

TP – Capteur de température Maxim-Dallas Thermochron DS1921K

Protocole de découverte Universal Plug and Play (UPnP)

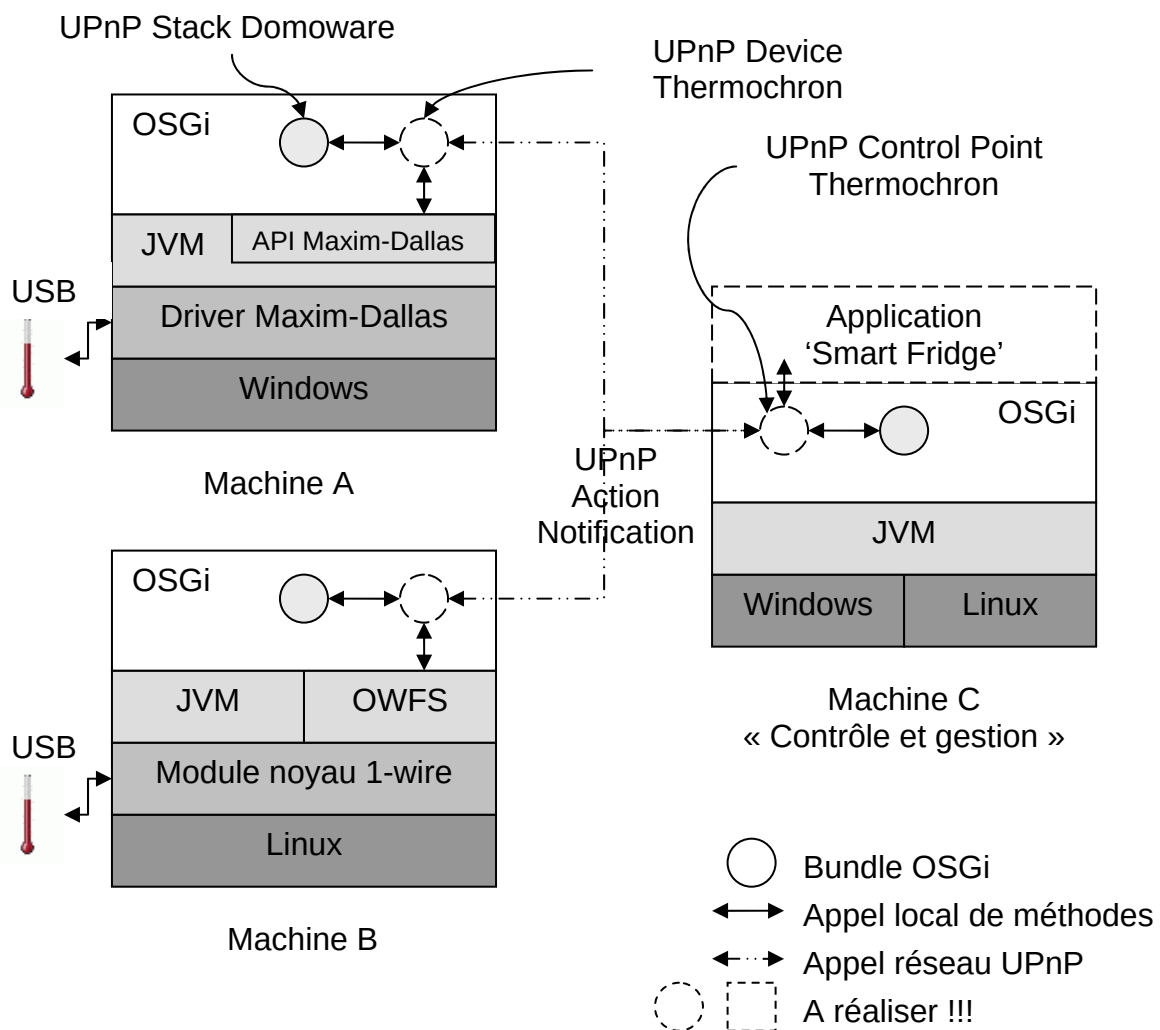
1. Objectifs

La mise en œuvre d'un système pervasif implique de prendre en compte tous les éléments d'infrastructure du niveau matériel jusqu'au niveau logiciel. Ce TP a pour objectifs de mettre en place les deux premiers éléments de l'application 'Smart Fridge' que sont :

- Le capteur de température Maxim-Dallas Thermochron DS1921K.
- Le protocole réseau de découverte Universal Plug and Play (UPnP), afin de découvrir et de piloter ce capteur à partir d'un environnement réseau local.

