[ARCHIVES] Premiers programmes en C# "étape par étape" avec une carte Netduino+ v2

[Mise à jour le : 20/3/2019]

1. Préambule

Pour mener à bien ce tutoriel vous devez disposer d'une carte Netduino plus 2 (ou Netduino 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** » téléchargeable 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 WILDERNESS LABS.

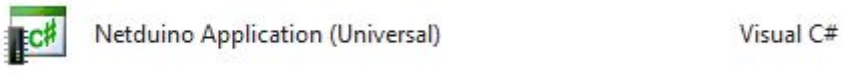
2. Premier programme : Blink

Cahier des charges du programme 1

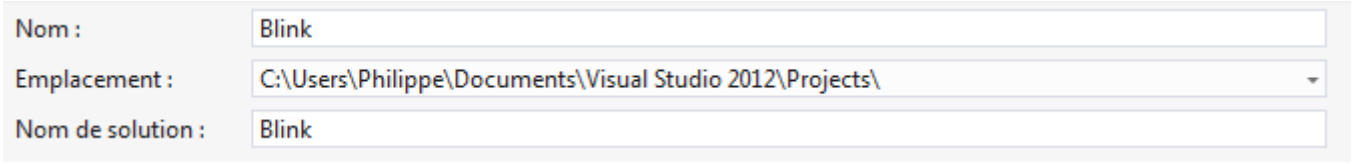
Faire clignoter la LED de la carte Netduino !

Étape 1 : Créer un projet

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



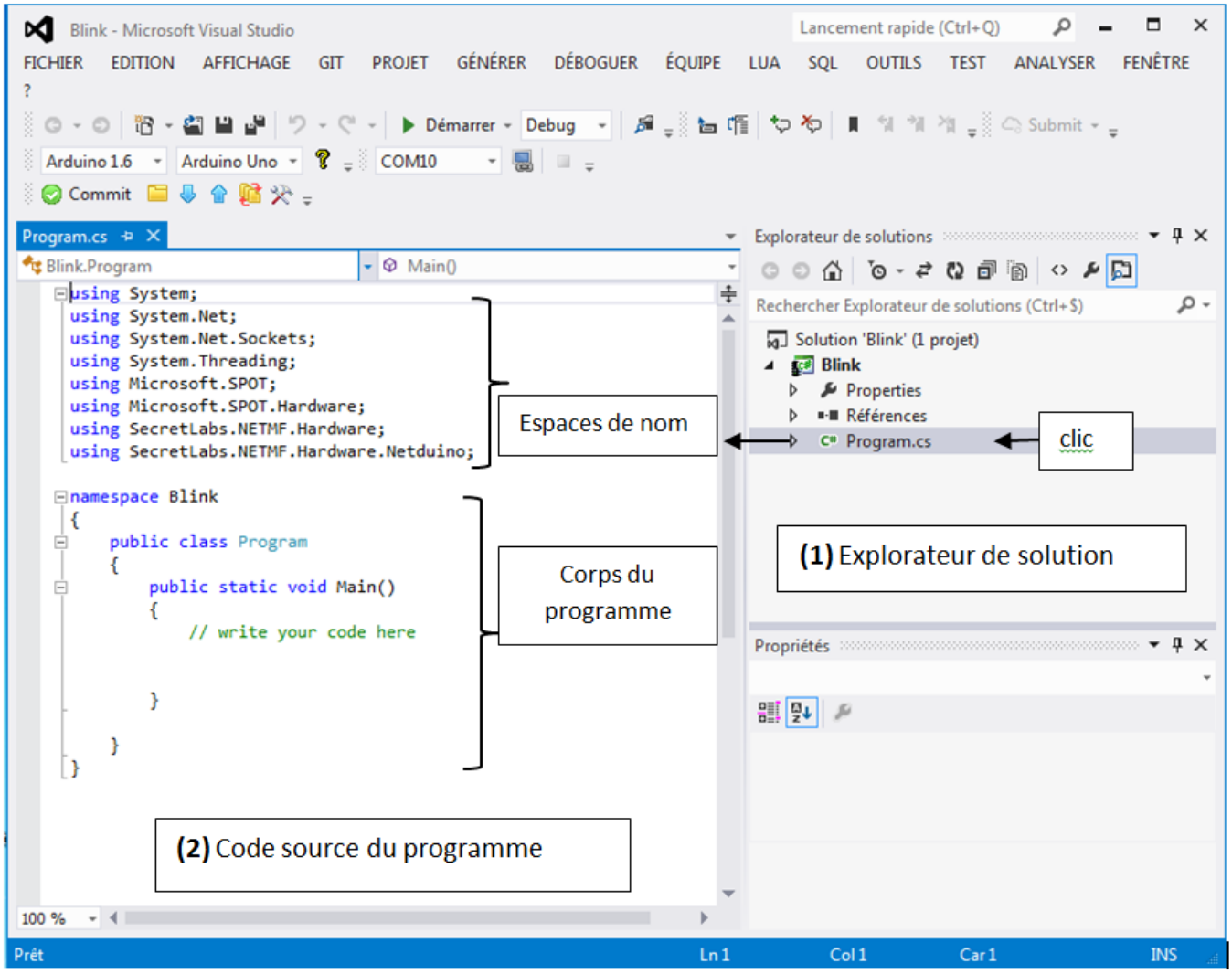
- Donner le nom "**Blink**" à l'application puis cliquer sur **Ok**.



