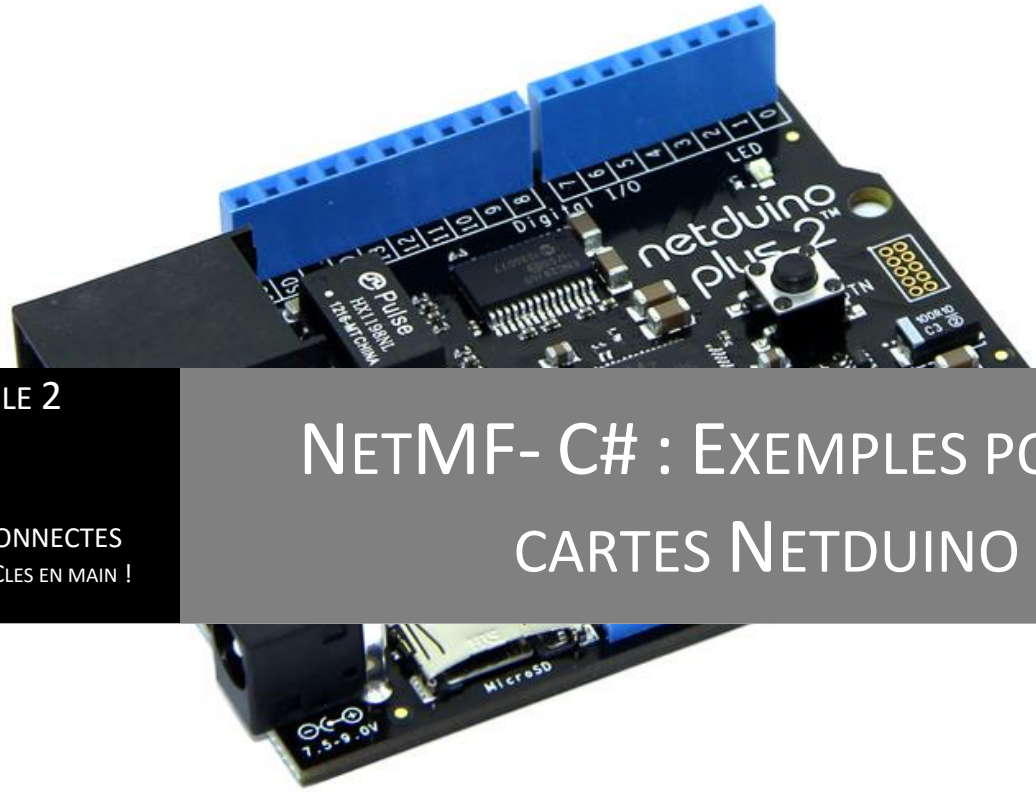
19/02/2016



FASCICULE 2
CLIENT
SERVEUR
OBJETS CONNECTES
CLES EN MAIN !

NETMF- C# : EXEMPLES POUR LES CARTES NETDUINO

V1.0.0

En cours de rédaction

philippemariano@gmail.com

Table des matières

Présentation

Ce document est le **deuxième tome** d'un recueil de programmes écrits en **C#**. Ils illustrent la mise en œuvre d'application **REST** destinées à transformer la carte Netduino plus 2 en **objet connecté** à un **réseau local** ou à **internet**. Sa configuration en **client et/ou en serveur http** est décrite à travers différents exemples.

Un premier tome est consacré à la mise en œuvre de cette même carte avec des capteurs et des prés-actionneurs.

Tous les programmes ont été compilés avec l'IDE **Visual Studio Express 2012**. Ils s'appuient sur la version **4.3 du microframework .NET**. Il est possible de réutiliser les fichiers sources avec d'autres cibles du même fabricant.

Les exemples de code écrit en C# sont adaptés du livre « **Getting Started with the Internet of Things »**. Ils sont utilisables sous licence **Apache Version 2.0**.

Ce document est divisé en **quatre chapitres** :

1. **Netduino plus 2 est un client sur un réseau local**
2. **Netduino plus 2 est un serveur sur un réseau local**
3. **Multithreading**
4. **L'internet des objets**

Les caractéristiques et la connectique de la carte Netduino plus 2 de la société Secret Labs sont présentées dans la suite de ce document.

Un shield Tinkerkit est également décrit en annexe. Il permet de relier facilement les différents capteurs et prés-actionneurs.

Les **projets** associés aux exemples sont répertoriés page suivante. Les sources de ces projets sont disponibles au téléchargement.

La documentation du microframework .NET 4.3 est consultable à partir du lien suivant:

http://msdn.microsoft.com/en-us/library/jj610646(v=vs.102).aspx

Le matériel

L'écosystème Netduino devrait plutôt être appelé l'écosystème .Net Micro Framework (NETMF). Le Micro Framework .Net est un sous-ensemble du Framework .Net créé par **Microsoft open source**. Netduino est une plateforme open source, développée par la société Secret Labs.

La carte Netduino plus 2 a la même configuration de broches que l'Arduino Uno V3. Elle est compatible avec un grand nombre de shields.

La gamme complète des matériels est disponible sur ici.

Caractéristiques des principales cartes (celles sur lesquelles les exemples ont été testés sont entourées ci-dessous)

mainboard	netduino 2	netduino plus 2	netduino 3 wi-fi
			
	more information »	more information »	more information »
processor and memory			
microcontroller	STMicro STM32F2	STMicro STM32F4	STMicro STM32F4
speed	120 MHz (Cortex-M3)	168 MHz (Cortex-M4)	168 MHz (Cortex-M4)
code storage	192 KB	384 KB	1408 KB
ram	60 KB	100+ KB	164+ KB
operating system	.NET Micro Framework 4.2 (or 4.3)	.NET Micro Framework 4.2 (or 4.3)	.NET Micro Framework 4.3
input and output			
networking	none	ethernet: 10 mbps	wi-fi: 802.11b/g/n including SSL/TLS 1.2 and WPA2
arduino shield compatibility	works with most arduino shields (some require .net mf drivers)	works with most arduino shields (some require .net mf drivers)	works with most arduino shields (some require .net mf drivers)
digital i/o	22 gpio, 6 pwm, 4 uart, i2c, spi	22 gpio, 6 pwm, 4 uart, i2c, spi	22 gpio, 6 pwm, 4 uart, i2c, spi
analog inputs	6 adc channels (12-bit)	6 adc channels (12-bit)	6 adc channels (12-bit)
storage	add-on: sd shields (up to 2 gb)	micro sd (up to 2 gb)	micro sd (up to 2 gb)
gobus ports	none	none	3 gobus ports
environmental			
operating temperature	0 - 70 °C (32 - 158 °F)	0 - 70 °C (32 - 158 °F)	0 - 70 °C (32 - 158 °F)
rohs	rohs compliant	rohs compliant	rohs compliant

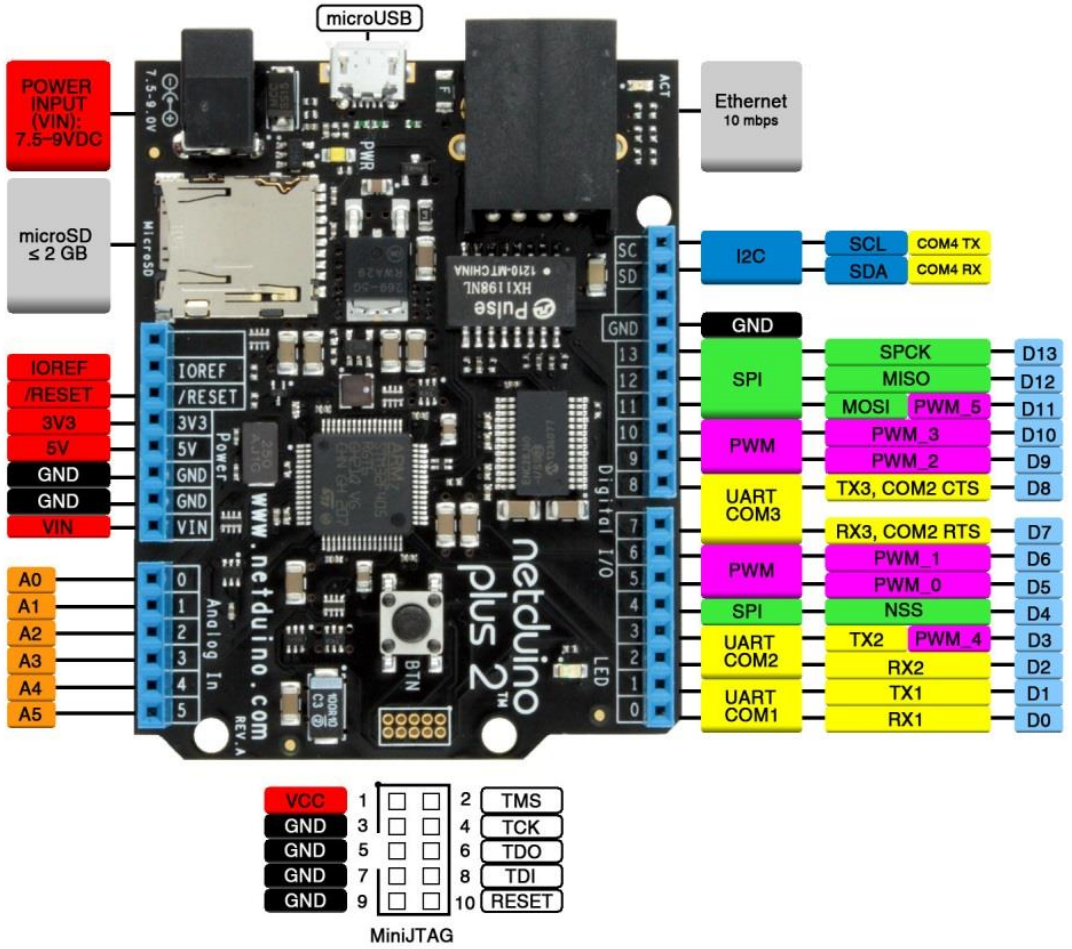
Identification des entrées / sorties de la carte Netduino plus 2

Netduino Plus2

Version: 4.2.1.0

Processor and memory
* STMicro 32-bit microcontroller
* Speed: 168MHz, Cortex-M4
* Code Storage: 384 KB
* RAM: 100+ KB

Power
* if VIN is present, powered by VIN
* if VIN is not present, powered by USB
* maximum GPIO current: 25mA per pin
* maximum mcu output current: 125mA
* output: 5 VDC and 3.3 VDC regulated
* input/output: VIN is unregulated
* total max current
  (incl. GPIO, power rail, microSD, Ethernet):
  - ~500 mA @ 5 V (USB)
  - ~380 mA @ 7.5 V (VIN)
  - ~300 mA @ 9 V (VIN)
  - ~250 mA @12 V (VIN)
* digital i/o are 3.3 V--but 5 V tolerant
* IOREF: 3.3 V (allows shields to adapt to the voltage provided from the board)
* 12-bit ADC: converts an anolog input voltage from 0 V to IOREF to a digital from 0 to 4095
* erase pad has ben removed



gutworks

Version 1.2-Nov 27, 2012

La carte Sensor Shield V2 Tinkerkit

Le module Sensor Shield V2 TinkerKit (fig2) permet de raccorder facilement et sans soudure des capteurs et des actuateurs sur une carte Netduino plus 2.

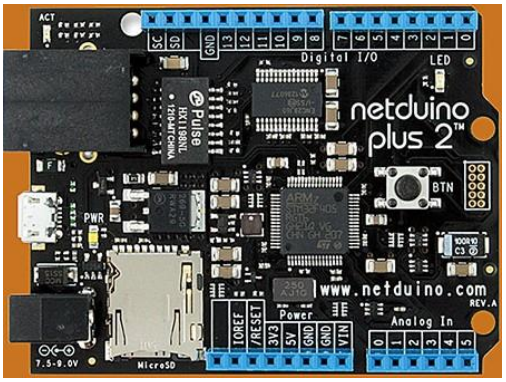


Figure 1 : Netduino plus 2

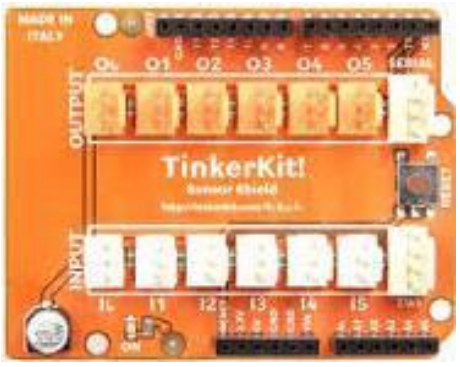


Figure 2 : Sensor Shield V2Tinkerkit

Table de correspondance des dénominations Netduino, Tinkerkit, Visual Studio

Correspondance entre les dénominations des connexions de la carte Netduino plus 2, celles du shield Tinkerkit et celles de Visual Studio (Classes Secrete Labs et NETMF).

Sorties Numériques [*Digital Output*]

Visual Studio	Netduino		Tinkerkit
| | v		Secondary Features	
	SC	SCL/UART 4TX	TWI
	SD	SDA/UART 4 RX	TWI
Pins.GPIO_PIN_D13	13	SPCK	-
Pins.GPIO_PIN_D12	12	MISO	-
Pins.GPIO_PIN_D11	11	PWM / MOSI	O0
Pins.GPIO_PIN_D10	10	PWM	O1
Pins.GPIO_PIN_D9	9	PWM	O2
Pins.GPIO_PIN_D8	8	UART 3 TX / UART 2 CTS	-
Pins.GPIO_PIN_D7	7	UART 3 RX / UART 2 RTS	-
Pins.GPIO_PIN_D6	6	PWM	O3
Pins.GPIO_PIN_D5	5	PWM	O4
Pins.GPIO_PIN_D4	4		-
Pins.GPIO_PIN_D3	3	UART 2 TX / PWM	O5
Pins.GPIO_PIN_D2	2	UART 2 RX	-
Pins.GPIO_PIN_D1	1	UART 1 TX	Serial
Pins.GPIO_PIN_D0	0	UART 1 RX	Serial
Pins.GPIO_ONBOARD_LED	-	LED	-

Entrées numérique [*Digital Input*]

Visual Studio	Netduino	Tinkerkit
Pins.GPIO_PIN_A5	5	I5
Pins.GPIO_PIN_A4	4	I4
Pins.GPIO_PIN_A3	3	I3
Pins.GPIO_PIN_A2	2	I2
Pins.GPIO_PIN_A1	1	I1
Pins.GPIO_PIN_A0	0	I0
Pins.GPIO_ONBOARD_SW1	BP	-

Entrées analogiques [*Analog Input*]

Visual Studio	Netduino	Tinkerkit
Cpu.AnalogChannel.ANALOG_5	5	I5
Cpu.AnalogChannel.ANALOG_4	4	I4
Cpu.AnalogChannel.ANALOG_3	3	I3
Cpu.AnalogChannel.ANALOG_2	2	I2
Cpu.AnalogChannel.ANALOG_1	1	I1
Cpu.AnalogChannel.ANALOG_0	0	I0

Précautions à prendre lors de la configuration des entrées / sorties numériques.

