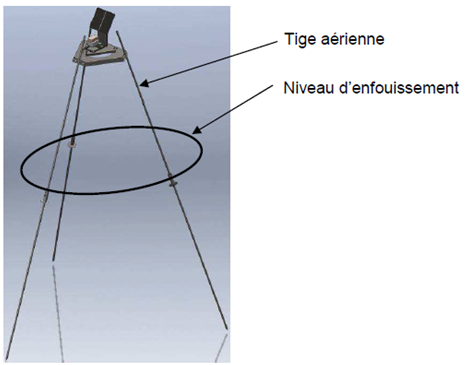
VAGUE2

**Lycée PE MARTIN**

**Bourges**

|  |  |
| --- | --- |
|  | CENTRE D’INTERET  **CI2 : Expérimenter et mesurer sur un système** |

# *Sujet du TP*



**Télémètre**

|  |
| --- |
| **bd14980_** TP de découverte.  **bd14980_** Application, mise en œuvre de savoirs/savoir-faire  **bd14980_** Recherche et validation de solutions.  **bd14980_** Evaluation.  C:\Users\Philippe\Desktop\imgres.jpg |
|
| Capacités Associer un modèle aux composants d’une chaine d’information et d’une chaîne d’énergie.  Traduire le comportement d’un système. |

**Compétences**

|  |  |
| --- | --- |
| *bd21301_ Associer un modèle à un système ou à son comportement* | *§B* |
| *bd21301_ Préciser ou justifier les limites de validité du modèle envisagé* | *§B, §C* |

**Problématique**

***Programmer la fonction « Traiter » d’un système de mesure d’une distance.***

**Conditions de déroulement de l’activité**

|  |  |  |
| --- | --- | --- |
| ***Phases de travail*** | **Objectif** | **Activité** |
| ***A) Mise en situation*** | ***Analyser*** *le système.* | *Associer un modèle aux composants d’une chaine d’information et d’une chaîne d’énergie. (Analyser la solution de l’exercice «****Mesure de la hauteur de marée en mangrove****»)* |
| ***B) Etude de la problématique*** | ***Programmer*** *la fonction « Traiter » d’un système de mesure d’une distance.* | *Description du comportement attendu avec un algorithme. Programmation d’une carte à microcontrôleur.* |
| ***C) Synthèse*** | ***Apporter*** *de nouvelles fonctionnalités à la fonction « Traiter ».* | *Modification d’un algorithme et programmation.* |

**RESSOURCES DOCUMENTAIRES, LOGICIELS ET MATERIELS**

Carte à microcontrôleur **Panda 2**, 1 potentiomètre (E-Block), 1 buzzer (E-Block), 1 LCD 2x16 équipé d’un module de pilotage série ELCD COMFILE. Logiciel Microsoft Visual Studio 2010 pro(C#) ou version Express



Document« **Guide du débutant pour C# et .NET Micro Framework**»

Répertoire du projet « Telemetre »

**Cours / TD** : Généralités sur les systèmes à microcontrôleur. TD1, TD2 et TD3 sur le type des variables.

**Cours / Exercice** : Généralités surles capteurs et la chaîne de mesure.

**A) Mise en situation** **🕐 :** **Durée** conseillée

**A1) Présentation**

|  |  |
| --- | --- |
|  | Le travail demandé dans ce TP fait suite à l’exercice « **Mesure de la hauteur de marée en mangrove** » traité dans le cours « **Généralités sur les capteurs et la chaîne de mesure** ».  Vous devez utiliser les résultats que vous avez obtenus ou reprendre la correction située dans le répertoire du TP.  Pour placer le répertoire du TP sur le bureau du PC. |

**A2) Matériel**

On souhaite **tester** l’algorithme **« Hauteur**» avec une carte **PANDA2.** Un afficheur **LCD** permettra de visualiser le résultat du traitement. Un potentiomètre simule la tension U image de la hauteur.

|  |  |
| --- | --- |
|  | **Connecteur PANDA2\*** |
| **U(V)** | **An3** |
| **Hauteur(cm)** | **Di1** |

****

**Potentiomètre**



**U(V**)

**Hauteur(cm)**

**PANDA 2**

**B) Etude de la problématique** [🕐 1h]

**B1) Algorithme**

L’algorithme a été écrit pendant le cours. Reprenez votre travail ou la correction.

