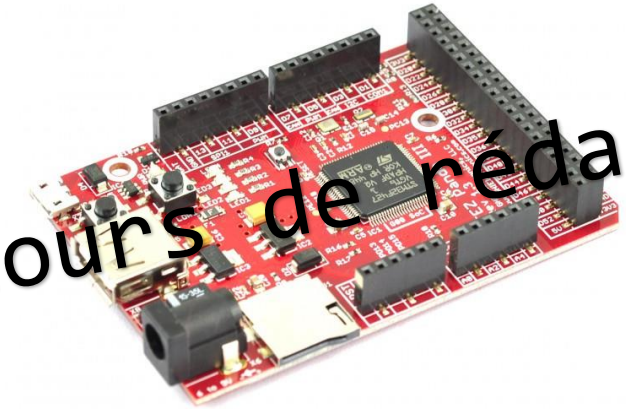
Mise à jour le
15/4/2017

Programmes en langage C# avec .NETMF 4.3

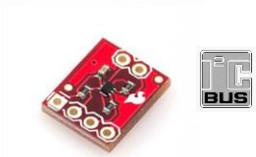
Testés sur les cartes FEZ (PANDA III et COBRA III)

En cours de rédaction



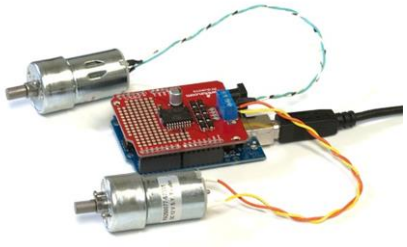
Combo accéléromètre gyroscope
ADXL345-ITG3200

Boussole



TMP102 (Température)

philippemariano@gmail.com



Ardumoto

MD25 : Carte de commande pour moteurs "CC »

Table des matières

Matériel – Logiciel - Documentation

- **Fabricant** : GHI Electronics
- **Distributeurs** : Mouser Lextronic Roboshop

Les matériels en 2017

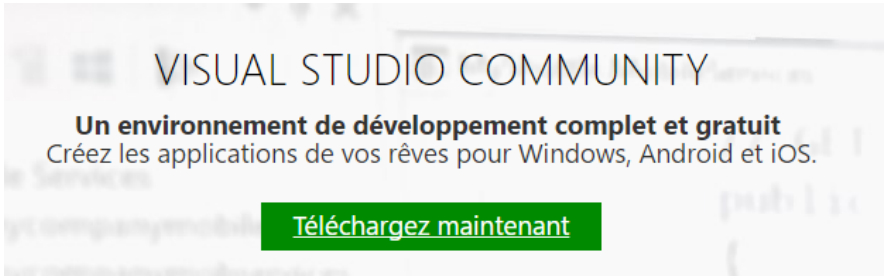
Connectique

	FEZ PANDA III	**FEZ COBRA III**	**GXP Gadgeteer Bridge (sur la carte Cobra)**	**Display NHVN Module**
				
Processor	**180 MHz 32-bit ARM Cortex-M4**	**120 MHz 32-bit ARM Cortex-M3**		
Core System Hardware	**G80 SoC Processor**	**G120 SoM**		
System Platform	**.Net Micro Framework**	**.Net Micro Framework**		
User Available Flash	**256 KB**	**2.87 MB**		
User Available RAM		**13.67 MB**		
GPIO	**53**	**60**		
PWM, Analog etc				
Prix	**$39.95**	**$59.95**		

Pour plus d'information, voir le sélecteur de produit sur le site du fabricant [Lien]

A rajouter Carte SSI + Hub I2C + Adaptateur + Shield Tinkerkit

- **Les logiciels**



GHI Electronics NETMF SDK 2016 R1

Un guide d'installation des logiciels est disponible sur le site GHI Electronics [lien]

- **La documentation**

.NET Micro Framework for Beginners (Nov 2015)	https://goo.gl/ffmqQu
Microsoft NETMF Platform SDK 4.3	https://goo.gl/m3auW3
GHI NETMF 4.3 SDK (Juin 2016)	https://goo.gl/fklCmX
".NET & Internet of Things" (ancienne version 2011)	https://goo.gl/KOc9op

Tableaux récapitulatifs des exemples de code

Lien hypertexte vers le code de l'exemple.

Nom du répertoire contenant le projet Visual Studio.

Lien hypertexte vers la page web décrivant **la classe spécifique** au circuit intégré ou au module (maintenue sur **Github**).

La classe spécifique au circuit intégré ou au module est inclue dans la bibliothèque **MicroToolsKit** (un NuGet sur **NuGet.org**).

La photo du montage à réaliser est dans le sous répertoire **Doc_A_Consulter** du répertoire du projet.

Les entrées, sorties numériques

	Visual Studio	Description (CI ou module)			
E/S 1	BlinkingLed	- Sortie numérique : faire clignoter la Led de la carte Panda 3			
E/S 2	PANDA_3_LED_BP	- E/S numériques : commander une Led avec un bouton-poussoir.			
E/S 3	PANDA_3_EasyStepperMot	- Sorties numériques : commander un moteur pas à pas avec une carte EasyStepper Driver Motor V4.4	☒	☒	

	Visual Studio	Description (CI ou module)			
INT 1	PANDA_3_INT	E numérique : commander une Led avec un bouton-poussoir.			

	Visual Studio	Description (CI ou module)			
PWM 1	PANDA_3_PWM	PWM : Faire varier la luminosité d'une Led			
PWM 2	PANDA_3_ARDUMOTO	PWM : Faire varier la vitesse d'un moteur à CC			☒
PWM 3	PANDA_3_SERVO	PWM : Régler la position d'un servomoteur de modélisme			

Les entrées analogiques

	Visual Studio	Description (CI ou module)			
Analog 1	PANDA_II_POT	Entrée : Régler la fréquence de clignotement d'une Led avec un potentiomètre.			
Analog 2	PANDA_II_Thermo	Entrée : Mesurer la température ambiante avec un module GHI FEZ thermomètre.			
Analog 3	PANDA_II_S_AN	Sortie : Rampe sur sortie analogique			

La communication série – Asynchrone - UART

	Visual Studio	Description (CI ou module)			
UART 1	PANDA_II_UART	UART : Transmettre une valeur numérique via une liaison RS232			☒
UART 2	PANDA_3_ELCD162	UART : Utiliser un afficheur Lcd à commande série ELCD-162	☒	☒	☒
UART 3a	PANDA_II_XBeeE	UART : Transmettre des données avec un module XBee			☒
UART 3b	PANDA_II_XBeeR	UART : Recevoir des données avec un module XBee	☒	☒	☒

La communication série – Synchrone – I²C

	Visual Studio	Description (CI ou module)			
I2C_1	PANDA_3_I2C	I²C : Chenillard sur huit Leds reliées à un port d'E/S PCF8574.	☒	☒	☒
I2C_2	PANDA_3_I2C_LCD	I²C : Commander un afficheur LCD à circuit PCF2119.	☒	☒	☒
I2C_3	PANDA_3_I2C_SRF08_US	I²C : Mesurer une distance avec un télémètre à ultrasons SRF08.	☒	☒	☒
I2C_4		I²C : Recopier l'état de boutons poussoirs sur des Leds via des PCF8574 (carte SSI).	☒	☒	☒
I2C_5	PANDA_II_I2C_HMC6352	I²C : Lire la direction donnée par une boussole HMC6352.	☒	☒	☒
I2C_6	PANDA_II_I2C_TMP102	I²C : Mesurer la température ambiante avec un capteur TMP102.	☒	☒	☒
I2C_7	PANDA_II_I2C_MD25	I²C : Commander deux motoréducteurs à C.C. équipés d'encodeurs avec une carte MD25.	☒	☒	☒
I2C_8	PANDA_II_I2C_TSL2561	I²C : Mesurer la luminosité ambiante avec un capteur TSL2561	☒	☒	☒
I2C 9	NetduinoMLX90614	I²C : Mesurer la température d'un objet avec un capteur MLX90614.	☒	☒	
I2C 10	NetduinoMCP3424	I²C : Acquérir des données issues de capteurs analogique avec un CAN MCP3424.	☒	☒	
	PANDA_II_I2C_ADXL345	I2C(Accéléromètre ADXL345/gyroscope ITG3200)			

La communication série – Synchrone – One Wire

	Visual Studio	Description (CI ou module)			
One_Wire_1	PANDA_II_1_Wire_DS18B20	OneWire : Mesurer la température ambiante avec un capteur **DS18B20**			

La gestion du temps – HTR

	Visual Studio	Description (CI ou module)			
	PANDA_3_HTR				

Les systèmes de fichier

	Visual Studio	Description (CI ou module)			
SD1	Panda_3_SD	Accès à un fichier sur une carte SD. (A venir)			

Divers

	Visual Studio	Description (CI ou module)			
	PANDA_3_HTR PANDA_3_CLAVIER_PC_USB PANDA_3_AFFICHEUR_GRAPHIQUE				

Correspondance entre les chapitres du document « **.NET & Internet of Things** » et les répertoires des projets Microsoft Visual C# 2010 EXPRESS

Exemple	Répertoires Visual C# 2010 Express	Chapitres du document « .NET & Internet of Things »	Remarques
EXEMPLE_A EXEMPLE_B EXEMPLE_C	1_Test_Ethernet 2_Test_IP 3_Test_DHCP 4_Test_HTTP 5_InternetOfThings 6_SendEmail	Chapitre 5 paragraphe 5.2 Connecting Ethernet Chapitre 5 paragraphe 5.2 Connecting Ethernet (Accessing the Internet) Chapitre 5 paragraphe 5.2 Connecting Ethernet (Using DHCP) Chapitre 6 paragraphe 6.1 User Datagram Protocole (UDP Transceive data with PC) Chapitre 7 FEZ – http Chapitre 10 Sensor Monitoring Chapitre 12 You've got mail and SMS	

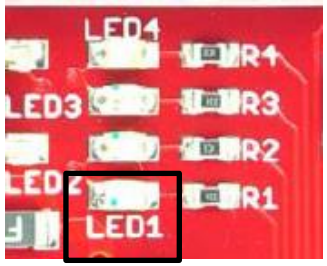
1. Les entrées, sorties numériques

1.1. Faire clignoter la « LED1 » de la carte FEZ PANDA 3 !

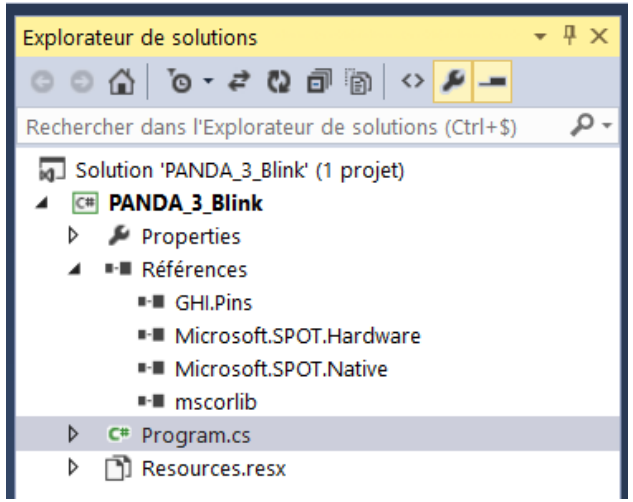
Code C#

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

namespace PANDA_3_Blink
{
    public class Program
    {
        public static void Main()
        {
            // Blink board LED
            bool ledState = false;
            var LED = new OutputPort(FEZPandaIII.Gpio.Led1, false);

            while (true)
            {
                Thread.Sleep(500); // Sleep for 500 milliseconds
                // toggle LED state
                ledState = !ledState;

                if (ledState){
                    Debug.Print("Led éteinte");
                }
                else {
                    Debug.Print("Led éclairée");
                }
                led.Write(ledState);
            }
        }
    }
}
```

Espaces de noms



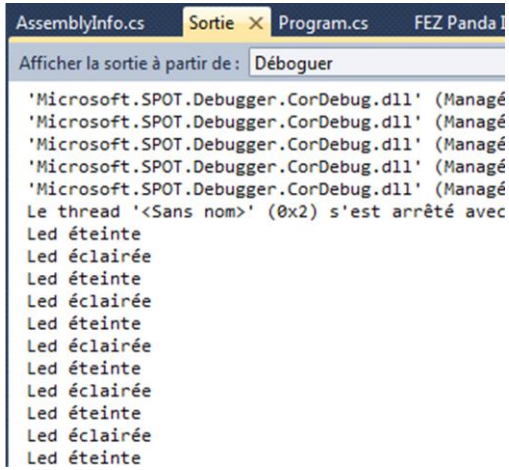
Extrait du manuel utilisateur de la carte FEZ PANDA 3



La construction de l'objet LED est simplifiée par l'auto complétion (IntelliSense) !

Pour illustrer l'utilisation du debugger !

Pas à pas
F10 : Step Over
F11 : Step into

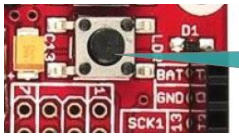


F12

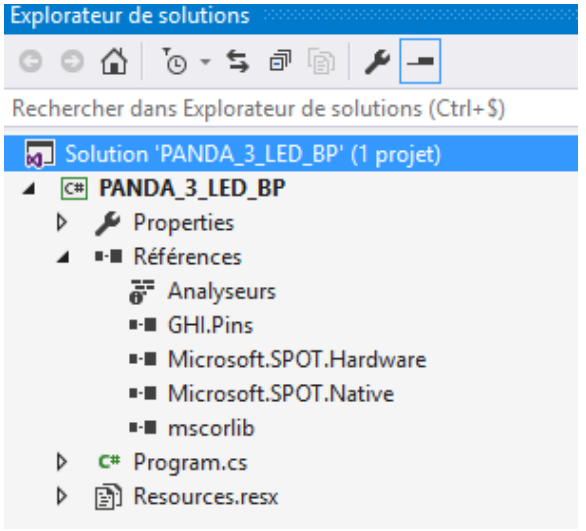
```
public class OutputPort : Port
    {
        public OutputPort(Cpu.Pin portId, bool initialState);
        protected OutputPort(Cpu.Pin portId, bool initialState, bool glitchFilter,
Port.ResistorMode resistor);

        public bool InitialState { get; }

        public void Write(bool state);
    }
```

.net Micro Framework

1.2. Commander une LED avec un bouton-poussoir !



Extrait doc carte FEZ PANDA 3

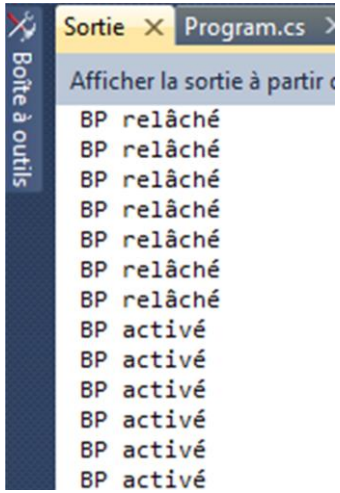


Code C#

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

namespace
{
    public class Program
    {
        public static void Main()
        {
            OutputPort LED = new OutputPort(FEZPandaIII.Gpio.Led1, false);
            InputPort Button = new InputPort(FEZPandaIII.Gpio.Ldr0, false, Port.ResistorMode.PullUp);

            while (true)
            {
                LED.Write(!Button.Read());
                if (!Button.Read())
                {
                    Debug.Print("BP activé");
                }
                else
                {
                    Debug.Print("BP relâché");
                }
                Thread.Sleep(10);
            }
        }
    }
}
```

Utilisation du debugger

Pas à pas
F10 : Step Over
F11 : Step into



F12

```
public class InputPort : Port
    {
        public InputPort(Cpu.Pin portId, bool glitchFilter, Port.ResistorMode resistor);
        protected InputPort(Cpu.Pin portId, bool initialState, bool glitchFilter, Port.ResistorMode
resistor);
        protected InputPort(Cpu.Pin portId, bool glitchFilter, Port.ResistorMode resistor,
Port.InterruptMode interruptMode);

        public bool GlitchFilter { get; set; }
        public Port.ResistorMode Resistor { get; set; }
    }
```

.net Micro Framework

1.3. Commander un moteur pas à pas avec une carte EasyDriverStepperMotor V4.4

code C#

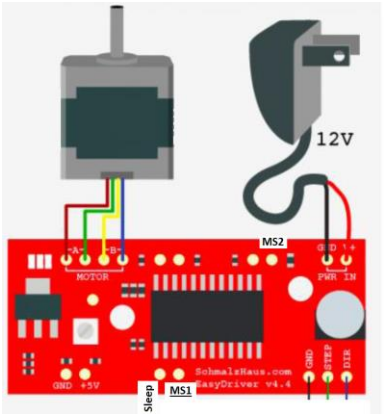
```
using System;
using System.Threading;
using Microsoft.SPOT;
using GHI.Pins;

using Microtoolskit.Hardware.MotorDrivers;

namespace PANDA_3_EasyStepperMot
{
    public class Program
    {
        public static void Main(){

            var stepper = new EasyStepperDriver(FEZPandaIII.Gpio.D13, FEZPandaIII.Gpio.D12, FEZPandaIII.Gpio.D2,
            FEZPandaIII.Gpio.D10, FEZPandaIII.Gpio.D11, FEZPandaIII.Gpio.D3);
            stepper.WakeUp();

          while (true) {
             // Exemples d'utilisation de la méthode Step() et des propriétés StepMode, StepDirection et Steptime
            stepper.EnableOutputs(); // Activation des sorties
            Debug.Print("Sleep= " + stepper.IsDriverSleep + " Enable= " + stepper.IsOutputsEnable);
            Debug.Print("Full Forward"); // 360° pour le moteur ITC-VNC-1
            stepper.Turn(nbpas, EasyStepperDriver.Direction.Forward, delay, EasyStepperDriver.Mode.Full);
            Debug.Print("Pas= " + stepper.Steps + " Mode= " + stepper.StepMode + " Dir= " + stepper.StepDirection +
            " time= " + stepper.StepDelay + "ms" + "\n");
            stepper.DisableOutputs(); Thread.Sleep(time); // Désactivation des sorties pendant la temporisation

            Debug.Print("Half Backward"); stepper.EnableOutputs(); // 180° pour le moteur ITC-VNC-1
            stepper.Turn(nbpas, EasyStepperDriver.Direction.Backward, delay, EasyStepperDriver.Mode.Half);
            Debug.Print("Pas= " + stepper.Steps + " Mode= " + stepper.StepMode + " Dir= " + stepper.StepDirection +
            " time= " + stepper.StepDelay + "ms" + "\n");
            stepper.DisableOutputs(); Thread.Sleep(time); // Désactivation des sorties pendant la temporisation

            Debug.Print("Quater Forward"); stepper.EnableOutputs();  // 90° pour le moteur ITC-VNC-1
            stepper.Turn(nbpas, EasyStepperDriver.Direction.Forward, delay, EasyStepperDriver.Mode.Quarter);
            Debug.Print("Pas= " + stepper.Steps + " Mode= " + stepper.StepMode + " Dir= " + stepper.StepDirection +
            " time= " + stepper.StepDelay + "ms" + "\n");
            stepper.DisableOutputs(); Thread.Sleep(time); // Désactivation des sorties pendant la temporisation

            Debug.Print("OneEighth Backward"); stepper.EnableOutputs(); // 45° pour le moteur ITC-VNC-1
            stepper.Turn(nbpas, EasyStepperDriver.Direction.Backward, 1, EasyStepperDriver.Mode.OneEighth);
            Debug.Print("Pas= " + stepper.Steps + " Mode= " + stepper.StepMode + " Dir= " + stepper.StepDirection +
            " time= " + stepper.StepDelay + "ms" + "\n");
            stepper.DisableOutputs(); Thread.Sleep(time); // Désactivation des sorties pendant la temporisation
            Thread.Sleep(2 * time);
          }
        }
    }
```