2. Architecture

L'architecture du système est la suivante :



Dans cette architecture, trois éléments seront à développer et tester :

- L'API Java fournie par Maxim-Dallas est à tester pour vérifier le bon fonctionnement du capteur.
- Le Device UPnP Thermochron qui se déclare sur le réseau local, permet d'être interrogé sur la température du capteur, et notifie des changements de température.
- Le Control Point UPnP Thermochron qui surveille le réseau local, et s'abonne aux notifications de changement de température.

3. Prérequis d'installation

Systeme

Le TP se déroulera au département sur la partition Windows.

Optionnel : Le TP est tout à fait possible sous Linux (pour les étudiants motivés ayant apportés leur portable ☺), mais tous les logiciels n'y sont alors pas installés. Je conseille de réaliser en premier le TP sous Windows et ensuite de le porter sous Linux.

Driver

Maxim-Dallas fournit un driver Windows USB. Ce driver est installé sur les partitions Windows et le port USB assigné au capteur est marqué par une pastille de couleur.

Optionnel : Sous Linux, pour reconnaître le capteur, il est nécessaire de compiler et d'installer le module noyau 1-wire.

Java

Sur la partition Windows, le JDK 1.6 de Sun (<http://java.sun.com/>) est installé dans le répertoire C:\Program Files\Java\jdk1.6.0_01. Il est nécessaire de positionner la variable d'environnement JAVA_HOME sur ce répertoire.

(Rappel :

- pour positionner une variable d'environnement sous Windows, menu 'Démarrer', click bouton droit sur 'Poste de travail', 'Propriétés', onglet 'Avancé', 'Variables d'environnement', et 'Nouveau' dans variables utilisateur
- pour positionner une variable d'environnement sous Linux, modifier le fichier .bashrc pour inclure JAVA_HOME=/monrepertoirejdk et export JAVA_HOME)

Maven

L'environnement de développement et d'exécution que nous allons utiliser requiert l'outil Apache Maven pour sa chaîne de compilation (<http://maven.apache.org/>). Sur la partition Windows, cet outil est installé dans C:\MACINFO\maven. Il est nécessaire d'ajouter le répertoire C:\MACINFO\maven\bin dans la variable PATH de l'environnement. Sur la partition Windows, cette variable appartenant à l'administrateur, cela est déjà réalisé.

OSGi – Felix

L'environnement de développement et d'exécution sera le même que dans le TP précédent, c.a.d. l'implantation Open Source Apache Felix des spécifications OSGi (<http://cwiki.apache.org/FELIX/>). Pour pouvoir développer sous cet environnement, il est nécessaire de le compiler.

Dans le répertoire C:\Temp\, créez un répertoire MACIF. Téléchargez l'archive suivante dans le répertoire MACIF :

<http://perso.citi.insa-lyon.fr/flemouel/Teaching/INSA/4TC/MACIF/TPSensor/felix-0.9.0-SNAPSHOT-20070610.zip>

(Cette archive correspond exactement au subversion des sources
svn checkout [http://svn.apache.org/repos/asf/felix/trunk felix](http://svn.apache.org/repos/asf/felix/trunk/felix))
Décompressez cette archive dans le répertoire et vérifiez que vous avez bien un nouveau répertoire felix.

Lancez l'invite de commande (menu 'Démarrer', 'Tous les programmes', 'Accessoires', 'Invite de commande') et tapez les commandes suivantes :

```
cd C:\Temp\MACIF\felix
mvn -Dpackaging=plugins clean install
mvn -Dpackaging=bundle clean install
mvn -Dpackaging=osgi-bundle clean install
```

Que s'est-il passé ?

Réponse : Maven vient de chercher et de résoudre toutes les dépendances logicielles qu'utilise chacune des parties de Felix. Il a chargé ces logiciels à partir d'un répertoire commun distant et les a installés dans le répertoire local \$HOME/.m2 (sous Windows, cela correspond à C:\Documents and Settings\login\.m2). Il a compilé le profil plugins : les artefacts de base, le profil bundle : le noyau de la distribution Felix et le profil osgi-bundle : tous (ou presque) les bundles additionnels.

