Université Nice Sophia Antipolis

Dream IT
Innovation & Technologies
http://www.dreamit-fr.org/

U-Health

Document :	*Intégration des objets dans le cloud d'objets*
Sous-tâche :	1.3
Numéro des Livrables :	D1.3
Date	13/12/2013
Rédacteurs :	*Rebecca Poustis (Polytech), Christian Brel (I3S), Stéphane Lavirotte (I3S-Polytech), Jean-Yves Tigli (I3S-Polytech)*
Coordinateurs :	Stéphane Lavirotte

Sommaire

1 Introduction

Suite à l'identification et la conception d'objets communicants, nous nous intéressons à la manière d'intégrer ces objets dans un cloud d'objets. L'objectif principal est de permettre la construction d'une application basée sur les objets communicants. Ces objets sont des objets de la vie quotidienne augmentés par différents capteurs. Ces objets rendent donc un service principal qui leur est propre comme par exemple, une lampe qui permettra d'allumer ou d'éteindre la pièce. En supplément, ces objets vont proposer d'autres services associés aux capteurs ajoutés à l'objet comme par exemple dans le cadre de la lampe, l'ajout d'un micro qui va permettre d'obtenir le niveau de bruit dans la pièce ou encore la reconnaissance d'un type de bruit particulier.

La construction d'une application à partir de ces objets est nécessaire pour obtenir des fonctionnalités de plus haut niveau. En combinant les données provenant des capteurs d'un matelas communiquant et les données en provenance du micro sur une lampe, nous pouvons corréler ces données afin de détecter des troubles du sommeil (ronflements, cycles de sommeil perturbés, etc…)

Avant de mettre en place une telle application, nous reprenons dans ce livrable la manière dont ont été développés les objets et nous précisons ceux qui ont été retenus pour le scénario final d'application dans le cadre du projet, en décrivant en détails le(s) service(s) qu'ils proposent.

2 Développement d'objets communicants

Nous décrivons dans cette partie la plateforme logicielle utilisée pour développer les différents objets. Cette plateforme s'appuie sur un protocole de communication qui est UPnP (Universal Plug-and-Play) que nous décrivons dans une seconde section. Dans une troisième section, nous décrivons de manière succincte la plateforme matérielle utilisée pour rendre les objets fonctionnels.

2.1 WComp

WComp est un environnement de prototypage logiciel rapide créé par l'équipe Rainbow (Laboratoire I3S, UMR 7271 Université Nice Sophia/CNRS).
Cette approche à composants légers a été développée pour la composition de services. Celle-ci permet de concevoir des applications d'Informatique Ambiante par assemblage de composants logiciels orchestrant des accès aux services de l'infrastructure ambiante. Ainsi ces composants légers gèrent les orchestrations dynamiques de services Web pour dispositifs.
WComp est au cœur de l'Ubiquarium (plateforme d'étude des usages des équipements d'informatique mobile en environnement simulé) du département SI de Polytech Nice.

WComp est un environnement de développement et d'exécution évènementiel utilisant des composants logiciels ou matériels. Les entités peuvent à la fois représenter des briques logicielles à la manière de Bean ou des entités matérielles UPnP. WComp permet l'assemblage de ces différents composants entre eux (assemblage de composants) et leur exécution à l'intérieur d'un container.
L'architecture de WComp s'organise autour de containers (pour l'exécution des assemblages de composants) et de designers. L'objectif des containers est de prendre en charge dynamiquement les services systèmes requis par les composants d'un assemblage comme l'instanciation, la désignation, la destruction de composants logiciels fonctionnels et de liaisons. Les designers permettent la manipulation dynamique des assemblages de composants au travers des containers qui les gèrent. Un designer graphique d'architecture permet, par exemple, de composer manuellement des assemblages de composants à partir d'une représentation graphique des flots d'événements. Il est particulièrement adapté à la description de l'application. Un designer d'aspects d'assemblage permet quant à lui, de décrire des schémas d'adaptation aux variations des objets présents. Ces derniers sont alors sélectionnables, applicables et tissables et permettent ainsi d'adapter dynamiquement l'application précédemment décrite aux variations dues à son contexte.
La Figure 1 représente l'interface qui permet de créer des assemblages de composants WComp.

