LEADYS

Rapport De Projet

Etude du Leap Motion et implémentation d'une application

Auteurs :
Mickael Campmas
Elsa Martel
Nicolas Noël
Nicolas Poelen

Encadrant :
Eric Bourreau

Dans le cadre du projet TER, Licence 3 Informatique.
Année universitaire 2014-2015

Remerciements

Nous tenons à remercier Monsieur Eric Bourreau pour son temps, ses idées et ses conseils qui nous ont permis de structurer nos pensées tout au long du projet.

Nous souhaitons aussi remercier Pierre-Gilles Leymarie, développeur du projet Gladys, pour la mise à disposition de son projet ainsi que le temps qu'il nous a accordé pour répondre à nos questions.

Enfin, nous voudrions remercier Mili-Mona Salokannel pour la réalisation du logo du projet.

Table des matières

Glossaire

Circle gesture
: Mouvement circulaire horizontal de la main au-dessus du Leap Motion.

GPIO
: Les ports GPIO (General Purpose Input/Output, littéralement Entrée/Sortie pour un Usage Général) sont des ports d'entrée/sortie très utilisés dans le monde des microcontrôleurs.

IFTTT
: Web service permettant de créer des chaînes d'actions conditionnelles, déclenchées par des changements sur d'autres web services (IF This Then That).

Key tap gesture
: Mouvement consistant à mimer avec un doigt une frappe de clavier au-dessus du Leap Motion.

Leadys
: (Leap motion & glaDYS). Ensemble du projet tutoré, consistant en une interface de domotique, basé sur le Leap Motion et d'un assistant domotique intelligent Gladys.

NPM
: Gestionnaire de paquets officiel pour Node.js (fausse abréviation de Node Package Manager).

Phillips Hue
: Dispositif d'ampoules LED relié en WiFi à une passerelle, elle-même reliée à son routeur/Box. Ces ampoules de 8,5Watts sont composées de 3 LED RGB capables de reproduire 16 millions de nuances de couleurs.

Screen tap gesture
: Mouvement consistant à mimer d'un doigt le toucher d'un écran tactile au-dessus du Leap Motion.

SDK
: Acronyme anglais pour Software Development Kit. Il s'agit d'une « trousse de développement logiciel », un ensemble d'outils permettant à des développeurs de créer de nouveaux logiciels. Elles contiennent notamment de nombreux outils et codes déjà optimisés par un ensemble de programmeurs ou par les fabricants d'un périphérique. Dans le cadre de notre projet, nous utilisons notamment le SDK Leap Motion.

Swipe Gesture
: Mouvement vertical ou horizontal de la main au-dessus du Leap Motion.

Introduction

Sorti en 2010, le Leap Motion est un capteur de mouvements qui permet d'interagir avec un ordinateur. Il peut potentiellement remplacer une souris, une manette de jeu voire un clavier. Il capte les mouvements des mains, des doigts et des phalanges, tout en prenant en compte leurs positions dans l'espace (distance par rapport au périphérique), l'angle des articulations... Ceci est rendu possible par la présence de 2 caméras et 3 LED infrarouge.

Ainsi, l'ordinateur est capable de représenter les mains dans un environnement en trois dimensions et d'effectuer des actions en fonction des mouvements de celles-ci. Nous avons donc étudié les avantages et les inconvénients de cet objet. Ensuite, nous avons cherché une application possible en nous inspirant de ce qui existe déjà.

Nous avons choisi de développer une application qui permet de contrôler des objets connectés situés dans une maison. Celle-ci va permettre de contrôler de la musique, des lampes connectées à l'aide de différents gestes inspirés de la langue des signes et aussi de consulter la météo. Elle pourra être utilisée sur les différents systèmes d'exploitation car nous avons choisi Java comme langage pour la portabilité de l'application.

Le but de ce projet est ainsi de démontrer qu'il est possible d'allier le Leap Motion à d'autres appareils connectés afin d'améliorer l'expérience utilisateur et d'en faire une interface homme-machine simple et « naturelle ». Il sera d'ailleurs associé à un Raspberry pi où un assistant domotique intelligent Gladys sera installé et prendra en compte les mouvements qui seront effectués au-dessus du périphérique.