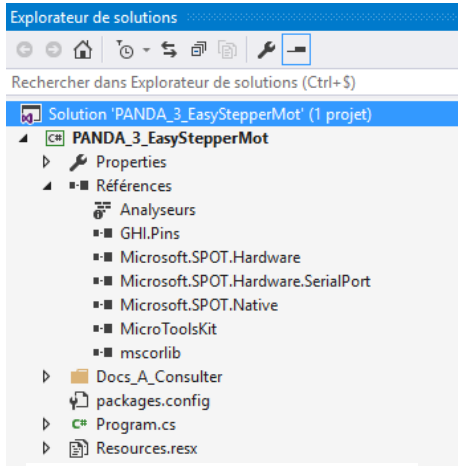
github SOCIAL CODING

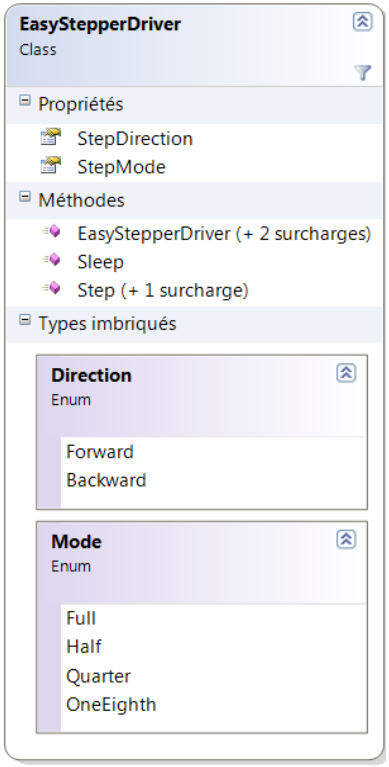
Page Web de la classe EasyStepperDriver

nuget

MicroToolsKit







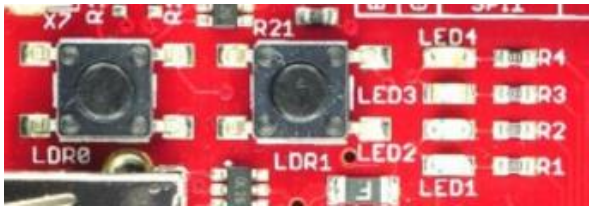
Connexions à la carte Panda 3 :	DIR -> Di13	Step -> Di12	*MS_2 -> Di11	*MS_1 -> Di10	*/Enable ->Di3	*/Sleep -> Di2

* Option

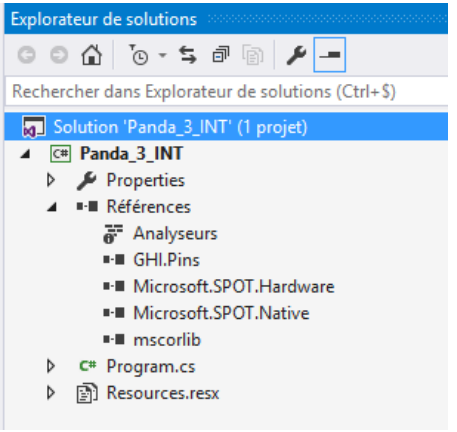
1.4. Utiliser une interruption (commande de la LED1 avec le bouton-poussoir LDR0 !)

Code C#

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

namespace Panda_3_INT
{
  public class Program
  {
      static OutputPort LED;

      public static void Main()
        {
         OutputPort LED = new OutputPort(FEZPandaIII.Gpio.Led1,false);
         // La broche générera une interruption sur chaque front (montant et descendant)
         InterruptPort BPLdr0 = new InterruptPort(FEZPandaIII.Gpio.Ldr0, true, Port.ResistorMode.PullUp, Port.InterruptMode.InterruptEdgeBoth);

         // Déclaration d'un gestionnaire d'interruption
         BPLdr0.OnInterrupt += new NativeEventHandler(BPLDR0_OnInterrupt);

         Thread.Sleep(Timeout.Infinite); // Après l'initialisation, Main n'est plus utilisé
      }

      static void BPLDR0_OnInterrupt (uint port, uint state, DateTime time)
      {
          LED.Write(state == 0);
          if (state == 0)
          {
              Debug.Print("BP Activé");
          }
          else
          {
              Debug.Print("BP Relâché");
          }
      }
  }
}
```

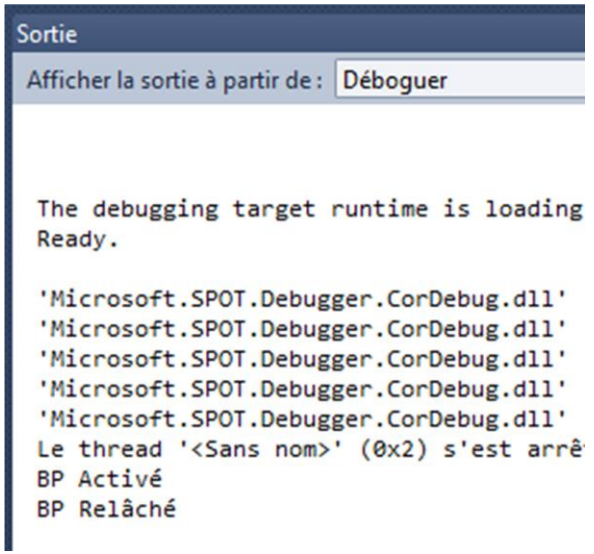


Extrait doc carte FEZ PANDA 3



Utilisation du debugger

Pas à pas
F10 : Step Over
F11 : Step into



Une action est affichée seulement lors d'une interruption.

F12

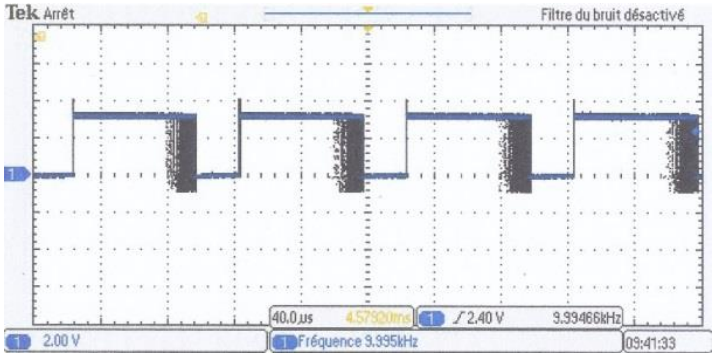
```
public sealed class InterruptPort : InputPort
    {
        public InterruptPort(Cpu.Pin portId, bool glitchFilter, Port.ResistorMode resistor,
Port.InterruptMode interrupt);
        public Port.InterruptMode Interrupt { get; set; }
        public void ClearInterrupt();
        public override void DisableInterrupt();
        public override void EnableInterrupt();
    }
```

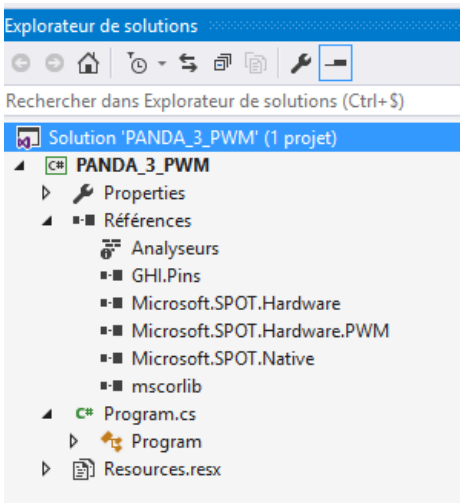
.net Micro Framework

1.5. PWM: Faire varier la luminosité de la LED1

Remarque : La luminosité de la LED1 croit puis décroit périodiquement (10% < α < 90%)

Extrait doc carte FEZ PANDA 3



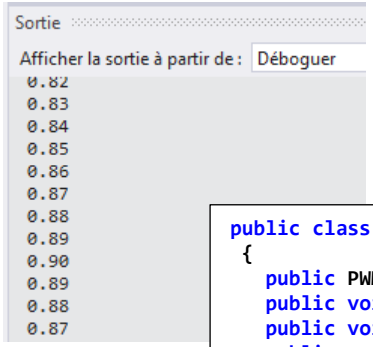


Code C#

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

namespace PANDA_3_PWM
{
    public class Program
    {
        public static void Main()
        {
            double step = 0.01;
            double level = 0.5; // Rapport cyclique
            var frequency = 10000;
            var Led = new PWM(FEZPandaIII.PwmOutput.Led1, frequency, level, false);

            Led.Start();

            while (true)
            {
                level = level + step;
                Led.DutyCycle = level;

                if ((level >= 0.9) || (level <= 0.1))
                {
                    step = step * -1; // Invert the step
                }
                Debug.Print(level.ToString("F2"));
                Thread.Sleep(10);
            }
        }
    }
}
```

Utilisation du debugger **pour visualiser le réglage du rapport cyclique.**

Pas à pas
F10 : Step Over
F11 : Step into



F12

```
public class PWM : IDisposable
 {
   public PWM(PWM.Pin pin);
   public void Dispose();
   public void Set(bool pinState);
   public void Set(int freq_Hz, byte dutyCycle);
  public void SetPulse(uint period_nanosecond, uint
highTime_nanosecond);
 }
```

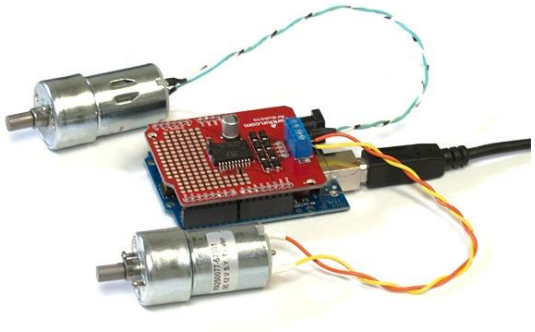
1.6. PWM : Commande d'un motoréducteur (GHM-16) équipé d'un codeur

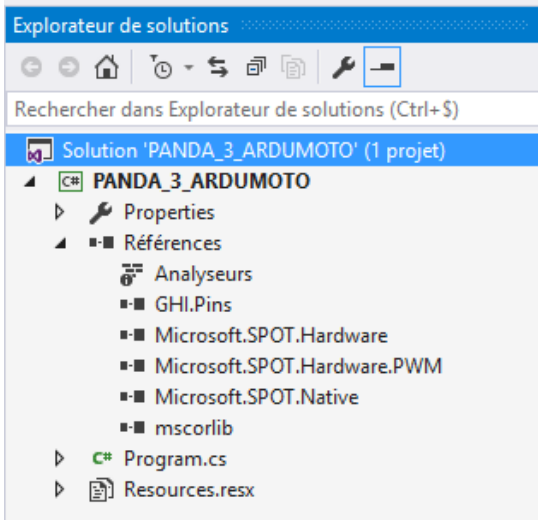
Code C#

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

using GHI.Pins;

namespace PANDA_3_ARDUMOTO
{
    public class Program
    {
        public static void Main()
        {
            // Mise en œuvre du shield Ardumoto
            // ATTENTION : Afin de conserver le port I²C de la carte, le shield Ardumoto
            // doit être modifié comme dans le tableau ci-dessus si le Moteur A est utilisé
            var duty = 0.1; // Rapport cyclique entre 0 et 1
            var freq = 1000; // Fréquence en Hz

            // Moteur 12V connecté sur la voie B de la carte Ardumoto
            var PWMB = new PWM(FEZPandaIII.PwmOutput.D11, freq, duty, false);
            var DIRB = new OutputPort(FEZPandaIII.Gpio.D13, true);

            PWMB.Start();

            while (true)
            { // Rampe d'accélération
                for (int i = 0; i < 9; i++)
                {
                    PWMB.DutyCycle = duty + 0.1 * i;
                    Thread.Sleep(500);
                }
                Thread.Sleep(4000); // Palier à vitesse constante
                PWMB.DutyCycle = 0.1;
            }
        }
    }
}
```

Connectique

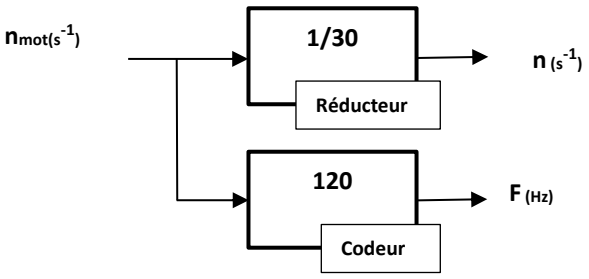
Panda 3	Ardumoto	Moteur
Di5	PWMA	A
Di12	DIRA	
Di11	PWMB	B
Di13	DIRB	



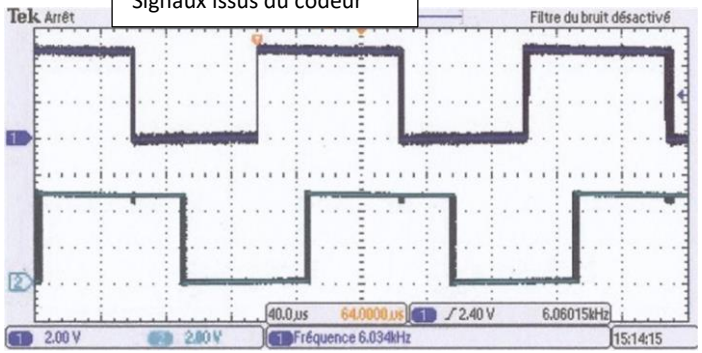


A tester

Motoréducteur Lynxmotion GHM-16 • Voltage: 6-12vdc • RPM: 200 • Couple: 0.78Kg-cm (10.83oz-in) • Réduction: 30:1 • Poids : 0.34 livres (154g) • Diamètre extérieur : 37mm • Diamètre (essieu): 6mm	
Encodeur en quadrature lynxmotion • Alimentation : 5V Cycles par révolution: 120 • Comptes en quadrature par révolution: 480 • Fréquence: 30kHz Red = +5vdc Black = Ground Green = Output A Yellow = Output B	 E4P-120-079-HT



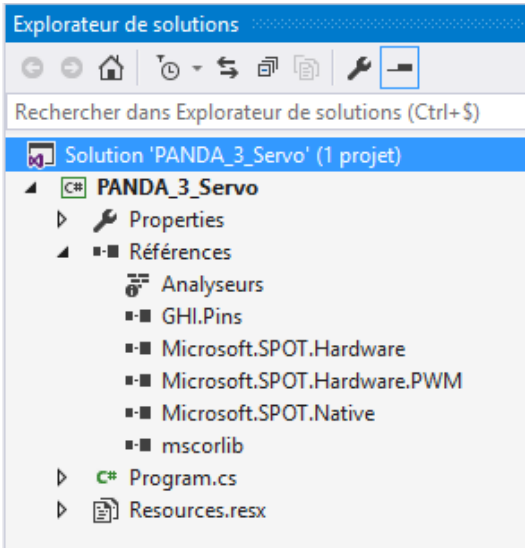
Signaux issus du codeur



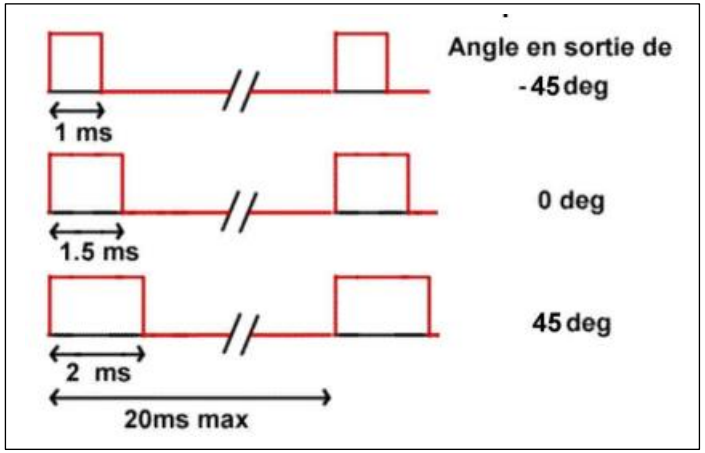
1.7. PWM : Commande d'un servomoteur

Code C#

A tester

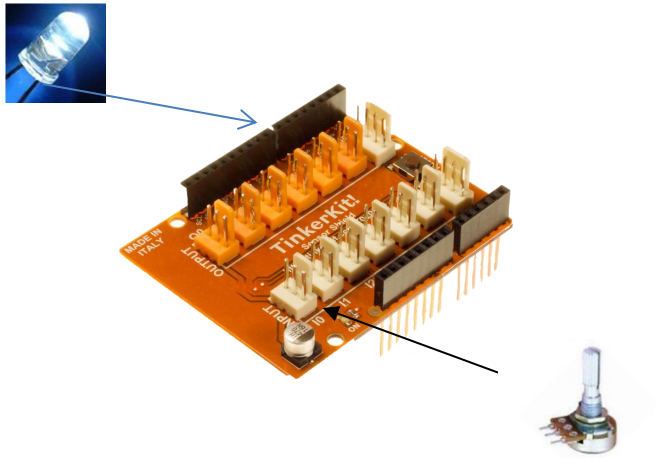


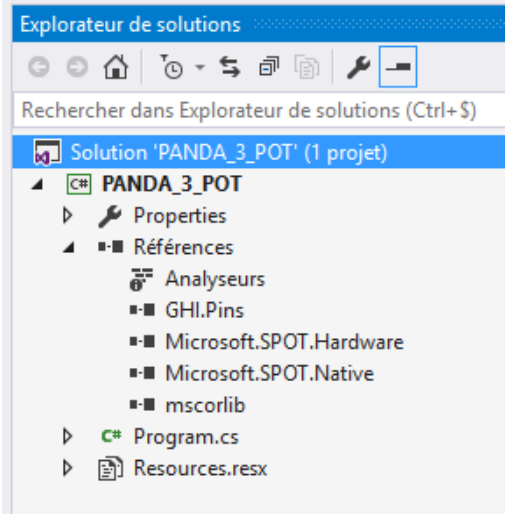

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using GHI.Pins;


namespace PANDA_3_Servo
{
    public class Program
    {
        public static void Main()
        {
            // Documentation de la classe PWM http://msdn.microsoft.com/en-us/library/microsoft.spot.hardware.pwm(v=vs.102).aspx
            var duty = 0.05; // Rapport cyclique entre 0 et 1
            var freq = 50; // Fréquence en Hz min pour ce type de servo

            // Configuration l'E/S numérique 5 de la carte Panda 3 en PWM
            // Connexion du Servomoteur au connecteur O4 de la carte TINKERKIT
            PWM Servo = new PWM(FEZPandaIII.PwmOutput.D5, freq, duty, false);

            Servo.Start();

            while (true)
            { // Déplacement angulaire de 90°
                for (var i = 0; i < 6; i++)
                {
                    Servo.DutyCycle = duty + i * 0.01;
                    Thread.Sleep(1000);
                }

                Servo.DutyCycle = duty;
            }
        }
    }
}
```



2. Les entrées, sorties analogiques

2.1. Régler la fréquence de clignotement d'une Led

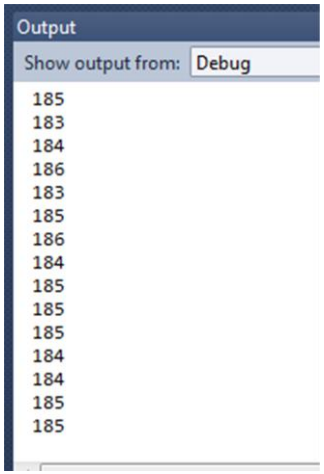




Code C#

```
using System;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.Hardware;
using GHIElectronics.NETMF.FEZ;

namespace Panda_II_Pot
{
    public class Program
    {
        public static void Main()
        {
            var led = new OutputPort(FEZPandaIII.Gpio.Led1, false);

            // Le potentiomètre est connecté à l'entrée I0 du shield Tinkerkit
            var Pot = new AnalogInput(FEZPandaIII.AnalogInput.A0);

            while (true)
            {
                var N = Pot.ReadRaw() / 4; // Résultat sur 10 bits
                Debug.Print(N.ToString());
                led.Write(true);
                Thread.Sleep(N);
                led.Write(false);
                Thread.Sleep(N);
            }       }
    }
}
```

Utilisation du **debugger** pour visualiser la valeur renvoyée par le CAN en fonction de la position du potentiomètre.