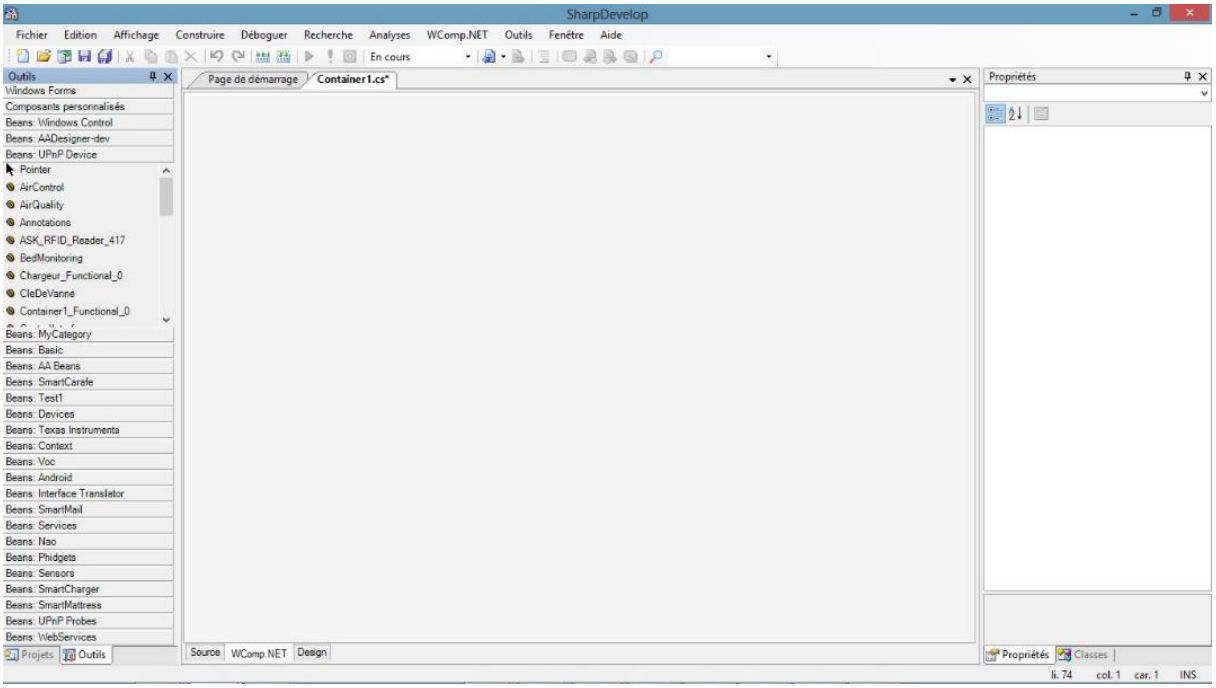


Figure 1 : Designer graphique intégré au logiciel SharpDevelop

2.2 UPnP

Comme nous avons pu le voir, WComp utilise le protocole UPnP. UPnP permet à des clients (points de contrôle) de découvrir et de rechercher par diffusion de messages des dispositifs (serveurs UPnP) sur un réseau local. UPnP permet également la communication (échange d'informations, sous la forme d'appel de méthodes (du client vers le dispositif) et d'envoi d'événements (du dispositif vers le client) entre un dispositif et un point de contrôle.

Afin d'augmenter les capacités des objets de la vie quotidienne, nous avons couplé à chaque objet communicant une carte Phidget avec un système Linux embarqué permettant d'héberger un container WComp qui s'occupe de la logique interne de l'objet. Ce lancement permet de rendre visible l'objet sur le réseau via la technologie UPnP. Voici une brève description de ces cartes Phidgets.

2.3 Cartes Phidgets

Pour fabriquer un objet communicant, nous utilisons des cartes Phidget 1072 avec Linux embarqué sur ARM. 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.

Il existe deux catégories de Phidgets : l'une sans système et l'autre avec système embarqué. La première est un ensemble de blocs de construction pour la technologie de détection et de contrôle à partir d'un ordinateur via un branchement USB. Il suffit simplement de brancher l'appareil pour recevoir les informations récupérées des capteurs à l'aide du logiciel Phidget Control Panel ou d'un autre programme. Cette carte Phidget est très utile pour développer et tester l'assemblage sur l'ordinateur à l'aide de WComp. La deuxième est un système embarqué. Elle permet ainsi de faire tourner WComp sur la plate-forme, sans l'interface graphique. De ce fait, les assemblages peuvent

être directement intégrés et lancés par la plate-forme et on peut avoir un objet communicant autonome. Grâce à la technologie UPnP, toutes les données renvoyées par la logique de l'assemblage, peuvent être collectées. Ces services peuvent ensuite être facilement gérés par une application informatique.

3 Intégration des objets dans un cloud d'objets

Dans le cadre d'une application comme le sujet porté par le projet U-Health, c'est-à-dire une application de prévention santé avec des objets non intrusifs, différents acteurs interviennent. Le patient va vouloir accéder à un monitoring de ces données de manière compréhensive, l'entourage du patient va vouloir être prévenu afin d'intervenir et d'aider au mieux son proche, le médecin va vouloir obtenir les données détaillées afin de prévenir tout risque santé puis intervenir sur certains objets du patient afin d'affiner le paramétrage par exemple. Nous pourrions élargir le nombre d'acteurs mais nous comprenons déjà que ces acteurs vont vouloir accéder à des objets présents en des lieux différents et construire une liaison entre ces objets. Par exemple, pour le médecin, il faudrait mettre en relation une application présente sur son ordinateur et visible sur son réseau local comme un service UPnP, avec les objets présents sur le réseau personnel au domicile du patient. Ainsi, nous avons besoin d'intégrer les objets dans un réseau plus global qui prendrait la forme d'un cloud d'objets.

