Premiers programmes en C# avec une carte BrainPad v1 ou v2 "Étape par Étape"

[Mise à jour le 4/9/2020]

- **Sources**
  - Site de GHI Electronics : Getting Started
- **Lectures connexes**
  - Les outils logiciels à installer pour programmer en C# sous TinyCLR-OS
- Un **guide en français** pour débuter

Préambule

Pour mener à bien ce tutoriel vous devez disposer d'une carte **BrainPad v1 ou v2**.

Microsoft **Visual Studio 2017 ou 2019** (Community) doit être installé sur le PC.

Le firmware de la carte doit être à jour. Si ce n'est pas le cas : suivez le "Guide d'installation" ou voir la vidéo "Updating BrainPad's Firmware - Tech Talk 041".

Les vidéos du cours sur les fondamentaux du langage C# sont accessibles sur le site MVA. Elles constituent un excellent préalable ou un complément à ce tutoriel.

---

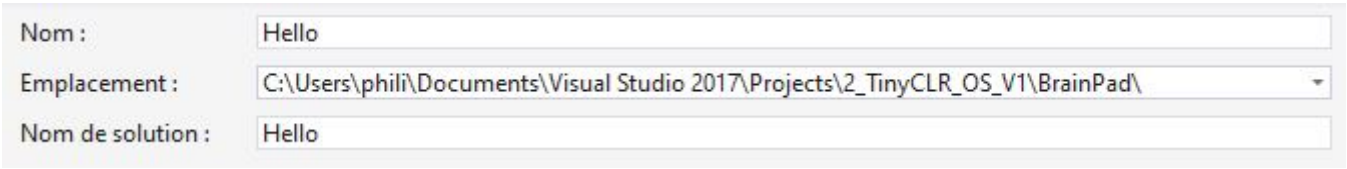
1. Un premier programme : Hello

Cahier des charges

Faire clignoter la LED **Light Bulb** (RVB) de la carte BrainPad et afficher le texte "**Hello, World !**" sur l'écran graphique.

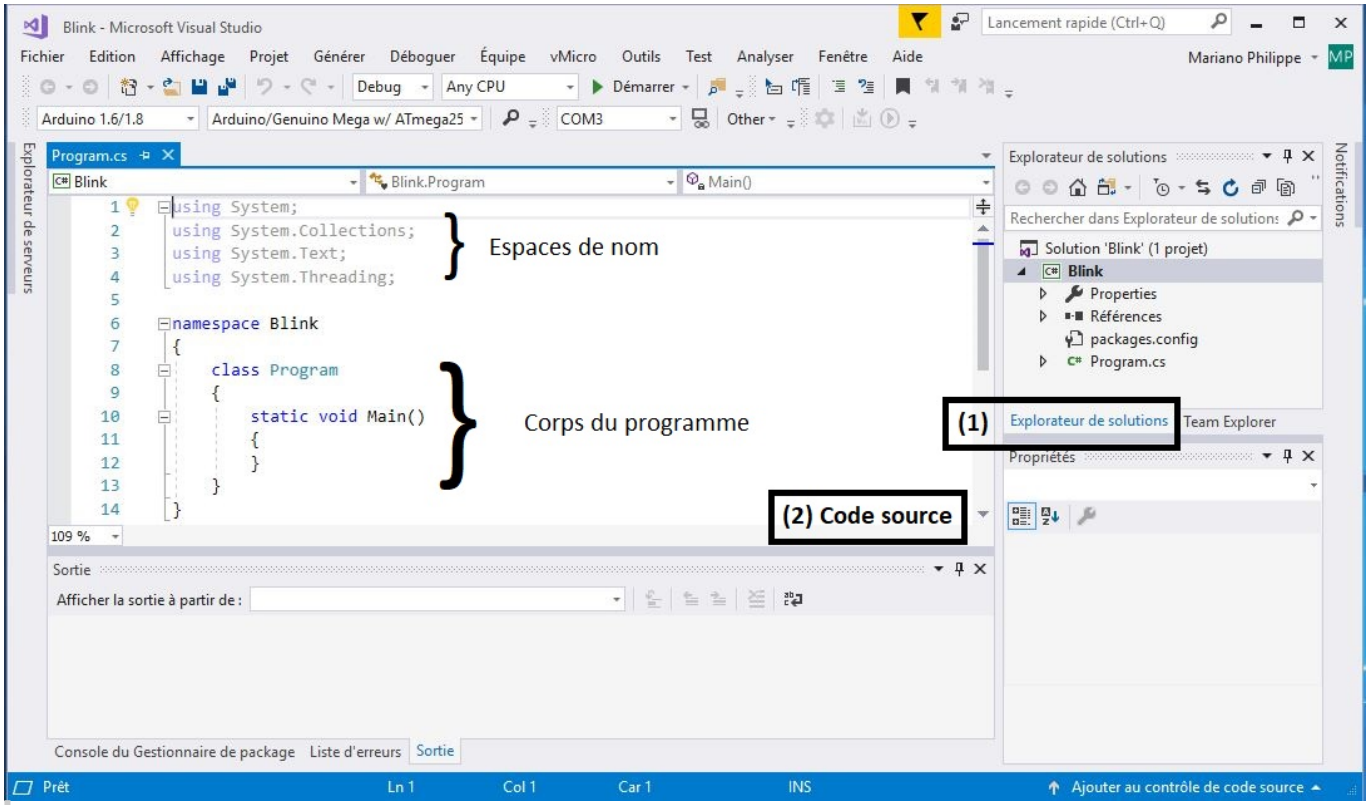
Etape 1 : Créer un projet avec le template BrainPad Application