Après une présentation du domaine d'étude, nous étudierons la problématique liée à l'utilisation du Leap Motion et analyserons ses possibles implémentations. Enfin, nous aborderons nos choix en terme de développement avec le rapport technique.

1 Domaine d'étude

1.1 Interactivité et objets connectés

L'avancée technique dans le domaine des objets connectés, et leur présence de plus en plus affirmée sur le marché, a fait naître le besoin de mettre différemment en interaction l'homme et la machine. Les techniques habituelles de contrôle de la machine ne sont pas toujours adaptées aux exigences des utilisateurs de ces nouveaux objets. Les ordinateurs et les périphériques qui leur sont habituellement associés (souris, claviers et autres) sont aujourd'hui largement concurrencés par les tablettes, les smartphones, mais aussi toutes sortes d'objets du quotidien qui désormais s'intègrent au sein d'un réseau domestique. Ces objets ont réinventé notre manière d'interagir avec la machine : tactile, reconnaissance vocale, reconnaissance de mouvements, capteurs en tous genres... C'est dans ce cadre que nous avons voulu penser le Leap Motion, non seulement comme un capteur de mouvements mais aussi comme un de ces objets connectés.

Il est à noter que la reconnaissance de mouvements, bien qu'étant déjà un domaine d'étude puisant sa source dans les années 80, a connu des avancées considérables à l'aube des années 2000, et s'est largement démocratisée cette dernière décennie notamment avec le succès de la Kinect. Le Leap Motion s'inscrit donc aussi dans la lignée de ces capteurs accessibles au « grand public » et non à un certain nombre d'initiés et ce malgré sa vocation qui se veut encore expérimentale et en attente de développements.

1.2 Relier les objets connectés

La multiplication des objets connectés a obligé les développeurs à repenser le réseau domestique qui autrefois, représentait la plupart du temps un simple lien entre des machines ayant des utilisations restreintes au « domaine informatique » et qui aujourd'hui, tend à devenir un véritable réseau de « domotique ». Les routeurs ou les ordinateurs personnels ne sont pas forcément les plaques tournantes les mieux adaptées à ces réseaux. De plus en plus de serveurs et de centrales de contrôle dédiés gèrent les diverses interactions des utilisateurs avec les objets connectés.

L'arrivée sur le devant de la scène du Raspberry Pi, un nano ordinateur conçu par David Braben, a été la source de nombreux développements et sa versatilité l'a mis en position de pouvoir jouer le rôle de « plate-forme de contrôle » reliant ces divers objets connectés. Aussi il existe d'ores et déjà des développements logiciels spécialisés dans le domaine de la domotique, conçu pour fonctionner sur un Raspberry Pi, tel que Gladys le groom connecté que nous avons utilisé au cours de ce projet, et dont le fonctionnement sera explicité plus en détail dans la partie suivante.

1.3 Le Multimédia au cœur du projet

Le Leap Motion et le Raspberry Pi sont tous les deux impliqués dans de nombreux projets liés au secteur du multimédia. En effet, ce secteur d'activité est très avide de

nouveaux moyens d'interaction et contrôle, d'autant que ses finalités sont variées et font souvent appel à la créativité de l'utilisateur. Aussi, de nombreuses applications amenées à inclure des interactions sonores ou visuelles, utilisent le Leap Motion pour son interactivité ludique avec l'utilisateur, ou le Raspberry Pi pour sa versatilité et sa portabilité qui lui permettent de s'adapter facilement au contexte d'une installation. Certains sites recensent même ces projets multimédias, et présentent des innovations intercroisant divers objets connectés. Notre projet fait appel à ce lien fort, existant entre les objets connectés et le secteur du multimédia, puisqu'il se propose de contrôler un lecteur audio et un système d'éclairage.

2 Problématique

2.1 Le Leap Motion

2.1.1 Limitations sur le marché

Le Leap Motion souffre depuis son lancement d'une difficulté à s'implanter. Des difficultés qui s'expliquent notamment par un manque de couverture médiatique : il reste encore un périphérique méconnu du grand public alors que son utilisation et même son packaging, l'y destine.