Utiliser **OBLIGATOIREMENT** les classes Secret Labs tel que cela est décrit dans les exemples sous peine de « planter » le firmware de la carte (il devra alors être réinstallé !).

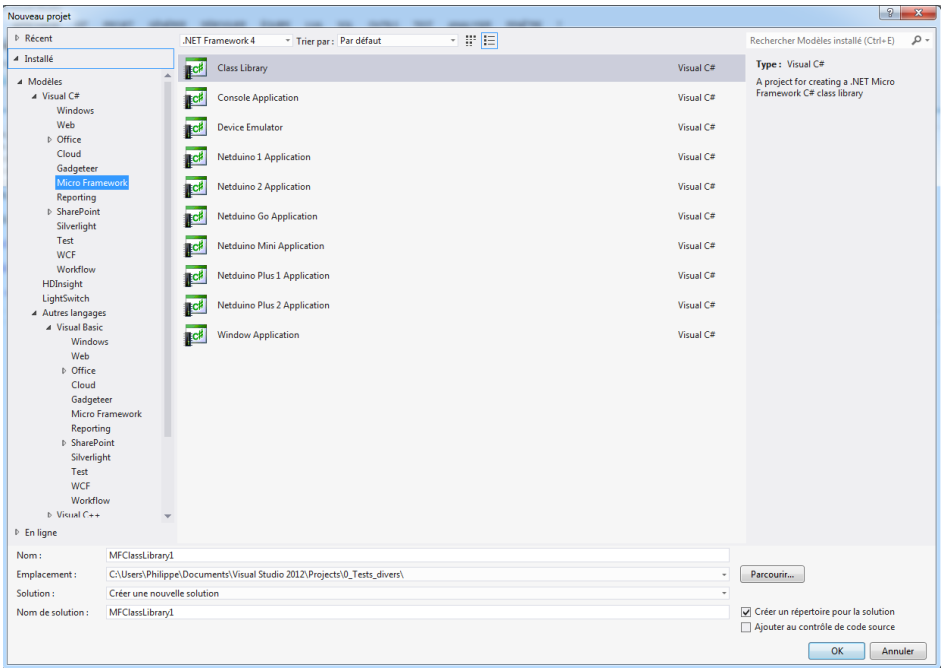
Le « Guide d'installation des logiciels » est téléchargeable ici

Les logiciels

➢ Visual Studio

La véritable force de l'écosystème NETMF est l'association du Micro Framework .NET et de l'IDE Visual Studio. Le Micro Framework .NET est un environnement de développement complet et bien documenté. NETMF contient des fonctionnalités avancées telles que les protocoles de communication, la gestion de fichiers, XML, des interfaces graphiques, le multithreading, etc.

Les cartes Netduino se programment en **C#** (syntaxe C, langage perçu comme une amélioration de Java) ou en Visual Basic avec l'environnement de développement intégré (IDE) **Microsoft Visual Studio** (disponible dans les versions **Community ou professionnel)**. Il est nécessaire d'installer le Micro Framework .NET correspondant à la version de l'IDE et le SDK Secret Labs correspondant à la carte ciblée.



Fonctionnalités particulièrement appréciables dans l'IDE Microsoft Visual Studio

- Environnement de développement intégré complet pour créer des **applications Web**, **Windows Desktop et cross-plateforme** iOS, **Android** et Windows,
- **Coloration** syntaxique,
- **Autocompletion** (Intellisense),
- **Template** de code,
- **Debugger in situ** (exécution du programme en pas à pas dans la carte avec retour de la valeur des variables dans l'IDE)

- **NotPad++ (ou autre)**

Notepad++ est un éditeur de source avec mise en relief de la syntaxe et mise en forme de cette dernière.
Ce logiciel vous permet également de colorer les mots définis par l'utilisateur. Vous pouvez ainsi imprimer votre code source en couleur. De plus, Notepad++ possède une fonction multi-vues qui permet à l'utilisateur d'éditer différents documents à la fois et même d'éditer le même document en synchronisant 2 vues différentes.
Il supporte entièrement le glisser-déposer: vous pouvez déposer le fichier pour l'ouvrir mais également glisser et visionner un document d'une vue à une autre. Ce logiciel fonctionne aussi vite que le bloc-note fourni par MS Windows.

Utilisé ici pour l'édition des sites web installés sur Wampserver.

- **WampServer**

WampServer est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

Les sites web développés dans ce document ont été testés sous **WampServeur (64bits et PHP 5.5) 2.5**

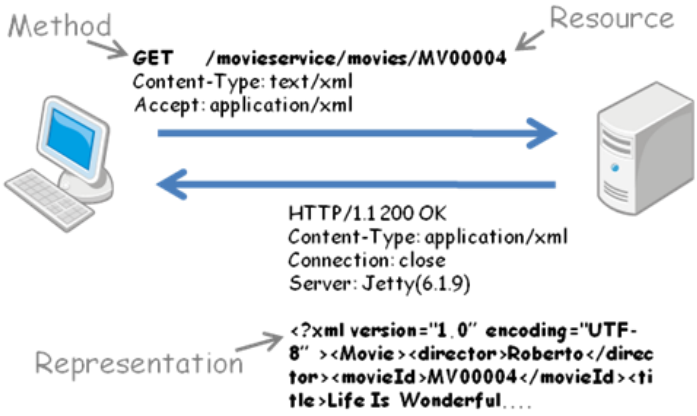
- **REST Console**

Une extension du navigateur Chrome pour tester ses développements REST.

Utilisé ici pour tester les applications serveur installées dans la carte Netduino.

L'architecture REST ((Representational State Transfer)

REST (Representational State Transfer) ou RESTful est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service).



Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière.

L'architecture REST utilise les spécifications originelles du **protocole HTTP**, plutôt que de réinventer une surcouche (comme le font SOAP ou XML-RPC par exemple).

1. **Règle n°1 : l'URI comme identifiant des ressources**
2. **Règle n°2 : les verbes HTTP comme identifiant des opérations**
3. **Règle n°3 : les réponses HTTP comme représentation des ressources**
4. **Règle n°4 : les liens comme relation entre ressources**
5. **Règle n°5 : un paramètre comme jeton d'authentification**

Pour en savoir plus : **Le blog PHP de Nicolas Hachet**

HTTP : Les scénarios Client / Serveur

*« L'**HyperText Transfer Protocol**, plus connu sous l'**abréviation HTTP** — littéralement « protocole de transfert **hypertexte** » — est un protocole de communication client-serveur développé pour le World Wide Web. » Wikipédia*

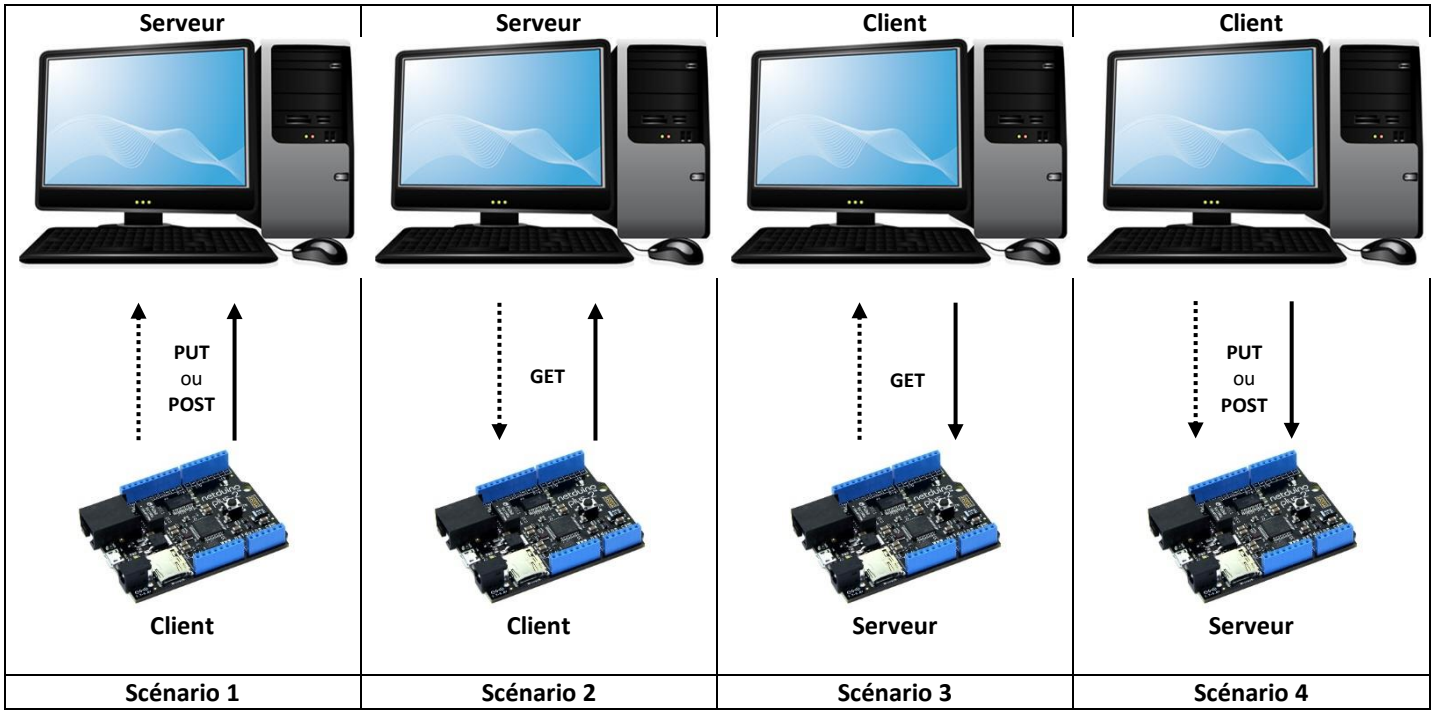
- **Requêtes HTTP**
  - PUT : Utilisée pour écrire dans une ressource
  - GET : Utilisée pour lire une ressource
  - POST : Utilisée pour créer une ressource

- **Scénarios**
  Lorsque la carte Netduino est connectée à un réseau TCP/IP, elle peut communiquer avec une machine (PC, etc) en tant que :
  1 - Client envoyant des données à un serveur.
  2 - Client recevant des données d'un serveur.
  3 - Serveur fournissant des données à un client.
  4 - Serveur acceptant des données d'un client.

Direction de la requête

Direction des données

Ces différents scénarios sont illustrés ci-dessous.



Scénario 1	Scénario 2	Scénario 3	Scénario 4

Remarque : Une machine cliente envoie des requêtes au serveur.

- **Applications**
  - Surveillance à distance (Schémas 1 et 3)
  Dans ce type d'application la carte Netduino **produit** des données (les valeurs des grandeurs physiques mesurées par des capteurs).

  - Commande à distance (Schémas 2 et 4)
  Dans ce type d'application la carte Netduino **consomme** des données.

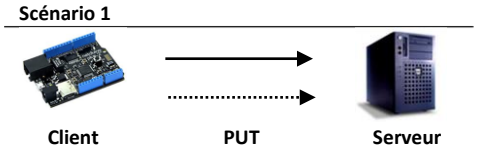
Le **sens de la requête** précise si la machine est un **client** ou un **serveur.** Le **sens des données** précise si la machine est un **producteur** ou un **consommateur.**

Tableau récapitulatif des exemples présentés dans ce document

EXEMPLE	Netduino	Requête programmée	Nom du répertoire projet Visual Studio 2012 Express	Nom du répertoire du site Web à installer sur un serveur Apache	Commentaires	page
Client_1a	**Client**	**PUT**	NetduinoPutClientLocal	ApachePUTGET	La carte Netduino plus 2 transmet périodiquement une donnée à un serveur Apache. Celui-ci la présente dans une page HTML.	
Client_1b			NetduinoPutClientLocalV2	ApachePUTGETV2	La carte Netduino plus 2 transmet plusieurs données à un serveur Apache. Celui-ci les présente dans une page HTML (+ CSS).	
Client_1c			NetduinoPutClientLocalAjax	ApachePUTGETAjax	La carte Netduino plus 2 transmet périodiquement une donnée à un serveur Apache. Celui-ci la présente dans une page HTML (+CSS3). Le navigateur charge périodiquement la page avec des requêtes asynchrones (Ajax).	
Client_2		**GET**	NetduinoGetClientLocal	ApacheGETBOOL	La carte Netduino plus 2 charge une valeur binaire, disponible sur un serveur Apache, et fait évoluer son programme en fonction de son état logique.	
Serveur_1	**Serveur**	-	HelloWebHtmlLocal		La carte Netduino plus 2 renvoie une page HTML en réponse à une requête GET. (Emise par un navigateur ou l'application REST Console)	
Serveur_2		-	VoltageMonitor		La carte Netduino plus 2 renvoie une donnée en réponse à une requête PUT. (Emise par l'application REST Console)	
Serveur_2a		-	VoltageMonitor2		La carte Netduino plus 2 intègre une donnée à une page Web et la transmet en réponse à requête GET. (Emise par un navigateur ou l'application REST Console)	
Serveur_3		-	LedController		La carte Netduino commande une led en fonction des requêtes PUT qu'elle reçoit. (Requêtes émises par l'application REST Console)	
Serveur_3a		-	LedControllerHTML		En réponse à une requête GET émise par un navigateur, la carte Netduino transmet une page web contenant le code javascript de deux requêtes PUT associées à des boutons. Il est alors possible de commander la led avec une souris.	
Serveur_4	**Client et Serveur**		MultitreadingNetduino		Multithreading. Commande de la fréquence de clignotement d'une Led.	
Serveur_4a			MultitreadingNetduinoV2		Multithreading. Commande de la fréquence de clignotement d'une Led. Une partie des pages du site est déporté sur un NAS.	