**B2) Programme C# et test**



**B21) Prise en main de Microsoft Visual Studio (si nécessaire)**

Suivez la démarche proposée dans le document ressource « **Guide du débutant pour C# et .NET Micro**

**Framework**» à partir du paragraphe « L’émulateur ».

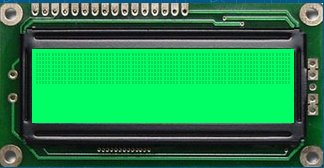
**B22) Programmation de l’algorithme en C# avec Microsoft Visual Studio**

La hauteur sera simulée par un potentiomètre relié à l’entrée An3 de la carte d’extension (FEZ Shield). (**Annexe 2**)

**Ouvrez** le projet Microsoft Visual Studio « **Telemetre** ».

On donne un exemple de programme écrit en C# en **annexe 3**. Vous pouvez vous inspirez de ce programme pour :

**- Configurer les entrées / sorties** de la carte PANDA 2. (**Annexe 2**)



**I= ???mA**

**H= ???cm**

**- Déclarer les variables**.

**- Programmer le traitement**.

**- Afficher les résultats comme ci-contre.**

**Contrainte**: Les informations affichées par

le LCD doivent être présentées comme sur la copie d’écran ci-dessus.

**Utilisation de l’environnement de développement intégré (IDE)**

* Utilisez l’**IntelliSense pour écrire votre code**.

Les touches à connaître avec l’IntelliSence sont :

- Le **point (.)**, pour afficher les membres d’un type ;

- Le **Tab (Tabulation)**, pour sélectionner l’élément surligné dans IntelliSence ;

- Le **Ctrl Espace** pour vérifier l’écriture d’un terme présent dans l’IntelliSence, et en finir la frappe si le mot (ou le début du mot) tapé n’a pas d’ambiguïté.

- Le **Tab Tab (double tabulation)** pour insérer un **extrait de code**. (bloc préprogrammé reprenant les structures de base du C#)

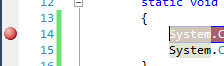
*Source : Coach C# Atelier 1*

* Touche **F6** pour **compiler** le code. (La dernière version du programme est sauvegardée)
* Touche **F5** pour **exécuter** le code.
* Pensez à utiliser le **debugger** pour mettre votre **programme au point**.
* La touche **F11** permet d’exécuter le programme en pas à pas dans la carte Panda2.
* Déclaration d’une constante

Exemple : const Single Kcan = 310.30303F;

Exemple d’affichage obtenu dans le debugger:





**Exemple**

* Un **clic gauche** dans la marge de l’éditeur de code permet d’insérer un point d’arrêt.

**Pour faire vérifier le fonctionnement de votre programme**

**C) Synthèse** [🕐 30mn]

****

**Buzzer**

**Modification 1**

**Modifiez** votre algorithme **puis** votre programme pour déclencher une alarme si la hauteur mesurée H(cm) est à l’extérieur de l’intervalle suivant :

**20 ≤ H(cm) ≤ 170**

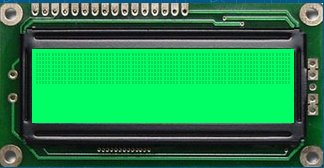
Remarque : La fréquence du signal émis par le buzzer doit être Fmin = 500Hz pour H < 20cm et Fmax = 1kHz pour H > 170cm. Pour cela, vous disposez d’une méthode **Set(Fréquence, rapport cyclique)** pour la régler et **Set(false)** pour désactiver le buzzer.

|  |  |
| --- | --- |
|  | **Connecteur PANDA2\*** |
| **Buzzer** | **Di5** |

Déclaration de l’objet Buzzer :

PWM Buzzer = new PWM(PWM.Pin.PWM5);

**Pour faire vérifier le fonctionnement de votre programme**



**H= ???cm**

**deltaH= ????**

**Modification 2**

On suppose que le capteur est placé à une hauteur de référence Href = 180cm.