Il est notamment difficile de lui trouver une utilisation pour laquelle il deviendrait véritablement utile, un domaine où il se substituerait à un outil existant en offrant une qualité et un confort supérieurs, voire un domaine complètement nouveau où il deviendrait indispensable.

Un autre problème rencontré par le Leap Motion et qui est directement relié au précédent est celui du manque d'applications et de développeurs. En effet, le manque de domaines d'applications trouve aussi ses racines dans un manque de développeurs et d'idées dans un domaine relativement jeune (réalité virtuelle/motion-tracking), et cela se remarque par un manque de diversité dans les applications disponibles sur la boutique en ligne : on y retrouve très souvent les mêmes (souris virtuelle, jeux-vidéo, dessin...).

2.1.2 Limitations physiques

Problèmes durant les tests

D'autres limitations peuvent se faire ressentir durant l'utilisation : de par son fonctionnement, le Leap Motion ne perçoit qu'une image en 2D. Si on présente ses mains ouvertes au-dessus du Leap Motion, paumes vers le sol, l'appareil ne détectera que le premier niveau, la partie inférieure des mains. Tout mouvement effectué au-dessus des mains ne sera pas enregistré, il en est de même si des doigts se superposent, c'est alors une « estimation » de la position des doigts qui est alors calculée par rapport à la dernière position connue, qui mène parfois à des situations ambigües telles que des articulations déboîtées. On peut alors difficilement faire reconnaître au Leap Motion des signes trop complexes, un interpréteur complet de langue des signes devient alors presque impossible.

Sa trop grande précision l'amène parfois à interpréter des mouvements involontaires, donnant alors des réactions qui n'ont pas été voulues au départ. Parfois, le périphérique ne reconnaîtra pas un mouvement, tout simplement à cause d'une mauvaise simulation d'une partie de la main qu'il ne voit plus ou bien parce que le mouvement n'était pas assez rapide pour être reconnu. Un certain apprentissage et un paramétrage plus fin des mouvements (même ceux déjà existants dans la SDK*) est alors nécessaire. Ce même défaut l'empêche d'ailleurs d'être « mobile » pour être par exemple accroché à un T-shirt.

Enfin, l'analyse des mouvements nécessite une grande puissance de calcul que ne peut offrir une tablette ou de l'informatique embarquée. Il ne peut donc être déporté et utilisé sur un Raspberry Pi, même dans sa deuxième version.

Fatigue

Un autre problème rencontré lors d'une utilisation continue est la fatigue inhérente au fait de devoir maintenir ses bras et mains en l'air constamment. Cet inconvénient, appelé syndrome« Gorilla arm »[1], limite donc l'appareil à des mouvements brefs pour éviter une gêne chez l'utilisateur. Pour cette raison et les raisons citées précédemment, il est impossible de l'utiliser dans un cadre thérapeutique[2].

2.2 Objets connectés

2.2.1 Interface homme-machine

De plus en plus présents, les objets connectés connaissent un plein essor grâce à l'émergence des smartphones et des réseaux domestiques. Les objets de domotiques connectés se voient alors dotés d'interfaces nombreuses et variées : interface web, applications mobiles, contrôle vocal...

Limites du contrôle vocal

Un contrôle vocal fiable, fluide et réactif est le but rêvé dans le domaine des interactions homme-machine. Toutefois, le système n'est pas encore véritablement précis et 100% fiable (notamment pour la langue française) et l'interaction avec un ordinateur via ce système reste encore très primaire. Il n'est par exemple pas possible de communiquer et d'activer des objets dans un domicile de manière naturelle, il faut encore se contenter de recherches internet et de commandes vocales limitées qui ne sont pas toujours comprises par la machine.

Ce dispositif reste, bien évidemment, inaccessible aux personnes sourdes/muettes : il faut alors penser à de nouvelles manière d'interagir avec son ordinateur.

2.2.2 Gladys

Principe : un groom connecté

Gladys[3] est un assistant domotique intelligent open-source développé par Pierre-Gilles Leymarie. Ce programme, fonctionnant sur un Raspberry Pi, propose une alternative à une intelligence artificielle contrôlée par la voix en repensant le problème : ce n'est pas à l'utilisateur de penser et donner des ordres, mais à l'assistant de prédire ses besoins et de réagir sans intervention de l'homme.