Pas à pas
F10 : Step Over
F11 : Step into

F12

```
public class AnalogIn : IDisposable
{
public AnalogIn(AnalogIn.Pin ain);
public void Dispose();
public int Read();
public void SetLinearScale(int minValue, int maxValue);
}
```

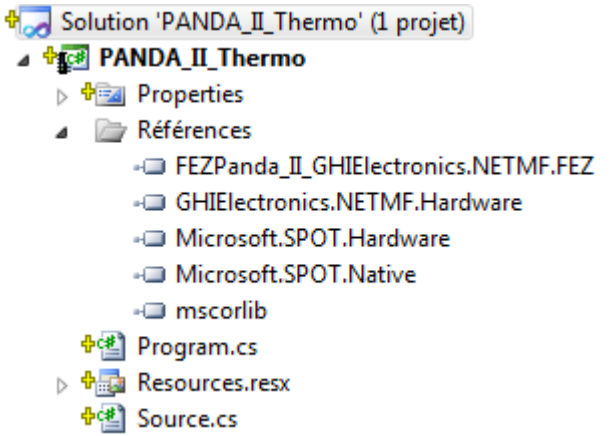
.net Micro Framework

2.2. Mesurer une température avec le module FEZ thermomètre (CTN)

Code C#

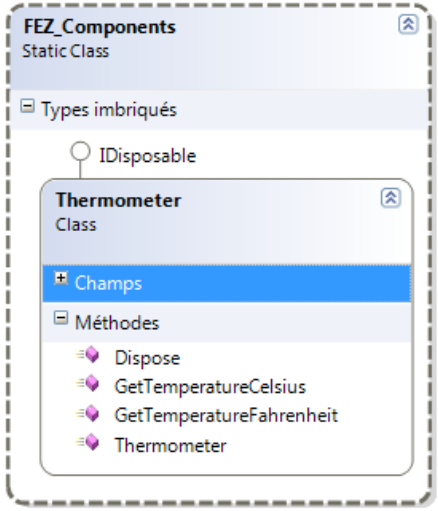
```
using System;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.FEZ;


namespace PANDA_II_Thermo
{
    public class Program
    {
        public static void Main()
        {
        // Create Thermometer object assigned to the a Thermometer Component connected to An0 with analog
        FEZ_Components.Thermometer myThermometer = new FEZ_Components.Thermometer(FEZ_Pin.AnalogIn.An2);
        int value = 0;

        while (true)
            {
            value = myThermometer.GetTemperatureCelsius();
            Debug.Print("myThermometer reading is: " + value.ToString());
            Thread.Sleep(1000);
            }
        }
    }
}
```

Module FEZ (Thermomètre)



Utilisation du debugger **pour visualiser la valeur de la température.**

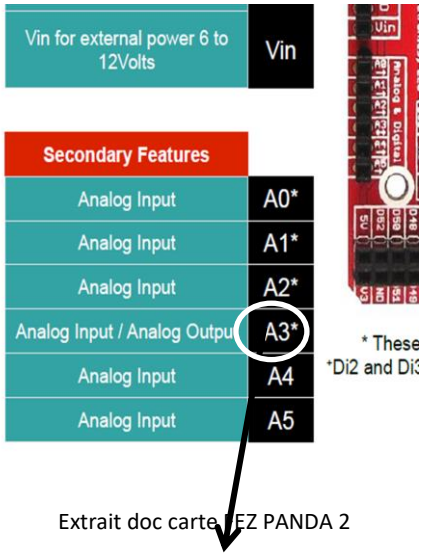
Pas à pas
F10 : Step Over
F11 : Step into



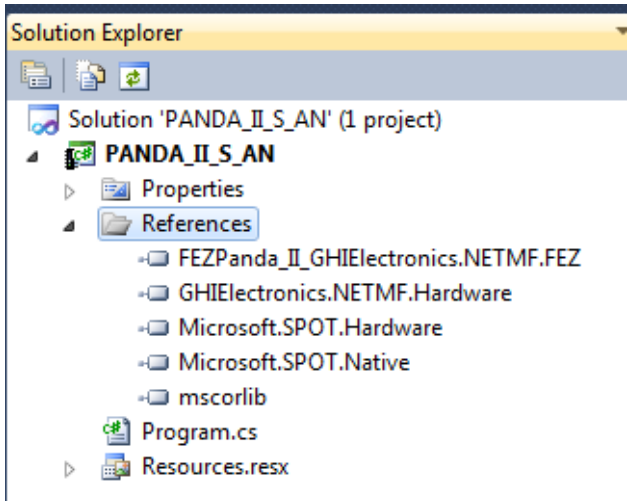
2.3. Générer un signal "rampe" sur la sortie analogique

Code C#

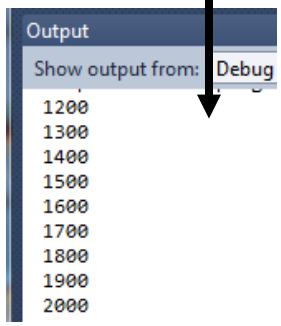
```
using System;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.Hardware;
using GHIElectronics.NETMF.FEZ;

namespace PANDA_II_S_AN
{
    public class Program
    {
        public static void Main()
        {
            AnalogOut VoltLevel = new AnalogOut((AnalogOut.Pin)FEZ_Pin.AnalogOut.An3);
            VoltLevel.SetLinearScale(0, 3300); // Mise à l'échelle

            while (true)
            {
                for (int i = 0; i < 10; i++)
                {
                    VoltLevel.Set(1000 + 100 * i);
                    Debug.Print((1000 + 100 * i).ToString());
                     Thread.Sleep(1);
                }
                VoltLevel.Set(1000);
            }
        }
    }
}
```

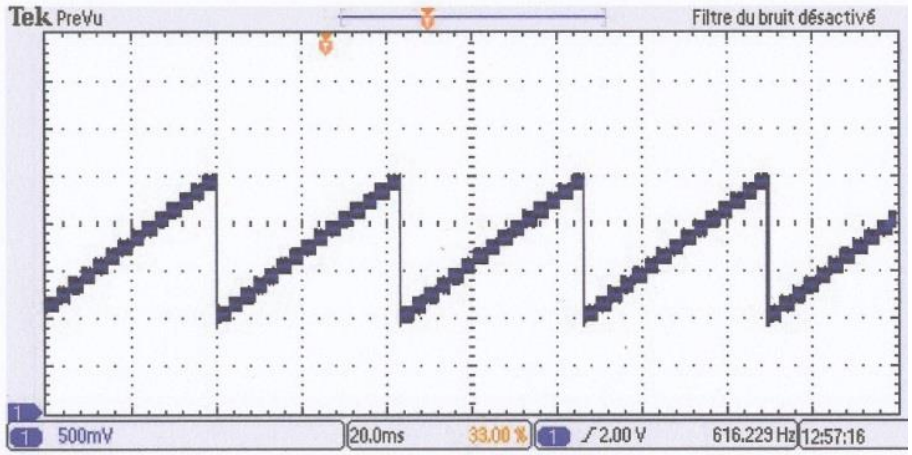


Extrait doc carte EZ PANDA 2



La construction de l'objet sortie An3 **semble complexe mais l'**IntelliSense **simplifie le travail !**





Utilisation du debugger **pour visualiser la valeur envoyée à la fonction CNA.**

Pas à pas
F10 : Step Over
F11 : Step into

Explications détaillées : **French Beginner Guide ebook1.03 (8/2010)**
(Chapitre 14 : Entrées/Sorties analogiques)

3. La communication série

3.1. UART : Transmettre une valeur numérique

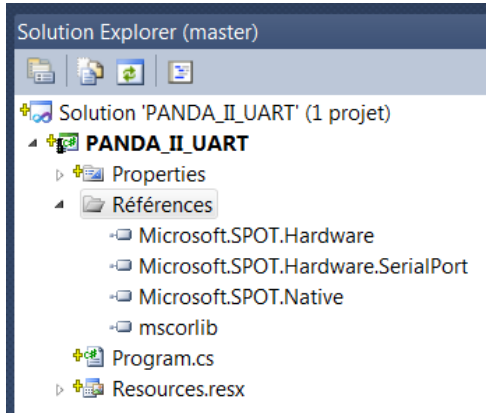
Code C#

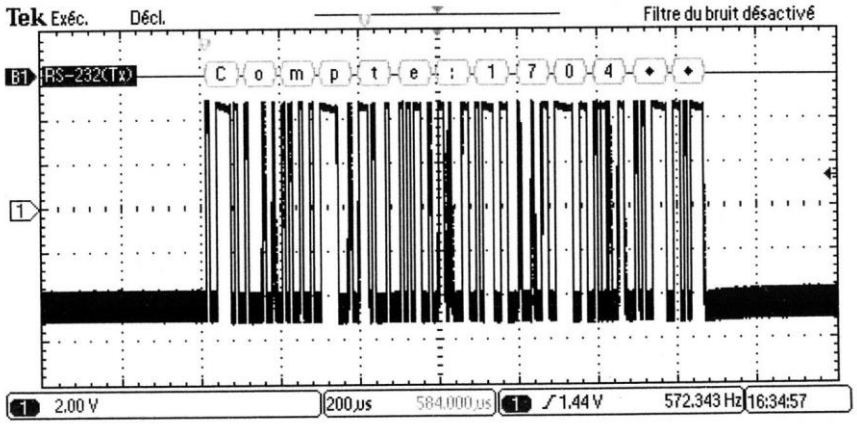
```
using System;
using System.Text;
using System.Threading;
using System.IO.Ports;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

namespace PANDA_II_UART
{
    public class Program
    {
        public static void Main()
        {
            SerialPort UART = new SerialPort("COM1", 115200);
            int counter = 0;
            UART.Open();

            while (true)
            {
                // Création d'une chaine de caractères
                string counter_string = "Compte:" + counter.ToString() + "\r\n";
                // Conversion de la chaîne en octets
                byte[] buffer = Encoding.UTF8.GetBytes(counter_string);
                // Envoie des octets au port série
                UART.Write(buffer, 0, buffer.Length);
                Debug.Print(counter_string);

                // Incrémentation du compteur
                counter++;
                // Attente de 100ms entre deux envois
                Thread.Sleep(100);
            }
        }
    }
}
```

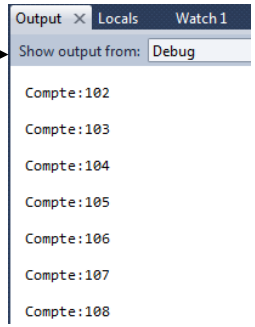
ıpt inputs
ı 2.2K pull up

	Secondary Features
Di7*	CAN Bus Channel 1 Output
Di6*	PWM
Di5*	PWM
Di4*	CAN Bus Channel 1 Input
Di3*	(Open Drain Pin*) I2C SCL
Di2*	(Open Drain Pin*) I2C SDA
Di1*	COM1 Output
Di0*	COM1 Input

Extrait doc carte FEZ PANDA 2







Utilisation du **debugger** pour visualiser la valeur envoyée sur le port COM.

Pas à pas
F10 : Step Over
F11 : Step into

```
public class SerialPort : IDisposable
    {
public SerialPort(string portName);
        public SerialPort(string portName, int baudRate);
…
public void Open();
        public int Read(byte[] buffer, int offset, int count);
        public int Write(byte[] buffer, int offset, int count);
…
```

F12

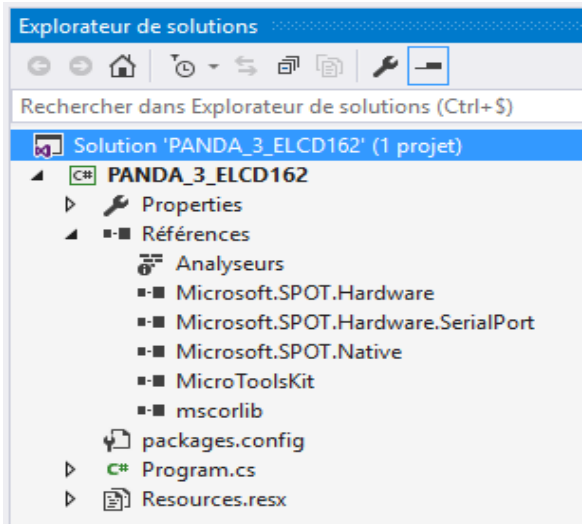
3.2. UART : Utiliser un afficheur LCD à commandes séries (Module COMFILE ELCD-162)

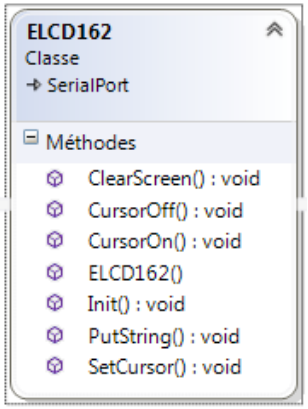
Code C#

```
using Microtoolskit.Hardware.Displays;

namespace PANDA_3_ELCD162
{
    public class Program
    {
        public static void Main()
        {
            // Création d'un objet "Afficheur LCD" connecté au COM1
            // de la carte PANDA 3 avec un convertisseur série (ELCD162)
            ELCD162 Lcd = new ELCD162("COM1");

            // Test des méthodes du Driver ELCD162
            Lcd.Init(); Lcd.ClearScreen(); Lcd.CursorOff();

            Lcd.SetCursor(3, 0);
            Lcd.PutString("Lycee PEM!");
            Lcd.SetCursor(6, 1);
            Lcd.PutString("SSI");
        }
    }
}
```

Description de la classe **SerialELCD162**

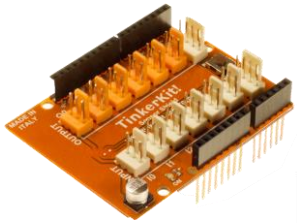
MicroToolsKit





Matériels

- Carte PANDA III
- Sensor Shield TINKERKIT V2
- Afficheur 2x16 + module ELCD162 COMFILE Série

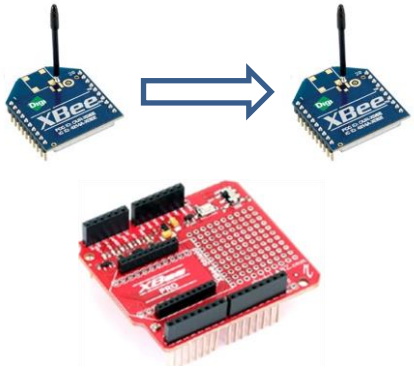


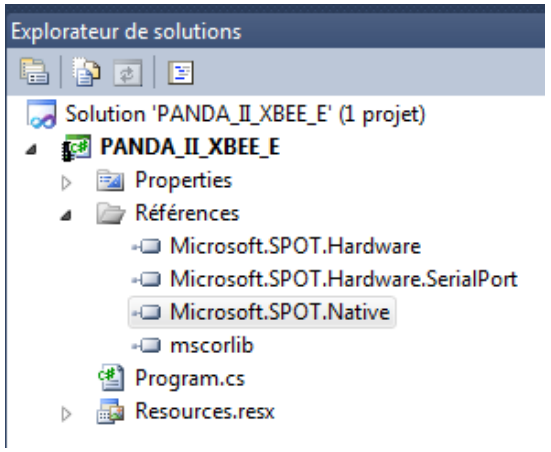
Shield TINKERKIT V2

Afficheur 2x16 équipé d'un
module ELCD162 COMFILE Série (19200b/s)