- Ouvrir l'**IDE Microsoft Visual Studio Community 2017 ou 2019** 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) sélectionnez :
  Installé → Visual C# → TinyCLR puis **TinyCLR Application**.
- Donner le nom **"Hello**" à l'application puis cliquer sur Ok.



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

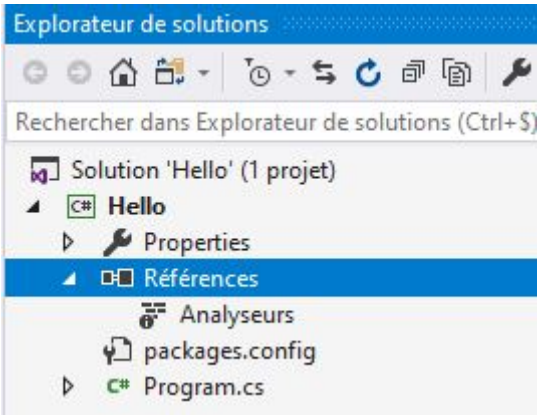
L'**IDE** est alors configuré comme sur la copie d'écran ci-dessous (ou un équivalent selon sa version) :



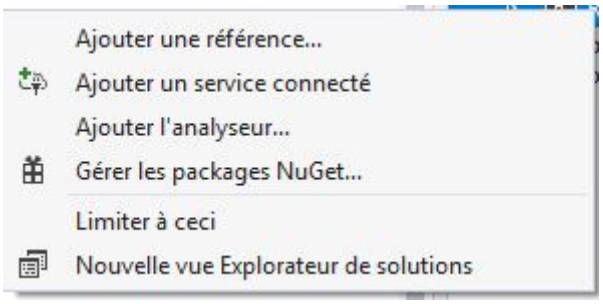
Le projet ***Blink*** est contenu dans la solution ***Blink***. Les espaces de nom sont des **racourcis**.

Etape 2 : Installer des bibliothèques dans le projet

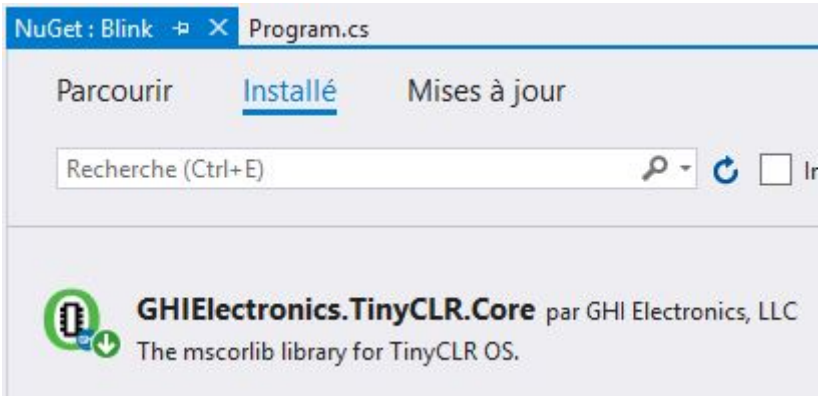
- Effectuer un **clic droit** sur **Références** dans l'explorateur de solution (s'il n'est pas visible : **Affichage → Explorateur de solution**)