4. Mise en place du capteur Maxim-Dallas Thermochron DS1921K

Dans cette partie du TP, nous allons nous occuper de la couche basse matérielle.

Détection

En premier lieu, nous allons vérifier que le capteur est bien reconnu.

Insérez le capteur dans le port USB marqué avec une pastille de couleur. Puis ouvrez le gestionnaire de périphériques (menu 'Démarrer', 'Poste de travail', click bouton droit, 'Propriétés', onglet 'Matériel', 'Gestionnaire de périphériques'). Dans le cas où le capteur n'est pas reconnu, vous aurez un périphérique USB avec un point d'interrogation. Dans le cas où il est reconnu, vous aurez dans la liste '1-Wire Devices' -> 'USB Host Adapter for 1-Wire Network'.

Optionnel : Sous Linux, pour vérifier que votre module noyau 1-wire a bien détecté le capteur, tapez `lsusb` (en tant que root). Vous devriez obtenir quelque chose comme cela :

```
Bus 002 Device 002: ID 04fa:2490 Dallas Semiconductor DS1490F
2-in-1 Fob, 1-Wire adapter
```

Vérification de bon fonctionnement

Dans le répertoire `C:\Temp\MACIF`, créez le répertoire `API`, et téléchargez l'API Java fournie par Maxim-Dallas ici :

http://perso.citi.insa-lyon.fr/flemouel/Teaching/INSA/4TC/MACIF/TPSensor/owapi_1_10.zip

ou à partir du site original (lecture conseillée) :

http://www.maxim-ic.com/products/ibutton/software/1wire/1wire_api.cfm

Décompressez cette archive dans le répertoire et vérifiez que vous avez bien les répertoires suivants :

- `docs` : avec la javadoc de l'API des capteurs
- `examples` : avec des programmes de tests
- `lib` : avec les jar contenant le code compilé
- `rxtx` : un driver port série / parallèle (non utile ici)
- `src` : avec les sources de l'API

Le capteur DS1921 que nous allons tester peut fonctionner de deux manières : capture de température à la demande, ou lancement d'une mission de capture de mesures.

Pour vérifier que le capteur marche correctement, déroulez l'exemple fourni en tapant les commandes suivantes sous l'invite de commande :

```
cd C:\Temp\MACIF\API\examples\Thermochron
run_initMission
run_dumpMission ahl
run_dumpMission k
```

A partir des questions posées, essayez de comprendre le fonctionnement du capteur.

Optionnel : Sous Linux, l'interaction avec le capteur pour s'effectuer avec le logiciel OWFS (<http://www.owfs.org/> ; <http://owfs.sourceforge.net/>). Le capteur sera alors monté dans `/proc`. L'accès à `/proc` est ensuite à programmer selon le langage de développement utilisé. OWFS fournit le support pour perl, python, php, tcl, C (Java en bêta dans le SVN).

5. Mise en place du protocole UPnP

Dans cette partie du TP, nous allons concevoir la partie réseau UPnP qui permet de déclarer et piloter un périphérique sur un réseau local. Dans un premier temps, nous allons mettre en place ce protocole indépendamment du capteur. Nous relierons les deux dans la partie suivante.

Lancement de la pile réseau UPnP et examen de son fonctionnement

Tout d'abord, démarrez Felix, dans une invite de commande, avec les commandes suivantes :

```
cd C:\Temp\MACIF\felix\main
```

```
java -jar bin\felix.jar
```

Entrez un nom de profil.

Jouez avec l'interface texte comme dans le TP précédent.

Que s'est-il passé ?

Réponse : Felix crée un répertoire local \$HOME/.felix (sous Windows, cela correspond à C:\Documents and Settings\login\.felix) dans lequel sera stocké tous les bundles que vous démarrez.