3.1 Retour sur le protocole UPnP

Comme décrit précédemment, UPnP est le protocole sur lequel nous nous appuyons afin de rendre l'objet « communiquant ». UPnP est un protocole réseau qui lui même s'appuie sur d'autres protocoles comme TCP/IP, UDP et HTTP. Ce protocole permet de mettre en relation des clients (serveurs UPnP) sur un réseau local. Effectivement, de part sa définition, il permet la découverte, à travers des points de contrôle, des différents dispositifs (serveurs UPnP) présents sur le réseau. Ainsi, un dispositif peut annoncer son apparition ou sa disparition à ces points de contrôle permettant d'avoir un état des dispositifs disponibles dans l'environnement. Cette découverte s'appuie notamment sur l'envoie de messages en UDP sur le réseau, messages qui ne sont pas pris en charge par les différents routeurs et donc ne peuvent sortir du réseau local sur lequel ces messages transitent.

Ainsi, en l'état, ce protocole permet la communication entre objets présents sur un réseau local et donc la construction d'une application de manière locale. La problématique est donc de mettre en relation plusieurs réseaux locaux (donc distants) afin de pouvoir interopérer avec des objets non disponibles sur son réseau local mais sur un autre réseau distant du sien.

3.2 Web des objets

Le problème présenté ci-dessus est adressé dans le domaine du Web des Objets. Ce domaine permet de mettre à disposition, sur le web, des objets qui seront alors accessibles grâce aux services qu'ils proposent. Ces objets se retrouvent alors exposer comme des ressources et les informations sur ces objets peuvent être échangées grâce à une architecture REST (Representational State Transfer) qui est basée sur le protocole HTTP. Dans ce cadre, [2] proposent une solution permettant

l'interopérabilité entre les objets disponibles sur le web. Cette approche propose 4 niveaux sur lesquels les développements d'objets pour le Web des Objets devraient s'appuyer. Ces 4 niveaux mettent en avant la volonté d'uniformiser le développement des services REST associés à chaque objet, la définition d'un modèle d'objets, la représentation de l'objet et enfin la manière de le retrouver et de l'adresser sur le réseau. Ainsi la mise en place d'une telle « norme » pourrait faciliter l'accès aux hubs sur lesquels sont greffés les objets et pourrait rendre interopérable les différents objets.

Dans la même problématique, [3] proposent des portails ayant la même interface logicielle et cela pour chaque technologie permettant de construire un « réseau du bâtiment ». Ainsi chaque portail permet d'exposer les objets qui l'adressent avec la technologie spécifique sur le web sous la forme de Web Services. Ainsi, ces portails permettent l'interopérabilité d'objets qui à priori n'ont pas été développés avec la même technologie et cela à travers le web c'est-à-dire pouvant aller opérer sur des objets en dehors du « réseau du bâtiment » lui-même.

Enfin, [1] proposent la plateforme « Lab of Things » permettant de mener des expérimentations sur des applications constituées d'objets connectés de plusieurs sites (maisons, laboratoires etc…) Ce « lab » propose tout d'abord un framework commun pour le développement d'applications utilisant les objets connectés et facilitant la mise en place d'un certain nombre de fonctionnalités comme le monitoring des applications à distance, le déploiement dans un cloud, la mise à jour des applications à distance etc... Il est construit au dessus de « HomeOS » qui permet d'abstraire la manipulation des objets et donc de simplifier la création d'applications utilisant les objets. Chaque expérimentation a un ordinateur dédié qui fait fonctionner « HomeOS » et sert de hub pour l'expérimentation. C'est ainsi que l'expérimentation peut utiliser différents objets présents sur différents sites distants.

3.3 WComp dans le cloud

Notre proposition se base sur les différents travaux présentés précédemment ainsi que sur la solution WComp que nous avons développée. Cette proposition, représentée sur la Figure 2, permet de construire des applications utilisant des objets communicants qui sont disponibles dans plusieurs réseaux locaux différents. Nous proposons l'ajout dans chaque réseau local d'un « Hub concentrateur » qui permet de récolter les informations en ce qui concerne l'apparition et la disparition d'un objet sur le réseau local, mais aussi de transmettre les évènements à l'extérieur du réseau et les appels de fonctionnalités provenant de l'extérieur du réseau vers les objets du réseau local. Ces Hubs sont alors connectés à un conteneur applicatif déployé sur le cloud qui permet de mettre en place la logique de l'application voulue. Ainsi, à la manière de ce que nous avons dans les containers WComp, nous pouvons construire des applications capables d'adresser des objets étant disponibles sur des réseaux locaux différents. Le réseau local à travers son Hub, peut interagir avec le conteneur applicatif, en utilisant afin de l'informer de toutes apparitions et disparitions d'objets de son réseau local. Ainsi, le conteneur connaît en permanence l'état des différents réseaux locaux en terme d'infrastructure et de services proposés par chacun des objets. Lorsque dans le conteneur applicatif, l'écoute d'un événement d'un objet est nécessaire, le conteneur peut alors prévenir le Hub du réseau local auquel appartient l'objet dont l'événement doit être écouté, qui va alors écouter l'événement en question (communication UPnP sur le réseau local) et renvoyer les informations au conteneur applicatifs. De la même manière, lorsqu'un appel de fonctionnalités doit être effectué sur

un objet, le conteneur envoi cet appel sur le Hub gérant l'objet en question, Hub qui, à son tour, va effectivement faire l'appel en question.