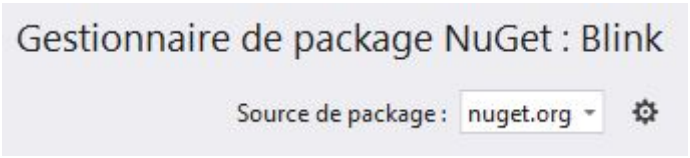
- Cliquer sur **“Gérer les packages NuGet...“**



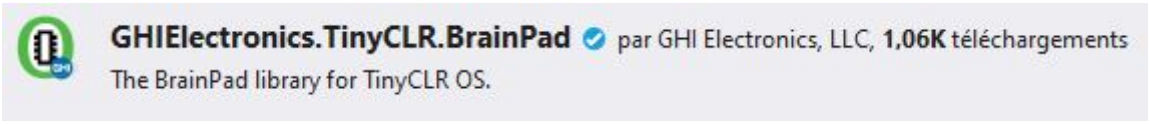
- Afficher la bibliothèque présente en vous plaçant sur l'onglet **“Installé”**.



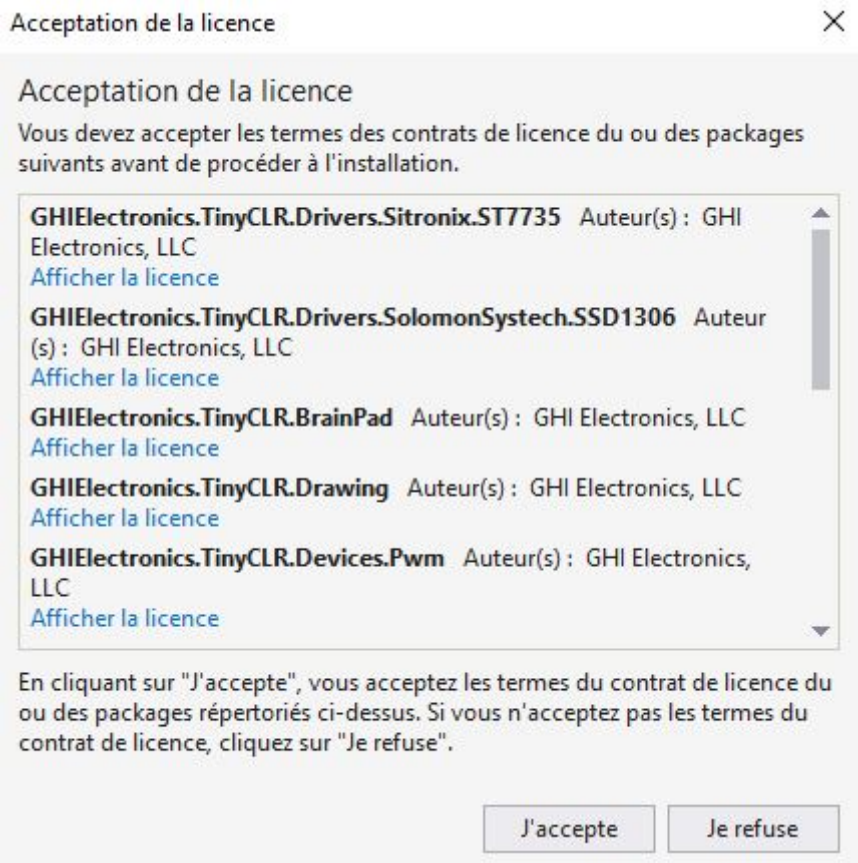
- Sélectionner **nuget.org** dans **“Source de package**”.



- Se placer dans l'onglet **“Parcourir”** et entrer **GHIElectronics.TinyCLR.BrainPad** .



- Cliquer sur ***GHIElectronics.TinyCLR.BrainPad*** puis “**Installer**” pour placer les biliothèques dans le projet.