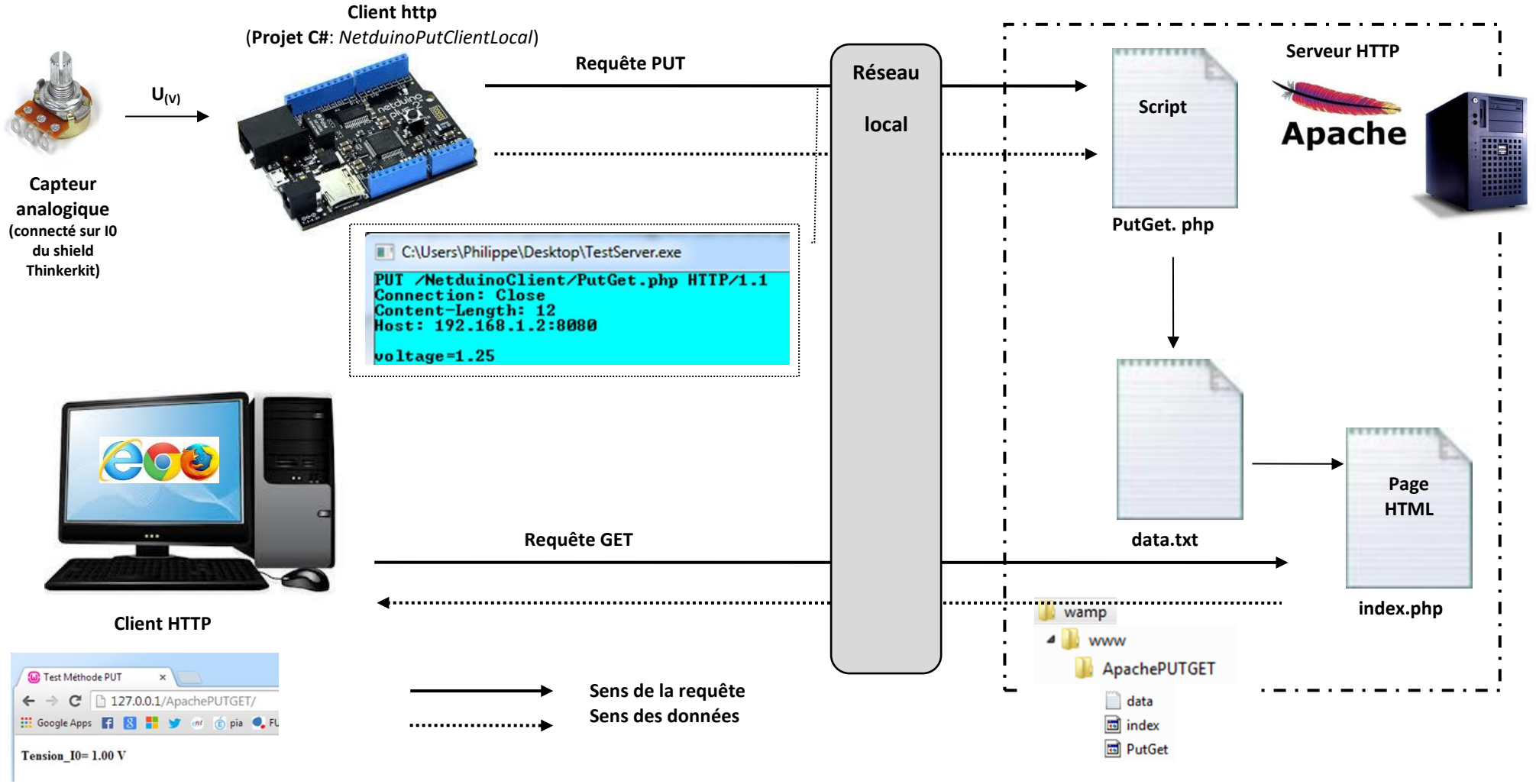
1. Netduino plus 2 est un client http sur un réseau local

1.1. Exemples d'applications liées à la surveillance à distance (scénario 1)



1.1.1. Surveiller la valeur d'une grandeur physique

A intervalles de temps réguliers, la carte Netduino plus 2 lit la tension issue du potentiomètre et la transmet au serveur "Apache" dans une requête PUT. Le script php *PutGet.php* sauvegarde cette valeur dans un fichier texte. Elle est alors disponible et peut être intégrée dans une page HTML.



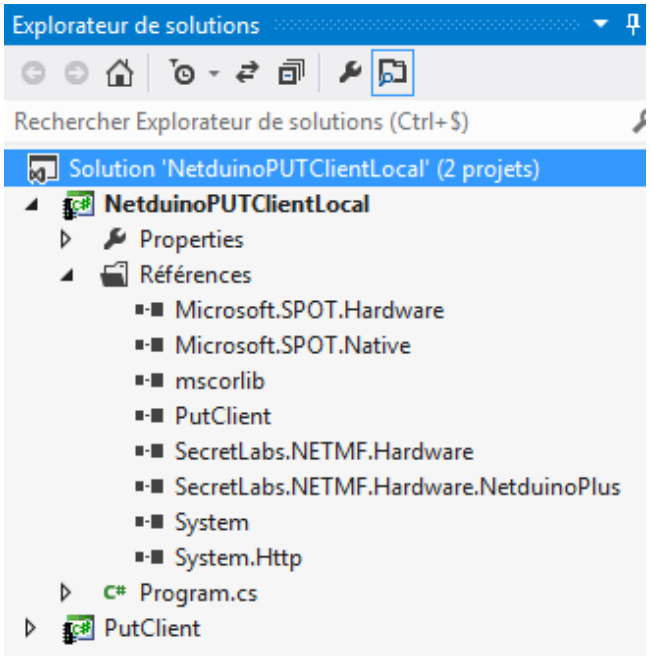
Code C# de l'exemple Client_1a

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using MFClassLibraryPut;


namespace NetduinoPUTClientLocal
{
   public class Program
   {
      public static void Main()
      { // Documentation des classes de NETMF 4.3
        // http://msdn.microsoft.com/en-us/library/bb417055.aspx

         // Url du serveur Apache destinataire des données
         const string baseUri = "http://192.168.1.2/ApachePUTGET/PutGet.php";

         // Durée entre deux transferts
         const int samplingPeriod = 10000;   // 10 secondes

         // Grandeur(s) physique(s) mesurée(s)
         var I0 = new AnalogInput(Cpu.AnalogChannel.ANALOG_0);

         while (true)
         {
            PutClient.WaitUntilNextPeriod(samplingPeriod);
            int ADCout = I0.ReadRaw() / 4; // Résultat de la mesure sur 10 bits
            double value = (ADCout * 3.3) / 1024; // Mise à l'échelle
            string sample = "Tension_I0=" + value.ToString("f"); // Construction du message
            Debug.Print("message: " + sample + "\n");
            PutClient.Send(baseUri, sample); // Emission de la requête PUT
         }
      }
   }
}
```

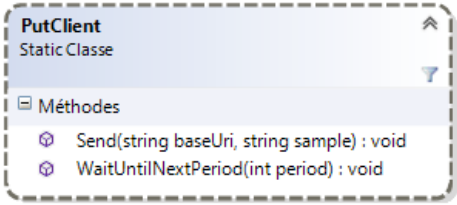
Fenêtre « Sortie » de l'IDE Visual Studio

```
Date: Wed, 13 Aug 2014 14:58:14 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 2
Connection: close
Content-Type: text/html


200
```

Exemple de réponse renvoyée par le serveur Apache



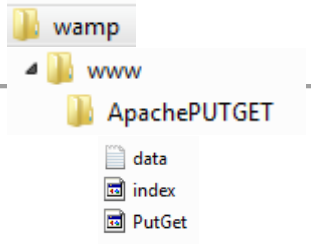

```
message: Tension_I0=3.30
```



Code du script PHP (PutGet.php)

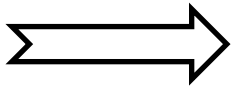
Remarque : Le script gère une méthode PUT ou une méthode GET. La valeur de la donnée « Tension_I0 » est mémorisée dans le fichier data.txt.

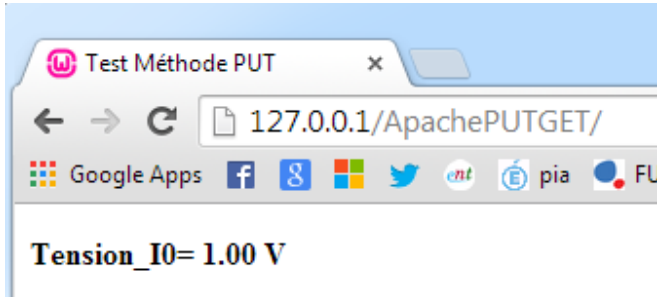
```
?php
    // detection du type de verbe
    if($_SERVER['REQUEST_METHOD'] == 'GET') {
            $Fichier_Sauvegarde = fopen("data.txt","w");
            fputs($Fichier_Sauvegarde,$_GET['Tension_I0']."\n");
            fclose($Fichier_Sauvegarde);
        }
        elseif($_SERVER['REQUEST_METHOD'] == 'PUT') {
    $Fichier_Sauvegarde = fopen("data.txt","w");
            // file_get_contents lit tout un fichier dans une chaîne
            parse_str(file_get_contents("php://input"),$put_vars); // La function parse_str analyse la chaîne de caractères
            // str comme s'il s'agissait des paramètres passés via l'URL.
            // Toutes les variables qu'elle repère sont alors créées, avec leurs valeurs respectives.
            fputs($Fichier_Sauvegarde,$put_vars['Tension_I0']."\n");
            fclose($Fichier_Sauvegarde);
        }
?>
```



Code de la page html (index.php)

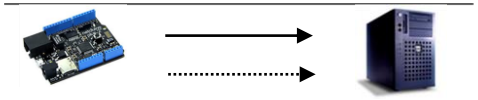
```
<!DOCTYPE html>
<html>
<head>
    <title>Test Méthode PUT</title>
    <meta content="text/html" charset="UTF-8">
</head>

<body>
<?php
    $fichier = fopen("data.txt", "r");
    $tension = fgets($fichier,10);
?>
<p><strong>Tension_I0= <?php echo $tension;?>V</strong></p>
<?php fclose($fichier );?>
</body>
</html>
```





1.1.2. Surveiller la valeur de plusieurs grandeurs physiques

Scénario 1



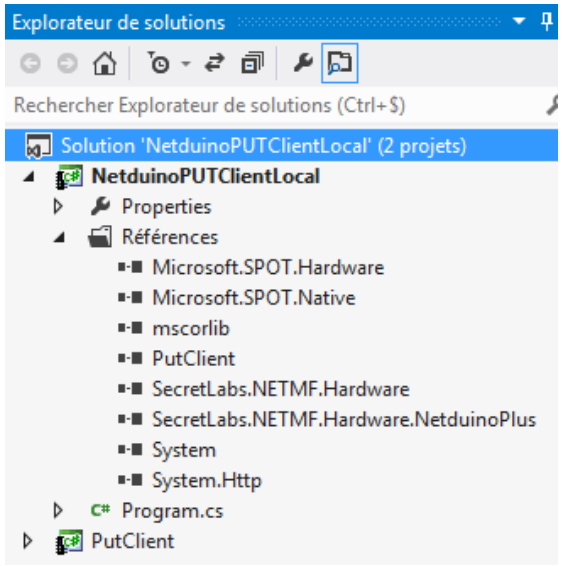
Code C# de l'exemple Client_1b

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using MFClassLibraryPut;

namespace NetduinoPUTClientLocal
{
   public class Program
   {
     public static void Main()
      { // Documentation des classes NETMF 4.3
        // http://msdn.microsoft.com/en-us/library/bb417055.aspx

          // Url du serveur Apache destinataire des données
          const string baseUri = "http://192.168.1.2/ApachePUTGETV2/PutGetV2.php";

          // Durée entre deux transferts
          const int samplingPeriod = 10000;   // 10 secondes

          // Grandeur(s) physique(s) mesurée(s)
          var I0 = new AnalogInput(Cpu.AnalogChannel.ANALOG_0);
          var I1 = new AnalogInput(Cpu.AnalogChannel.ANALOG_1);

          while (true)
          {
             PutClient.WaitUntilNextPeriod(samplingPeriod);
             int ADCout = I0.ReadRaw() / 4; // Résultat sur 10 bits
             double valeur0 = (ADCout * 3.3) / 1023; // Mise à l'échelle
             ADCout = I1.ReadRaw() / 4; // Résultat sur 10 bits
             double valeur1 = (ADCout * 3.3) / 1023; // Mise à l'échelle
             string sample = "TensionI0=" + valeur0.ToString("f") + "&" + "TensionI1=" + valeur1.ToString("f");
             Debug.Print("message: " + sample + "\n");
             PutClient.Send(baseUri, sample); // Emission requête PUT
          }
      }
   }
}
```

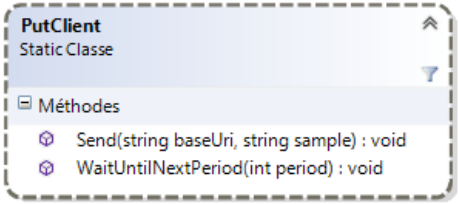
Fenêtre « Sortie » de l'IDE Visual Studio

```
Date: Wed, 13 Aug 2014 14:58:14 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 2
Connection: close
Content-Type: text/html


200
```

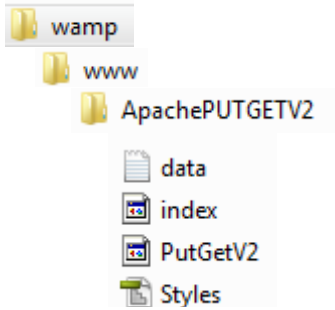
Exemple de réponse renvoyée par le serveur Apache




```
message: TensionI0=1.00&TensionI1=2.16
```