**Modifiez** votre algorithme **puis** votre programme pour afficher la hauteur d’eau ou de sédiment **ΔH**(cm).

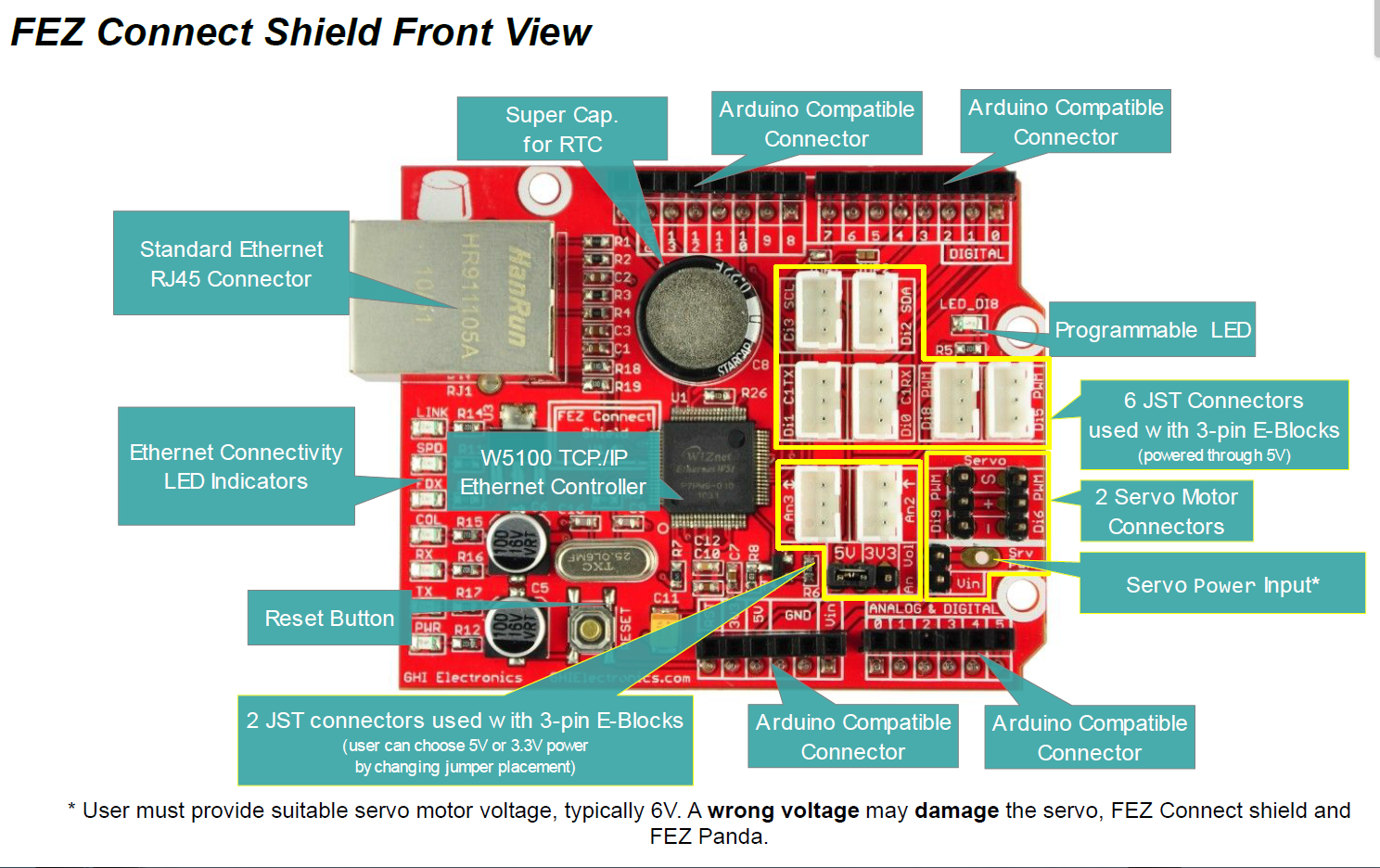
**Pour faire vérifier le fonctionnement de votre programme**