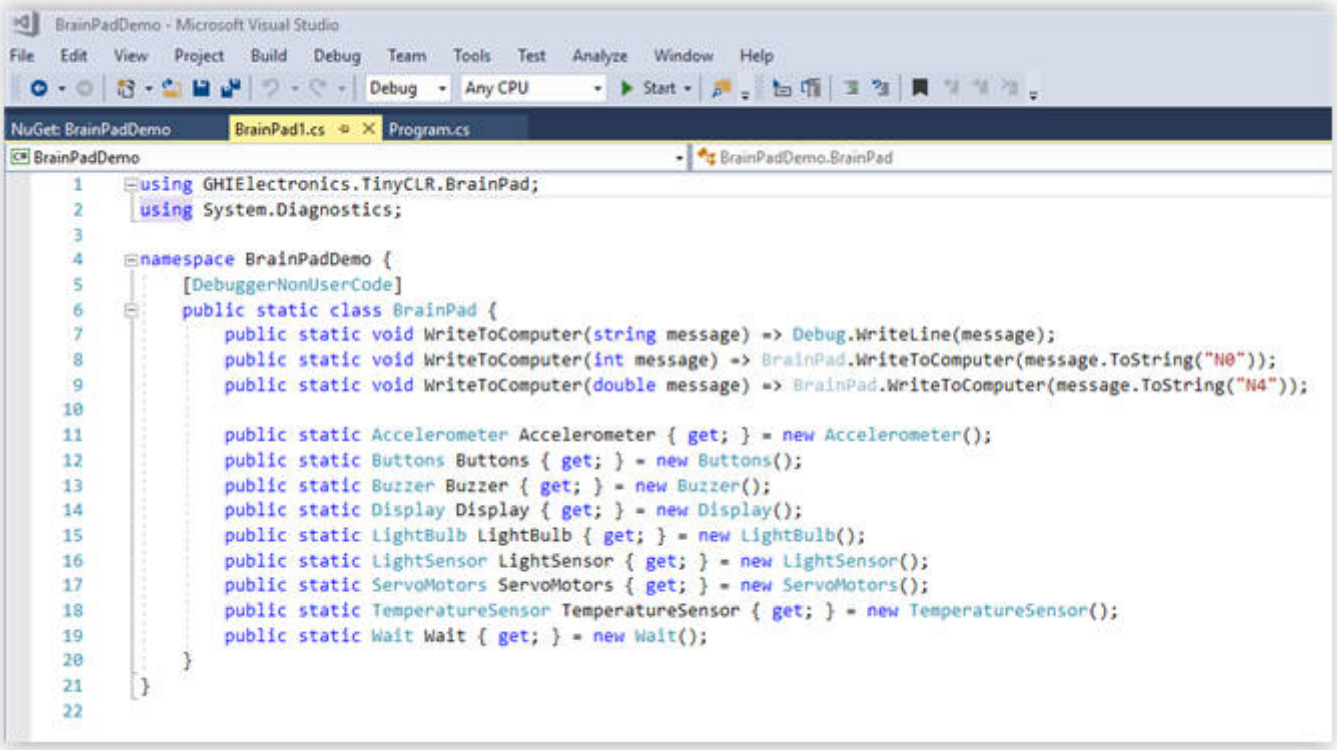
- Cliquer sur **“J'accepte”**. La liste des bibliothèques intallées est accessible dans **Références**.
- Fermer le gestionnaire de paquets

Etape 3 : Ajouter le code "Brainpad Helper"

Le code BrainPad Helper fournit les définitions nécessaires aux objets de la carte BrainPad.

- Sélectionner le nom du projet (Hello) puis clic-droit “**Ajouter → Ajouter un nouvel élément …**” ou (**Ctrl-Maj-A**).
- Dans la boîte de dialogue “**Ajouter un nouvel élément**”, cliquer sur **BrainPad Helper**, puis sur le bouton Ajouter.

- Vous verrez un onglet étiqueté **BrainPad1.cs** avec un contenu comme celui représenté ci-dessous.



- Fermer la fenêtre

Etape 4 : Editer le code source du programme

- Ecrire le code source du programme **“Hello”** ci-dessous dans l'onglet ***Program.cs***.

hello.cs

```
using System;
using System.Collections;
using System.Text;
using System.Threading;

namespace Hello
{
    class Program
    {
        static void Main()
        {
            BrainPad.Display.DrawText(0, 0, "Hello!");
            BrainPad.Display.RefreshScreen();

            while (true)
            {
                BrainPad.LightBulb.TurnWhite();
                BrainPad.Wait.Seconds(1);
                BrainPad.LightBulb.TurnOff();
                BrainPad.Wait.Seconds(1);
            }
        }
```

```
    }
}
```

Etape 5 : Télécharger et exécuter le programme dans la carte

- La carte étant connectée au PC avec le câble USB, le programme est compilé puis transféré en cliquant sur le bouton 

ou sur **F5**.
- La LED RVB **Light Bulb** clignote et le texte “**Hello!**” s'affiche sur l'écran comme ci-dessous.