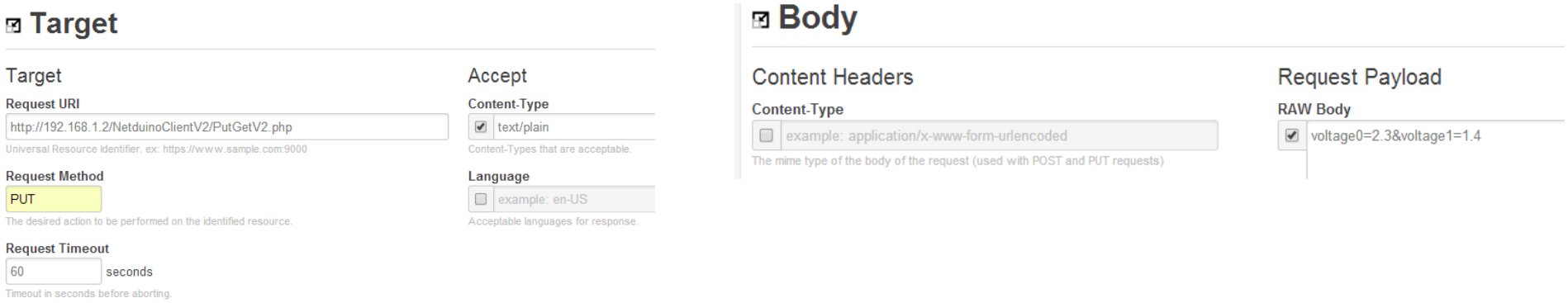
Code du script PHP (PutGetV2.php)




```php
<?php
if($_SERVER['REQUEST_METHOD'] == 'GET')
    { // detection du type de verbe
            $Fichier_Sauvegarde = fopen("data.txt","w");
            foreach($_GET as $cle=>$valeur)
            {
                fputs($Fichier_Sauvegarde,$valeur."\n");
                echo ($_GET[$cle]."\n"; // Visu dans l'extension REST Console de Chrome (Response Preview)
            }
            fclose($Fichier_Sauvegarde);
    }

    elseif($_SERVER['REQUEST_METHOD'] == 'PUT')
    {// detection du type de verbe
       $Fichier_Sauvegarde = fopen("data.txt","w");
            // file_get_contents lit tout un fichier dans une chaîne
            parse_str(file_get_contents("php://input"),$put_vars); // La function parse_str analyse la chaîne de caractères
            // str comme s'il s'agissait des paramètres passés via l'URL.
            // Toutes les variables qu'elle repère sont alors créées, avec leurs valeurs respectives.
            foreach($put_vars as $cle=>$valeur)
            {
                fputs($Fichier_Sauvegarde,$valeur."\n");
                echo $put_vars[$cle]."\n"; // Visu dans l'extension REST Console de Chrome (Response Preview)
            }
            fclose($Fichier_Sauvegarde);
    }
?>
```

Test du script PHP (PutGetV2.php) avec l'extension **« REST Console »** du navigateur Chrome

REST Console version 4.0.2



Code de la page html (index.php)

```
<!DOCTYPE html>
<html>
<head>
    <title>Test Méthode PUT</title>
    <meta content="text/html" charset="UTF-8">
    <link href="styles.css" rel="stylesheet" />
</head>

<body>
<?php
    $fichier = fopen("data.txt", "r");
    $tension0 = fgets($fichier,10);
    $tension1 = fgets($fichier,10);
?>
<div class="page">
<table>
  <tr>
    <td colspan="2"><h1>Carte Netduino  plus 2 + Tinkerkit<h1></td>
  </tr>
  <tr>
    <td>Tension sur I0 =</strong></td>
    <td><strong><?php echo $tension0;?>V<strong></td>
  </tr>
  <tr>
    <td>Tension sur I1 =</td>
    <td><strong><?php echo $tension1;?>V</strong></td>
  </tr>
</table>
</div>
<?php fclose($fichier );?>
</body>
</html>
```

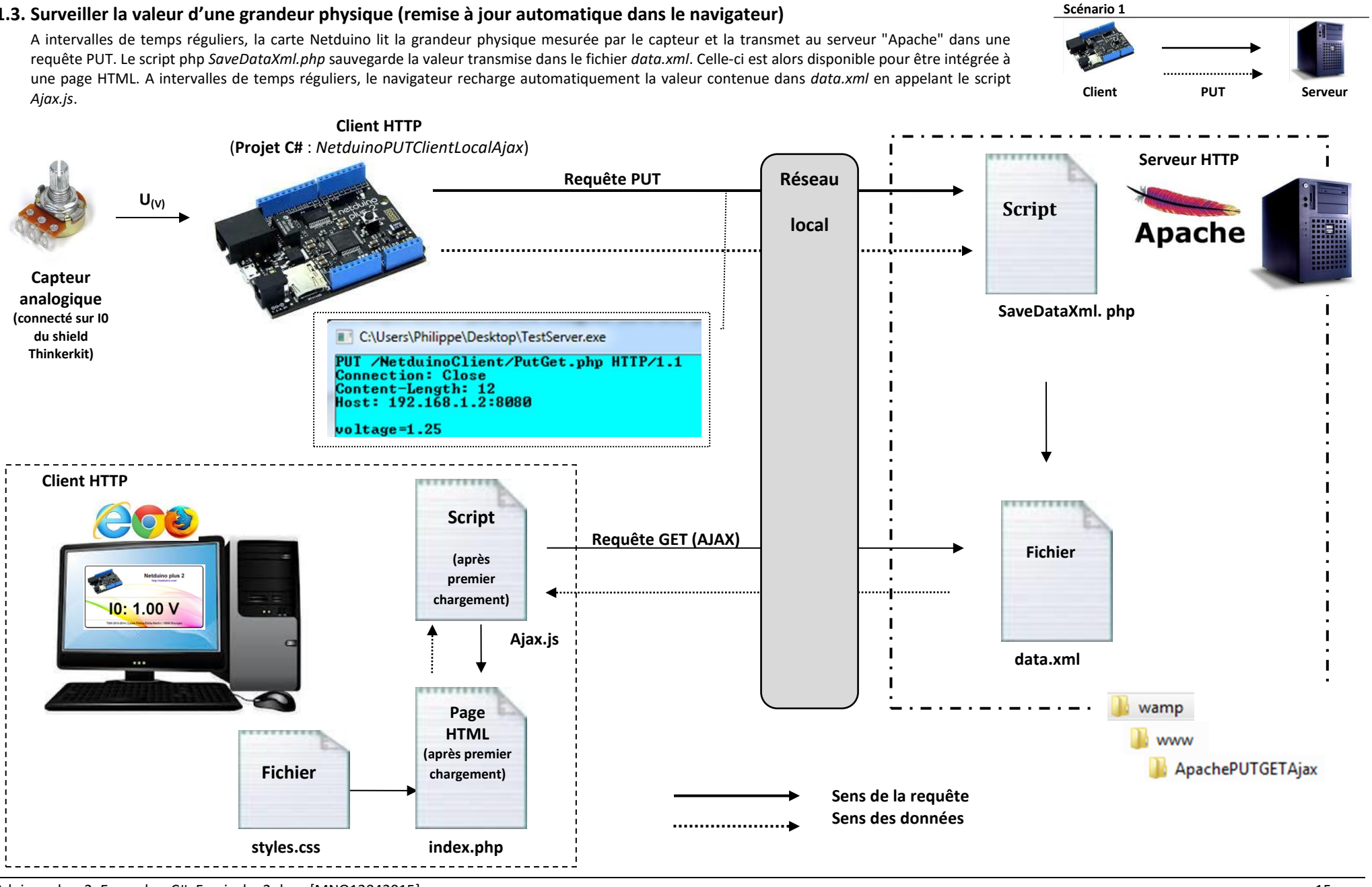
Code de la page de style (Styles.css)

```
body
{
   background-color: #01B0F0;
   font-family: Arial, Helvetica, sans-serif;
   color: white;
   text-align: center;
}

.page
{
   margin: 0 auto; /* Centrage de la page sur l'écran */
   width:300px;
}

h1
{
   font-size: medium;
}

table
{
   background-color: #C710DB;
   border:2px solid #FFFFFF;
}

td
{
   border:1px solid #FFFFFF;
   padding:6px;
}
```

Variante : transmission de plusieurs valeurs au format .csv

1.1.3. Surveiller la valeur d'une grandeur physique (remise à jour automatique dans le navigateur)

A intervalles de temps réguliers, la carte Netduino lit la grandeur physique mesurée par le capteur et la transmet au serveur "Apache" dans une requête PUT. Le script php *SaveDataXml.php* sauvegarde la valeur transmise dans le fichier *data.xml*. Celle-ci est alors disponible pour être intégrée à une page HTML. A intervalles de temps réguliers, le navigateur recharge automatiquement la valeur contenue dans *data.xml* en appelant le script *Ajax.js*.



Code C# de l'exemple Client_1c

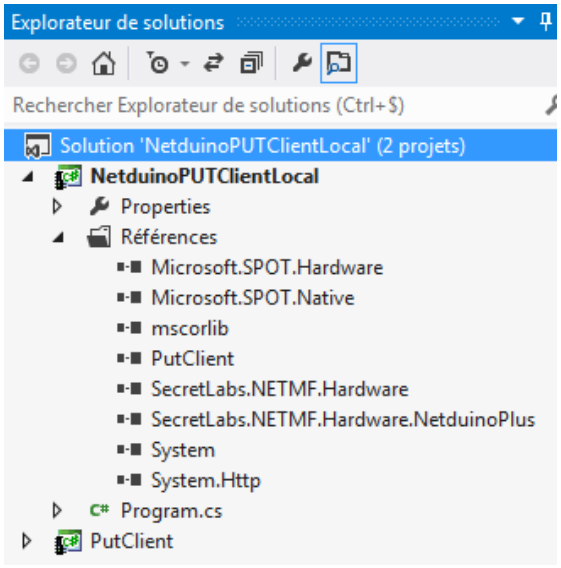
```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using MFClassLibraryPut;


namespace NetduinoPUTClientLocal
{
   public class Program
   {
      public static void Main()
      { // Documentation des classes NETMF 4.3
        // http://msdn.microsoft.com/en-us/library/bb417055.aspx

         // Url du serveur Apache destinataire des données
         const string baseUri = "http://192.168.1.2/ApachePUTGETAjax/mesScripts/SaveDataXml.php";

         // Durée entre deux transferts
         const int samplingPeriod = 1000;   // 1 secondes

         // Grandeur(s) physique(s) mesurée(s)
         var I0 = new AnalogInput(Cpu.AnalogChannel.ANALOG_0);

         while (true)
         {
            PutClient.WaitUntilNextPeriod(samplingPeriod);
            int ADCout = I0.ReadRaw() / 4; // Résultat de la mesure sur 10 bits
            double value = (ADCout * 3.3) / 1023; // Mise à l'échelle
            string sample = "Tension_I0=" + value.ToString("f");
            Debug.Print("message: " + sample + "\n");
            PutClient.Send(baseUri, sample); // Emission requête PUT
         }
      }
   }
}
```

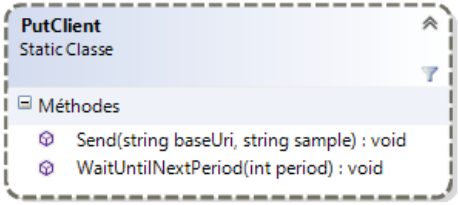
Fenêtre « Sortie » de l'IDE Visual Studio

```
Date: Wed, 13 Aug 2014 14:58:14 GMT
Server: Apache/2.4.9 (Win64) PHP/5.5.12
X-Powered-By: PHP/5.5.12
Content-Length: 2
Connection: close
Content-Type: text/html


200
```

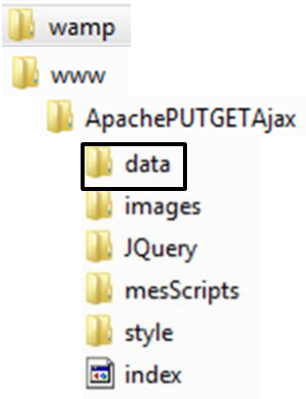
Exemple de réponse renvoyée par le serveur Apache

```
message: Tension_I0=1.00
```





Code du script PHP (SaveDataXml. php)



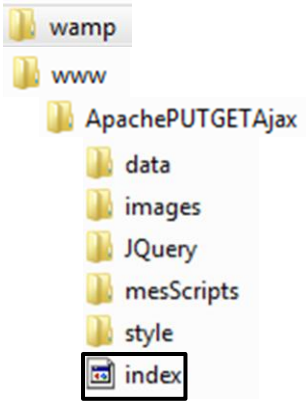

```php
<?php
    if($_SERVER['REQUEST_METHOD'] == 'GET')
    {
            // Lecture du ficher xml
            $xml = simplexml_load_file("../data/data.xml");

            // Modification des éléments
            $xml->donnee->I0=$_GET["Tension_I0"];

            // Ecriture du fichier xml
            $chxml = $xml->asxml("../data/data.xml");
    }
    elseif($_SERVER['REQUEST_METHOD'] == 'PUT')
    {
       // Lecture du ficher xml
            $xml = simplexml_load_file("../data/data.xml");

            // Modification des éléments
            // file_get_contents lit tout un fichier dans une chaîne
            parse_str(file_get_contents("php://input"),$put_vars); // La function parse_str analyse la chaîne de caractères
            // str comme s'il s'agissait des paramètres passés via l'URL.
            // Toutes les variables qu'elle repère sont alors créées, avec leurs valeurs respectives.
            $xml->donnee->I0=$put_vars['Tension_I0'];

            // Ecriture du fichier xml
            $chxml = $xml->asxml("../data/data.xml");
    }
?>
```

Code du fichier XML (data.xml)

```xml
<?xml version="1.0" encoding="utf-8"?>
<netduino>
    <donnee>
            <I0>1.00</I0>
    </donnee>
</netduino>
```

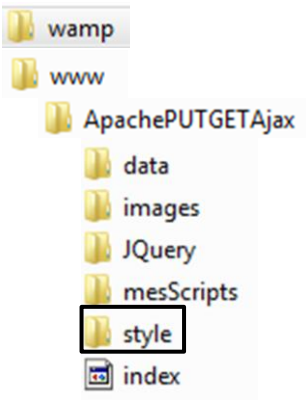
Code de la page html (index.php)

```
<DOCTYPE html>
<html>
<head>
    <title> Netduino plus 2 </title>
    <link rel="stylesheet" href="style/Styles.css">
</head>

<body>
<?php // Ouverture du fichier contenant la donnée
    $xml = simplexml_load_file("data/data.xml");
    $I0 = (float)($xml->donnee[0]->I0);
?>
    <div class="page">
        <div class="header">
            <div class="logo"><img src="images/netduino.jpg" /></div>
            <div class="titre">
                <div class="texte">
                    Netduino plus 2<br>
                    <a href="http://netduino.com/" target="_blank">http://netduino.com/</a>
                </div>
            </div>
        </div>
        <hr />
        <div class="content">
            <span>I0:</span>
            <div id="I0"><?php echo $I0 ?></div>
            <span>V</span>
        </div>
        <hr />
        <div class="footer">TSIN 2013-2014 | Lycée Pierre-Emile-Martin | 18000 Bourges </div>
    </div>
    <script src="JQuery/jquery-1.8.2.min.js"></script>
    <script src="mesScripts/Ajax.js"></script> <!-- Rafraichissement de la page -->
</body>
</html>
```



