O. Carillo – V. Hourdin – S. Lavirotte – G. Rey – J.-Y. Tigli
Polytech'Nice Sophia
SI5 – Master 2 IFI IAM - 2015-2016

Rendez vos objets communicants et interactifs avec des Phidgets

1 Kit de Développement pour plateformes Phidgets

Les phidgets forment une « bibliothèque matérielle » de capteurs et d'actionneurs permettant de construire des interfaces physiques à partir d'un ensemble de « widgets » physiques.

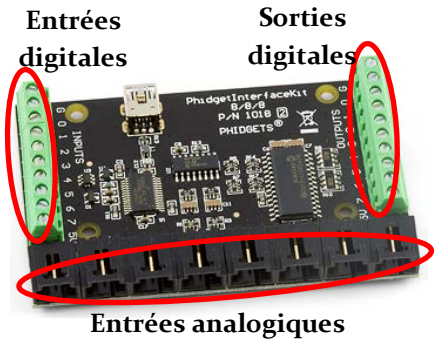
http://www.phidgets.com/

Une grande variété de capteurs et d'actionneurs sont disponibles :

- Capteurs : température, luminosité, infrarouge, son, intensité électrique, poids, …
- Actionneurs : boutons pression, potentiomètre, variateur, …

Les différents capteurs se connectent sur une plate-forme appelée Interface Kit suivant deux modes possibles :

- Digital (entrée et sortie) : pour y connecter des LEDs par exemple (sortie),
- Analogique : pour connecter les capteurs et actionneurs.



1.1 Installation des Pilotes

Installez les drivers et bibliothèques pour Phidgets en sélectionnant le paquetage adapté à votre machine :

http://www.phidgets.com/docs/Operating_System_Support

Il existe deux grands types de plateformes fournies par Phidget : des plateformes qui sont de simples dispositifs USB que l'on connecte à un ordinateur et des plateformes embarquées autonomes avec un CPU et une interface Ethernet. Afin de rendre ce deuxième type de plateforme Phidget découvrable quand vous la connectez (et que nous utiliserons dans la deuxième partie du TD), vous devez aussi installer le logiciel de découverte d'Apple, Bonjour :

https://www.apple.com/fr/support/bonjour/

1.2 Environnements de Développement

1.2.1 Installation de l'environnement .NET

Vous avez déjà installé l'environnement .NET 3.5 (SharpDevelop 3.2.1). Donc vous devez disposer des outils nécessaires pour réaliser cette partie.

1.2.2 Développement C# pour Phidgets

L'installation du driver pour Phidgets a installé les bibliothèques nécessaires au développement pour les différents environnements, et en particulier pour l'environnement .NET.

Commencez par créer une solution pour votre programme de test de l'interface Phidgets : Fichier / Nouveau / Solution / C# / Application Windows / Application Console

Afin de pouvoir utiliser la bibliothèque Phidgets depuis votre application, vous devez utiliser les inclusions suivantes :

```
using Phidgets;
using Phidgets.Events;
```

Pour fournir la référence de la bibliothèque Phidget au projet, vous effectuerez un clic droit sur les références et sélectionnerez l'ajout d'une référence et sélectionnerez la librairie Phidgte21.dll dans les composant du GAC (Global Assembly Cache).

Ensuite, vous devrez créer un objet de type InterfaceKit qui permet la connexion à la plate-forme Phidgets ainsi que la récupération des données et l'envoi des informations à la plate-forme.

Rendez vos objets communicants et interactifs avec des Phidgets

```
static InterfaceKit ifKit; //Déclaration en dehors du main

ifKit = new InterfaceKit(); //Instanciation dans la fonction main
```

La programmation événementielle étant recommandée pour s'interfacer avec la plate-forme, vous connecterez les fonctions handlers sur les fonctions de bases ainsi que sur les fonctions permettant de récupérer les informations des senseurs de la plate-forme

```
//Hook the basic event handlers
ifKit.Attach += new AttachEventHandler(ifKit_Attach);
ifKit.Detach += new DetachEventHandler(ifKit_Detach);
ifKit.Error += new ErrorEventHandler(ifKit_Error);

//Hook the phidget specific event handlers
ifKit.SensorChange += new SensorChangeEventHandler(ifKit_SensorChange);
```

Enfin, vous réaliserez l'ouverture de la connexion à l'aide de la fonction `ifKit.open()` avec une attente de la connexion de la plate-forme à l'ordinateur `ifKit.waitForAttachment()`.