- **Étude de l'exemple**

L'application démarre en exécutant **une seule fois** le code ci-dessous.

init.cs

```
    BrainPad.Display.DrawText(0, 0, "Hello!");
    BrainPad.Display.RefreshScreen();
```

La méthode **DrawText()** de l'objet Display placé sur la carte BrainPad affiche en blanc le texte “Hello!” à la position 0,0 de l'écran.

Puis, le code placé entre les accolades de la boucle **while()** s'exécute **indéfiniment**.

while.cs

```
while (true)
{
     BrainPad.LightBulb.TurnWhite();
     BrainPad.Wait.Seconds(1);
     BrainPad.LightBulb.TurnOff();
     BrainPad.Wait.Seconds(1);
}
```

La Led Bulb est éclairée puis éteinte successivement pendant 1s.

Exercice 1 : Modifiez le programme pour que la LED émette un flash de 100ms toutes les 1s. Affichez votre nom (prénom) à la position (10,10). Remarque : l'afficheur n'accepte pas les caractères accentués.

---

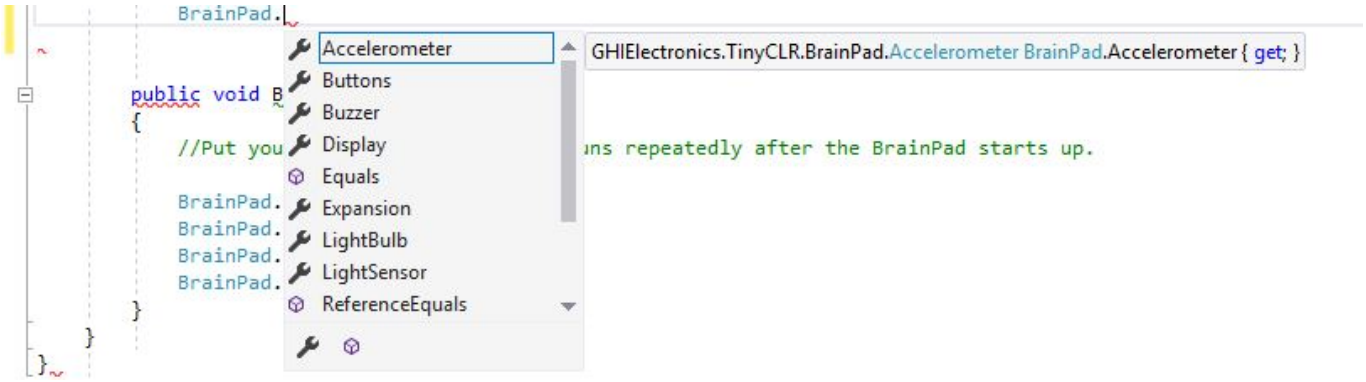
2. L'objet BrainPad

Dans le monde “réel”, une voiture, une table, un téléviseur sont des objets. Un **objet** peut être décrit par son **état** (la voiture est une Citroën bleue) et par son **comportement** (la voiture avance).

Dans le monde “virtuel” tout peut être assimilé à un objet : même une personne ! Les concepts d'**objet**, d'**état** et de **comportement** sont repris dans les langages de programmation dits “Objet” comme le **C#** utilisé pour programmer la carte BrainPad.