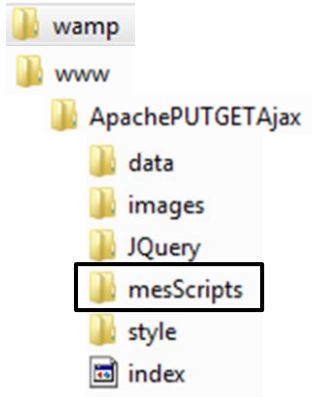
Code de la feuille de style (Styles.css)

```
/* --------------------- page --------------------- */
.page {
        background-image:url("../images/white.jpg");
        margin:0 auto;/* Centrage de la page sur l'écran */
        width:650px;
        border: black solid 3px;
        border-radius:15px;
        box-shadow:1px 2px 4px;
        font-family:"Helvetica Neue", Helvetica, Arial, sans-serif;
        font-weight:bold;
        text-align:center;
        color:black;
}
/* --------------------- Lien --------------------- */
a{
        font-size:15px;
        text-decoration:none;
}

a:hover{
        color:blue;
}

/* --------------------- En-tête --------------------- */
.header{
}

.logo {
        margin-left:5px;
        margin-top:5px;
        margin-bottom:5px;
        margin-right:8px;
        display:inline-block;
        background-color:blue;
}

.titre{
        margin:5px;
        width: 416px;
        height: 155px;
        display:inline-block;
        vertical-align:top;
}

.texte{
        padding-top:30px;
        font-size:30px;
}
/* --------------------- Contenu --------------------- */
.content{
        font-size:90px;
}

#I0{
        display:inline;
}

/* --------------------- Pied de page ----------------- */
.footer{
        min-height:30px;
        font-size:15px;
}
```



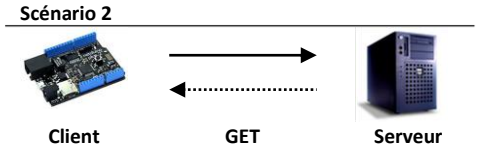
Code du script JS (Ajax.js)

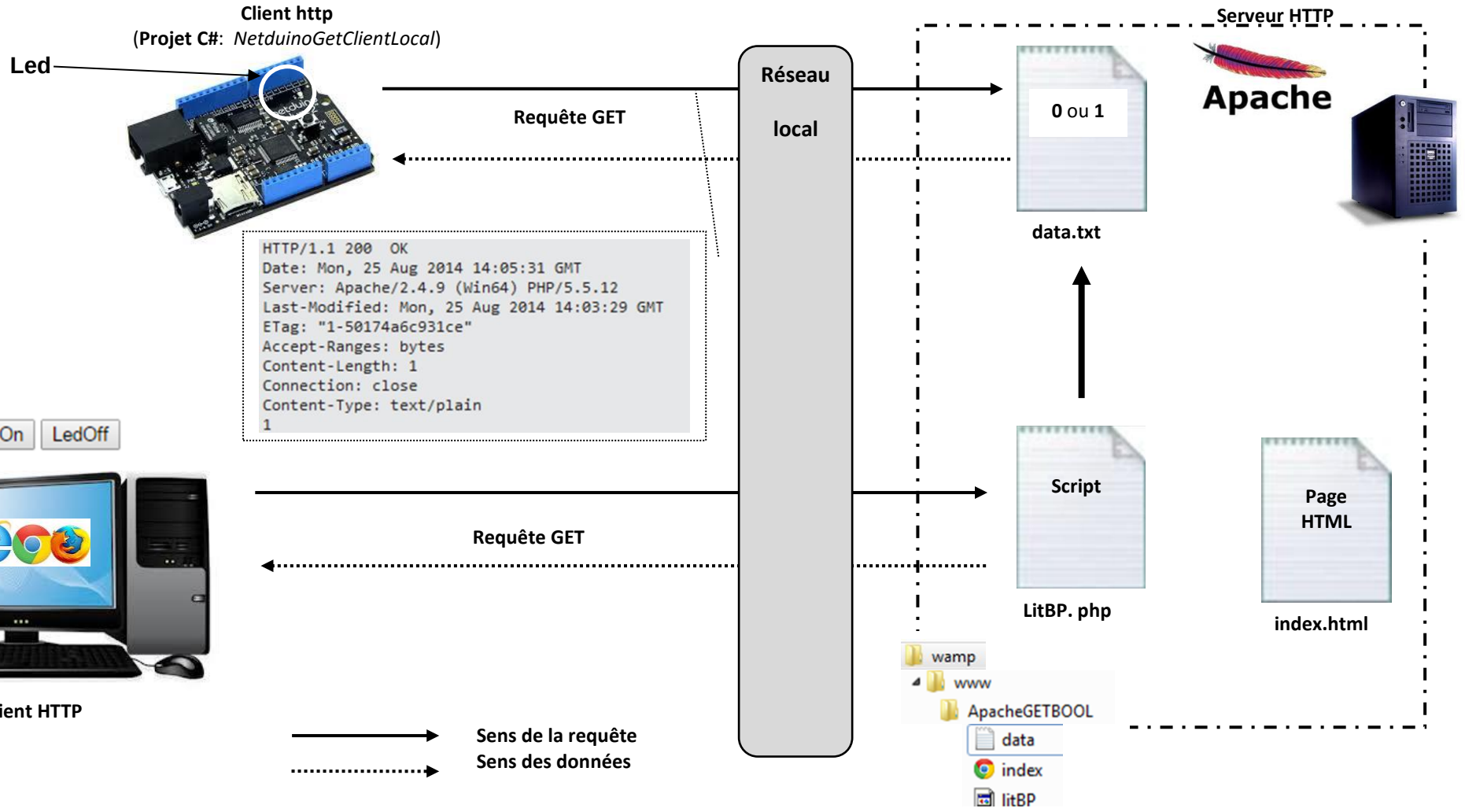
```
$(document).ready(function(){
    var FREQ = 1000;            // Fréquence de rafraîchissement(en ms).
// ----------------------------------------------------------------------------
    function startAJAXcalls(){ /* setTimeout(mafonction, duree); mafonction est */
            setTimeout(function(){ /* appelée quand duree  est dépassé  */
                    getXMLData();
                    startAJAXcalls();
            },
                    FREQ
            );
    }
// ----------------------------------------------------------------------------
    getXMLData();               // Appel de la fonction getXMLData au chargement de la page
    startAJAXcalls();  // Démarrage de la fonction startAJAXcalls

// Lecture des données dans le fichier ../data/data.xml et affichage de la donnee
// dans la page index.php
// ----------------------------------------------------------------------------
    function getXMLData(){
            $.ajax({
                    url:"data/data.xml",
                    cache:false,
                    dataType:"xml",
                    success:function(xml){
                            $("#I0").empty(); /*Vide l'élément identifié par I0 avant la mise à jour */
                            $(xml).find("donnee").each(function(){
                                    var $I0 =  $(this).find("I0").text();
                                    $("#I0").html($I0);
                            });
                    }
            });
    }
// ----------------------------------------------------------------------------
});
```



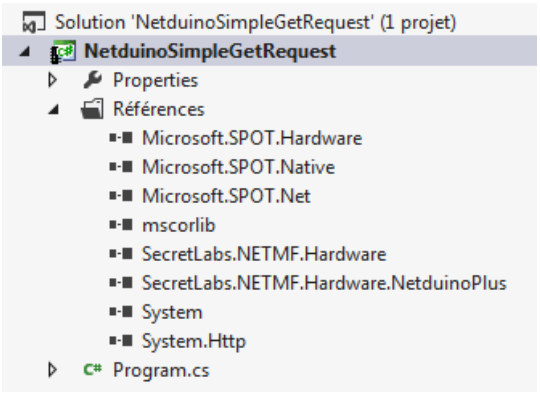
1.2. Exemple d'application liée à la commande à distance (scénario 2)

La carte Netduino plus 2 envoie périodiquement une requête GET au serveur Apache. Celui-ci renvoi le contenu du fichier *data.txt*. Si la donnée contenue dans ce fichier est égale à « 1 », la LED de la carte Netduino s'éclaire sinon elle s'éteint. Le fichier *data.txt* est modifié par le script LitBP.php en fonction des données transmises par le formulaire (les boutons LedOn, LedOff) de la page *index.html*.





Code C# de l'exemple Client_2




```
using System;
using System.Threading;
using System.IO;
using System.Net;
using System.Text;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

namespace NetduinoSimpleGetRequest
{
   public class Program
   {
      // Ressources http://msdn.microsoft.com/en-us/library/bb417055.aspx

      public static void Main()
      {
         var requestUri = "http://192.168.1.2/ApacheGETBOOL/data.txt";
         var Led = new OutputPort(Pins.ONBOARD_LED,false);

         int ressourceReseau = 0;

         while (true)
         {
            ressourceReseau = Receive(requestUri);

            if (ressourceReseau == 1)
            {
               Led.Write(true);
            }
            else
            {
               Led.Write(false);
            }
            Thread.Sleep(500); // Lecture de la resource sur le serveur toutes les 0,5s
         }
      }
```

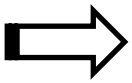
```
      private static int Receive(string requestUri)
      {
         int value = 0;

         using (var request = (HttpWebRequest)WebRequest.Create(requestUri))
         {
            request.Method = "GET";

            // headers
            request.KeepAlive = false; //false si pb lors d'un transfert sur
                                       //un réseau local (serveur Apache)!

            // send request and receive response
            using (var response = (HttpWebResponse)request.GetResponse())
            {
               // consume response
               value = HandleResponse(response);
            }
         }
         return value;
      }
```

```
        public static int HandleResponse(HttpWebResponse response)
        {
            // response status line
            Debug.Print("HTTP/" + response.ProtocolVersion + " " +
                        response.StatusCode + " " +
                        response.StatusDescription);

            // response headers
            string[] headers = response.Headers.AllKeys;
            foreach (string name in headers)
            {
                Debug.Print(name + ": " + response.Headers[name]);
            }

            // response body
            var buffer = new byte[(int)response.ContentLength];
            Stream stream = response.GetResponseStream();
            int toRead = buffer.Length;
            while (toRead > 0)
            {
                // already read: buffer.Length - toRead
                int read = stream.Read(buffer, buffer.Length - toRead, toRead);
                toRead = toRead - read;
            }
            // Processing the received value
            char[] chars = Encoding.UTF8.GetChars(buffer);
            Debug.Print(new string(chars));
            int result = Convert.ToInt32(new string(chars));
            return result;
        }
    }
}
```

Partie à modifier en fonction du contenu de la ressource

Code de la page html (index.html)

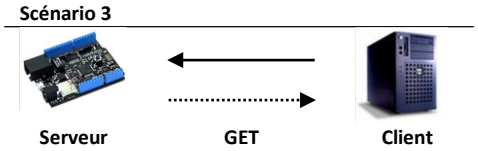
```
<!DOCTYPE html>
<html>
<head>
        <title>Test Méthode GET</title>
        <meta content="text/html" charset="UTF-8">
</head>

<body>
<h1>Netduino plus 2</h1>
<h3>Commande de la Led</h3>
<form action="litBP.php" method="GET">
        <input type="submit" name="Envbool" value="LedOn" />
        <input type="submit" name="Envbool" value="LedOff" />
</form>
</body>
</html>
```



Code du script PHP (litBP.php)

```
<?php
        $Fichier_Sauvegarde = fopen("data.txt","w");

        if ($_GET['Envbool']== "LedOn")
        {
                fputs($Fichier_Sauvegarde,"1");
        }
        else
        {
                fputs($Fichier_Sauvegarde,"0");
        }

        fclose($Fichier_Sauvegarde);
        header('Location: index.html');
?>
```

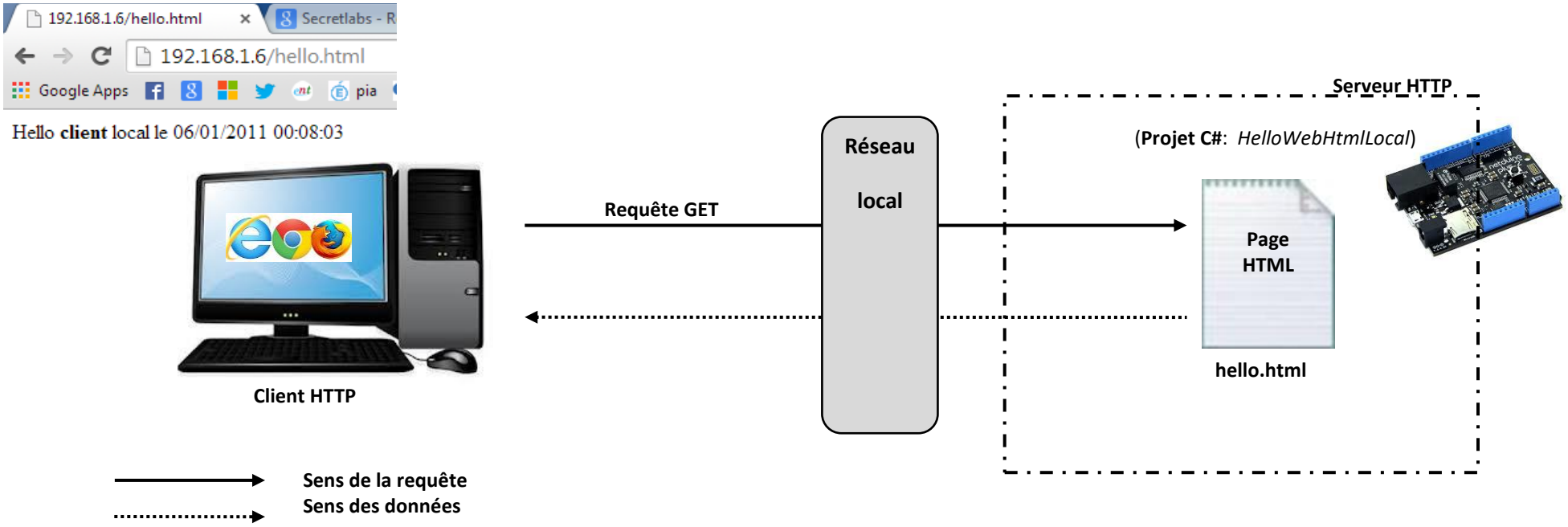
2. Netduino plus 2 est un serveur http sur un réseau local

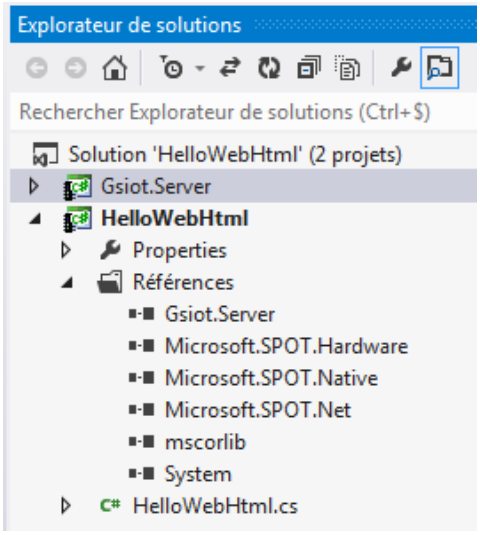


2.1. Exemples d'applications liées à la surveillance à distance (scénario 3)

2.1.1. Affichage d'une page web

La carte Netduino plus 2 renvoie la page ***Hello.html*** après la réception de la requête GET. Toute autre demande renvoie « Erreur 404 ».






```
DHCP enabled: True
MAC address: 5C-86-4A-00-DB-06
Device address: 192.168.1.6
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.6/
```

L'URI affectée à la carte netduino plus 2 par le serveur DHCP est affichée dans le Debugger !

