessous.



Relancez le programme (touche F5 ou 

). Celui-ci s'arrête sur ***while***. Vous pouvez l'exécuter le code ligne par ligne (**mode pas à pas**) en appuyant sur la **touche F10**. (Un appui exécute une ligne)

Résultat attendu dans la fenêtre de sortie et comportement de la LED correspondant !



3. Deuxième programme : "MesureAngle" (entrée analogique et sortie numérique)

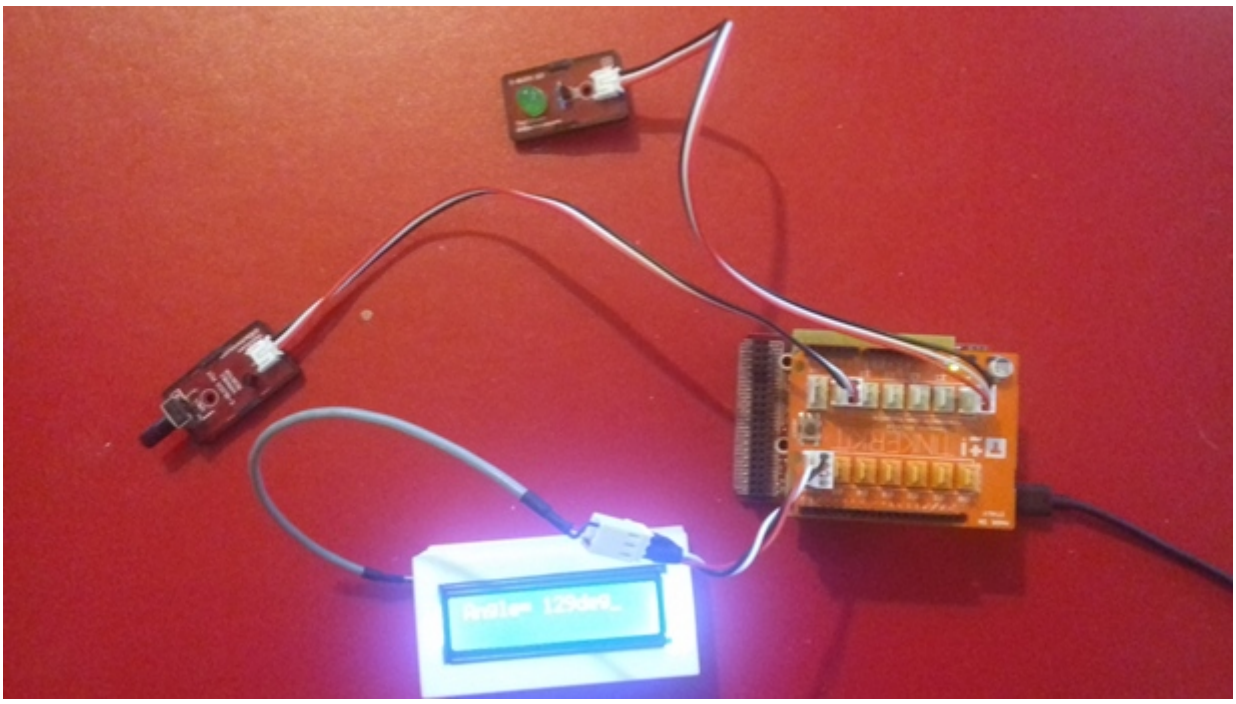
Cahier des charges du programme 2

Afficher la position angulaire d'un axe sur un lcd.

Matériels

- Carte Panda 3 et shield Tinkerkit
- Potentiomètre
- Afficheur LCD 2 lignes de 16 caractères et module ELCD162

Montage à réaliser



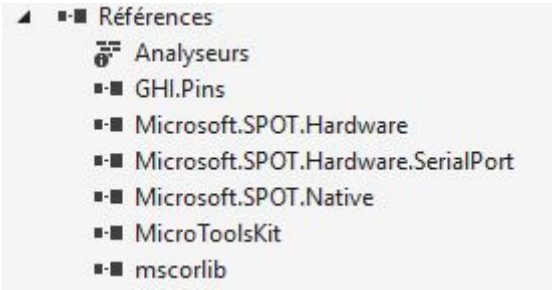
Etape 1 : Créer le projet "MesureAngle"