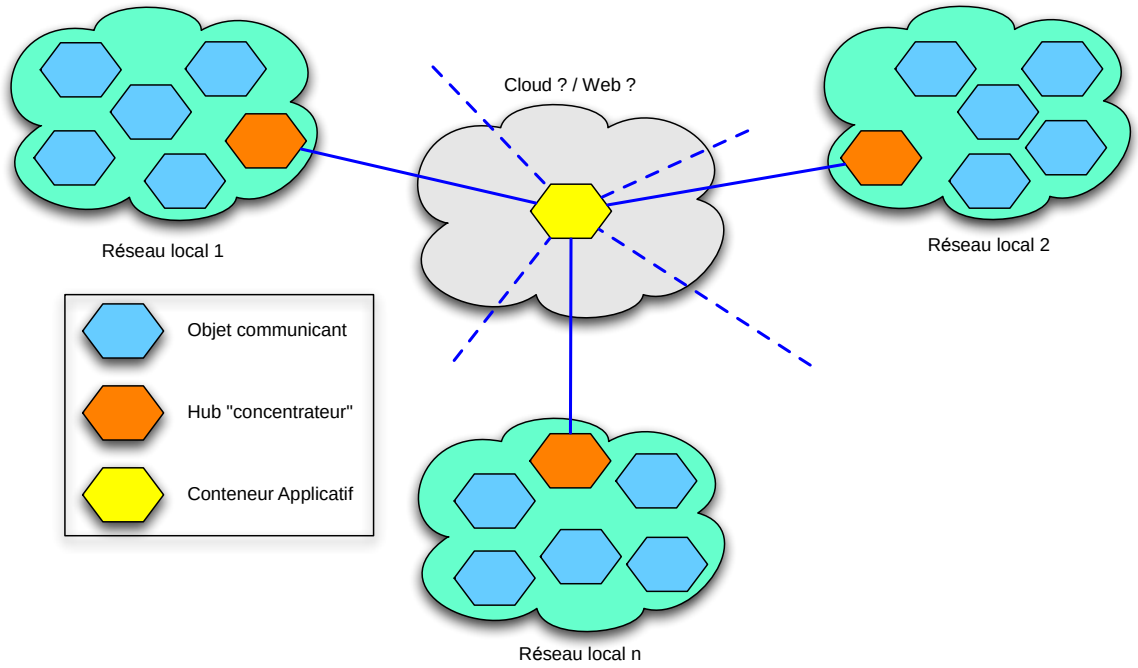


Figure 2 : Solution proposée pour la mise en place d'une application utilisant des objets communicants disponible sur plusieurs réseaux locaux différents

L'application se retrouve ainsi répartie sur plusieurs réseaux locaux et sa gestion s'effectue grâce à des concentrateurs qui relaient les différentes informations nécessaires à son bon fonctionnement, de la même manière que nous pourrions le faire sur une application déployée sur le réseau local.

3.4 Vers l'auto-adaptation de l'application dans le cloud

Cette solution permet ainsi à une application de pouvoir utiliser des objets présents sur différents réseaux locaux. Ainsi, une première perspective sera de permettre à l'application de sélectionner l'objet à utiliser en fonction de règles de sélection. Effectivement, puisque cette application a à disposition un ensemble d'objets, dont certains peuvent potentiellement remplir les mêmes fonctionnalités, nous pouvons à partir de l'expression de règles à définir (fonction de la qualité de services, de la charge d'utilisation de l'objet etc…) permettre à l'application de sélectionner le « meilleur » objet sur lequel bâtir ces propres fonctionnalités. Nous pourrions alors utiliser des règles se rapprochant des Aspect d'Assemblages et de toute la mise en application de ceux-ci pour permettre à l'application de s'adapter à son environnement (ici les objets présents sur différents sites) de manière dynamique.

4 Objets développés

Trois objets communicants ont été développés dans le cadre de la chambre d'une personne. Ces objets correspondent à la volonté que nous avons de travailler avec des objets de la vie quotidienne augmentés de capteurs. Ces objets seront intégrés dans un scénario permettant de créer une application de prévention des troubles du sommeil. Nous décrivons dans cette partie les objets retenus pour ce scénario.

4.1 SmartMattress

4.1.1 Description

Le SmartMattress est un matelas composé d'un maillage de capteurs de pressions. Le placement des capteurs a dû être recherché et les différentes méthodes pour obtenir une réponse des capteurs ont été mises en place sur le matelas.

Ensuite, une amélioration des placements des capteurs a été nécessaire. Cette amélioration avait pour but d'optimiser le placement des capteurs « dans » le matelas afin d'obtenir un résultat nous permettant d'étudier, suivant les valeurs renvoyées par les capteurs, la position de la personne sur la matelas.