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

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

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 raccourcis .

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

Étape 2 : Éditer le code 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

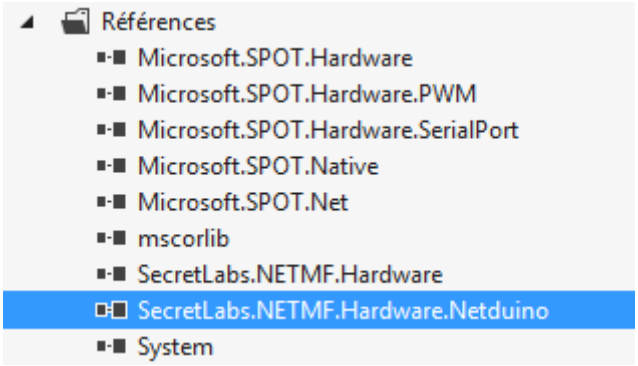
installés par défaut et de la structure minimum du corps de programme (**repère 2 de la copie d'écran ci-dessus**). Nous allons commencer par changer un de ces espaces de noms et la référence vers la bibliothèque lui correspondant.

Pour le moment l'espace de noms suivant : ***SecretLabs.NETMF.Hardware.Netduino*** n'est pas destiné à la bonne cible puisque nous avons une Netduino plus 2 (ou Netduino 3 Wifi).

Le modifier en ***SecretLabs.NETMF.Hardware.NetduinoPlus;*** dans le code source du programme (**repère 2 ci-dessus**).

Visual Studio le souligne 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** !

Déplier le répertoire "**Références**" (References) dans l'**explorateur de solution** et supprimer SecretLabs.NETMF.Hardware.Netduino par un clic droit puis "Supprimer" (Remove).



Pour ajouter la nouvelle bibliothèque: effectuer un clic droit sur "Références" (Reference) puis "Ajouter une référence" (Add Reference) et dans la liste, sélectionner : **SecretLabs.NETMF.Hardware.NetduinoPlus**. 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

Entrer 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.

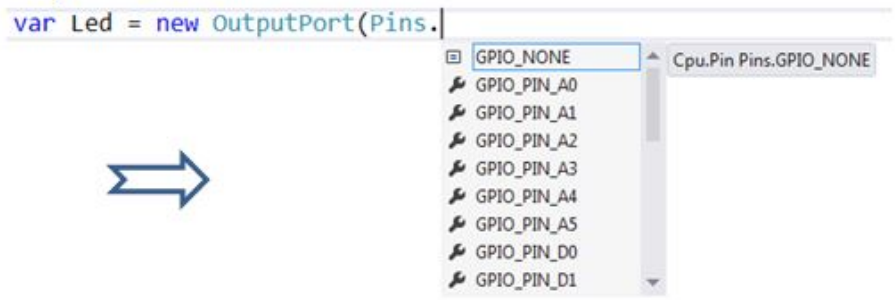
Il n'est pas nécessaire d'écrire complètement un mot : un **appui sur la touche tabulation** 

permet de l'insérer lorsqu'il est sélectionné.