Code C# de l'exemple Serveur_1

```
using System;
using Gsiot.Server;

public class HelloWebHtml
{
   public static void Main()
   {
      var webServer = new HttpServer ❶
      { ❷
         RequestRouting =
         {
            { "GET /hello.html"❸, HandleGetHelloHtml },❹
            { "GET /*", context =>
                        { context.SetResponse("Erreur 404", "text/plain"); }❺
            }
         }
      };
      webServer.Run();❻
   }

   static void HandleGetHelloHtml(RequestHandlerContext context) ❹
   {
      string s =
         "<html>\r\n" +
         "\t<body>\r\n" +
         "\t\tHello <strong>client</strong> local le " +
            DateTime.Now + "\r\n" +
         "\t</body>\r\n" +
         "</html>";
       context.SetResponse(s, "text/html");
   }
}
```

```
DHCP enabled: True
MAC address: 5C-86-4A-00-DB-06
Device address: 192.168.1.6
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.6/
```

❶ Construction du serveur avec un **initialiseur** d'objet.
❷ Par défaut, le serveur écoute sur le port 80. Celui-ci peut être changé avec l'attribut Port = x de l'objet server.
❸ Déclaration de la collection des requêtes et de leurs réponses ❹ dans l'attribut *RequestRouting*.
❺ Comme c'est le cas ici, la réponse associée à la requête peut être décrite par une expression lambda.
❻ Activation du serveur. Il attend les requêtes.

Remarques

- Les caractères \r\n (à la ligne) et \t (tabulation) sont optionnels. Il effectue seulement la mise en forme du code lors de son affichage dans le navigateur.

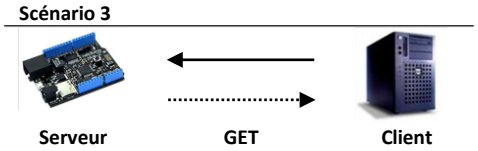
- Les sources du serveur sont dans le projet Gsiot.Server.

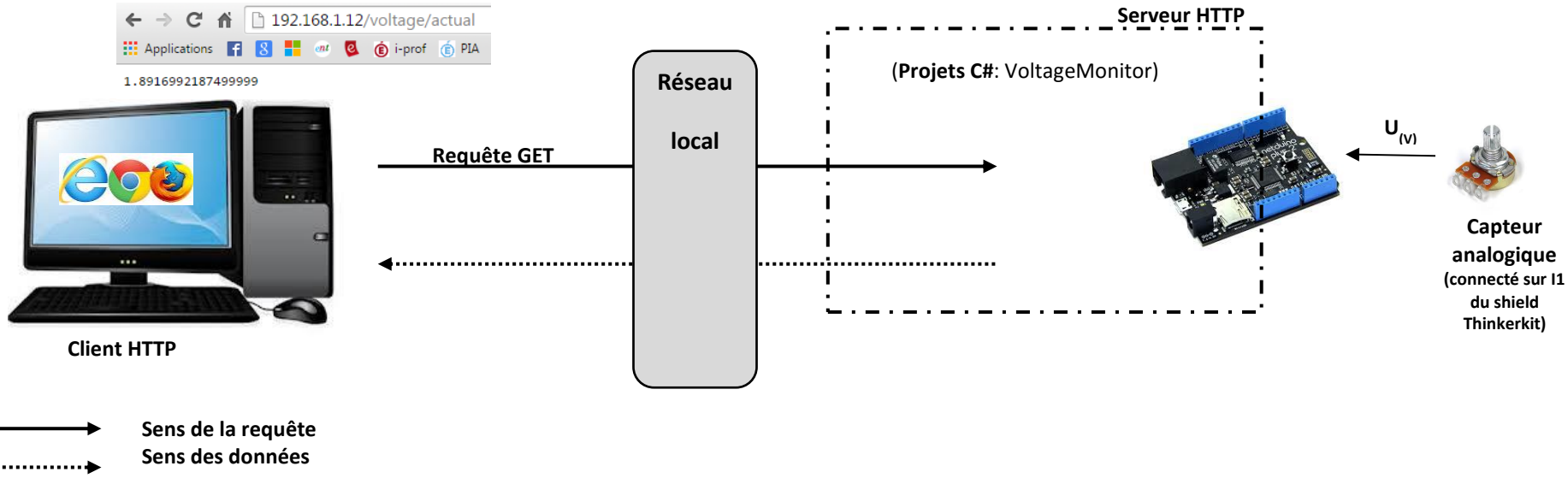
Amélioration

- Mémorisation de la page web sur une carte SD.

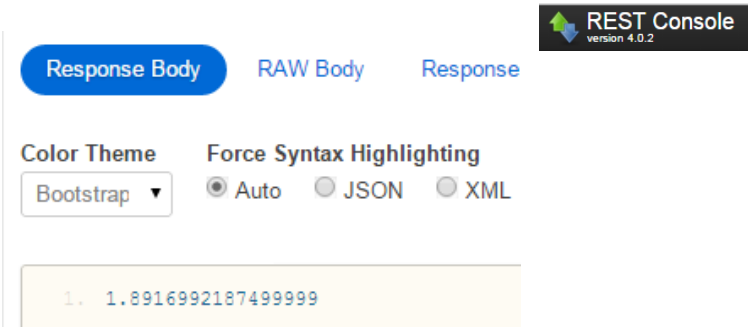
2.1.2. Affichage d'une grandeur physique intégrée à une page web

A - Affichage de la tension issue d'un potentiomètre dans un navigateur





Test avec l'extension **« REST Console »** du navigateur **Chrome**

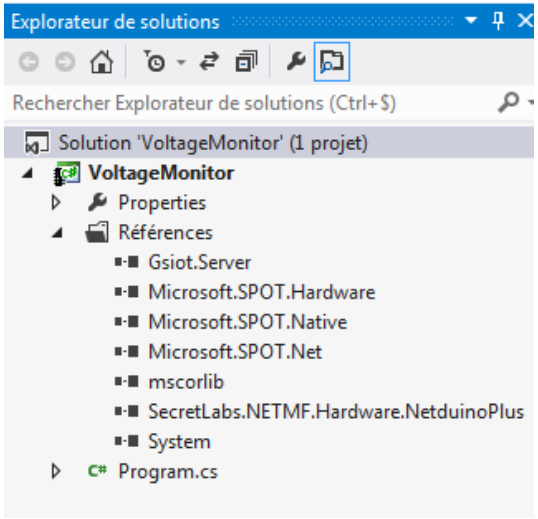


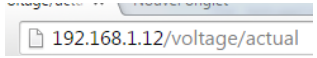
Code C# de l'exemple Serveur_2

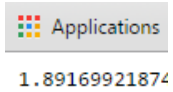
```
using Gsiot.Server;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

namespace VoltageMonitor
{
    public class Program
    {
        public static void Main()
        {
            // ground and power for the potentiometer

            var voltageSensor = new AnalogSensor ❶ // Initialiseur d'objets
            {
                InputPin = Cpu.AnalogChannel.ANALOG_1,
                MinValue = 0.0,
                MaxValue = 3.3
            };

            var webServer = new HttpServer ❷
            {
                RequestRouting =
                {
                    {
                        "GET /voltage/actual",❸
                         new MeasuredVariable
                         {
                          FromSensor = voltageSensor.HandleGet
                         }.HandleRequest ❹
                     },

                     {
                         "GET /hello", context => {context.SetResponse("Hello","text/plain"); }
                     }
                }
            };

            webServer.Run();❺
        }
    }
}
```

```
DHCP enabled: True
MAC address: 00-04-A3-00-00-00
Device address: 192.168.1.12
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.12/
```

L'adresse IP affectée à la carte netduino plus 2 par le serveur DHCP est affichée dans la fenêtre « Sortie » de Visual Studio !







❶ Déclaration et initialisation de l'entrée analogique à mesurer.

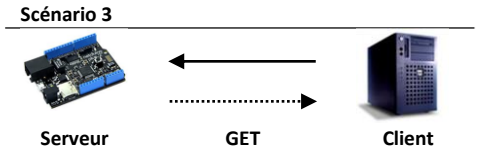
❷ Construction du serveur avec un initialiseur d'objet.

❸ Déclaration de la collection des requêtes et de leurs réponses ❹ dans l'attribut *RequestRouting*.

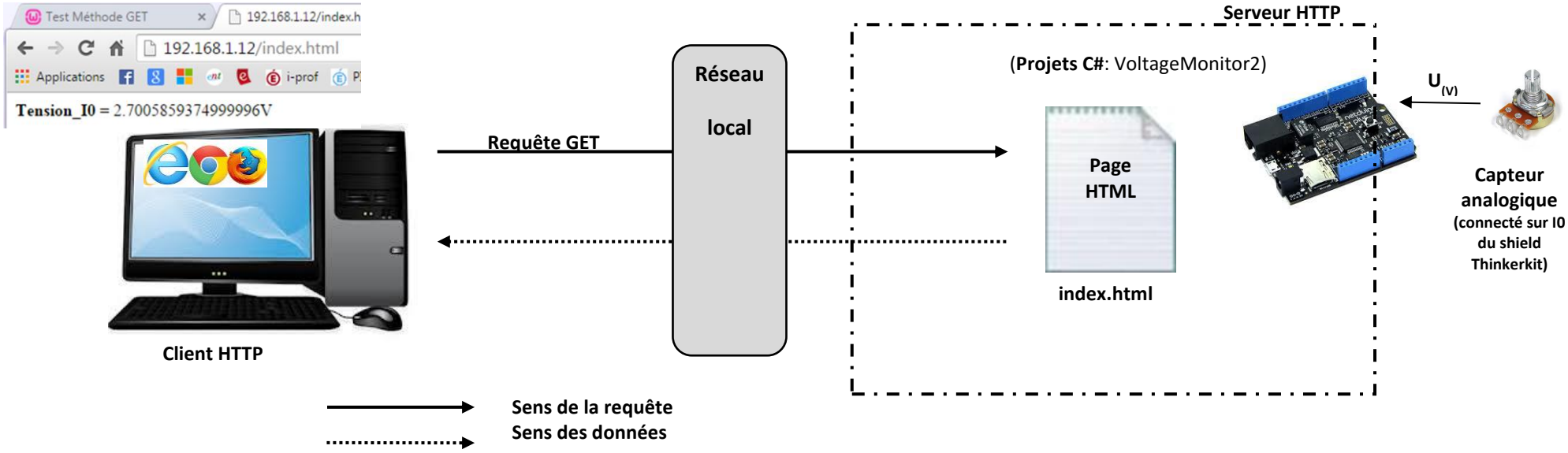
❹ La méthode *HandleGet* lit la valeur de la tension présente sur l'entrée analogique ANALOG_1 et la passe à l'attribut *FromSensor* . La méthode HandleRequest de l'objet `MeasuredVariable` gère la réponse à la requête `GET /voltage/actual`.

❺ Activation du serveur.

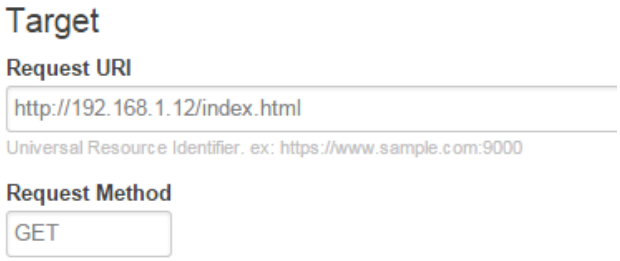
B – Modification de l'exemple Serveur_2



La valeur de la grandeur physique mesurée par la carte Netduino plus 2 est intégrée à une page web avant d'être transmise au navigateur.