Démarrez ensuite les bundles suivants, dans cet ordre :

1. Shell GUI
2. Shell GUI Plugin
3. Apache Felix Log Service
4. Servlet 2.1 API
5. OSGi R4 Compendium
6. HTTP Service
7. UPnP Extra
8. UPnP Basedriver
9. UPnP Tester
10. UPnP BinaryLight
11. UPnP Clock
12. UPnP TV

(Rappel : pour démarrer un bundle, vous devez taper sous Felix : start URL. Par exemple, pour démarrer Shell GUI, vous devez taper start file:C:\Temp\MACIF\felix\shell.gui\target\org.apache.felix.shell.gui-0.9.0-incubator-SNAPSHOT.jar (CTRL-C CTRL-V pour le couper coller ☺))

Que s'est-il passé ?

Réponse : Les bundles 1 et 2 ne sont pas indispensables mais ils fournissent une interface graphique plus conviviale pour travailler ☺ (le bundle 2 dépend du 1). Les bundles 3 et 4 sont requis pour le bundle 5. Le bundle 5 contient la spécification des interfaces de tous les services, notamment les services UPnP. Le bundle 6 lance un serveur Web (requis ici uniquement parce que le bundle 10 lance une servlet). Le bundle 8 contient les spécifications de la pile UPnP et le bundle 8 son implantation (le bundle 8 requiert le 7). Le bundle 9 lance un Control Point UPnP générique (il requiert la pile UPnP).

Lorsque vous démarrez les bundles 10-11-12, regardez bien le Control Point UPnP générique pour noter l'apparition des Devices UPnP correspondants. Vous pouvez ensuite examiner les variables UPnP présentes dans ces Devices et invoquer les actions UPnP.

Testez en particulier :

- Invoquer l'action UPnP d'allumage et d'extinction de la lampe
- Invoquer l'action de récupération de l'heure sur l'horloge
- Invoquer l'action UPnP d'allumage et d'extinction de la télévision
- Arrêter et démarrer le bundle UPnP Clock et constater bien la disparition et l'apparition de l'heure dans la télévision

Développement du bundle UPnP Device Thermochron

Dans le répertoire `C:\Temp\MACIF\felix`, créez le répertoire `upnp.sample.sensor`. Dans ce répertoire, en vous inspirant par exemple du code du Device BinaryLight dans `upnp.sample.binarylight`, développez un Device UPnP Thermochron qui diffuse une température aléatoire toute les 5 secondes.

Les caractéristiques demandées pour le Device UPnP Thermochron sont les suivantes :

- Un Device UPnP
 - o id : `uuid:Felix-Sensor` (par exemple, comme vous voulez)
 - o type : `urn:schemas-upnp-org:device:Sensor:1`
- Un Service UPnP
 - o id : `urn:upnp-org:serviceId:GrabTemperature:1` (par exemple)
 - o type : `urn:schemas-upnp-org:service:Grab:1`
- Une Variable UPnP :
 - o name : `Temperature`
 - o class : `Double`
 - o type : `TYPE_FLOAT`
- Une Action UPnP
 - o name : `GetTemperature`

(Rappel : Eclipse est présent dans l'environnement ☺)

Développement du bundle UPnP Control Point Thermochron

Dans le répertoire `C:\Temp\MACIF\felix`, créez le répertoire `upnp.sample.sensor.controlpoint`. Regardez par exemple le code du Device TV dans `upnp.sample.tv` pour la notification d'apparition et de disparition de l'heure. Ensuite développez un Control Point UPnP Thermochron qui s'enregistre comme UPnP Subscriber sur les Devices UPnP Thermochron (selon le type de Device et Service) et qui est notifié par le mécanisme d'UPnP Event Listener des changements de la variable `Temperature`.

(Indice : Commencez par copier dans `upnp.sample.tv`, les classes `Activator` et `TvDevice` et retirez le superflus).

6. Intégration du capteur et du protocole de découverte

Maintenant que vous avez deux bundles UPnP qui marchent magnifiquement ☺, l'heure est venue de remplacer la génération de température aléatoire par les réels appels de méthodes sur le capteur.

Pour commencer, lisez la javadoc de l'API fournie dans `C:\Temp\MACIF\API\doc\index.html`. Regardez en particulier le package `com.dalsemi.onewire.container` et la classe `OneWireContainer21` qui correspond au Thermochron DS1921.

Examinez ensuite le code source des fichiers exemples que vous avez utilisé pour tester le capteur dans `C:\Temp\MACIF\API\examples\Thermochron\src`.

Pour utiliser maintenant cette API, vous allez rencontrer deux problèmes : un au développement et un à l'exécution.