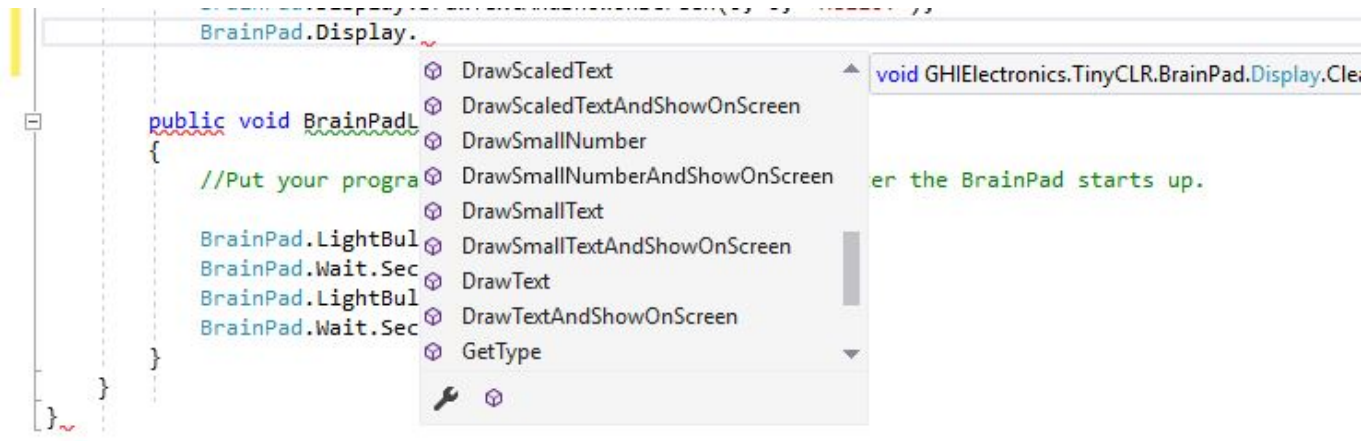
La carte BrainPad réelle est contrôlée par l'objet BrainPad virtuel. Cet objet est décrit par du code développé par la société GHI Electronics.

L'accès aux fonctionnalités de la carte BrainPad nécessite d'entrer le mot BrainPad **chaque fois qu'on y fait référence**. En faisant suivre ce mot par un **point** on obtient la liste des objets qui “composent” la carte tels que : **Display**, **Buzzer**, **Accelerometer** etc.



Tous ces “objets” sont accessibles à partir de leur nom. Il suffit d'en sélectionner un pour accéder à la liste de ses fonctionnalités.

Par exemple, en sélectionnant Display on obtient une liste comme celle représentée ci-dessous :



L'IDE Visual Studio simplifie la programmation en proposant automatiquement la liste des fonctionnalités d'un objet. Vous allez utiliser cette aide pour écrire le prochain programme.

> La description des classes de la bibliothèque BrainPad est disponible ici.

3. Bulb : un second programme pour se familiariser avec l'autocompletion

1. **Créer** un nouveau projet et nommez-le **Bulb**. **Installer** les bibliothèques et le fichier “BrainPad Helper”.

2. Le code source doit ressembler à la copie d'écran ci-dessous.

sourcebulb.cs

```
using System;
using System.Collections;
using System.Text;
using System.Threading;

namespace Bulb
{
    class Program
    {
        static void Main()
        {
        }
    }
}
```

3. **Copier** l'extrait de code ci-dessous entre les accolades de Main().

mainbulb.cs

```
static void Main()
{
   BrainPad.LightBulb.TurnOff();
   BrainPad.LightBulb.TurnWhite();
```

```
    BrainPad.LightBulb.TurnOff();
    BrainPad.LightBulb.TurnWhite();
}
```

4. **Télécharger** et exécuter le code en cliquant sur Démarrer ou en appuyant sur **F5**.

Problème : la Led ne clignote pas, mais reste constamment éclairée. Ceci vient du fait que le programme fonctionne très rapidement. Nous n'avons donc pas le temps de voir la Led s'éclairer puis s'éteindre. Il est nécessaire de ralentir le programme !

Pour visualiser ce que fait le programme lorsqu'il s'exécute lentement, nous allons le faire fonctionner en mode pas-à-pas.

4. Le mode pas-à-pas pour tester un programme

Ce mode de fonctionnement permet de mettre un programme “au point” (**déboguer**). Dans ce mode, vous pouvez l'arrêter, le redémarrer ou le mettre en pause.

1. **Cliquer** sur l'icône “Arrêter”.

2. **Placer** un point d’arrêt en cliquant à gauche de la première ligne comme ci-dessous.

3. **Relancer** le programme (touche **F5** ou Démarrer). Celui-ci s’arrête. Vous pouvez exécuter le code ligne par ligne (mode **pas à pas**) en appuyant sur la **touche F10**. (Un appui exécute une ligne).

Note : cliquer sur le point d'arrêt pour le supprimer.

Exercice 2 : Modifier le programme pour que la Led clignote 2 fois par seconde (couleur bleue). On l'éteindra au démarrage de la carte.

---

5. BP : un troisième programme pour se familiariser avec la structure si... alors ... sinon

Algorithme

---

si (condition) **alors** Action1 **sinon** Action2 **fin s**i

---

L'action 1 est exécutée si la condition est vraie.