La Figure 3 représente le matelas sous le drap. Chaque capteur est relié à la plate-forme Phidget qui elle-même est incluse dans le matelas. On obtient ainsi un matelas autonome ayant gardé sa forme et sa fonction (c.f. Figure 4).

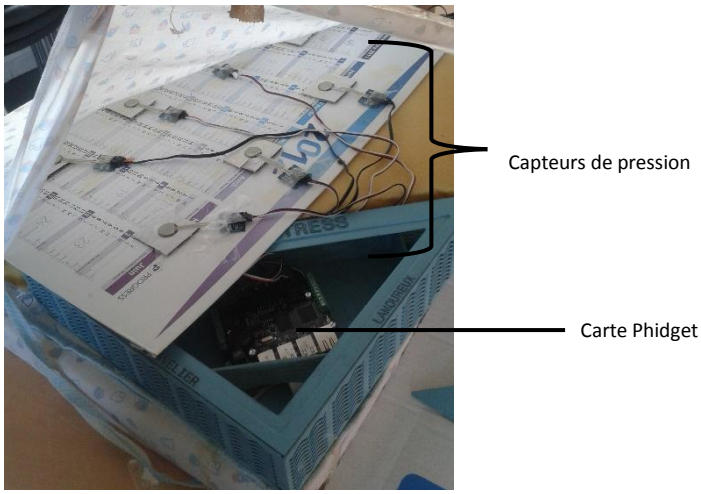
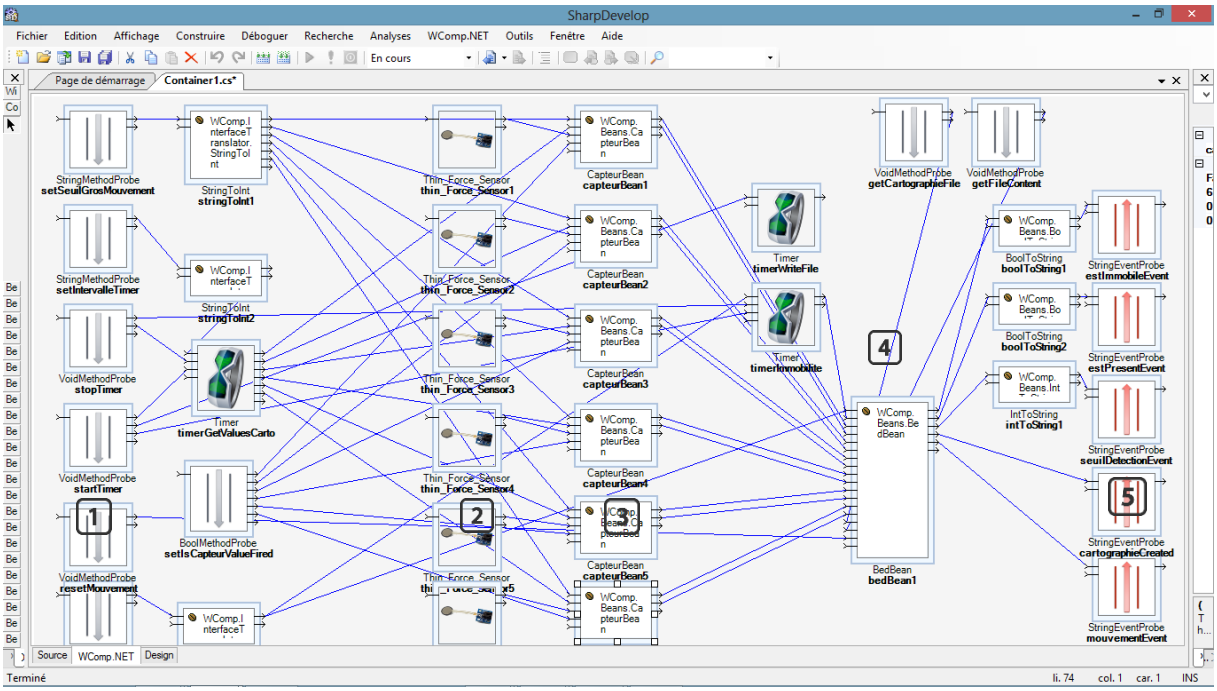


Figure 3 : Positionnement des capteurs du matelas

Figure 4 : SmartMattress finalisé

4.1.2 Assemblage du SmartMattress



1 Dans cette colonne se trouve les appels de méthodes qui feront partie de l'interface UPnP du service lié au Container WComp

2 Tous les beans présents dans cette colonne sont ceux qui représentent tous les capteurs de pression branchés sur la Phidget du SmartMattress.

3 Ces beans sont les beans que j'ai dû créer pour jouer le rôle de filtre, que ce soit pour la valeur du capteur aussi bien que pour les informations qu'on veut en tirer (numéro de port du capteur et la valeur renvoyée).

4 Ce bean est celui qui contient la logique mise en oeuvre et permettant l'interprétation des valeurs reçues par les capteurs. En effet, plusieurs événements sont renvoyés si un mouvement est survenu sur le matelas, ou encore si la personne est présente sur le matelas mais immobile.