Installation de l'API Maxim-Dallas comme artéfact dans le répertoire local Maven

Dans le code de votre UPnP Device Thermochron, modifiez juste la classe dans laquelle vous générez la température pour y inclure la ligne suivante :

```
import com.dalsemi.onewire.container.OneWireContainer21;
```

Recompilez votre bundle.

Que se passe-t-il ?

Réponse : le compilateur Java vous informe qu'il ne trouve pas le package `com.dalsemi.onewire.container`. La solution à ce problème est que la chaîne de compilation (Maven) doit fournir le bon `CLASSPATH` au compilateur. Comme décrit juste au début, Maven va chercher les dépendances d'un logiciel dans un répertoire local `$HOME/.m2` (ou distant s'il ne le trouve pas en local). Il faut donc y installer l'API Maxim-Dallas.

Téléchargez l'artéfact Maven Maxim-Dallas

<http://perso.citi.insa-lyon.fr/flemouel/Teaching/INSA/4TC/MACIF/TPSensor/maven-artifact-maxim-dallas.zip> et décompressez le dans le répertoire `$HOME/.m2` (sous Windows, cela correspond à `C:\Documents and Settings\login\.m2`).

Modifiez le fichier `pom.xml` de votre projet `upnp.sample.sensor` pour y inclure la dépendance suivante :

```
<dependency>
  <groupId>maxim-dallas</groupId>
  <artifactId>com.dalsemi.onewire</artifactId>
  <version>1.10</version>
  <scope>provided</scope>
</dependency>
```

Recompilez et, normalement, tout devrait bien se passer ☺

(Note : vous pouvez regarder la structure de l'archive zip pour voir comment j'ai transformé l'API Maxim-Dallas donnée en un artéfact Maven).

Installation de l'API Maxim-Dallas comme bundle sous Felix

Après avoir compilé votre bundle UPnP Device Thermochron, vous devrez le tester sur Felix. Mettez à jour le bundle, et redémarrez le.

Que se passe-t-il ?

Réponse : Felix vous informe qu'il ne peut pas démarrer votre bundle car la dépendance `com.dalsemi.onewire` (que vous venez d'introduire juste avant ☺) n'est pas satisfaite sur Felix. Et en effet, aucun bundle ne fournit l'implantation de l'API Maxim-Dallas. Il faut donc installer un bundle la réalisant.

Dans `C:\Temp\MACIF\felix`, créez les répertoires `com.dalsemi.onewire` et `com.dalsemi.onewire/target`. Téléchargez le bundle OSGi Maven Maxim-Dallas dans le répertoire `target` <http://perso.citi.insa-lyon.fr/flemouel/Teaching/INSA/4TC/MACIF/TPSensor/com.dalsemi.onewire-1.10.jar>.

Démarrez ce bundle sous Felix :

```
start
```

```
file:C:\Temp\MACIF\felix\com.dalsemi.onewire\target\com.dalsemi.onewire-1.10.jar
```

Démarrez ensuite votre bundle UPnP Device Thermochron et, normalement, tout devrait bien se passer ☺

(Note : vous pouvez regarder la structure de l'archive jar pour voir comment j'ai transformé l'API Maxim-Dallas donnée en un bundle OSGi).

Développement de l'appel de l'API Maxim-Dallas dans le Device UPnP Thermochron

Il ne vous reste ***plus*** maintenant qu'à remplacer la génération de température aléatoire par des appels réels de méthodes sur le capteur.

Les classes à utiliser sont

```
com.dalsemi.onewire.OneWireAccessProvider
```

```
com.dalsemi.onewire.adapter.DSPortAdapter
```

```
com.dalsemi.onewire.container.OneWireContainer21
```

Indices de développement :

- Il est plus simple d'utiliser le capteur en mode à la demande qu'en mode mission.
- La référence de l'adapter est à trouver une seule fois.
- La référence de l'iButton est à trouver à chaque demande de température.
- Notifier que lorsqu'il y a un changement de température est plus judicieux que notifier à chaque fois.

Attention !!!

N'oubliez pas que vous avez mis en œuvre ce TP sur `C:\Temp` qui est un répertoire local !!! Si vous souhaitez récupérer votre travail, faites-en une archive et copiez-la sur une partition réseau !!!