Signification de quelques icônes

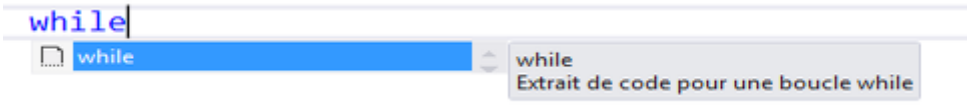
	Classe
	Constante
	Enumération
	Méthode ou fonction
	Mot clé du langage
	Propriété

Exemple :



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

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



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

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

Complétez la structure ***while*** avec le code ci-dessous :

```
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
```

Remarque : les commentaires (le texte après le double slash) **peuvent être omis**.

Étape 3 : Générer la solution

Dans l'éditeur, sélectionner : **Générer (Build) → Générer la solution (Build Solution) ou [Ctrl+Maj+B]**. S'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.

Étape 4 : Transférer le programme dans la carte Netduino 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

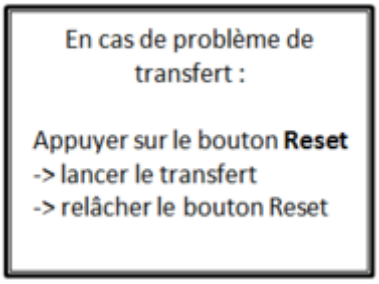
mise au point ou pour remplacer un périphérique d'affichage.

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

2. Transfert et exécution du programme dans la carte Netduino

Connecter la carte au PC. Vérifier que l'éditeur est en mode Debug. Pour transférer et exécuter le programme dans la carte, cliquer sur Démarrer (Start) ou appuyer 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 !



Étape 5 : Tester le programme en mode pas-à-pas

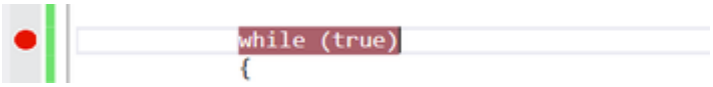
Pour mettre un programme "**au point**" (**déboguer**), vous pouvez l'arrêter, le redémarrer ou le mettre en pause.

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
```

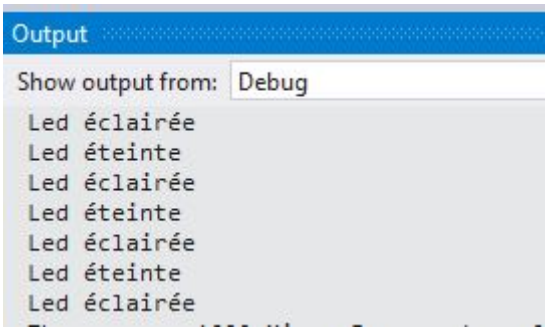