L'action 2 est exécutée si la condition est fausse. (L'action 2 peut être omise).

Cahier des charges

Si le bouton (**D**)own est appuyé la led s'éclaire en vert.

---

1. **Créer** un nouveau projet et le nommer **BP**. Installer les bibliotèques et le code "BrainPad Helper".

2. **Copier** le code source ci-dessous et l'exécuter.

Brainpad1.cs

```
using System;
using System.Collections;
using System.Text;
using System.Threading;

namespace BP
{
    class Program
    {
        static void Main()
        {
            BrainPad.LightBulb.TurnOff();
```

```
            while (true)
            {
                if (BrainPad.Buttons.IsDownPressed())
                {
                    BrainPad.LightBulb.TurnGreen();
                }
            }
        }
    }
}
```

Il semble qu'il y ait un problème : la Led ne s'éteint pas lorsqu'on relâche le bouton. Pour cela : il faut lui demander !

3. **Modifier** le programme comme ci-dessous.

Brainpad2.cs

```
using System;
using System.Collections;
using System.Text;
using System.Threading;

namespace BP
{
    class Program
    {
        static void Main()
        {
            BrainPad.LightBulb.TurnOff();
            while (true)
            {
                if (BrainPad.Buttons.IsDownPressed())
                {
                    BrainPad.LightBulb.TurnGreen();
                }
                else
                {
                    BrainPad.LightBulb.TurnOff();
                }
            }
        }
    }
}
```

Exercice 3 : Modifier le programme pour que la Led s'éclaire en bleu si l'on appuie sur (**L**)eft, en blanc si l'on appuie sur (**U**)p et en rouge si l'on appui sur (**R**)ight. On l'éteindra au démarrage de la

carte.

Contrainte : utiliser une structure alternative imbriquée dans le test !

Synthèse : "Jouer une partition avec le buzzer"

Les **méthodes** décrites dans le tableau ci-dessous permettent de contrôler le buzzer.

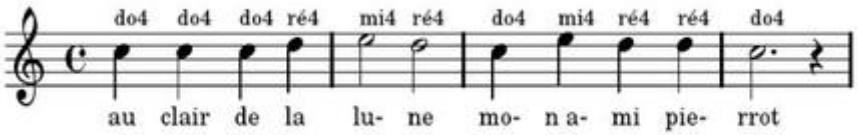
	Synthaxe	Description
S	*void* **StartBuzzing**(*double* Frequency)	Joue une fréquence.
S	*void* **StopBuzzing**()	Coupe le son.

Remarque : **void** signifie que la méthode ne renvoie pas d'informations. **double** signifie que la valeur à placer entre les parenthèses est un réel.

Exemple : BrainPad.Buzzer.StartBuzzing(523); (Joue un Do4)

Vous allez utiliser ces méthodes pour écrire le programme "**AuClairDeLaLune**" dont la partition est donné ci-dessous (ou un autre de votre choix) .

Partition



Les fréquences correspondant aux notes et le rythme à attribuer à une note sont précisés dans les ressources ci-dessous.

Ressources

- Fréquence des notes : lien
- Le rythme des notes de musique (ronde, blanche, noire...): lien
- La description des classes de la **bibliothèque BrainPad** est disponible ici.

Exercice 4

Version 1a : La carte joue seule la partition (une fois). (Nom du projet : **ClairLuneV1a**)

Version 1b : La carte joue seule la partition (une fois) et le texte de la chanson s'affiche sur l'écran. (Nom du projet : **ClairLuneV1b**) [Video]

Remarques : Une noire dure 0,5s

---

pour aller plus loin...

Version 2 : L'utilisateur joue la partition avec le clavier. (Nom du projet **ClairLuneV2**)

Indications : Utiliser des évènements pour gérer les boutons-poussoir (**voir prof**)

Events

Many modules generate useful events. Type +=<tab><tab> to add a handler to an event, e.g.: button.ButtonPressed +=<tab><tab>

Exemple

```
BrainPad.Buttons.WhenLeftButtonPressed += Buttons_WhenLeftButtonPressed;

private static void Buttons_WhenLeftButtonPressed()
        {
            play("do");
        }
```

Sources

Les sources des exemples et des exercices (compilés avec **Visual Studio Community 2017 ou 2019**) sont téléchargeables ici.