5 Cette colonne regroupe tous les événements renvoyés et visibles, par cet assemblage, dans l'interface du service UPnP correspondant à cet assemblage.

4.2 SmartLight

4.2.1 Description

Pour la conception d'une lampe, la Smart AirLight et la Snorlex issues du livrable précédent ont été reprises. Il a fallu décider laquelle des deux garder et si des fonctionnalités pouvaient être reprises de celles non-utilisées. Le choix s'est porté sur la Snorlex, renommée SmartLight, contenant un capteur sonore et un capteur sensitif permettant d'allumer et d'éteindre la lampe en la « touchant » (c.f. Figure 5).

Aucune application dédiée n'a été développée pour cet objet. Nous pouvons directement relier la lampe au Plotter déjà existant pour visualiser la courbe du bruit pendant le sommeil de l'individu.

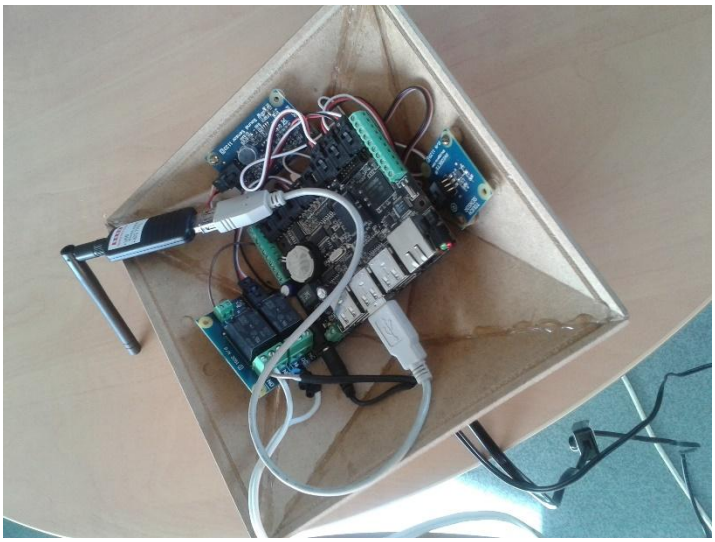


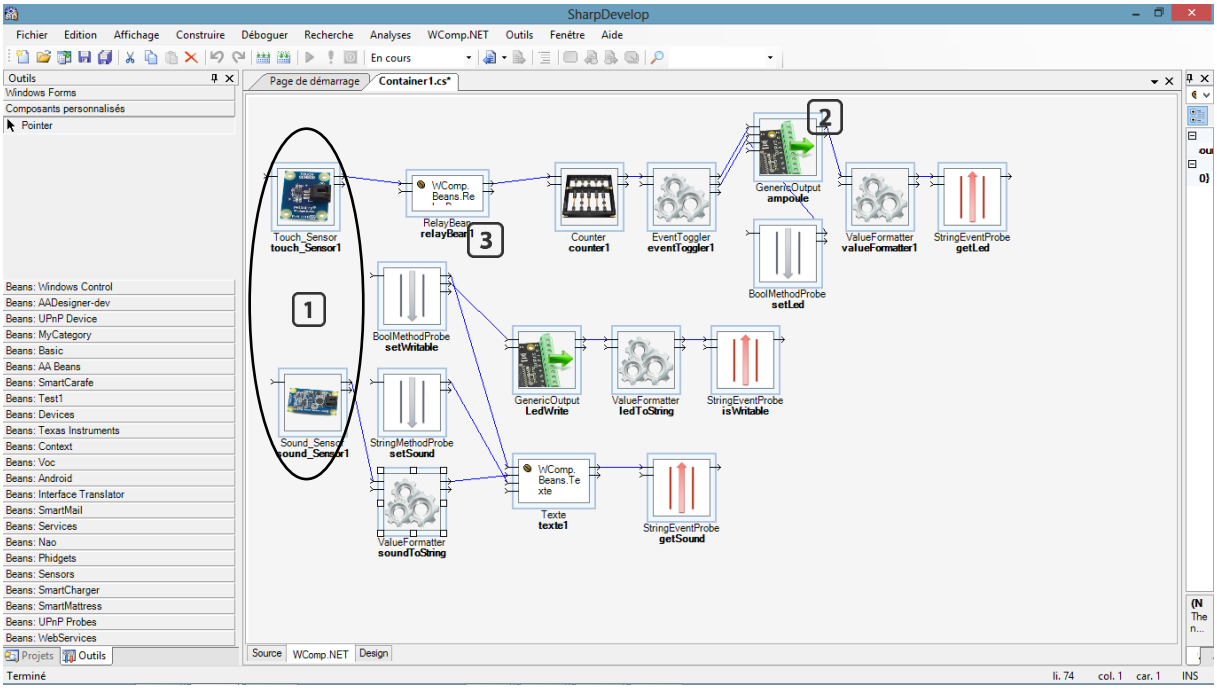
Figure 5 : Aperçu de la Phidget de SmartLight

La Figure 5 montre la phidget mise à l'intérieur de la lampe, on peut y voir une clef wifi, un relai pour l'ampoule de 240V, un capteur sonore en haut et un capteur sensitif à droite.