Placer un **point d'arrêt** (rond rouge) en cliquant à gauche du mot ***while*** comme ci-dessous.



Relancer le programme (touche F5 ou 

). Celui-ci s'arrête sur ***while***. Vous pouvez 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 !



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

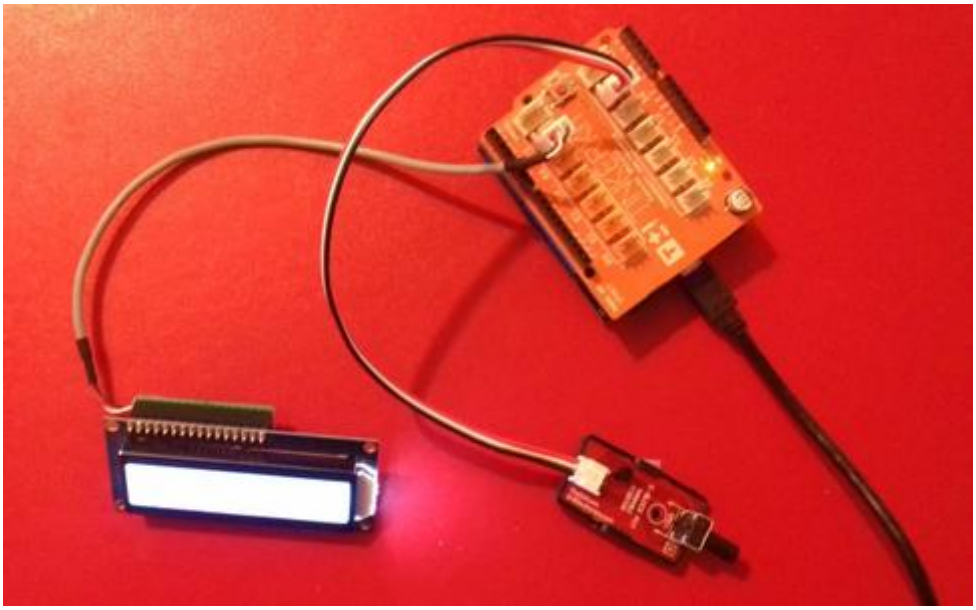
Cahier des charges du programme 2

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

Matériels

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

Montage à réaliser



Étape 1 : Créer le projet "MesureAngle"

Reprendre la démarche vue dans le programme 1. Nommer le projet : *MesureAngle*.

Étape 2 : Éditer le code du fichier Program.cs

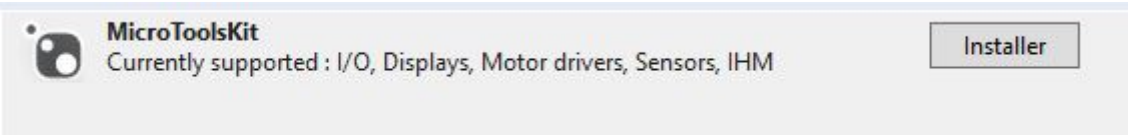
1. Sélection des “bibliothèques” et déclaration des “Espaces de noms”

a. Modifier 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. Sélectionner “*Gérer les packages NuGet*”
3. Dans “*Source de package*”, sélectionner “*nuget.org*”
4. Positionner la fenêtre sur *Parcourir* et entrer **Microtoolskit** dans la barre de “*Recherche en ligne*”

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



b. Modifier 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;
using Microsoft.SPOT.Hardware;

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

Compléter 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(Cpu.AnalogChannel.ANALOG_5); // Connecté sur I5 Tinkerkit
// http://webge.github.io/ELCD162/ (Description de la classe ELCD162)
```

```
// http://webge.github.io/ELCD162/ (Description de la classe ELCD162)
ELCD162 Lcd = new ELCD162(); // Connecté sur O5 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);
}
```