Reprenezla démarche vue dans le programme 1. Nommez le projet *MesureAngle*

Etape 2 : Editer le code du fichier Program.cs

1. Sélection des “bibliothèques” et déclaration des “Espaces de noms”

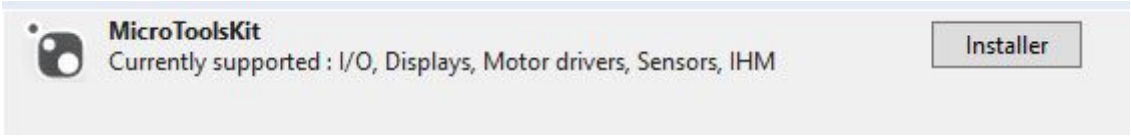
a. Modifiez la liste des bibliothèques du projet comme ci-dessous :



Remarque : La bibliothèque **MicroToolskit** est à télécharger sur Nuget.org en suivant la procédure suivante.

1. clic droit sur “*Références*”
2. Sélectionnez “*Gérer les packages NuGet*”
3. Dans “*Source de package*”, sélectionnez “*nuget.org*”
5. Positionnez la fenêtre sur *Parcourir* et entrez **Microtoolskit** dans la barre de “*Recherche en ligne*”

La bibliothèque apparaît comme sur la copie d'écran ci-dessous. Cliquez sur “*Installer*”.



b. Modifiez la liste des espaces de noms (on conserve seulement ceux qui sont utiles au projet !)

using.cs

```
using System;
using System.Threading;
using Microsoft.SPOT.Hardware;
using GHI.Pins;

using Microtoolskit.Hardware.Displays;
```

Remarque : le contenu de la bibliothèque MicroToolsKit est accessible en effectuant un clic droit sur son nom dans "*Référence*" et en sélectionnant "*Afficher dans l'explorateur d'objets*".

2a. Partie déclarative du programme

Complétez le fichier "Program.cs" avec les extraits de code ci-dessous:

a. Déclaration des constantes

const.cs

```
public static void Main()
{
  // Constantes
  const UInt16 delay = 500;   // En ms
```

b. Déclaration des variables

var.cs

```
// A la suite du code précédent, entrer :
// Variables
var N = 0;
var Angle = 0;
```

c. Configuration des entrées, sorties

es.cs

```
// A la suite du code précédent, entrer :
// Configuration des E/S
// http://msdn.microsoft.com/en-us/library/hh421132.aspx (Description
```

```
de la classe AnalogInput)
 AnalogInput Potentiometre = new
AnalogInput(FEZPandaIII.AnalogInput.A5); // Connecté sur I5 Tinkerkit
 // http://webge.github.io/ELCD162/ (Description de la classe ELCD162)
 ELCD162 Lcd = new ELCD162("COM1"); // Connecté sur SERIAL Tinkerkit
```

d. Initialisation de l'afficheur

init.cs

```
 // A la suite du code précédent, entrer :
// Initialisation
Lcd.Init(); Lcd.ClearScreen(); Lcd.CursorOff();
```

2b. Partie exécutive du programme

while.cs

```
 // A la suite du code précédent, entrer :
 while (true)
 {
   N = Potentiometre.ReadRaw();
   Angle = (270 * N ) / 4096;
   Lcd.ClearScreen();
   Lcd.PutString("Angle =" + Angle.ToString("F1") + "deg");
   Thread.Sleep(delay);
 }
```

Etape 3 : Générer la solution

Reprenez la démarche vue dans le programme 1.

Etape 4 : Transférer le programme et l'exécuter

Reprenez la démarche vue dans le programme 1.

Exercice

Modifiez le programme pour qu'une LED simulant une alarme (à rajouter sur la carte) s'éclaire si l'angle n'est pas compris dans l'intervalle [**120°, 150°C**]. En C#, le **OU** logique se code **||**, le **ET** logique se code **&&**.

Sources des exemples Les sources des exemples (compilées avec Visual Studio Community 2015) sont téléchargeables ici.