Le résultat final de la SmartLight est représenté sur la photographie de la Figure 6. Sur le devant, un dessin de marche/arrêt est dessiné puisqu'elle s'allume et s'éteint grâce au touché.

Figure 6 : SmartLight finalisée

4.2.2 Assemblage de la SmartLight



1 Les deux composants graphiques, représentant les deux capteurs présents sur la Phidget de la lampe.

2 Bean représentant la sortie digitale où est branchée l'ampoule

3 Bean créé pour filtrer les valeurs reçues par le capteur de toucher.

4.3 SmartCharger

4.3.1 Description

Le SmartCharger est un objet dont la fonctionnalité principale est de pouvoir connecter un téléphone afin de le recharger. Des capteurs supplémentaires ont été ajoutés comme un capteur de qualité d'air, un capteur d'humidité et de température, et un capteur de luminosité. Pour éviter un flux d'informations trop important sur le réseau, un prétraitement et un regroupement des informations sont effectués pour limiter les envois de messages sur le réseau trop nombreux. De plus, une autre idée était que ce chargeur pouvait faire lumière d'ambiance ; une guirlande de leds RGB a donc été ajoutée.

Sur la carte Phidget de la Figure 7, on peut voir quatre capteurs branchés sur les entrées analogiques (un capteur de son, un capteur de température, un capteur d'humidité et un capteur de luminosité) deux clefs usb l'une pour la wifi l'autre pour la qualité de l'air en fonction du CO2 (Figure 9). Sur la gauche de l'image, on peut aussi voir une guirlande de leds branchée sur les sorties digitales de la carte. La Figure 8 présente le capteur de qualité d'air connecté en USB à la carte Phidget. La Figure 9 présente l'objet finalisé.

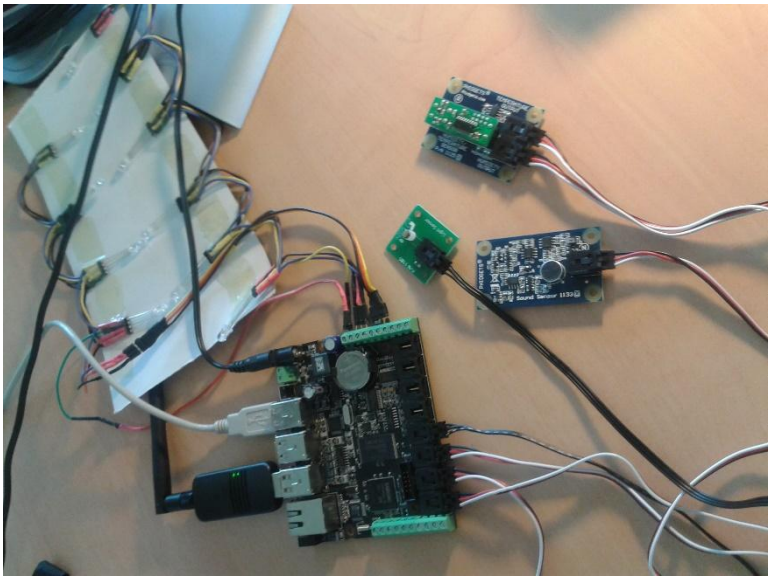
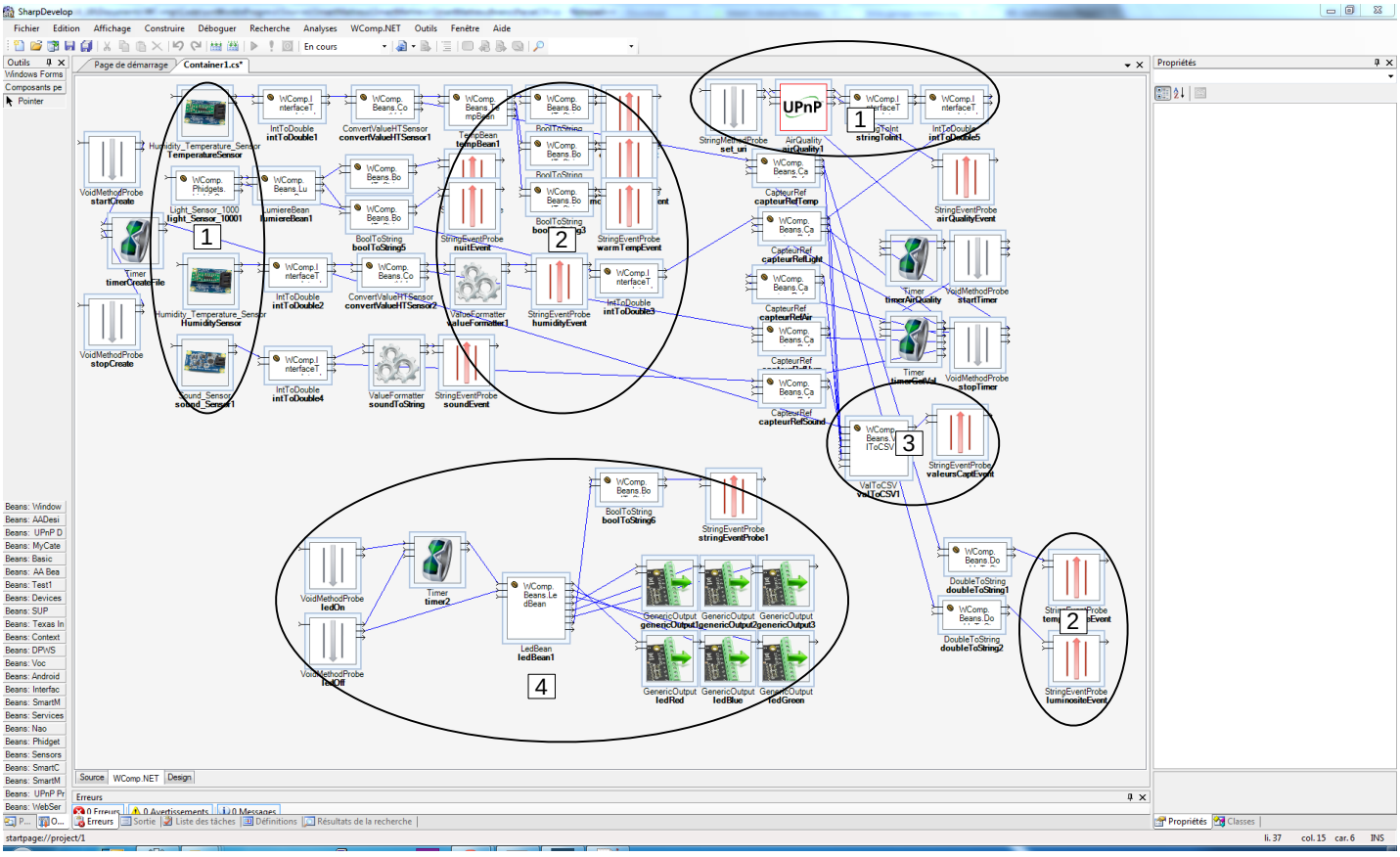