Test avec l'extension **« REST Console »** du navigateur **Chrome**



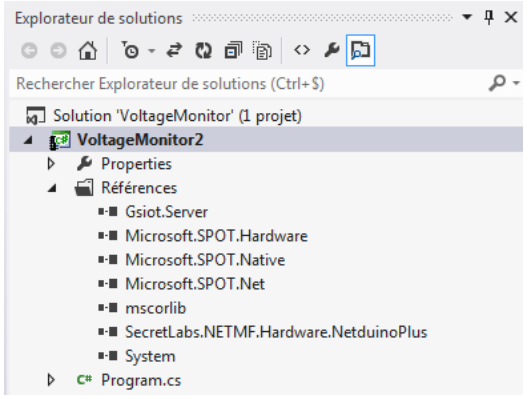
Code C# de l'exemple Serveur_2a

```
using Gsiot.Server;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

namespace VoltageMonitor
{
    public class Program
    {
        public static void Main()
        {
            // ground and power for the potentiometer

            var voltageSensor = new AnalogSensor  // Initialiseur d'objets
            {
                InputPin = Cpu.AnalogChannel.ANALOG_1,
                MinValue = 0.0,
                MaxValue = 3.3
            };

            var webServer = new HttpServer
            {
                RequestRouting =
                { // Renvoie la valeur de la tension
                   {
                       "GET /voltage/actual",
                        new MeasuredVariable
                        {
                         FromSensor = voltageSensor.HandleGet
                        }.HandleRequest
                   },
                   { // Renvoie la valeur de la tension inclue dans une page web
                        "GET /index.html", HandleRequestVoltage
                   },
                   {// Renvoie Hello
                        "GET /hello", context => { context.SetResponse("Hello","text/plain"); }
                   },
                   {// Renvoie Erreur 404
                        "GET /*", context => { context.SetResponse("Erreur 404","text/plain"); }
                   },
                }
            };

            webServer.Run();
        }
```

```
DHCP enabled: True
MAC address: 00-04-A3-00-00-00
Device address: 192.168.1.12
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.12/
```

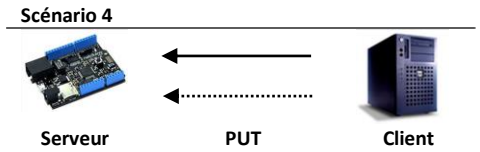
L'adresse IP affectée à la carte netduino plus 2 par le serveur DHCP est affichée dans la fenêtre « Sortie » de Visual Studio !

```
        static void HandleRequestVoltage(RequestHandlerContext context)
        {
            var voltageSensor = new AnalogSensor  // Initialiseur d'objets
            {
                InputPin = Cpu.AnalogChannel.ANALOG_0,
                MinValue = 0.0,
                MaxValue = 3.3
            };

            string s =
                "<html>\r\n" +
                "\t<body>\r\n" +
                "\t\t <strong>Tension = </strong>" +
                   voltageSensor.HandleGet().ToString() + "V" + "\r\n" +
                "\t</body>\r\n" +
                "</html>";
            context.SetResponse(s, "text/html");
        }
    }
}
```



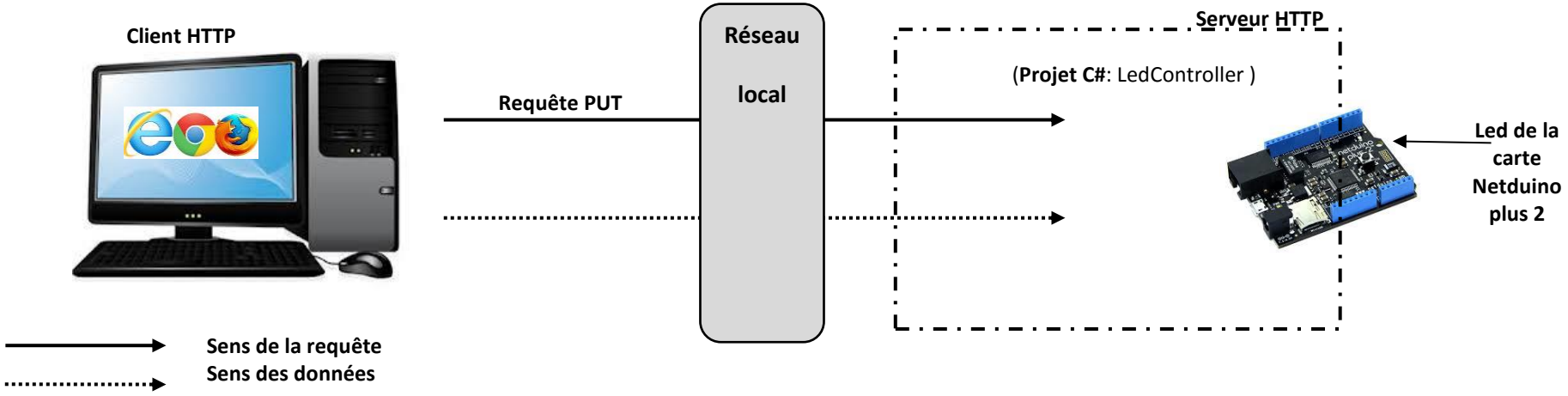
Code commenté : Voir Serveur2 et Serveur 3

2.2. Exemples d'application liées à la commande à distance (scénario 4)

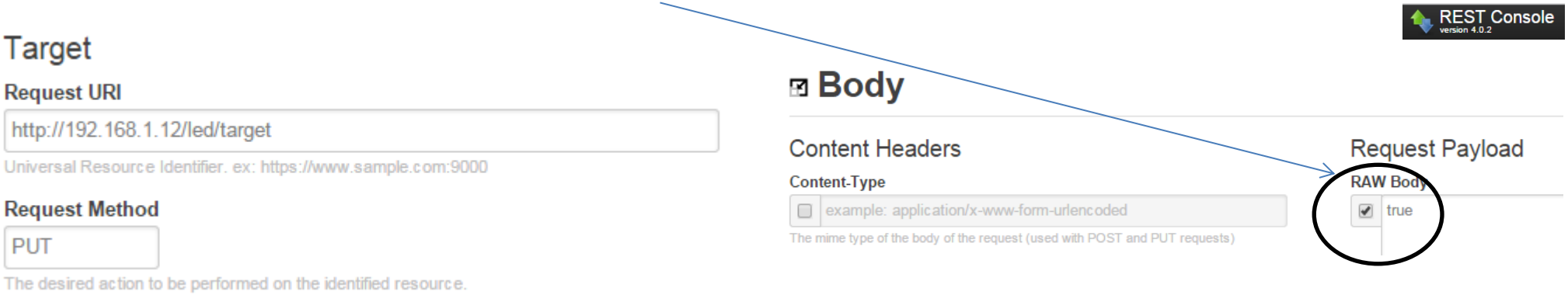


2.2.1 Commande de la Led de la carte Netduino (V 1)

L'envoie de la requête ci-dessous avec la **console REST** entraine l 'éclairage de la Led de la carte Netduino plus 2.



Test avec l'extension **« REST Console »** du navigateur **Chrome (Eclairage de la Led)**

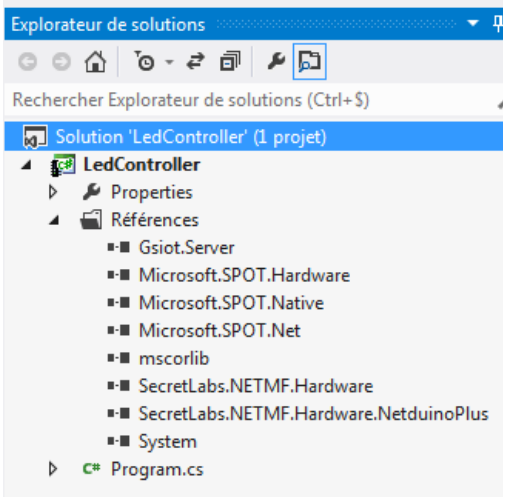


Code C# de l'exemple Serveur3

```
using Gsiot.Server;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

namespace LedController
{
    public class Program
    {
        public static void Main()
        {
            var ledActuator = new DigitalActuator ❶
            {
                OutputPin = Pins.ONBOARD_LED
            };

            var webServer = new HttpServer ❷
            {

                RequestRouting =
            {
                {
                    "PUT /led/target", ❸
                    new ManipulatedVariable ❹
                    {
                        FromHttpRequest =
                            CSharpRepresentation.TryDeserializeBool,
                        ToActuator = ledActuator.HandlePut
                    }.HandleRequest
                }
            }
            };

            webServer.Run();❺
        }

    }

}
```

```
DHCP enabled: True
MAC address: 00-04-A3-00-00-00
Device address: 192.168.1.12
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.12/
```

L'adresse IP affectée à la carte netduino plus 2 par le serveur DHCP est affichée dans la fenêtre « Sortie » de Visual Studio !



❶ Déclaration et initialisation de la sortie logique à commander.

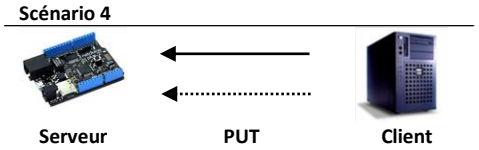
❷ Construction du serveur avec un initialiseur d'objet.

❸ Déclaration de la requête et de sa réponse ❹ dans l'attribut *RequestRouting*.

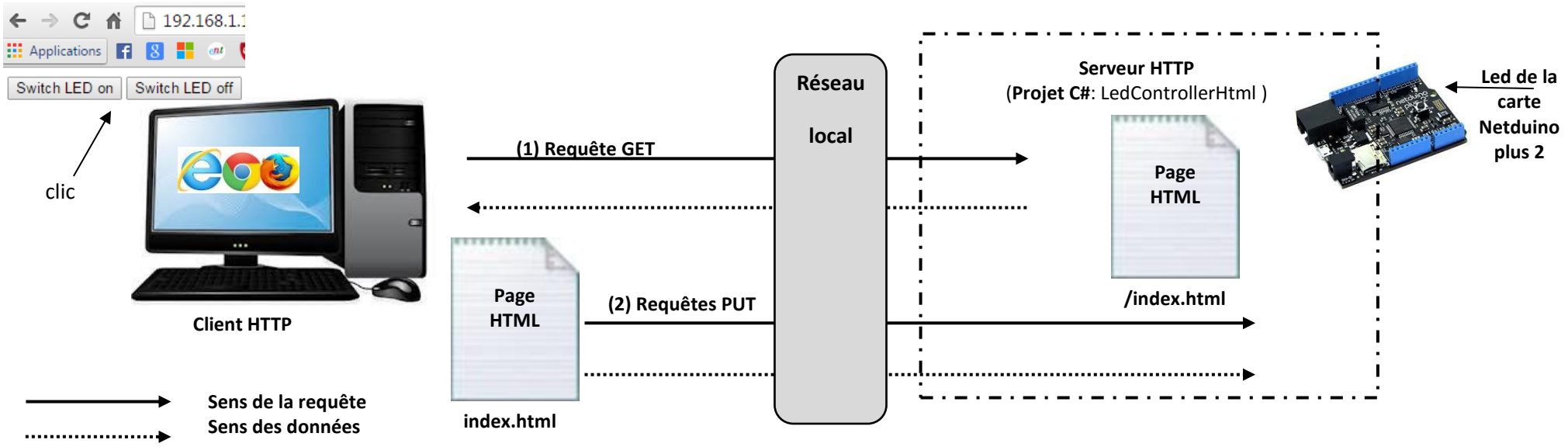
❹ A finir : `réponse à la requête PUT /led/target.`

❺ Activation du serveur. Celui-ci attend les requêtes.

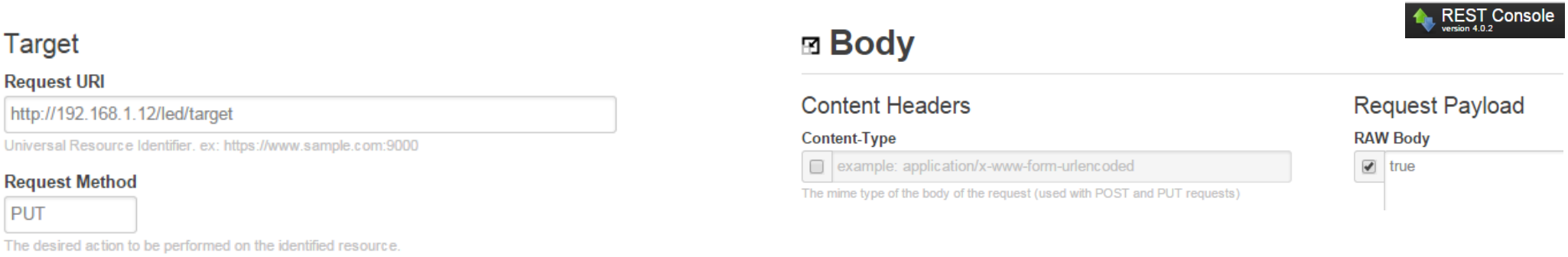
2.2.2 Commande de la Led de la carte Netduino (V2)



(1) Le navigateur charge la page index.html intégrée à la carte Netduino plus 2 (requête GET). L'utilisateur commande la Led de la carte Netduino avec Switch LED on et Switch LED off. (2) Un évènement "clic" sur un des boutons envoie une requête PUT.