**Annexe 1 : Les types de donnée reconnus par MSVS (langage C#)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Short Name** | **.Net**  **Class** | **Signed** | **Width**  **(bytes)** | **Range** |
| **byte** | Byte | non | 1 | 0 to 255 |
| **sbyte** | SByte | oui | 1 | -128 to 127 |
| **int** | Int32 | oui | 4 | 231 to 231 – 1  -2147483648 to 2147483647 |
| **uint** | UInt32 | non | 4 | 0 to 232 – 1  0 to 4294967295 |
| **short** | Int16 | oui | 2 | -32768 to 32767 |
| **ushort** | UInt16 | non | 2 | 0 to 65535 |
| **long** | Int64 | oui | 8 | 263 to 263 - 1 |
| **ulong** | UInt64 | non | 8 | 0 to 264 - 1 |
| **float** | Single | oui | 4 | -3.402823e38 to 3.402823e38 |
| **double** | Double | oui | 8 | -1.79769313486232e308 to 1.79769313486232e308 |
| **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 |  |
| **bool** | Boolean | - | 1 | Vrai(true) ou faux (false) |

**Extrait de la documentation Microsoft**

**Annexe 2 : FEZ Connect Shield**



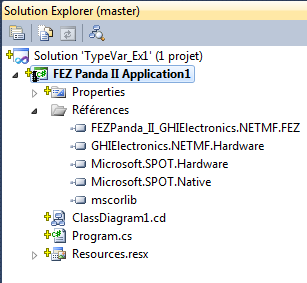
Position de l’afficheur LCD

Position du potentiomètre

simulant H(cm)

Position du Buzzer

**Annexe 3 : Exemple de programme écrit en C#**

****

using System;

using System.Threading;

**\*Remarque** : Dans Visual Studio, on sélectionne la méthode Read() puis F12 pour accéder à sa définition :

public int Read();

La valeur renvoyée est un type **int = Int32**. Elle ne peut pas être placée dans N (déclarée comme un Int16)

* **CASTING**

using Microsoft.SPOT;

using Microsoft.SPOT.Hardware;

using GHIElectronics.NETMF.FEZ;

using GHIElectronics.NETMF.Hardware;

namespace TypeVarEx2

{

public class Program

{

public static void Main()

{

// Variables

UInt16 NU = 0, NI = 0;

UInt32 NP = 0;

Single U1 = 0, I1 = 0, P1 = 0, P2 = 0;

// Configuration des entrées / sorties

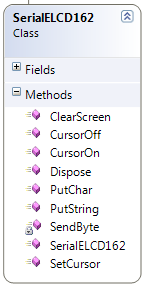
// Entrées analogiques

AnalogIn In\_Ua = new AnalogIn(AnalogIn.Pin.Ain2);

AnalogIn In\_I = new AnalogIn(AnalogIn.Pin.Ain3);

// Sortie série asynchrone

FEZ\_Components.SerialELCD162 LCD = new FEZ\_Components.SerialELCD162((byte)(FEZ\_Pin.Digital.Di5));



while (true)

{

NU = (UInt16)In\_Ua.Read();

**Lecture des Entrées**

NI = (UInt16)In\_I.Read();

// Solution 1: Coefficients appliqués avant la multiplication

U1 = (Single)9.9 \* (Single)NU / 512;

I1 = (Single)1.1 \* (Single)NI /1024;

P1 = U1 \* I1;

// Solution 2: Coefficient appliqué après la multiplication

NP = (UInt32)(NU \* NI);

P2 = NP \* (Single)20.771E-6;

Debug.Print("NU=" + NU.ToString() + " et Ua=" + U1.ToString() +

**Debugger**

" NI=" + NI.ToString() + " et I=" + I1.ToString() +

" NP=" + NP.ToString() + " et P1=" + P1.ToString() + "et P2=" + P2.ToString());

// Formatage du textes pour affichage sur le LCD

// http://msdn.microsoft.com/en-us/library/dwhawy9k(v=vs.110).aspx

LCD.ClearScreen();

LCD.PutString("Ua=" + U1.ToString("N1") + "V " + "I=" + I1.ToString("N1") + "A");

LCD.SetCursor(0, 1);

LCD.PutString("P=" + P2.ToString("N1") + "W");

Thread.Sleep(1000); // Temporisation 1s

}

}

}

}