Étape 3 : Générer la solution

Reprendre la démarche vue dans le programme 1.

Étape 4 : Transférer le programme et l'exécuter

Reprendre la démarche vue dans le programme 1.

Exercice 1

> Écrire l'algorithme corrspondant à ce programme.

Exercice 2

> Modifier 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 **&&**. Compléter l'algorithme précédent.

Exercice 3

> Modifier le programme pour qu'un Buzzer (commandé en PWM), simulant une alarme, se déclenche si l'angle n'est pas compris dans l'intervalle [**120°, 150°C**].Compléter l'algorithme précédent.

4.Troisième programme : "MesureTemperature" (entrée numérique et sortie numérique)

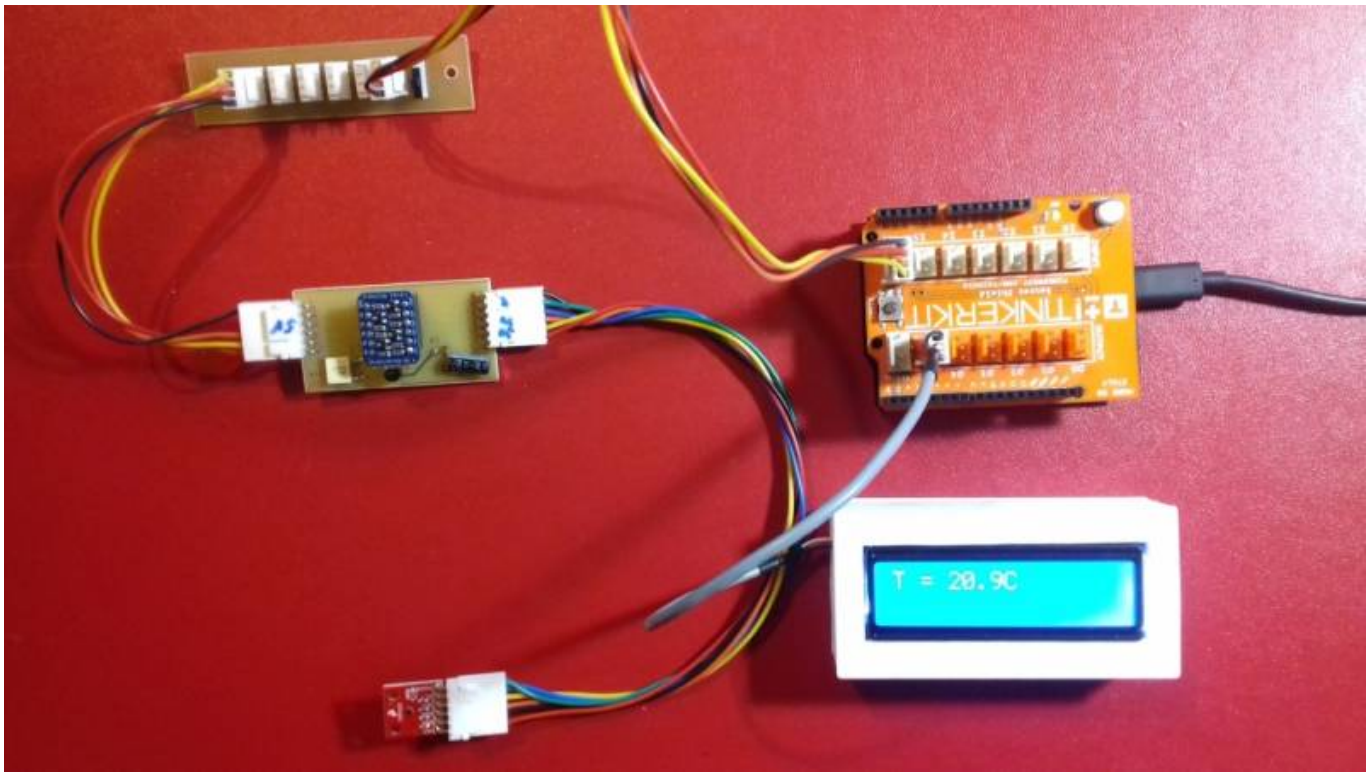
Cahier des charges du programme 3

Mesurer la température ambiante et l'afficher sur un LCD.

Matériels

- Carte Netduino et shield Tinkerkit
- Capteur de température TMP102 (I2C) + adaptateur de tension 3,3V ←→ 5V
- Afficheur LCD 2 lignes de 16 caractères et module ELCD162

Montage à réaliser



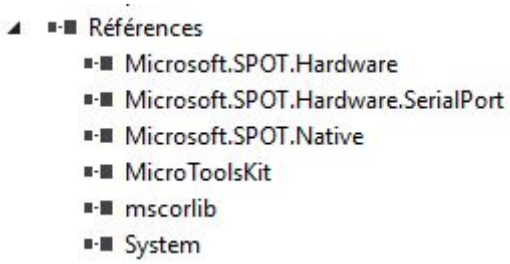
Étape 1 : Créer le projet "MesureTemperature"

Reprendre la démarche vue dans le programme 1. Nommer le projet : *MesureTemperature*.

Étape 2 : Éditer le code du fichier Program.cs

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

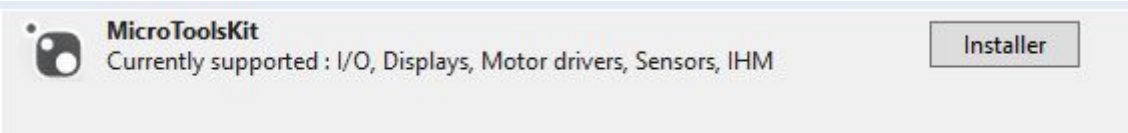
a. Modifier la liste des bibliothèques du projet comme ci-dessous :



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

1. clic droit sur “*Références*”
2. Sélectionner “*Gérer les packages NuGet*”
3. Dans “*Source de package*”, sélectionner “*nuget.org*”
4. Positionner 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. Cliquer sur “*Installer*”.