Il ne vous reste plus qu'à coder l'ensemble des fonctions handlers :

```
static void ifKit_Attach(object sender, AttachEventArgs e)
static void ifKit_Detach(object sender, DetachEventArgs e)
static void ifKit_Error(object sender, ErrorEventArgs e)
static void ifKit_SensorChange(object sender, SensorChangeEventArgs e)
```

Pour cette dernière méthode, vous pouvez récupérer le port du capteur (e.Index) et la valeur reçu pour ce capteur (e.Value).

Vous testerez que votre code fonctionne bien à l'aide d'un des capteurs fournis et que vous êtes bien capables d'afficher les valeurs à chaque changement.

1.2.3 Limites des développements sans intergiciel

Vous avez pu grâce à cette première partie avoir un premier aperçu de la facilité d'utilisation de la plate-forme Phidgets. Toutefois, développer une application pour un objet communicant est une activité qui va nécessiter de mettre en place une logique applicative sur l'objet, la mise en place d'interfaces de communication ou d'activation des fonctionnalités, le tout dans un contexte de prototypage plus que relevant du développement d'une application complète conçue.

Pour effectuer ce prototypage rapide, nous allons vous présentez l'intergiciel WComp, développé au sein de l'Equipe Rainbow du laboratoire I3S de l'Université de Nice Sophia Antipolis.

2 Découverte de WComp

Vous pouvez vous référer à la documentation technique de référence disponible en ligne ainsi qu'aux vidéos de démonstration disponibles à l'adresse suivante pour l'installation et la prise en main de l'environnement WComp :

```
https://download.wcomp.fr/Tutorial/ETIA/TD1/SharpWComp-3.0.0.1153.msi
http://www.wcomp.fr/videos
```

Commencez par démarrer SharpWComp et créez un fichier WComp :

- Fichier / Nouveau / Fichier …
- WComp.Net / C# Container -> crée un nouveau fichier Container.cs (onglet en haut de la zone de travail)
- Pour pouvoir manipuler les composants il faut activer la représentation graphique du Container (onglet WComp.NET en bas de la zone de travail, passage sur cet onglet automatique en SharpWComp 3.0).

N'oubliez pas que vous pouvez sauvegarder vos assemblages de composants avec l'option Export du menu WComp.NET.

Rendez vos objets communicants et interactifs avec des Phidgets

3 Composition Locale : Modèle LCA

Nous allons voir comment utiliser et créer des composants : composants fournis avec SharpWComp, ou encore créer vos propres composants pour répondre à vos besoins. Nous verrons aussi dans cette section comment relier un événement émis par un composant à un appel de méthode d'un autre composant ce qui permet d'établir des communications entre les composants.

3.1 Mise en place d'une application par assemblage de composants

3.1.1 Evènements simples

Nous souhaitons pouvoir allumer et éteindre une LED de la plate-forme Phidget à l'aide de 2 buttons. Créer les 2 boutons : un bouton LedOn et un bouton LedOff, et les relier au composant. Créer les 2 boutons : un bouton LedOn et un bouton LedOff, et les relier au composant `GenericOutput` de la catégorie Phidget afin d'appeler les méthodes `On()` et `Off()` du dispositif. Vous modifierez les propriétés du composant `GenericOutput` pour que le Port corresponde au port de la LED que vous souhaitez commander.