3.3. UART : Transmettre des données avec des modules XBEE (Emission/Réception)

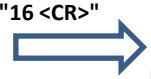
Code C#

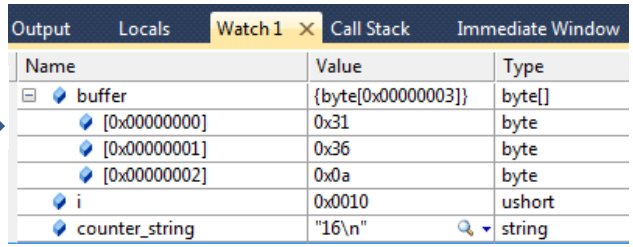




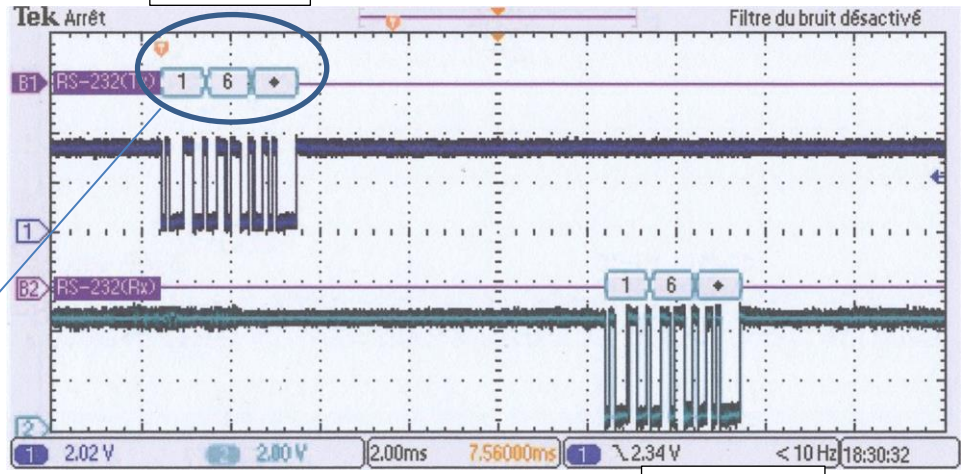

```
using System;
using System.Text;
using System.IO.Ports;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;



namespace PANDA_II_XBEE_E
{
    public class Program
    {
        public static void Main()
        {
            // Emetteur
            // Création d'un port série pour la communication avec le module XBEE. XBEE_RX sur Di0 et XBEE_TX sur Di1.
            SerialPort xbee_E = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);
            xbee_E.Open();

            UInt16 i=0; // Valeur à transmettre

            // Création d'une chaîne de caractère et d'un buffer d'émission
            string counter_string;
            byte[] buffer;

            // Envoi de la valeur toutes les 2s
            while (true)
            {
                for (i = 0; i <= 65535; i++)
                {   // \n = <CR> (délimite la fin de la valeur)
                    counter_string = i.ToString() + "\n";
                    buffer = Encoding.UTF8.GetBytes(counter_string);
                    xbee_E.Write(buffer, 0, buffer.Length);
                    Thread.Sleep(2000);
                }
            }
        }
    }
}
```

Transmission de "16 <CR>"





Emission



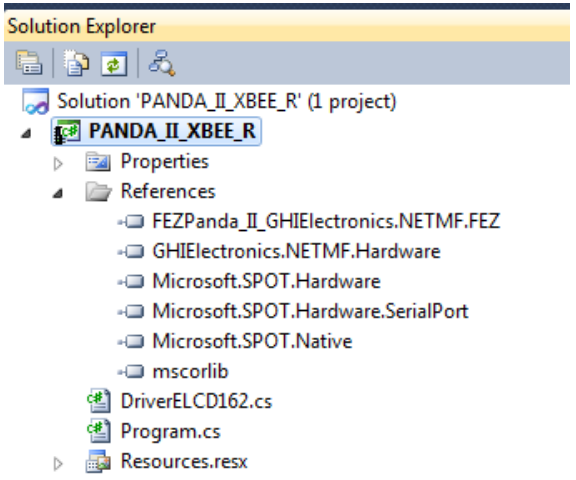
Réception

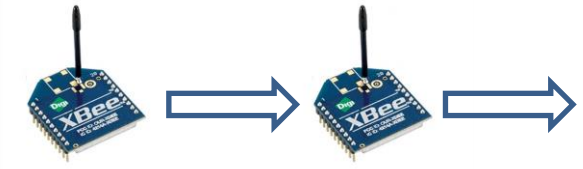
Explications détaillées : **French Beginner Guide ebook1.03 (8/2010)**
(Chapitre 17 : Interfaces séries / 17.1 UART)

Transmettre des données avec des modules XBEE (**Emission**/Réception)

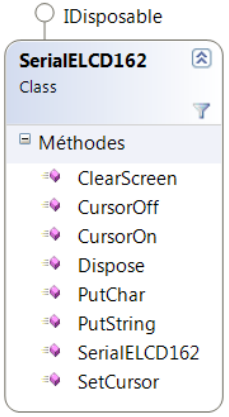
```
using System;
using System.Text;
using System.IO.Ports;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.FEZ;

namespace PANDA_II_XBEE_R{
    public class Program{
        public static void Main(){
            // Récepteur
            // Création d'un port série pour la communication avec le module XBEE. XBEE_RX sur Di0 et XBEE_TX sur Di1.
            SerialPort xbee_R = new SerialPort ("COM1", 9600, Parity.None, 8, StopBits.One);
            xbee_R.Open();
            // Création d'un port série logiciel pour l'afficheur LCD
            FEZ_Components.SerialELCD162 ELCD162 = new FEZ_Components.SerialELCD162((byte)FEZ_Pin.Digital.Di5);
            // Création d'une chaîne de caractères et d'un buffer de réception
            byte[] buffer = new byte[6] {0,0,0,0,0,0};
            char[] chars;
            // Initialisation du LCD
            ELCD162.ClearScreen(); ELCD162.CursorOff();

            while (true)
            {
                int pos = 0; bool Aff = false;
                // Lecture de la valeur transmise
                do{
                    xbee_R.Read(buffer, pos++, 1);
                    Aff = true;
                } while (xbee_R.BytesToRead > 0);

                // Affichage
                if (Aff)
                {
                    int i;
                    chars = Encoding.UTF8.GetChars(buffer);
                    for (i = 0; i < buffer.Length; i++){
                        if ((buffer[i] < 0x30) || (buffer[i] > 0x39)){break;}
                    }
                    string str = new string(chars, 0, i);
                    ELCD162.ClearScreen();ELCD162.PutString(str);
                   for (int j = 0; j < buffer.Length; j++){
                        buffer[j] = 0;
                    }
                }
            }
        }
    }
}
```

Description de la classe **SerialELCD162**





Afficheur 2x16 équipé d'un module COMFILE Série (19200b/s)

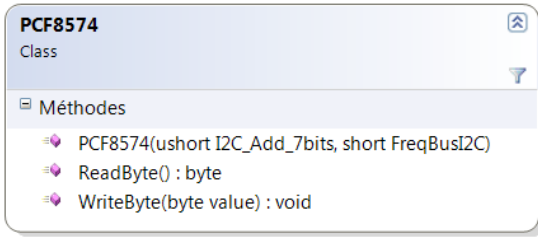


Explications détaillées : **French Beginner Guide ebook1.03 (8/2010) (Chapitre 17 : Interfaces séries / 17.1 UART)**

Le fichier DriverELCD162.cs gère une liaison **RS232 Soft**.
L'afficheur peut être placé sur une broche E/S quelconque.

3.4. I2C : Chenillard sur huit LED reliées à un port d'E/S PCF8574

Code C#




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

using Microtoolskit.Hardware.IO;

namespace PANDA_3_I2C
{
    public class Program
    {
        public static void Main()
        {
            // Paramètres du bus I2C
            byte addLeds_I2C = 0x38; // Adresse (7 bits) du PCF8574 relié aux Leds de la carte SSI
            Int16 Freq = 400; // Fréquence d'horloge du bus I2C en kHz

            byte stateLED = 0xFE; // Etat initial des LED. (Un 0 logique => Led éclairée)

            // Création d'un objet Led
            PCF8574 Leds = new PCF8574(addLeds_I2C, Freq);

            Leds.WriteByte(stateLED); // Initialisation de l'état des Leds
            Thread.Sleep(500);

            while (true)
             {
                // Modification de l'état des LED
                if (stateLED != 0)
                {
                    stateLED = (byte)(stateLED << 1);
                }
                else
                {
                    stateLED = 0xFF;
                }

                //Ecriture sur les Leds
                Leds.WriteByte(stateLED);
                Debug.Print(stateLED.ToString());
                Thread.Sleep(500); // Pour la simulation
             }
        }
    }
}
```

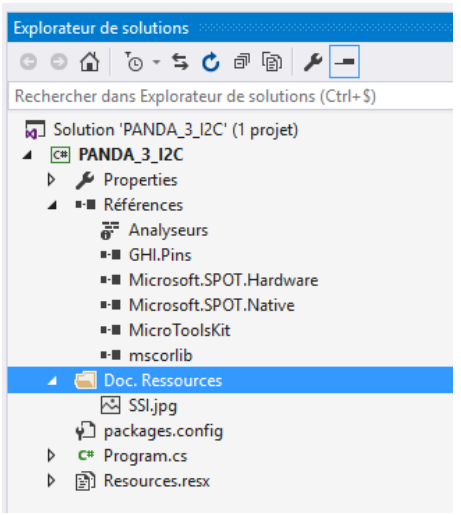
nuget

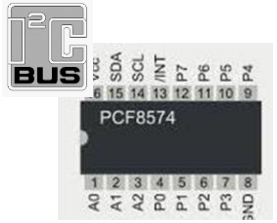
MicroToolsKit

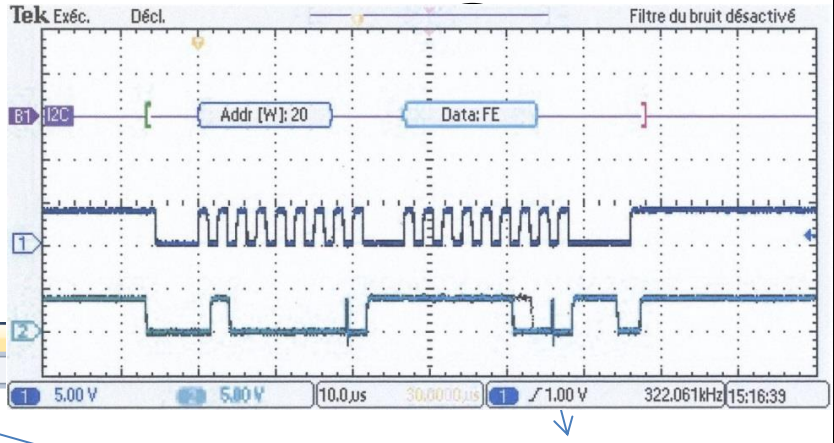
upt inputs
:h 2.2K pull up

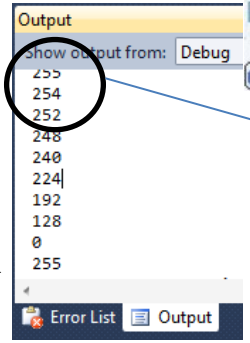
	Secondary Features
Di7*	CAN Bus Channel 1 Output
Di6*	PWM
Di5*	PWM
Di4*	CAN Bus Channel 1 Input
Di3*	(Open Drain Pin*) I2C SCL
Di2*	(Open Drain Pin*) I2C SDA
Di1*	COM1 Output
Di0*	COM1 Input

Extrait doc carte FEZ PANDA 3









Utilisation du **debugger** pour visualiser les valeurs envoyées au bus I2C

Pas à pas
F10 : Step Over
F11 : Step into

Rem : L'éclairage d'une LED est commandé par un niveau logique 0.

Matériels
- Carte PANDA III
- Sensor Shield TINKERKIT V2
- Carte SSI + Hub I²C

github SOCIAL CODING

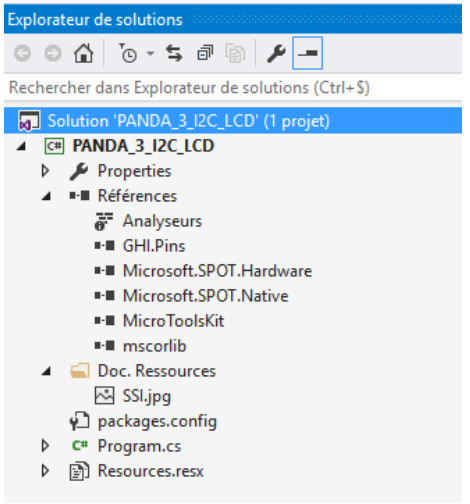
Description de la classe **PCF8574**

3.5. I2C : Commander un LCD I2C à PCF2119 BATRON ou MIDAS

Code C#

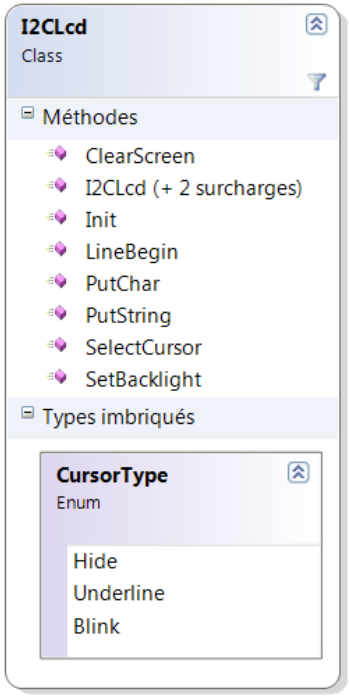
```
using System.Threading;

using Microtoolskit.Hardware.Displays;

namespace PANDA_3_I2C_LCD
{
    public class Program
        {
            byte y = 0x5A;

            // Création d'un objet Lcd I²C Midas
            I2CLcd Lcd = new I2CLcd();

            // Initialisation du Lcd I2C
            Lcd.Init();

            // Message
            Lcd.ClearScreen();
            Lcd.PutString(3, 0, "SSI...");
            Lcd.PutChar(11, 0, 0x4E);
            Lcd.PutString(2, 1, "Bonjour!!");

            // Jauges linéaires virtuelles
            for (byte w = 0x5a; w < 0x60; w++)
                Lcd.PutChar((byte)(w - 0x51), 1, w);

            while (true)
            {
                // Démo Widgets
                Lcd.PutChar(14, 0, 0x11);
                Lcd.PutChar(15, 0, 0x21);
                Lcd.PutChar(13, 0, 0x4C);
                Thread.Sleep(200);
                Lcd.PutChar(14, 0, 0x21);
                Lcd.PutChar(15, 0, 0x11);
                Lcd.PutChar(13, 0, 0x4B);
                Thread.Sleep(200);

                // Démo jauge linéaire virtuelle
                Lcd.PutChar(0, 0, (byte)y);
                y++;
                if (y > 0x5F) y = 0x5A;
            }
        }
    }
```

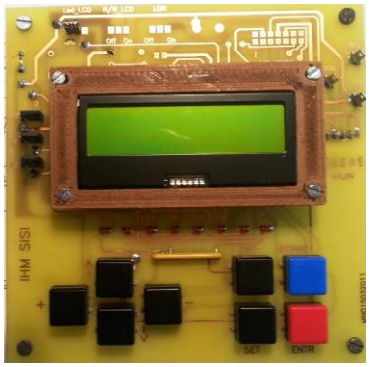
I2C Batron 2 x16



Description de la classe **I2CLCD**

MicroToolsKit





IHM SSI

Matériels

- Carte PANDA III
- Sensor Shield TINKERKIT V2
- Carte SSI + Hub I²C

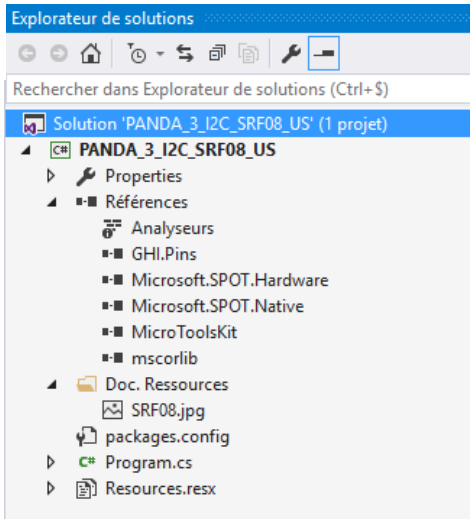
3.6. I2C : Mesurer une distance avec un télémètre à ultrasons SRF08

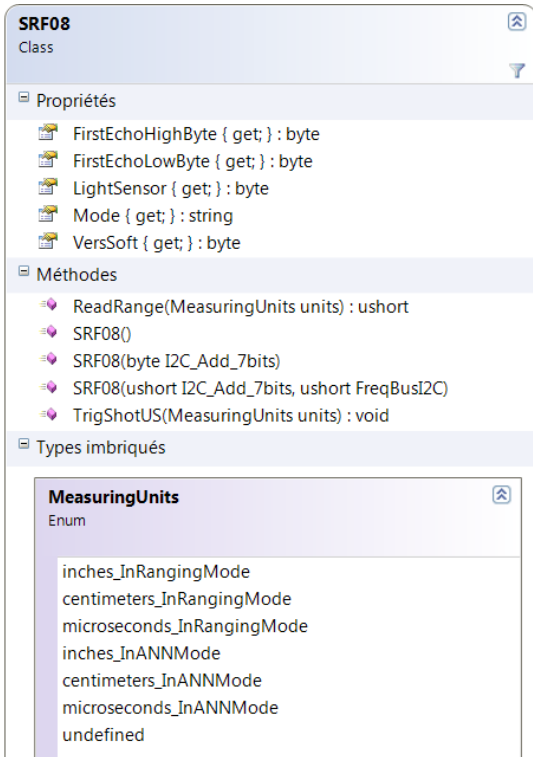
Mesure de la distance entre le télémètre et un obstacle et affichage sur un LCD à `commandes séries(RS232)`

Code C#

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microtoolskit.Hardware.Sensors;

namespace PANDA_II_I2C_SRF08_US
{
    public class Program
    {
            // Paramètres du bus I2C
            byte addTelem_I2C = 0x70; // Adresse (7 bits) du télémètre SRF08
            UInt16 Freq = 400; // Fréquence d'horloge du bus I2C en kHz

            // Création d'un objet télémètre SRF08
            SRF08 I2CTelemeter = new SRF08(addTelem_I2C, Freq);

            // Affichage de la version du software du télémètre
            Debug.Print("VerSoft: " + I2CTelemeter.VersSoft);
            // Lecture et affichage de la luminosité
            Debug.Print("Light: " + I2CTelemeter.LightSensor);
            // Affichage du mode de mesure: Ranging ou ANN
            Debug.Print("Mode: " + I2CTelemeter.Mode);
            Debug.Print("_____________________________________");
            Thread.Sleep(2000);

            while (true){
                // Déclenchement, lecture et affichage de la distance en cm
                Debug.Print("Distance: " + I2CTelemeter.ReadRange(SRF08.MeasuringUnits.centimeters_InRangingMode) + "cm");
                // Déclenchement, lecture des registres correspondant au premier echo
                Debug.Print("1st Echo HighByte: " + I2CTelemeter.FirstEchoHighByte + "  " + "1st Echo LowByte: " + I2CTelemeter.FirstEchoLowByte);
                // Déclenchement, lecture et affichage de la distance en inch
                Debug.Print("Distance: " + I2CTelemeter.ReadRange(SRF08.MeasuringUnits.inches_InRangingMode) + "inches");
                // Lecture des registres correspondant au premier echo
                Debug.Print("1st Echo HighByte: " + I2CTelemeter.FirstEchoHighByte + "  " + "1st Echo LowByte: " + I2CTelemeter.FirstEchoLowByte);
                // Déclenchement, lecture et affichage de la distance en microsecondes
                Debug.Print("Distance: " + I2CTelemeter.ReadRange(SRF08.MeasuringUnits.microseconds_InRangingMode) + "µs");
                // Lecture des registres correspondant au premier echo
                Debug.Print("1st Echo HighByte: " + I2CTelemeter.FirstEchoHighByte + "  " + "1st Echo LowByte: " + I2CTelemeter.FirstEchoLowByte);
                // Lecture et affichage de la luminosité
                Debug.Print("Light: " + I2CTelemeter.LightSensor);
                Debug.Print("_____________________________________");
                // Afichage du mode de mesure: Ranging ou ANN
                Debug.Print("Mode: " + I2CTelemeter.Mode);
                Thread.Sleep(1000);
            }
        }
    }
}
```