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

using.cs

```
using System;
using System.Threading;
using Microtoolskit.Hardware.Sensors;
using Microtoolskit.Hardware.Displays;
```

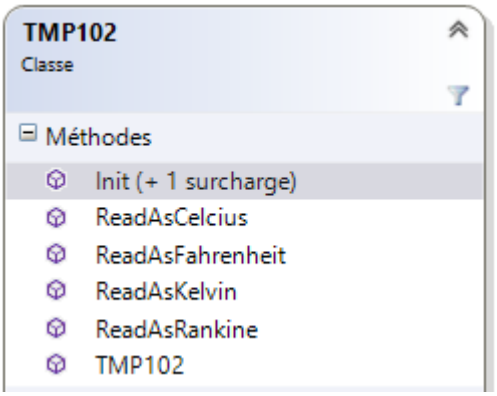
Remarque : le contenu de la bibliothèque MicroToolsKit est accessible en effectuant un clic droit sur son nom dans “*Référence*” et en sélectionnant “*Afficher dans l'explorateur d'objets*”.

2a. Partie déclarative du programme

Cette partie est donnée plus loin dans ce document.

2b. Partie exécutive du programme (le corps du programme) dans une boucle infinie

Informations complémentaires pour la mise en œuvre du capteur TMP102



1. Les méthodes accessibles à un objet de type TMP102 sont listées ci-contre.
Remarque : pour information, leur description est présentée sur le site github dans le §2.4 de la page accessible ici

2. Le capteur est initialisé en appliquant la méthode *Init()* à l'objet *ModuleMesureTemp*.

3. Une variable temperature est destinée à recevoir le résultat de la mesure.

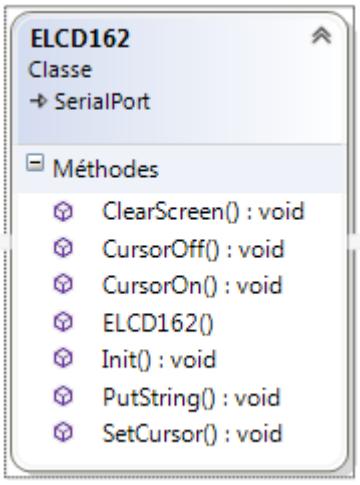
Exemple :

```
var temperature = 0.0;
```

4. La mesure est déclenchée par la méthode *ReadAsCelcius()* appliquée à l'objet *ModuleMesureTemp* (résultat en °C).

5. L'affichage se fera sur un LCD à commande série de type ELCD162.

Informations complémentaires pour la mise en œuvre de l'afficheur ELCD162



1. Les méthodes accessibles à un objet de type ELCD162 sont listées ci-contre.
Remarque : pour information, leur description est présentée sur le site github dans le §2.4 de la page accessible ici

2. L'afficheur est initialisé en appliquant la méthode *Init()* à l'objet *Lcd*.

3. L'écriture sur l'afficheur se fait avec la méthode *PutString()* appliquée à l'objet *Lcd*. Le texte est mis en forme en utilisant l'opérateur de concaténation "+". Le nombre de chiffres après la virgule est limité en passant le texte *Fx* (x = nombre de décimales) à la méthode *ToString* comme dans l'exemple ci-dessous.

Exemple : Pour afficher T = xx.xC (xx.x étant la température mesurée) on écrira :

```
Lcd.PutString("T = " + temperature.ToString("F1") + "C");
```

> **Exercice 4** : compléter le fichier *Program.cs* du projet avec le code ci-dessous et renseigner les parties manquantes puis rédiger l'algorithme correspondant à ce code.

```
namespace MesureTemperature
{
    public class Program
    {
        public static void Main()
        {
            // Variables
            var temperature = 0.0; // variable destinée à recevoir le
résultat de la mesure de température

            // Création des objets virtuels
            var ModuleMesureTemp = new TMP102();  // Création du capteur
virtuel
            var Lcd = new ELCD162(); // Création de l'afficheur virtuel

            // Initialisations
            ModuleMesureTemp.Init();
            Lcd.Init(); Lcd.ClearScreen(); Lcd.CursorOff(); // COM2


            while (true)
            {
                // Lire la température
                                                    // <- A compléter

                // Effacer le lcd
                                                    // <- A compléter

                // Afficher la température sur le lcd
                                                    // <- A compléter

                Thread.Sleep(1000); // Attendre une seconde entre chaque
mesure
            }
        }
    }
```

```
}
```

Étape 3 : Générer la solution

Reprendre la démarche vue dans le programme 1.

Étape 4 : Transférer le programme et l'exécuter

Reprendre la démarche vue dans le programme 1.

Exercice 5

> Remplacer le LCD par un afficheur graphique. (voir prof)

Sources des exemples

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