3.1.2 Evènements complexes

Avoir deux boutons pour contrôler une des LEDs peut ne pas s'avérer intéressant surtout si on doit le faire pour toutes les LEDs connectées à la plate-forme. Nous souhaitons maintenant pouvoir contrôler la LED à l'aide d'une `CheckBox`.

Créer `1 checkBox` et la relier au composant `GenericOutput` afin de piloter le dispositif à l'aide de la méthode `SetOutputValue(boolean)`. Nous pouvons constater que l'événement émis par la checkBox (`CheckStateChanged`) n'envoie pas de données (la valeur de la case cochée). La fonction `SetOutputValue` nécessite par contre un paramètre booléen. Il faut donc sélectionner le mode « *signature incompatible* » pour voir cette fonction. Le panneau suivant vous permettra de faire appel à une fonction de « get » sur le composant émettant l'événement pour compléter la signature de la méthode appelée.

Vous devriez maintenant pouvoir contrôler la LED. Mais comment faire pour contrôler l'allumage de cette LED en fonction de capteurs ou actionneurs connectés à la plate-forme Phidgets ?

3.2 Création d'un Composant Bean

Il peut s'avérer nécessaire de créer vos propres composants si aucun composant existant ne correspond à vos besoins. L'environnement offre la possibilité de créer un composant à partir d'un squelette de code. Nous allons donc créer un composant permettant d'envoyer un événement lorsqu'un seuil numérique est dépassé.

3.2.1 Création et compilation d'un composant

Après avoir quitté et redémarré SharpWComp, vous pouvez créer une nouvelle solution pour la création du composant souhaité :

- Fichier / Nouveau / Solution …
- WComp.NET / WComp Bean Combine (nommer cette solution Blink)

Ajoutez un fichier à votre nouvelle solution :

- Clic droit sur votre solution / Ajouter / Nouveau Fichier …
- WComp.Net / C# Bean with Thread -> crée un nouveau fichier BeanThread1.cs (que l'on peut renommer en Blink.cs)

Il ne vous reste plus qu'à modifier le squelette de code pour donner à ce nouveau composant le comportement souhaité.

Nous allons faire un composant qui nous permettra de faire clignoter une des lumières du feu tricolore. Ce composant disposera tout d'abord d'une **propriété** `sleepProperty` permettant d'indiquer la fréquence d'émission

de ces événements. Pour coder cette propriété, inspirez-vous du code contenu dans le squelette de code d'un composant Bean sans Thread.

Ce composant aura une **méthode** `MethodBlink` qui, en fonction de la valeur d'un paramètre booléen, démarrera ou arrêtera le Thread à l'aide des fonctions `Start` et `Stop`. Vous veillerez à terminer par l'émission d'un événement faux quand vous arrêtez le clignotement. La méthode `Stop` est utilisée par SharpWComp pour terminer proprement le thread créé par le Bean.

Ecrire le code correspondant, le compiler et ajouter la librairie créée dans le dossier contenant les composants (dossier `C:\Program Files (x86)\SharpDevelop\3.0\Beans\`). Un nouveau composant `Blink` est maintenant disponible pour vos assemblages. Si vous n'avez rien précisé dans la directive [Bean] vous trouverez votre composant dans la catégorie Beans : Basic. Si vous souhaiter créer une catégorie particulière pour y ranger ce nouveau composant, vous devrez spécifier comme directive en tête de votre classe :

```
[Bean(Category="Demo")]
```

3.2.2 Créer un assemblage pour la mise en œuvre de l'application

Créer un assemblage à l'aide du composant de `ThresholdInt` que vous avez développé ainsi qu'avec les composants correspondants aux capteurs connectés à votre plate-forme. Vous utiliserez un capteur pour régler la valeur de `ThresholdValue` et un autre capteur pour faire varier la valeur envoyée. Quand le seul fixé par le premier capteur sera dépassé, cela aura pour conséquence d'allumer une LED.

Vous pouvez complexifier l'exemple en mettant plusieurs `ThresholdInt` pour allumer des Led au fur et à mesure de l'augmentation de la valeur.