Description de la classe **SRF08**





Matériels

- Carte PANDA III
- Sensor Shield TINKERKIT V2
- Hub I²C
- SRF08

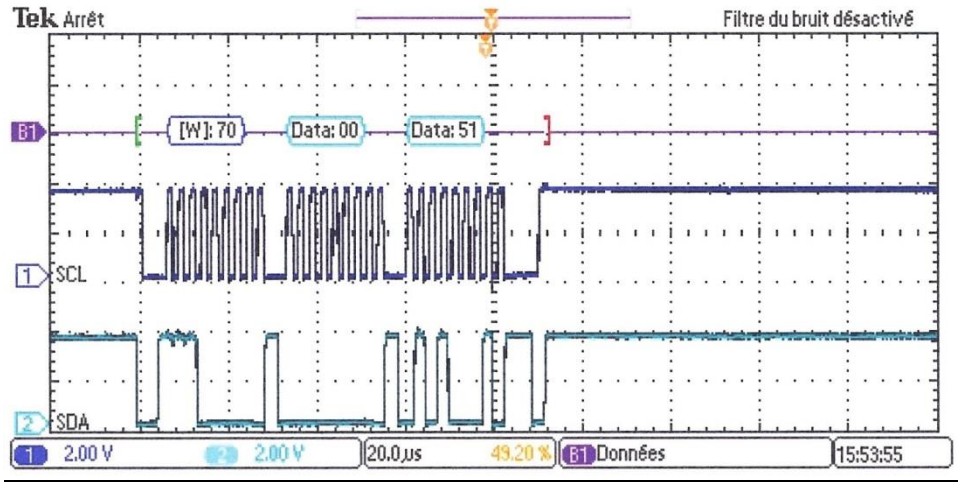
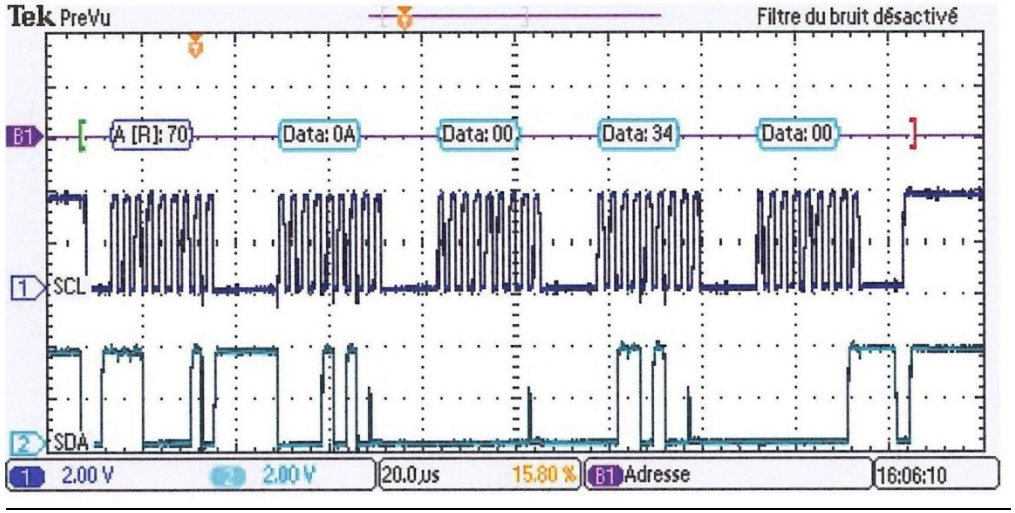
MicroToolsKit

SRF08
Transducteur à ultrasons

```
_____________________________________
Distance: 55cm
1st Echo HighByte: 0  1st Echo LowByte: 55
Distance: 21inches
1st Echo HighByte: 0  1st Echo LowByte: 21
Distance: 6824µs
1st Echo HighByte: 12  1st Echo LowByte: 168
Light: 139
_____________________________________
Mode: Ranging
```

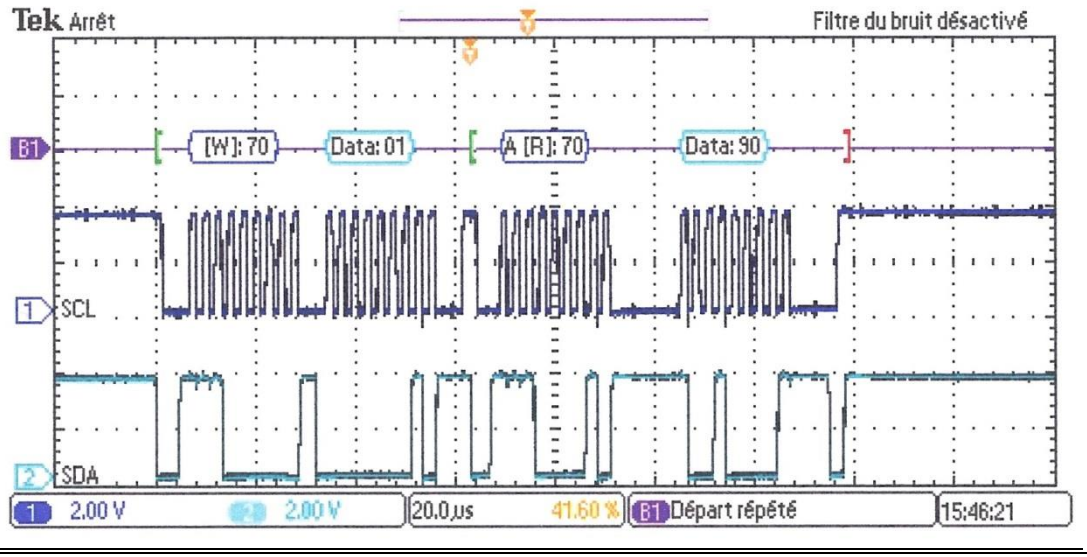
2C : Mesurer une distance avec un télémètre à ultrasons SRF08 (Suite)

Chronogrammes

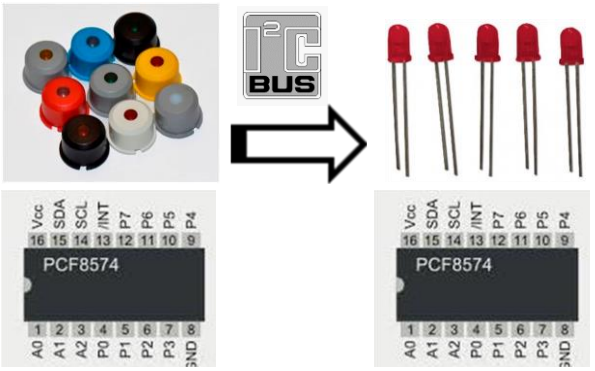
Méthode	Transaction	Chronogrammes
ReadRange(SRF08.UnitType.centimeters)	**Déclenchement du tir US** Start **@SLA + W** **N° registre** **Commande** Stop	
	Attente 75ms	....
	Lecture de la mesure Start **@SLA + R** **Reg0 (Software Rev)** **Reg1 (Light Sensor)** **Reg2 (1st Echo low)** **Reg3(1st Echo High)** Stop	

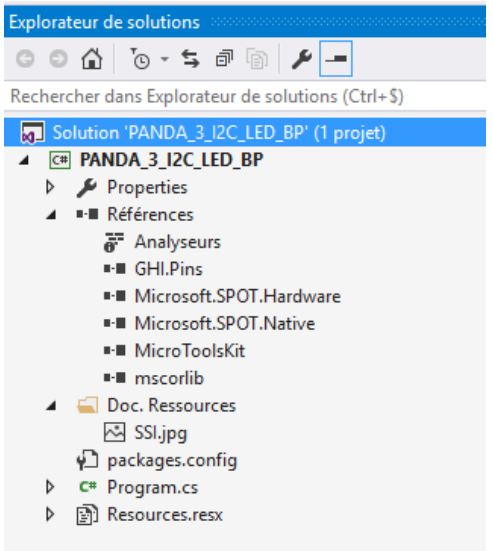
I2C : Mesurer une distance avec un télémètre à ultrasons SRF08 (Suite)

Chronogrammes

Propriété	Transaction	Chronogrammes
`I2CTelemeter.LightSensor`	Start **@SLA+W** **n°registre** Repeated Start **@SLA+R** **Donnée située en n° registre** Stop	

3.7. I2C : Recopier l'état de BP (PCF8574) sur des LED (PCF8574)





Code C#

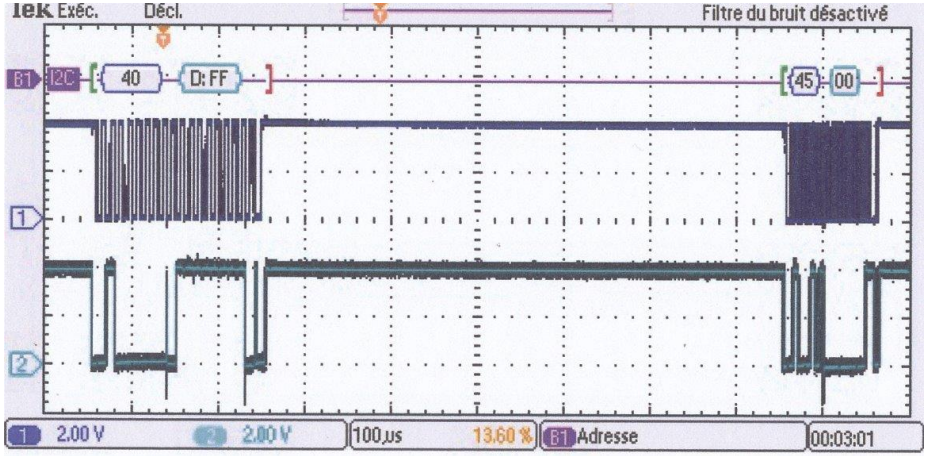
Description de la classe **PCF8574**

```
using System;
using Microtoolskit.Hardware.IO;

namespace PANDA_3_I2C_LED_BP
{
    public class Program
    {
        public static void Main()
        {
            // Paramètres du bus I2C
            byte addLeds_I2C = 0x38; // Adresse (7 bits) du PCF8574 relié aux LED
            byte addBPs_I2C = 0x3f;  // Adresse (7 bits) du PCF8574 relié aux BP
            Int16 FreqLed = 100; // Fréquence d'horloge du bus I2C en kHz
            Int16 FreqBP = 200; // Elle peut être différente pour chaque composant

           // Création d'un objet Leds
            PCF8574 Leds = new PCF8574(addLeds_I2C, FreqLed);

            // Création d'un objet BPs
            PCF8574 BPs = new PCF8574(addBPs_I2C, FreqBP);


            byte stateLED = 0xFF; // Etat initial des LED
            Leds.WriteByte(stateLED);

            while (true)
             {
                // Lecture des BPs
                stateLED = BPs.ReadByte();
                stateLED = (byte)~stateLED;
                // Ecriture sur les Leds
                Leds.WriteByte(stateLED);
             }
        }
    }
}
```

MicroToolsKit

PCF8574
Class

Méthodes
- PCF8574(ushort I2C_Add_7bits, short FreqBusI2C)
- ReadByte() : byte
- WriteByte(byte value) : void

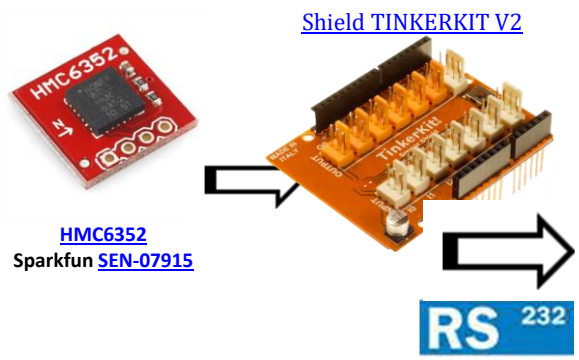


Modification de la fréquence du BUS I2C !

3.8. I2C : Afficher la direction donnée par une boussole HMC6352 sur un LCD à commandes séries

Code C#

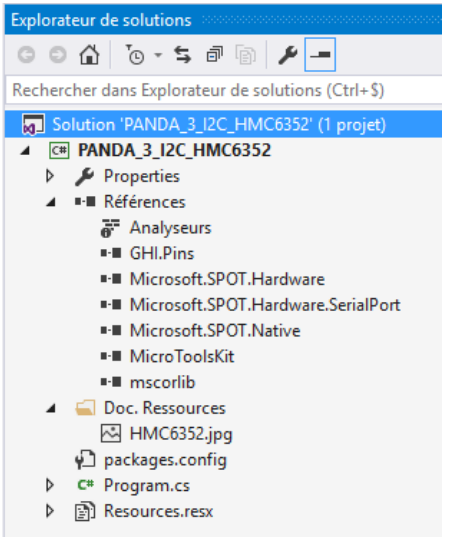
```
using System;
using System.Threading;
using Microsoft.SPOT;

using Microtoolskit.Hardware.Sensors;
using Microtoolskit.Hardware.Displays;


namespace PANDA_3_I2C_HMC6352
{
    public class Program
    {
        public static void Main()
        {// Remarque: la boussole HMC6352 dispose de résistances de tirage sur SDA et SCL !
            // Paramètres du bus I²C
            byte addCompass_I2C = 0x21; // Adresse (7 bits) du circuit HMC6352
            UInt16 Freq = 100;

            // Création d'un objet boussole HMC6352
            HMC6352 compass = new HMC6352(addCompass_I2C, Freq);

            // Création d'un objet "Afficheur LCD" connecté au COM1
            // de la carte PANDA 3 avec un convertisseur série (ELCD162)
            ELCD162 Lcd = new ELCD162("COM1");

            // Initialisation du Lcd à commandes séries
            Lcd.Init(); Lcd.ClearScreen(); Lcd.CursorOff();

            // Affichage de la version du software et de l'adresse de la boussole
            float lastHeading = (int)compass.GetHeading();
            byte ver = compass.ReadEeprom( HMC6352Compass.EEPROMAddress.SoftwareVersion);
            Debug.Print("Software Version = " + ver.ToString());
            Lcd.SetCursor(0, 0); Lcd.PutString("SofVers = " + ver.ToString());
            byte i2cAddr = compass.ReadEeprom(HMC6352Compass.EEPROMAddress.SlaveAddress);
            Debug.Print("Slave Address = " + i2cAddr.ToString());
            Lcd.SetCursor(0, 1); Lcd.PutString("2*sla + /W = " + i2cAddr.ToString());

            HMC6352.OperationalMode mode = compass.GetOperationalMode();
            Debug.Print("Operational Mode = " + mode.ToString());

            HMC6352.Frequency frequency = compass.GetFrequency();
            Debug.Print("Frequency = " + frequency.ToString());
            Thread.Sleep(5000);
```



Shield TINKERKIT V2

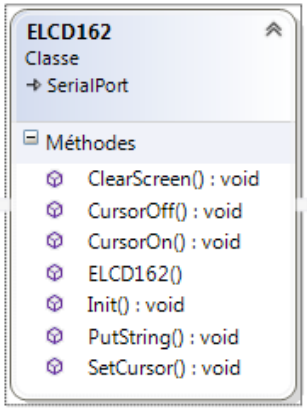
[HMC6352](#)
Sparkfun [SEN-07915](#)

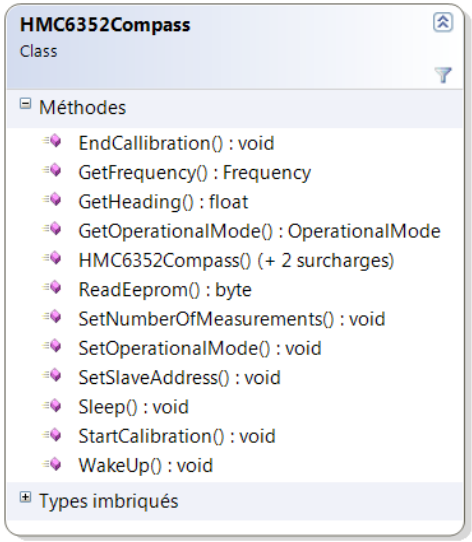
Afficheur 2x16 équipé d'un
[module ELCD162 COMFILE Série](#) (19200b/s)



Matériels

- Carte PANDA III
- Sensor [Shield TINKERKIT V2](#)
- Hub I²C
- HMC6352
- LCD + Module COMFILE Série





[MicroToolsKit](#)

Description des classes
HMC6352 et [ELCD162](#)

I2C : Afficher la direction donnée par une boussole HMC6352 sur un LCD à commandes séries

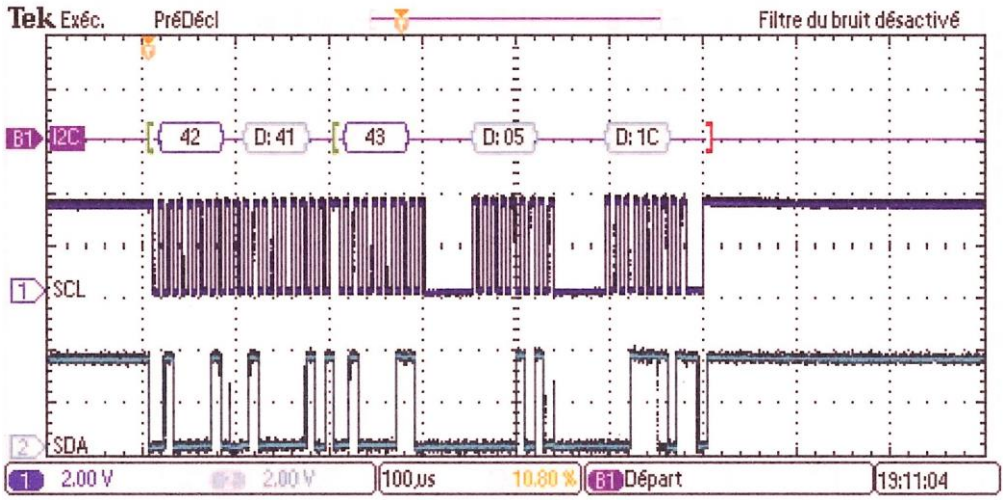
```
            while (true)
            {
                Thread.Sleep(500);

                float heading = compass.GetHeading();
                if (heading != lastHeading)
                {
                    Lcd.ClearScreen();
                    Lcd.PutString("Dir: " + heading.ToString(“N1));
                    lastHeading = heading;
                    Debug.Print(heading.ToString"N2"));
                }
            }
        }
    }
}
```