Figure 7 : Phidget du SmartCharger

Figure 8 : Capteur de qualité d'air connecté en USB

Figure 9 : SmartCharger finalisé



4.3.2 Assemblage du SmartCharger

1 Composants représentants les capteurs sur la carte Phidgets

2 Evènements qui renvoient les différentes valeurs des capteurs de manière individuelle

3 Composant qui regroupe toutes les valeurs afin de les envoyer en une seule fois
4 Composants pour gérer la guirlande de leds

4.4 Application Android Réveil

Afin de compléter le scénario, nous avons développé une application Android permettant de simuler la gestion du Réveil sur le téléphone. Cette application Android Réveil lance un service UPnP. Pour cela, les codes sources déjà existant d'une application réveil ont été repris et l'ajout du service UPnP effectué.
L'application étant un service UPnP à elle toute seule, nous pouvons ainsi la considérer comme un objet communicant. Cependant, comme il n'avait pas de capteurs externes comme les autres objets décrits précédemment, il n'a pas été nécessaire d'embarquer sur le téléphone un container WComp, l'application se suffisant à elle-même.

5 Conclusion

Après l'étude des différents objets de la vie quotidienne effectuée dans le livrable précédent, nous avons pu sélectionner une sous-partie de ces objets afin de les rendre robuste et adaptés à un scénario sur la prévention des troubles du sommeil. Pour le développement de ces objets, nous avons utilisé la plateforme logiciel WComp ainsi que des plateformes matérielles Phidgets pour les interactions physique au niveau de l'objet. Chaque objet propose alors un service et utilise le protocole UPnP afin de se déclarer sur le réseau et de déclarer son service.

Les objets fonctionnent donc en total autonomie et offre à la fois un service intrinsèque à leur état d'objet de la vie quotidienne, mais aussi des fonctionnalités supplémentaires issues l'augmentation de l'objet par des capteurs.

A partir de ces objets, nous devons mettre en place un scénario et approfondir l'utilisation de la plateforme WComp afin de permettre la construction d'une application dans laquelle ces objets seront inclus. Cette application ne sera pas figée, puisqu'elle devra s'adapter en fonction de l'apparition ou la disparition des objets dans l'environnement.

6 Bibliographie

[1] Bernheim Brush A.J., Filippov E., huang D ;, Jung J., Mahajan R., Martinez F., Mazhar K., Phanishayee A., Samuel A., Scott J. and Preet Singh R. Lab of Things : A Platform for Conducting Studies with Connected Devices in Multiple Homes. The *2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (UbiComp 2013), 2013.

[2] Blackstock M. and Lea R. Toward Interoperability in a Web of Things. The *2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (UbiComp 2013), 2013.

[3] Bovet G. and Hennebert J. Offering Web-of-Things Connectivity to Building Networks. The *2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (UbiComp 2013), 2013.