Test avec l'extension **« REST Console » du navigateur Chrome**

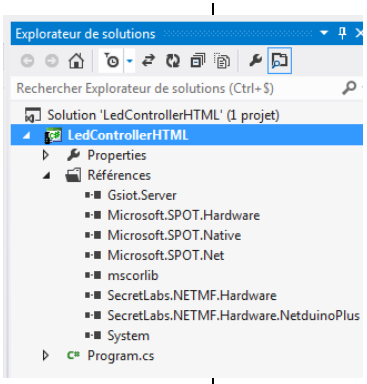


Code C# de l'exemple Serveur 3a

```
using Gsiot.Server;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

namespace LedControllerHTML
{
    public class Program
    {
        public static void Main()
        {
            var ledActuator = new DigitalActuator
            {
                OutputPin = Pins.ONBOARD_LED
            };

            var webServer = new HttpServer
            {
                RequestRouting =
            {
                {
                    "PUT /led/target",
                    new ManipulatedVariable
                    {
                        FromHttpRequest =
                            CSharpRepresentation.TryDeserializeBool,
                        ToActuator = ledActuator.HandlePut
                    }.HandleRequest
                },
                {
                    "GET /index.html",
                    HandleLedTargetHtml
                },
                   {
                    "GET /*",Context => {Context.SetResponse("Erreur
404","text/plain");}
                   }
            }
            };

            webServer.Run();
        }
```

```
DHCP enabled: True
MAC address: 00-04-A3-00-00-00
Device address: 192.168.1.12
Gateway address: 192.168.1.254
Primary DNS address: 192.168.1.254
Base Uri: http://192.168.1.12/
```

L'adresse IP affectée à la carte netduino plus 2 par le serveur DHCP est affichée dans la fenêtre « Sortie » de Visual Studio !




```
    static void HandleLedTargetHtml(RequestHandlerContext context)
      {
          string requestUri = context.BuildRequestUri("/led/target");❶
          var script =
              @"<html>❷
              <head>
                <script type=""text/javascript"">
                  var request;
                  try {❸
                    request = new XMLHttpRequest();
                  } catch (e) {
                    request = new ActiveXObject('Microsoft.XMLHTTP');
                  }
                  function put (content) {❹
                    request.open('PUT', '" + requestUri + @"');
                    request.setRequestHeader('Content-Type', 'text/plain');
                    request.send(content);
                  }
                </script>
              </head>
              <body>
                <p> ❺
                  <input type=""button"" value=""Switch LED on""
                    onclick=""put('true')""/>
                  <input type=""button"" value=""Switch LED off""
                    onclick=""put('false')""/>
                </p>
              </body>
           </html>";
          context.SetResponse(script, "text/html");
      }
  }
}
```

Page Web

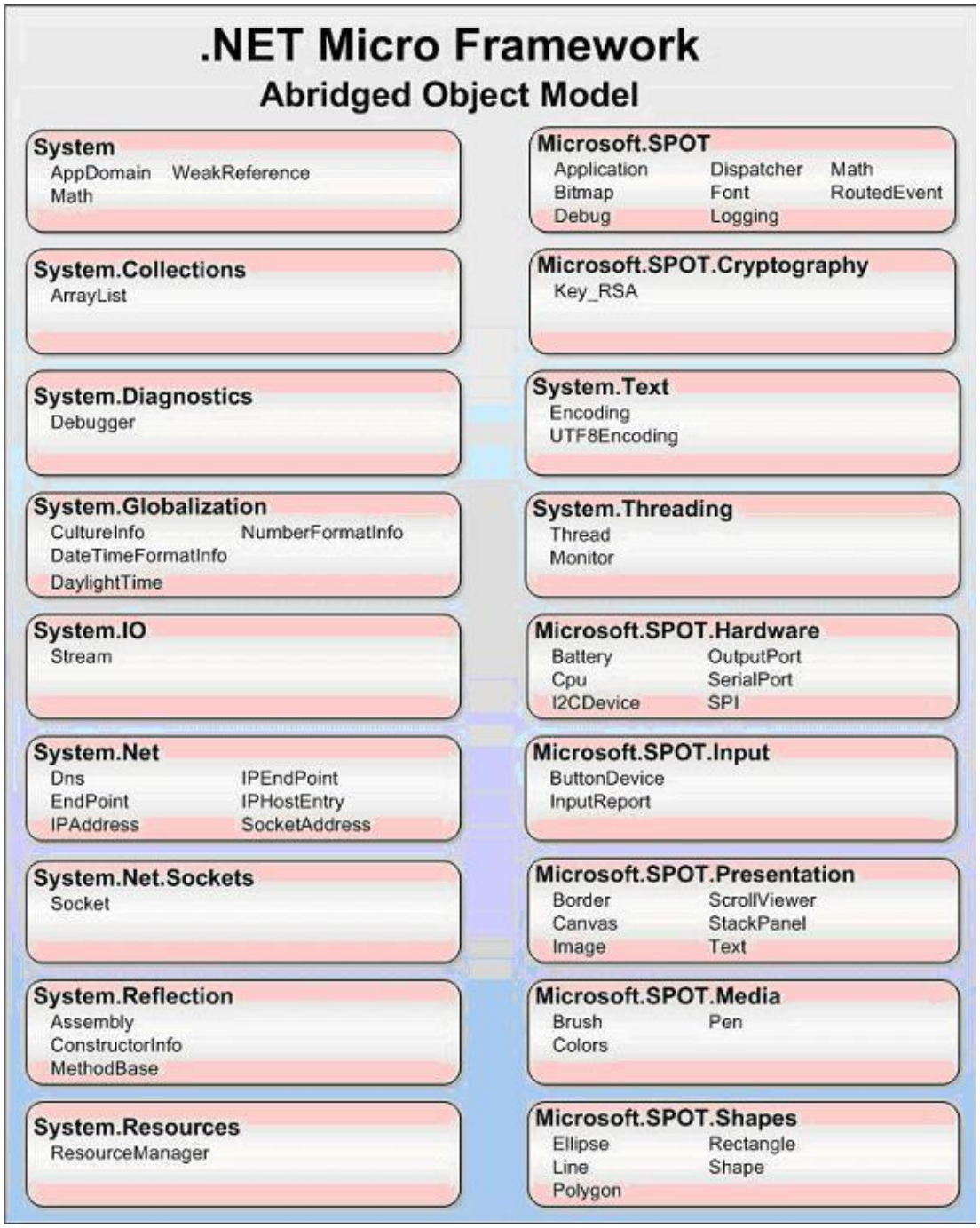
❶ A terminer
❷ A terminer
❸ Création d'un objet XMLHttpRequest ou ActiveX(internet Explorer).
❹ Création et envoie de la requête PUT.
❺ Un clic sur un des bouton-poussoir appelle la fonction put() .

3. **Multithreading**
4. **L'internet des objets**

Annexes

A1 - API Reference for .NET Micro Framework

http://msdn.microsoft.com/en-us/library/ee435793.aspx

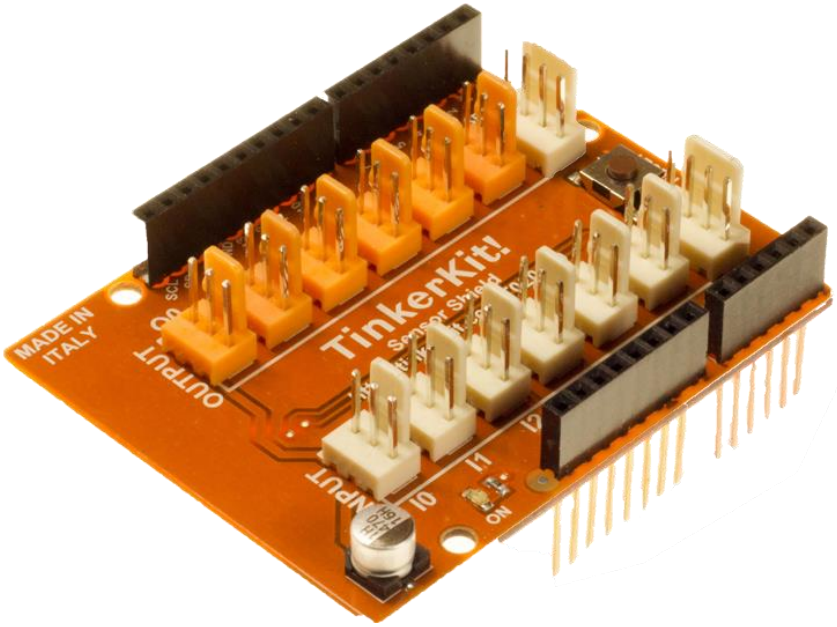


A2 - Les types reconnus par Microsoft Visual Studio et le .NET Microframework

Short Name	.Net Struct/Class	Signed	Width (bytes)	Range	Exemple d'utilisation
bool	Boolean	-	1	Vrai (true) ou faux (false)	bool present = false; Boolean present = false;
byte	Byte	non	1	0 to 255	
sbyte	SByte	oui	1	-128 to 127	
int	Int32	oui	4	2^{31} to $2^{31} - 1$ -2147483648 to 2147483647	int valeur = -164; Int32 valeur = -164;
uint	UInt32	non	4	0 to $2^{32} - 1$ 0 to 4294967295	uint compte = 42; UInt32 compte = 42;
short	Int16	oui	2	-32768 to 32767	
ushort	UInt16	non	2	0 to 65535	
long	Int64	oui	8	2^{63} to 2^{63} - 1	long attente = 421; Int64 attente = 421;
ulong	UInt64	non	8	0 to 2^{64} - 1	
float	Single	oui	4	-3.402823e38 to 3.402823e38	float nombre = 0.45F; Single nombre = 0.45F;
double	Double	oui	8	-1.79769313486232e^{308} to 1.79769313486232e^{308}	double nombre = 0.45; Double nombre = 0.45;
décimal	Decimal	oui	12	±1.0 × 10e−28 to ±7.9 × 10e28 Precise fractional or integral type that can represent decimal numbers with 29 significant digits	
char	Char	-	2	0 à 65535	char lettre = 'A'; Char lettre = 'A';
String	**string**	-	2 par caractère		string couleur = "rouge"; String couleur = "rouge";

Extrait de la documentation Microsoft

A3 - Shield Tinkerkit



The **Sensor Shield v.2** allows you to hook up the TinkerKit **SENSORS** and **ACTUATORS** directly to the NetDuino, without the use of the breadboard.

It has 12 standard TinkeKit 3pin connectors. The 6 labeled **I0** through **I5** are **Analog Inputs**. The ones labeled **O0** through **05** are **Outputs** connected to the PWM capable outputs of the Netduino Board (it is possible to change these to Digital Inputs, in which case they will report either **HIGH** or **LOW**, but nothing in between).

- Pin **11** on the Netduino is **O0** on the shield.
- Pin **10** on the Netduino is **O1** on the shield.
- Pin **9** on the Netduino is **O2** on the shield.
- Pin **6** on the Netduino is **O3** on the shield.
- Pin **5** on the Netduino is **O4** on the shield.
- Pin **3** on the Netduino is **O5** on the shield.

Module description: A green LED signals that the shield is correctly powered, a standard 6mm pushbutton allows you to RESET the board.

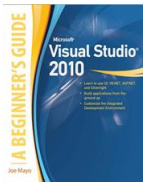
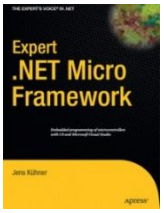
The **4pin TWI socket** allows communication to any device supporting the I2C protocol through the Wire library on Netduino. 5V and Ground are provided on the socket.

The **4pin SERIAL socket** allows the board to communicate with other devices that support serial communication. 5V and Ground are provided on the socket for your convenience.
Note: If you are sending or receiving data to and from the computer this serial connector is not available.

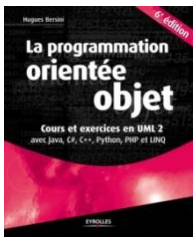
Two mounting holes are provided in the same position found on the Netduino board. A third hole allows you to see the led connected to pin 13 of the Netduino.

Bibliographie

PDF téléchargeables gratuitement sur internet

	**Visual Studio 2010**	Microsoft
	**Expert .NET Micro Framework**	E-Book
	**Getting_Started_with_the_Internet_of_Things**	Make

Livres

	**Cours et Exercices en UML 2**	Hugues Bersini	EYROLLES
	**C#5 et Visual Studio 2013** **Les fondamentaux du langage**	Sébastien Putier	ENI

Numériques payants

Netduino Measurement Electronics: hardware and software [Kindle Edition]	Amazon
Professional's Guide To .NET Micro Framework Application Development [Kindle Edition]	Amazon
Netduino Home Automation Projects [Kindle Edition]	Amazon

Webographie

Blogs

Comparatif	Arduino, Netduino, Raspberry Pi et Beaglebone Black	http://www.dragonflythingworks.com/
	Client et serveur http, multithreading	http://www.gsiot.info/

Matériels, documentation, forums, projets

netduino	http://netduino.com/
CodePlex Project Hosting for Open Source Software	https://www.codeplex.com
github SOCIAL CODING	https://github.com/
CODE PROJECT For those who code	http://www.codeproject.com/

Ressources logicielles

Gestionnaire de paquets à installer dans Visual studio
A partir de outils -> Extensions et mises à jour.

nuget

WPF-like library for simple graphic-UI application using Netduino (Plus) 2 and the FTDI FT800 Eve board.

http://cetmicrowpf.codeplex.com/

FTDI Chip

VM800B43A-PL

Netduino plus 2 GPS

http://n2plusgps.codeplex.com/

Adafruit Ultimate GPS Logger Shield

Shared library for the Netduino hardware and components (MCP23017, DS1307).

http://ndl.codeplex.com/

µPLibrary is an helper library composed of managed drivers for common hardware that you can interface to your .Net Micro Framework board and some other useful components (PIR, TMP102, DS1307, SHT1x, anémomètre, etc.).

http://uplibrary.codeplex.com/

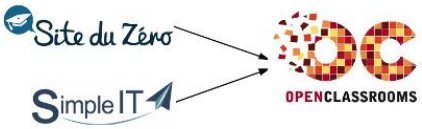
The 'netduino Helpers' is a C# library providing hardware drivers and utility classes

http://netduinohelpers.codeplex.com/

Examples

Project Netduino data logger for race motorcycle based on Netduino + programmed using C# and .NetMF.	http://datalogger.codeplex.com/
Station météo	http://mfweatherstation.codeplex.com/

Technologie .NET

Télécharger		http://goo.gl/bzSSo1
Guide de référence du programmeur		https://goo.gl/acQpwi
Débuter en C# (Vidéos en Français) 		http://goo.gl/59AKhM
Débuter en C# (Cours en ligne)		http://goo.gl/jvSh6K
Plus loin avec le Coach C# (Cours en ligne)		http://goo.gl/YTY98z
Centre de développement Visual C# (Ressources)		http://goo.gl/MKjqSR
NETMF (Ressources, Téléchargement)		http://www.netmf.com/
Le Blog NETMF(Actualités)		http://blogs.msdn.com/b/netmfteam/

Sujets avancés

Timer	https://goo.gl/gaaJQY
Multithreading	http://goo.gl/1hUFcJ

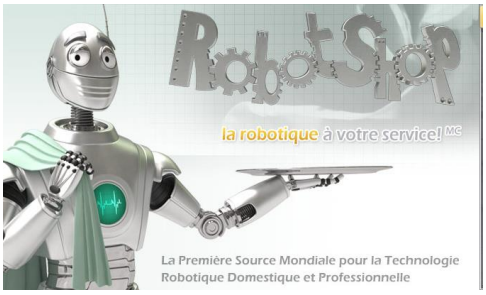
Distributeurs

Mouser Electronics
http://www.mouser.fr/

Generation Robots
France
www.generationrobots.com
Telephone: +33 5 56 39 37 05

Lextronic
France
www.lextronic.fr
Telephone: 01-45-76-83-88

Roboshop
http://www.robotshop.com/



Gotronic

http://www.gotronic.fr/

Index

Aucune entrée d'index n'a été trouvée.