Utilisation du debugger **pour visualiser la communication avec la boussole.**

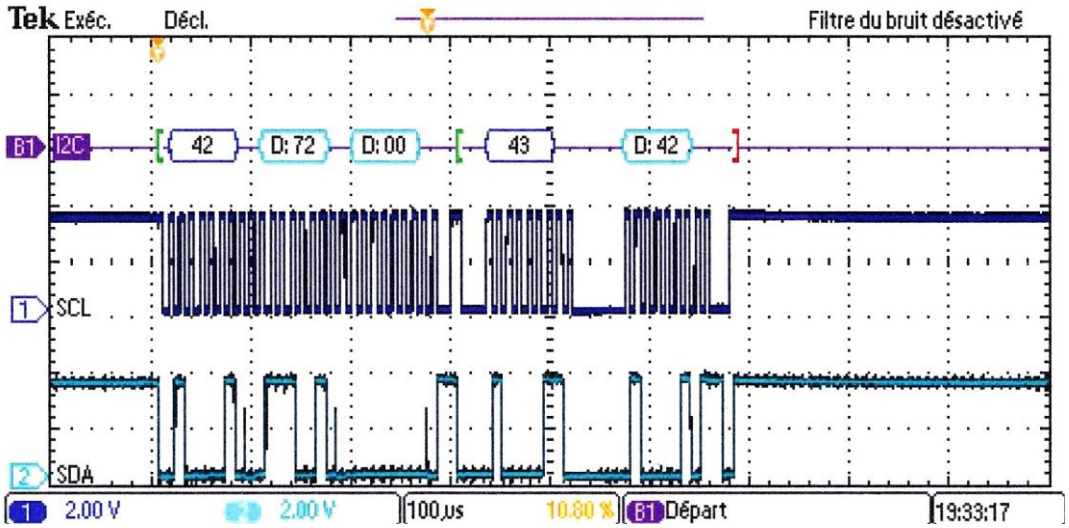
Pas à pas
F10 : Step Over

```
Output
Show output from: Debug
The thread '<No Name>' (0x2)
Software Version = 5
Slave Address = 68
Operational Mode = 0
Frequency = 2
2048
140.199997
140
140
140.800003
140.100006
140.399994
140.399994
140
140
```

Première **mesure** erronée !



Le µC demande à lire l'angle mesuré par la boussole (commande **41h = 'A')**
La boussole répond $051C_{(16)} = 1308_{(10)} = 130{,}8°$

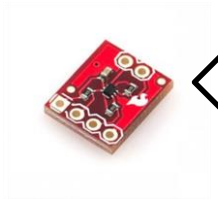


Le µC demande l'adresse de la boussole (commande **72h = 'r' 00h = 0)** .La boussole répond **42h : SLA+W. Son adresse SLA est donc 21h**

3.9. I2C : Mesurer la température ambiante avec un capteur TMP102

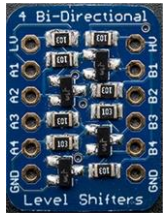
Code C#

```
using System;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.FEZ;
using Sensor;

namespace TMP102_Tester
{
   public class Program
   {
      public static void Main()
      {
         TMP102 CPTTEMP102 = new TMP102();
         CPTTEMP102.Init(TMP102.ADD0.Gnd);

         while (true)
         {
            CPTTEMP102.Read();
            Debug.Print("Temperature: " + CPTTEMP102.asCelcius() + " C");

            // Sleep for 1000 milliseconds
            Thread.Sleep(1000);
         }
      }
   }
}
```



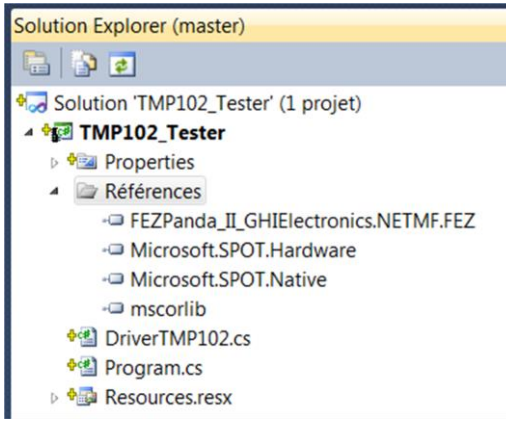
[TMP102]
Sparkfun [SEN-11931]

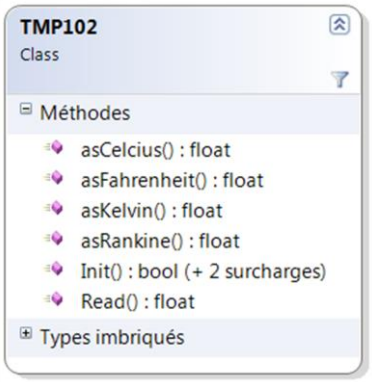
Alimentation: 1,4 à 3,6 Vcc
Consommation: 10 µA maxi (1 µA en veille)
Plage de mesure: -40°C à +125°C
Précision: 0,5 °C (de -25°C à +85°C)
Résolution: 0,0625°C
Interface série 2 fils
Dimensions: 13 x 10 mm

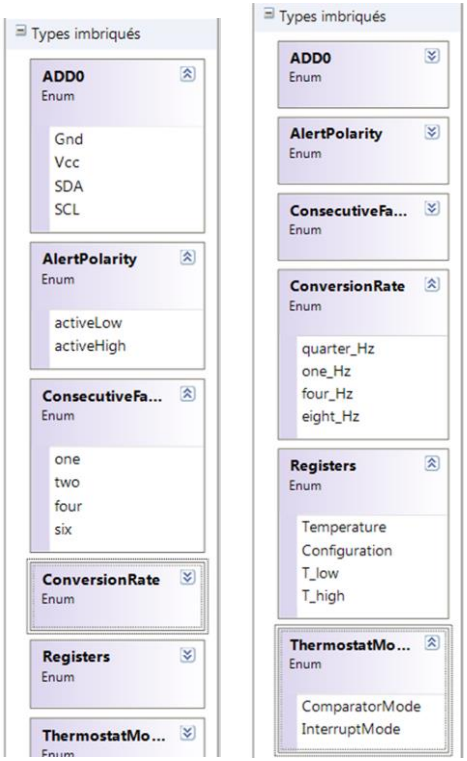


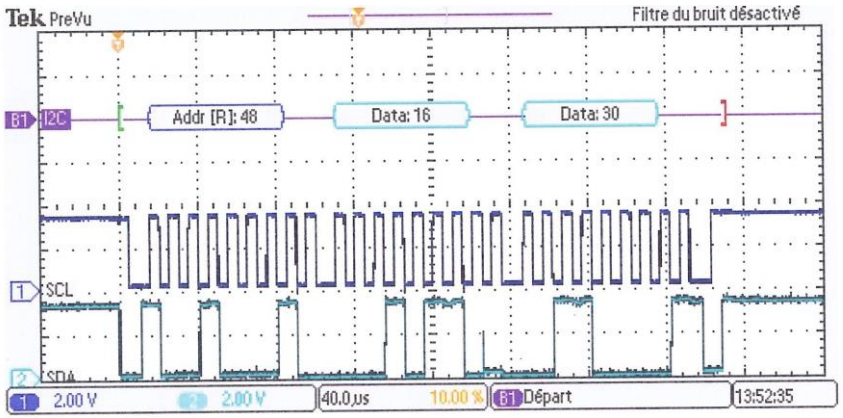
Convertisseur
5V <-> 3.3V
(Adafruit BSS138)

PANDA2









[Description de la classe **TMP102**]

3.10. I2C : Commander deux motoréducteurs, équipés d'encodeurs, avec une carte Devantech MD25

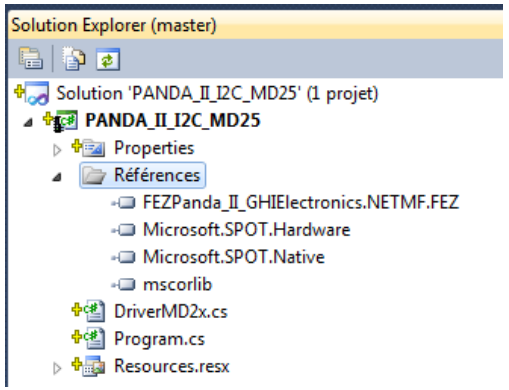
Code C# de l'exemple 20

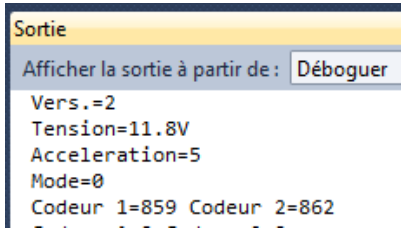
```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.FEZ;
using MD2x;

namespace PANDA_II_I2C_MD25
{
    public class Program
    {
        public static void Main()
        {
            // Programme de test de la carte MD23/MD25
            // Création d'un objet MotorControl (carte MD23 ou MD25)
            // avec l'adresse 0x58 et la fréquence de bus F = 100kHz
            MotorControlMD2x CarteMD23 = new MotorControlMD2x();

            // Lecture et affichage de certains registres de la carte
            Debug.Print("Vers.=" + CarteMD23.SoftRev.ToString());
            Debug.Print("Tension="+((Single)CarteMD23.Battery/10).ToString("N1")+"V");
            Debug.Print("Acceleration=" + CarteMD23.AccelerationRate.ToString());
            Debug.Print("Mode=" + CarteMD23.Mode.ToString());


            while (true)
            {
                // Essai 1: Rotation des moteurs pendant 2s
                // ----------------------------------------------------------------
                CarteMD23.SetSpeedTurn(140, 140); // Réglage de la vitesse des moteurs
                Thread.Sleep(2000); // Rotation pendant 2s
                // Lecture des registres et affichage de la valeur des codeurs
                Debug.Print("Codeur 1=" + CarteMD23.Encoder1.ToString() + " " + "Codeur 2=" + CarteMD23.Encoder2.ToString());
                CarteMD23.StopMotor(); // Arrêt des moteurs
                Thread.Sleep(3000);

                // Essai 2 : Rotation des moteurs jusqu'à ce que la distance recherchée soit atteinte
                // ----------------------------------------------------------------
                CarteMD23.RazEncoders(); // Remise à zéro des codeurs
                CarteMD23.SetSpeedTurn(140, 140); // Réglage de la vitesse des moteurs
                while (CarteMD23.Encoder1 < 2000)
                {
                    Debug.Print("Codeur 1=" + CarteMD23.Encoder1.ToString() + " " + "Codeur 2=" + CarteMD23.Encoder2.ToString());
                }
                CarteMD23.StopMotor();  // Arrêt des moteurs
                Thread.Sleep(5000);
            }
        }
    }
}
```

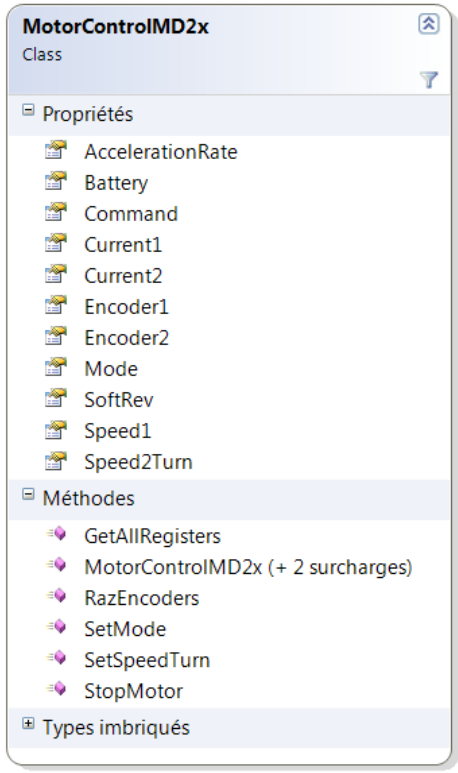
[Description de la classe MotorControlMD2x]

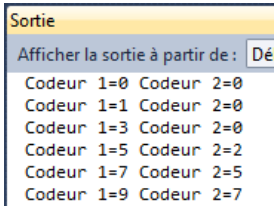
ATTENTION : les fichiers à télécharger sont destinés à une carte **Netduino plus 2**.

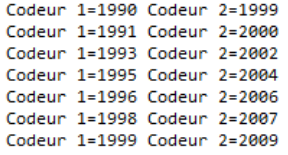
[MD25] : [Ensemble de pilotage pour moteurs "DCM2"]





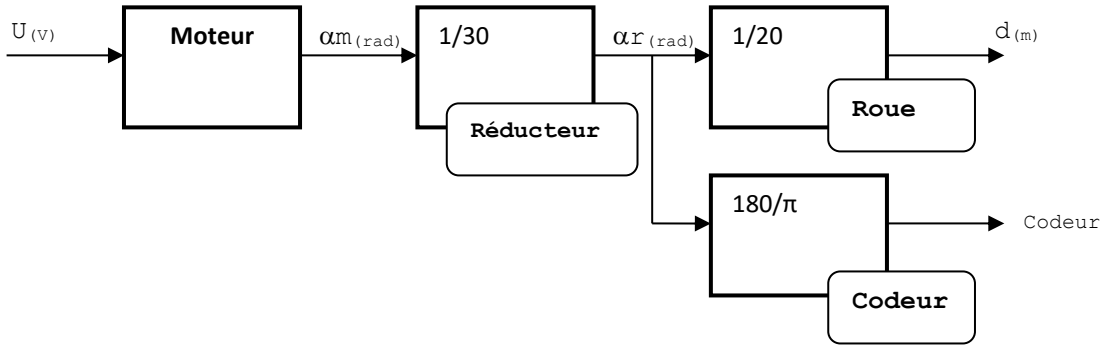




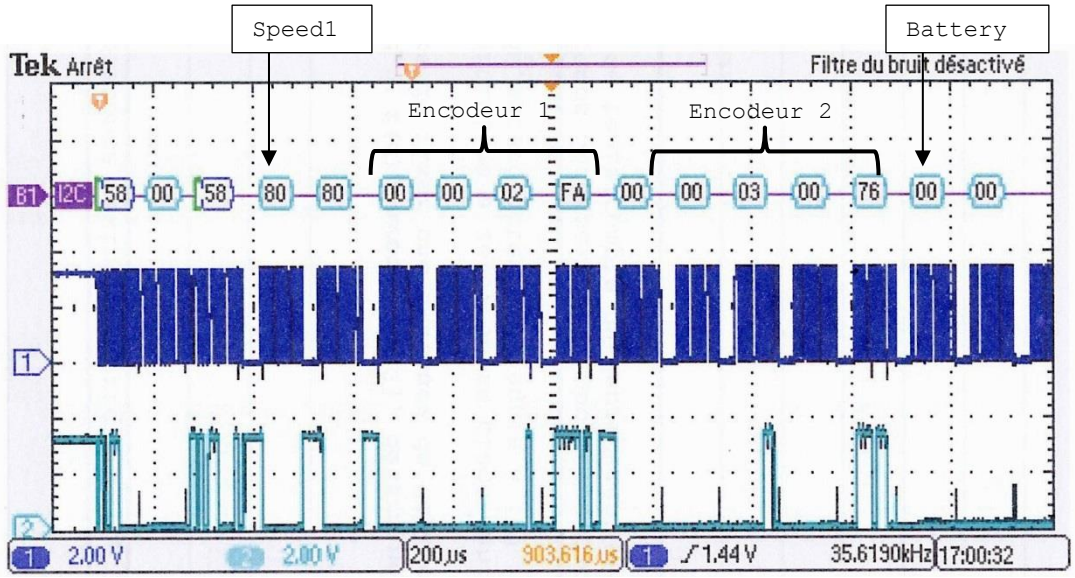


Explications détaillées : French Beginner Guide ebook1.03 (8/2010)
(Chapitre 17: Interfaces séries /17.3 I2C)

I2C : Commander deux motoréducteurs, équipés d'encodeurs, avec une carte Devantech MD25 (suite)



Lecture des registres de la carte MD23/MD25(partielle)



3.11. I2C : Mesurer la luminosité ambiante avec un capteur TSL2561

Code C#

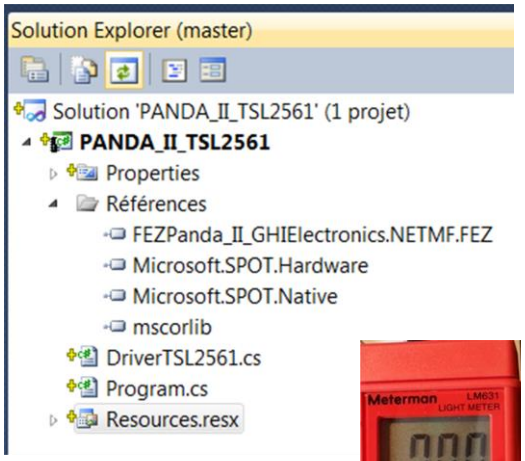
```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.FEZ;
using LightSensor;
namespace PANDA_II_TSL2561
{
    public class Program
    {
        public static void Main()
        {   // Création d'un objet TSL2561 (carte MD23 ou MD25)
            // avec l'adresse 0x29 et la fréquence de bus F = 100kHz
            TSL2561 I2CLightSensor = new TSL2561();
            I2CLightSensor.Init(TSL2561.Gain.x1,TSL2561.IntegrationTime._13MS);
            // Réglage des seuils
            I2CLightSensor.threshLowhigh = 0x20; I2CLightSensor.threshLowLow = 0x10;
            I2CLightSensor.threshhighhigh = 0x40; I2CLightSensor.threshhighLow = 0x20;
            TSL2561 I2CLightSensor = new TSL2561();
            I2CLightSensor.Init(TSL2561.Gain.x1,TSL2561.IntegrationTime._13MS);
            // Réglage des seuils
            I2CLightSensor.ThreshLowHigh = 0x20; I2CLightSensor.ThreshLowLow = 0x10;
            I2CLightSensor.ThreshHighHigh = 0x40; I2CLightSensor.ThreshHighLow = 0x20;

            while (true)
            { // Affichage du contenu des registres
                Debug.Print("Lecture des registres");
                Debug.Print("---------------------");
                Debug.Print("00h Control              : " + I2CLightSensor.Control);
                Debug.Print("01h Timing               : " + I2CLightSensor.Timing);
                Debug.Print("02h ThreshLowLow         : " + I2CLightSensor.ThreshLowLow);
                Debug.Print("03h ThreshLowhigh        : " + I2CLightSensor.ThreshLowHigh);
                Debug.Print("04h ThreshhighLow        : " + I2CLightSensor.ThreshHighLow);
                Debug.Print("05h Threshhighhigh       : " + I2CLightSensor.ThreshHighHigh);
                Debug.Print("06h Interrupt            : " + I2CLightSensor.Interrupt);
                Debug.Print("0Ah Part number / Rev Id : " + I2CLightSensor.Id);
                Debug.Print("--------------------------");
                Debug.Print("Valeurs des cannaux 0 et 1");
                Debug.Print("--------------------------");
                Debug.Print("Canal 0: " + I2CLightSensor.Channel0); Debug.Print("Canal 1: " + I2CLightSensor.Channel1);
                if (I2CLightSensor.CalculateLux()!=0){
                     Debug.Print("Luminosité: " + I2CLightSensor.CalculateLux()+"lux");

                } Else {
                    Debug.Print("Sensor overload");}
                Thread.Sleep(1000);
            }
        }
    }
}
```

github SOCIAL CODING

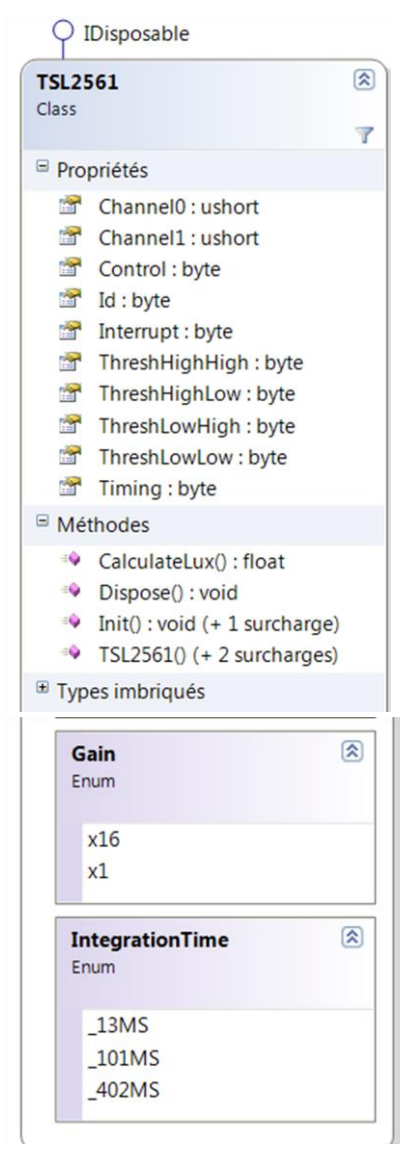
Description de la classe TSL2561

ATTENTION : les fichiers à télécharger sont destinés à une carte **Netduino plus 2**.

TSL2561
(Grove SEN-10171P)




```
Lecture des registres
---------------------
00h Control              : 3
01h Timing               : 0
02h ThreshLowLow         : 16
03h ThreshLowhigh        : 32
04h ThreshhighLow        : 32
05h Threshhighhigh       : 64
06h Interrupt            : 0
0Ah Part number / Rev Id : 17
--------------------------
Valeurs des cannaux 0 et 1
--------------------------
Canal 0: 185
Canal 1: 90
Luminosité: 729lux
```

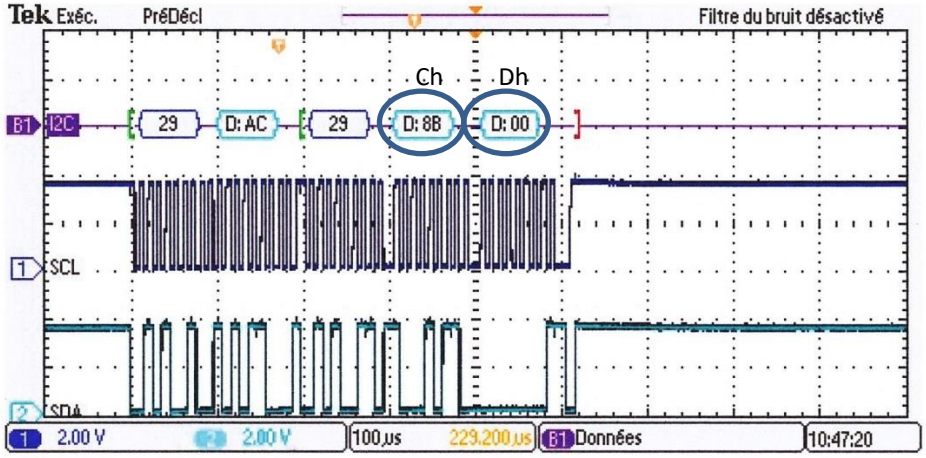


Explications détaillées : French Beginner Guide ebook1.03 (8/2010)
(Chapitre 17: Interfaces séries /17.3 I2C)

Mesurer la luminosité ambiante avec un capteur TSL2561 (suite)

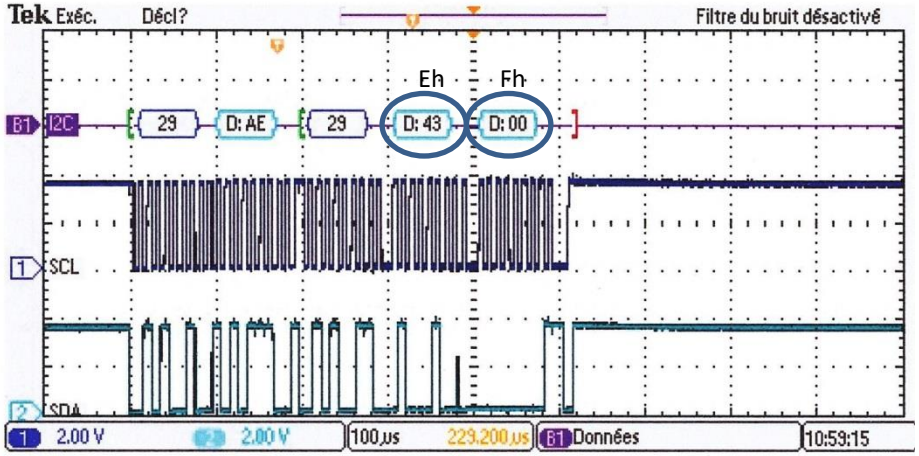
Lecture du canal 0 en mode mot : registres Ch (**Low byte**) et Dh (**High byte**)

Canal 0 = 008Bh



Lecture du canal 1 en mode mot : registres Eh (**Low byte**) et Fh (**High byte**)

Canal 1 = 0043h

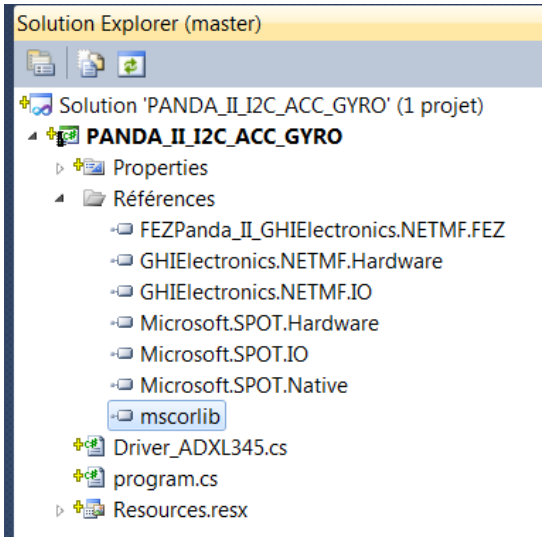


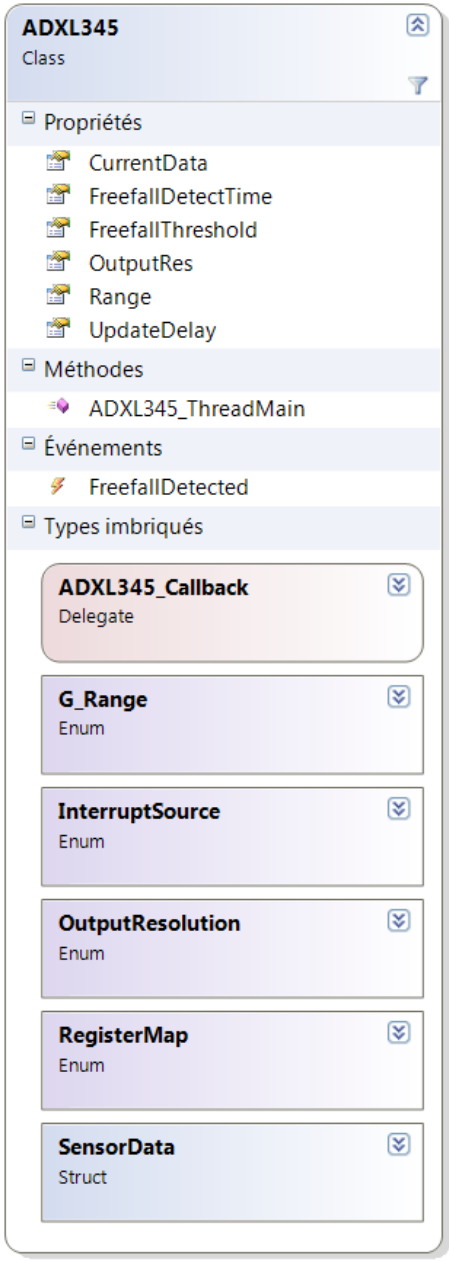
La méthode **calculateLux()** renvoie 420 lux pour ces valeurs des canaux.

3.12. I2C : Accéléromètre ADXL345 + Gyroscope ITG3200

ADXL345 + ITG3200

(Sparkfun SEN-10121)





C#

```
using System;
using System.IO;using System.IO.Ports;
using System.Text;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.IO;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.Hardware;
using GHIElectronics.NETMF.FEZ;
using GHIElectronics.NETMF.IO;

using Heffsoft.Sensors.Accelerometer;

namespace FlightComputer
{
    public class Program
    {
        public static void Main()
        {
            // Disable the GC Debug messages during output
            Debug.EnableGCMessages(false);
            Thread.Sleep(100);

            // Create the I2C bus device
            I2CDevice i2c_bus = new I2CDevice(null);

            // Create the accelerometer driver instance
            ADXL345 accelSensor = new ADXL345(i2c_bus,
                new InterruptPort((Cpu.Pin)FEZ_Pin.Interrupt.Di2, false, Port.ResistorMode.Disabled,
                Port.InterruptMode.InterruptEdgeHigh), 400,false);

            // Set the sensor to fixed resolution, updating every 10ms
            accelSensor.OutputRes = ADXL345.OutputResolution.FixedResoultion;
            accelSensor.UpdateDelay = 10;

            // Set a callback for the free fall detection
            accelSensor.FreefallDetected += new ADXL345.ADXL345_Callback(accelSensor_FreefallDetected);
```

Non testé

I2C : Accéléromètre + Gyroscope ADXL345 (suite)

```
        // Loop forever
            while(true)
            {
                // Read the data from the driver into a SensorData structure
                ADXL345.SensorData data = accelSensor.CurrentData;

                // Write the information out the debug port
                Debug.Print("Accel X:" + data.X + " Y:" + data.Y + " Z:" + data.Z + "\r\n");

                // Wait around a bit
                Thread.Sleep(100);
            }
        }

        private static void accelSensor_FreefallDetected(ADXL345 sender)
        {
            // We are in free fall, better tell someone
            Debug.Print("Free Fall Detected!");
        }
    }
}
```

Description de la classe ??????

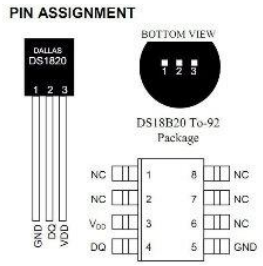
ATTENTION : les fichiers à télécharger sont destinés à une carte **Netduino plus 2**.

Explications détaillées : French Beginner Guide ebook1.03 (8/2010)
(Chapitre 17: Interfaces séries /17.3 I2C)

3.13. Un fil : Mesurer la température ambiante avec un capteur DS18B20 (OneWire)

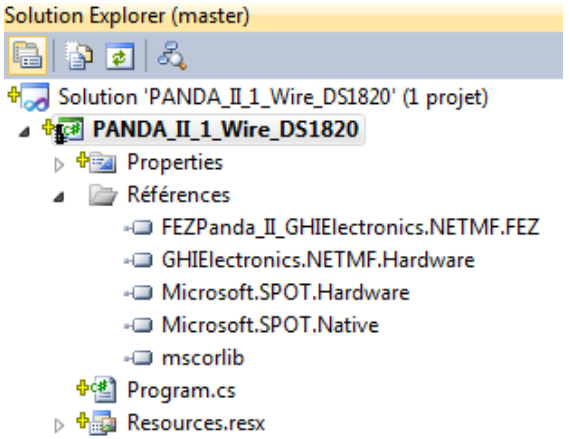
Code C#

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using GHIElectronics.NETMF.Hardware;
using GHIElectronics.NETMF.FEZ;

namespace PANDA_II_1_Wire_DS18B20
{
    public class Program
    {
        public static void Main()
        {
            // Change this your correct pin!
            Cpu.Pin myPin = (Cpu.Pin)FEZ_Pin.Digital

            OneWire ow = new OneWire(myPin);
            ushort temperature;

            // read every second
            while (true)
            {
                if (ow.Reset())
                {
                    ow.WriteByte(0xCC);     // Skip ROM, we only have one device
                    ow.WriteByte(0x44);     // Start temperature conversion

                    while (ow.ReadByte() == 0) ;   // wait while busy

                    ow.Reset();
                    ow.WriteByte(0xCC);     // skip ROM
                    ow.WriteByte(0xBE);     // Read Scratchpad

                    temperature = (byte)ow.ReadByte();           // LSB
                    temperature |= (ushort)(ow.ReadByte() << 8); // MSB

                    Debug.Print("Temperature: " + temperature / 16);
                    Thread.Sleep(1000);
                }
                else
                {
                    Debug.Print("Device is not detected.");
                }

                Thread.Sleep(1000);
            }
        }
    }
}
```



[DS18B20](#)
[DFRobot](#)

One Wire



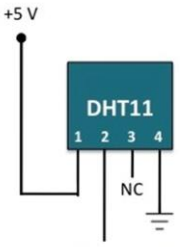

```
...public OneWire(Cpu.Pin pin);

...public static byte CalculateCRC(byte[] buffer, int offset, int count);
...public static ushort CalculateCRC16(byte[] buffer, int offset, int count, ushort seed);
...public void Dispose();
...public void Read(byte[] buffer, int offset, int count);
...public byte ReadBit();
...public byte ReadByte();
...public bool Reset();
...public bool Search_GetNextDevice(byte[] romRegistrationNumber);
...public bool Search_IsDevicePresent(byte[] romRegistrationNumber);
...public void Search_Restart();
...public void Write(byte[] buffer, int offset, int count);
...public void WriteBit(byte value);
...public void WriteByte(byte value);
```

Explications détaillées : **French Beginner Guide ebook1.03 (8/2010) (Chapitre 17: Interfaces séries /17.4 OneWire)**

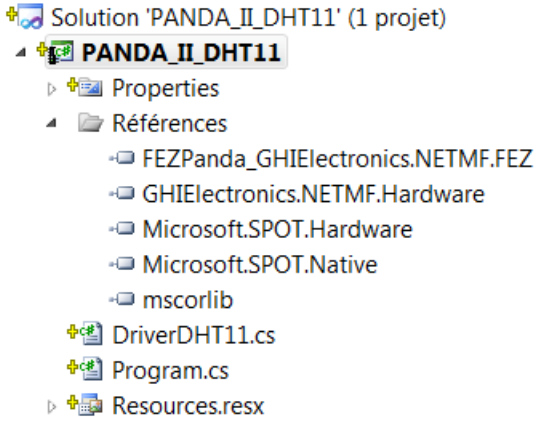
3.14. Un fil : Mesurer la température et l'humidité ambiantes avec un capteur DHT11 (1 fil spécifique non compatible OneWire)

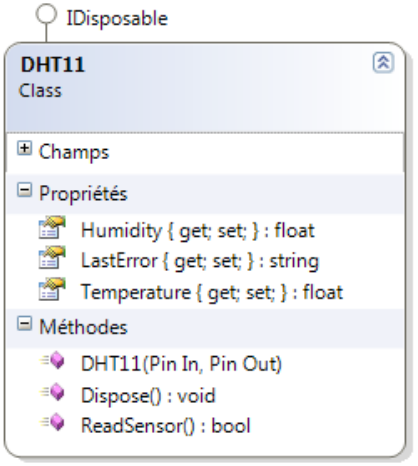
Code C#

```
using System;
using System.Threading;

using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.FEZ;
using GHIElectronics.NETMF.Hardware;

namespace PANDA_II_DHT11
{
    public class Program
    {
        public static void Main()
        {
            DHT11 MyDHT11 = new DHT11((Cpu.Pin)FEZ_Pin.Digital.Di6, (Cpu.Pin)FEZ_Pin.Digital.Di7);

            while (true)
            {
                if (MyDHT11.ReadSensor())
                {
                    Debug.Print("Temperature = " + MyDHT11.Temperature.ToString() + "°C");
                    Debug.Print("Humidity    = " + MyDHT11.Humidity.ToString() + "%");
                }
                else Debug.Print("DHT11 Error : " + MyDHT11.LastError);

                Thread.Sleep(10000);
            }
        }

    }
}
```



DHT11
Module Groove SEN11301P





Eviter de câbler le capteur avant d'avoir programmé la carte.

Web

HTTP: Page Web embarquée dans la carte FEZ-PANDA2

Explications détaillées : .Net & Internet of Things (Chapitre 7 : FEZ - HTTP)
Fichier source du projet dans le répertoire : FEZ_PANDA II_Reseaux\ 5_TestConnectEthernet_HTTP

Client

Philippe-Maison

SERVEUR
Connexion : adresse statique

FEZ PANDA II + CONNECT Shield

Câble Ethernet

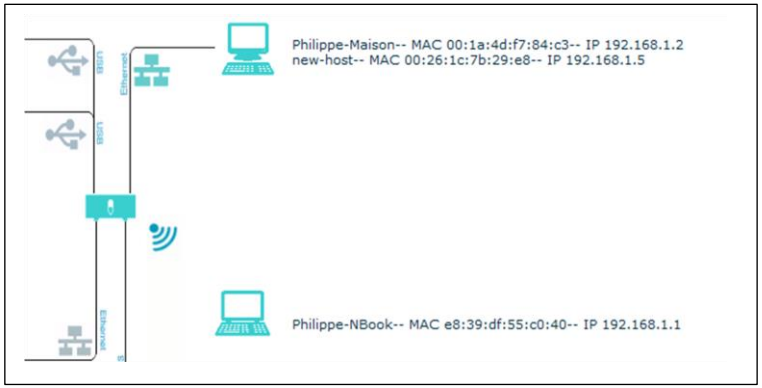
CPL

Visualisée sur

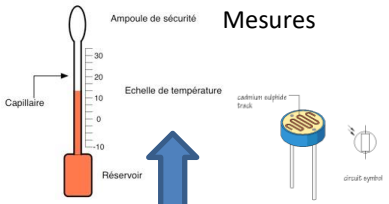
Visualisée sur

Client

Philippe-NBook



HTTP : Objets connectés



Mesures

SERVEUR

Connexion : DHCP
Lecture des capteurs
Transfert vers ThingSpeak
Affichage dans WebGE

FEZ PANDA II + CONNECT Shield

Câble Ethernet

Mise en forme

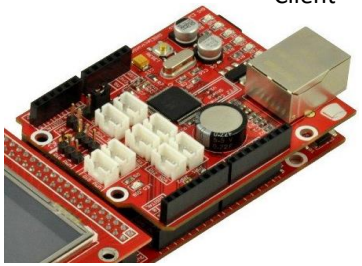
ThingSpeak

Site stockage et mise en forme des données

Internet

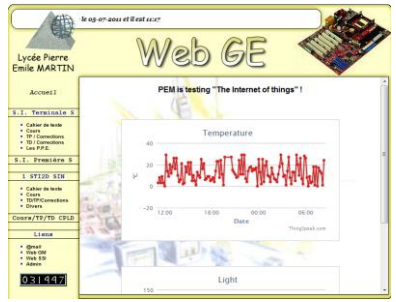
CPL

Client



Philippe-Maison

Présentation



[I2D SIN -> Divers]

Visualisée sur

Philippe-NBook Client

Explications: **.Net & Internet of Things (Chapitre 10 Sensor Monitoring)**

Fichier source **du projet dans le répertoire : FEZ_PANDA II_Reseau\6_InterOfThings**

SMTP : Envoi de l'état d'un capteur simulant l'état d'une porte par mail / sms

Explications : **.Net & Internet of Things (Chapitre 12 You've got mail and SMS)**

Fichier source du projet **dans le répertoire FEZ_PANDA II_Reseau \7_SendEmail**

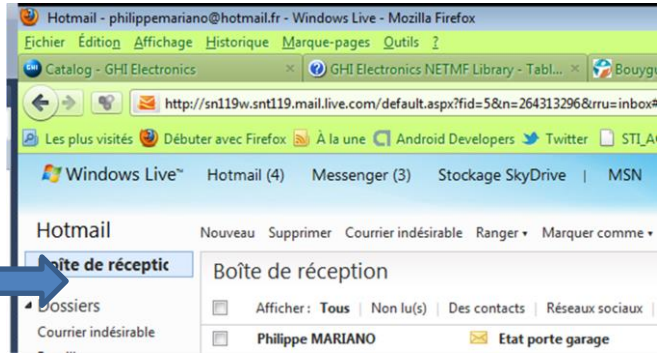
SERVEUR

Connexion : DHCP
Lecture état porte de garage (bouton-poussoir)
Envoi d'un mail à philippemariano@hotmail.fr

FEZ PANDA + CONNECT Shield

Câble Ethernet

Internet



CPL

Consulté sur

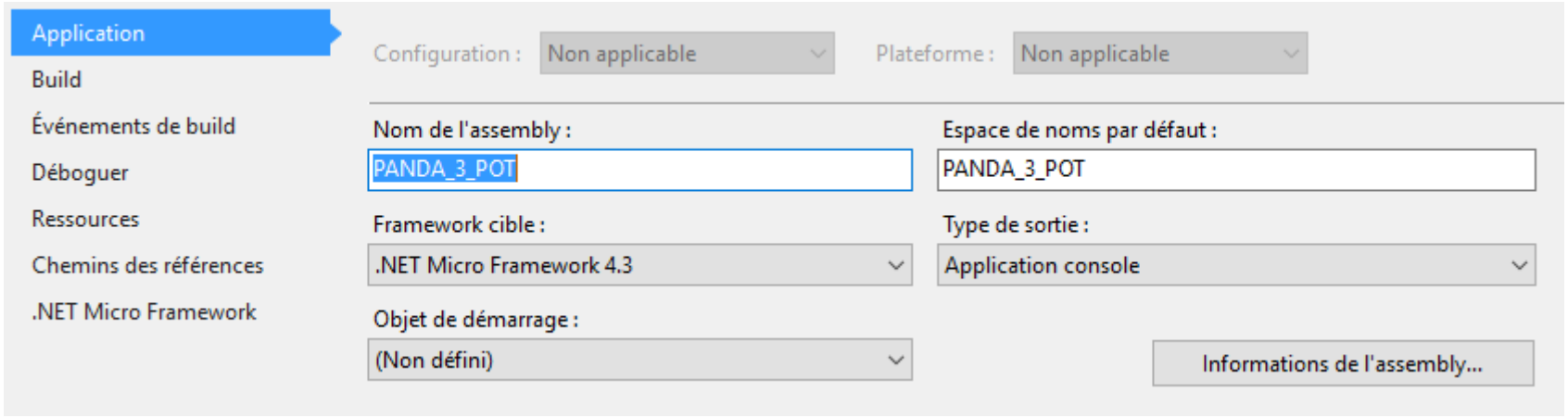
Client

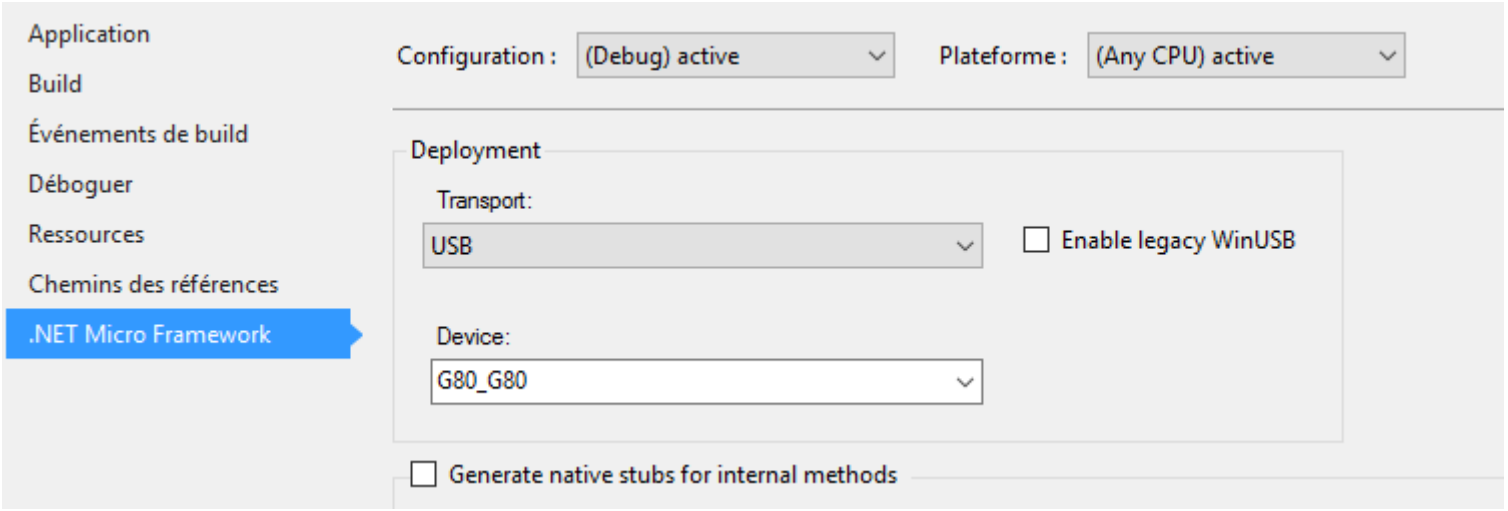
Philippe-Maison

Philippe-NBook Client

Annexes

A1 - Configuration des projets dans Visual studio 2015





A2 – Namespaces GHI

Namespaces

https://goo.gl/LvQOIw

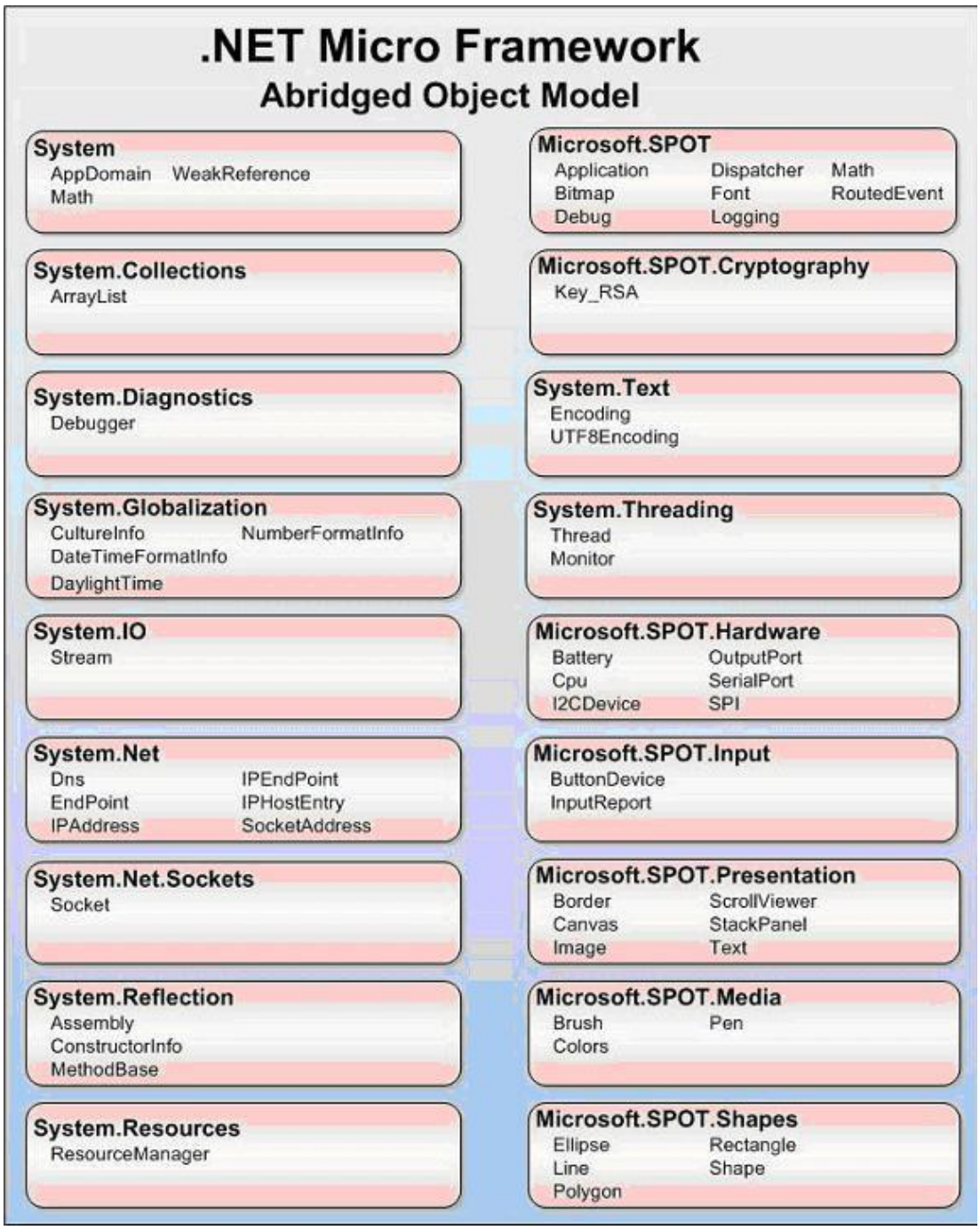
GHI Electronics NETMF v4.3 library for **.NET Micro Framework** provides extensions to the available Micro Framework assemblies. These extensions are specific to GHI Electronics related products.
This library provides the user with greater ability to have more innovative and flexible products. The additional featrures include but not limited to:

- **USB Host Driver:** This is where you can connect USB devices to your platform and use them on low and high level.
  To get started see `GHI.Usb.Host.Controller`. Some of the supported devices are:
  - USB Mass Storage.
  - USB HID Devices: Mouse, Keyboard, Joystick, etc.
  - Printers.
  - Serial Devices.
  - and many more...
- **USB Client Driver:** USB Client (USBC known as Device or Slave), is where your .Net Micro Framework device becomes a USB slave and connected to a master host such as PC running Windows. The USB Client can be used for debugging and deployment of applications through Microsoft Visual Studio but with this driver you can change how it works.
  To get started see `GHI.Usb.Client.Controller`
- **RLP (Runtime Loadable Procedure):** Provides the ability to execute native C/Assembly code.To get started see `GHI.Processor.RuntimeLoadableProcedures`.
- **Hardware Extensions:** Provides more hardware capabilities. For example:
  - Register access.
  - **SD Card Driver:** To get started see `GHI.IO.Storage.Removable`.
- **and many more...**

Namespace	Description	Namespace	Description
GHI.Glide	**Contains the core functionality.**	GHI.Processor	**Provides access to some of the available**
GHI.Glide.Display	**Contains the core classes that Glide uses to build visual displays.**	GHI.SQLite	**Databae access using SQLite.**
GHI.Glide.Geom	**Contains geometry classes, such as points and rectangles.**	GHI.Usb	**USB Host and Client access.**
GHI.Glide.UI	**Contains component classes such as Button, Image and ComboBox.**	GHI.Usb.Client	**USB client access.**
GHI.IO	**Provides access to some of the available I/O and bus services.**	GHI.Usb.Descriptors	**Descriptors for USB devices.**
GHI.IO.Storage	**Provides access to some of the available storage services.**	GHI.Usb.Host	**USB host access.**
GHI.Networking	**Supports network-related items.**	GHI.Utilities	**Helper utilities and methods.**
GHI.Pins	**Provides pin definitions for the System on Modules.**	System	

A3 - API Reference for .NET Micro Framework 4.3

https://goo.gl/dIvfxy

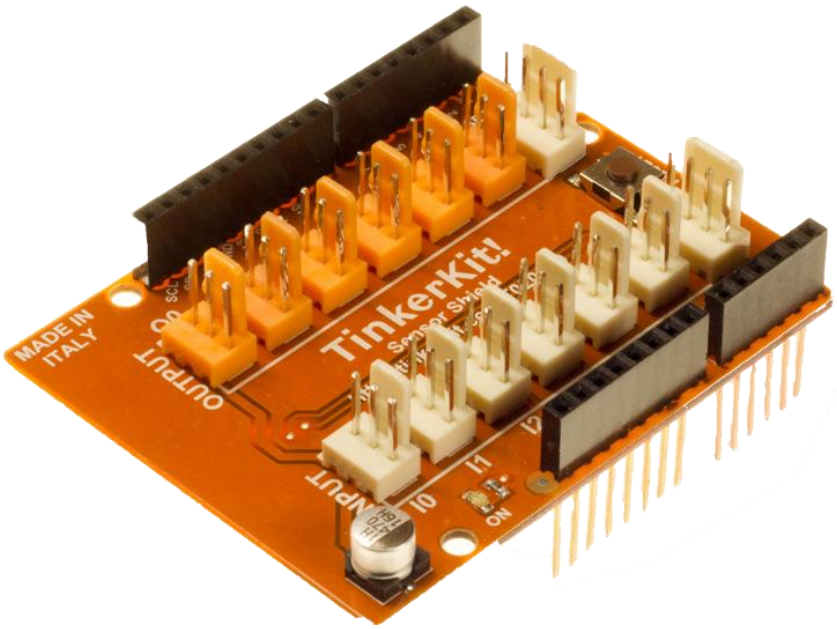


A4 – Les types reconnus par Microsoft Visual Studio et le .NET Microframework

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

A5 – Le 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.**

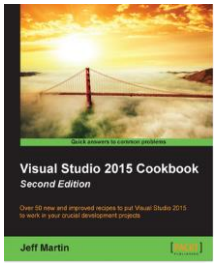
Module description: A green LED signals that the shield is correctly powered, a standard 6mm pushbutton allows you to RESET the board.

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.

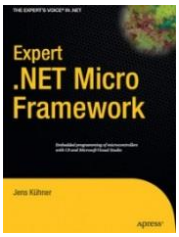
Bibliographie

PDF téléchargeables



Visual Studio 2015

http://it-ebooks.info/book/1472846854/



Expert .NET Micro Framework

http://it-ebooks.info/book/2053/

Webographie

Matériels, documentation, liens pour le téléchargement des outils logiciels

GHI ELECTRONICS (FEZ PANDA et Gadgeteers)

https://www.ghielectronics.com/

Projets, Communauté

GHI ELECTRONICS (FEZ PANDA et Gadgeteers)

https://www.ghielectronics.com/community

Codeplex Project Hosting for Open Source Software

https://www.codeplex.com/

GitHub

https://github.com/

Technologie .NET

Microsoft C# Express (téléchargement)

http://msdn.microsoft.com/fr-fr/express/aa975050

Centre de développement Visual C# (Ressources)

http://msdn.microsoft.com/fr-fr/vcsharp/aa336706

NETMF (Ressources, Téléchargement)

http://www.netmf.com/

Débuter en C#

http://fr.openclassrooms.com/informatique/cours/apprenez-a-developper-en-c/creer-un-projet-avec-visual-c-2010-express
http://fr.openclassrooms.com/informatique/cours/apprenez-a-developper-en-c/creer-un-projet-avec-visual-c-2010-express
http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-c-sur-net

Le Blog NETMF(Actualités)

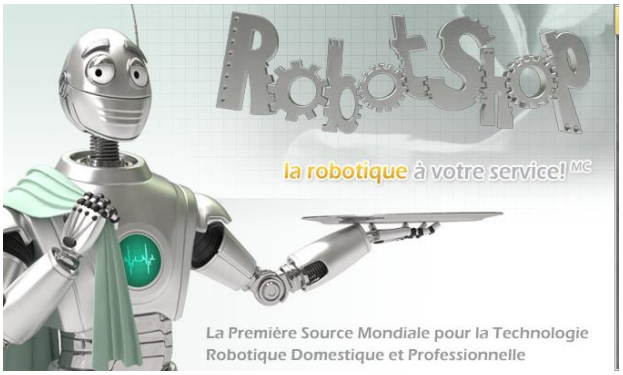
http://blogs.msdn.com/b/netmfteam/

Distributeurs

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