1. David Pogue, 2012. *Why Touch Screens Will Not Take Over*, [online] Scientific American, <http ://www.scientificamerican.com/article/why-touch-screens-will-not-take-over/>[consulté le 18 janvier 2015]

2. NaturalPad, 2014. *[TEST] Le Leap Motion*, [online] <http ://naturalpad.fr/leap-motion/>[consulté le 8 février 2015]

3. Gilles Leymarie, 2015. *Gladys Project*, [online] <http ://gladysproject.com/>[consulté le 5 février 2015]

Prenons un scénario classique : A 7h, l'utilisateur doit être réveillé. Gladys démarre un réveil avec une musique et lui donne la météo. Il est 18h, l'utilisateur rentre du travail. Gladys va, à son arrivée, allumer les lumières et démarrer une liste de musique.

L'assistant propose ainsi une manière de centraliser et connecter les objets connectés, tout en automatisant leurs activations et leurs associations. Il peut aussi être utilisé avec des services similaires tels qu'IFTTT* [4].

Ajout d'une interface plus naturelle : Leadys*

Toutefois, un problème peut se poser : qu'en est-il des imprévus ? Comment pourrait-on éteindre et rallumer les lumières, arrêter et reprendre la musique à la volée ?

Gladys propose d'ores et déjà une interface Web permettant de répondre à ce besoin. Toutefois, ne pourrait-on pas disposer d'un autre système, accessible plus rapidement et proposant une interaction plus naturelle et personnelle avec Gladys ?

C'est ici que le Leap Motion intervient : grâce à de simples mouvements et signes, il serait possible de réaliser ces actions sans avoir à passer par une interface web. Le périphérique serait connecté à un ordinateur et communiquerait via le réseau Wi-Fi avec Gladys, palliant ainsi le manque de puissance de calcul fourni par le Raspberry Pi pour faire fonctionner le Leap Motion.

En s'inspirant de la langue des signes française, nous pouvons ainsi établir des signes simples et facilement reconnaissables par le Leap Motion afin d'indiquer à Gladys quel objet nous souhaitons utiliser : nous avons ici un **sujet**. En utilisant la collection de mouvements comprise dans la SDK*, nous pouvons indiquer à Gladys quelle action nous souhaitons effectuer avec cet objet, nous avons alors un **verbe**. En ajoutant éventuellement un chiffre (nombre de doigts) pour, par exemple, indiquer quelle lampe nous souhaitons activer, nous avons un **complément**.

De cette manière nous pouvons construire des **phrases** simples et ainsi interagir de manière naturelle avec Gladys, tout en proposant une alternative aux commandes vocales aux personnes sourdes/muettes et en résolvant les problèmes rencontrés par le Leap Motion durant les phases de tests grâce à l'utilisation des signes simples.

4. IFTTT, 2011. *About IFTTT.* [online] <https ://ifttt.com/wtf>[consulté le 2 mars 2015]

3 Choix et rapport technique

Notre projet comporte deux parties : une partie serveur sur le Raspberry Pi qui régit la domotique, ainsi qu'une partie client qui reconnaît les mouvements envoyés par le Leap Motion et qui envoie les données au serveur. Nous avons décidé de nous séparer en deux groupes : deux personnes s'occupant du client et les deux autres du serveur.

3.1 Le SDK

3.1.1 Classes du Leap Motion

Le contrôleur (classe `Controller`) se connecte au Leap Motion à sa création, il récupère alors toutes les données envoyées par le Leap Motion.

L'écouteur (classe `Listener`) fournit un mécanisme basé sur les événements pour répondre aux changements d'état du contrôleur, il n'est pas obligatoire. Plusieurs écouteurs peuvent s'attacher au contrôleur, celui-ci appelle alors les méthodes des écouteurs lorsqu'un événement survient. Dans notre programme, nous créerons une nouvelle classe héritant de `Listener`.

Les mains sont représentées par une classe `Hand` qui possède des doigts (`Finger`) et le bras (`Arm`), ces doigts possèdent aussi des phalanges (`Bone`). Une classe `Tool` permet de gérer les outils. Chacun de ces objets enregistre la position et l'angle de ce qu'il représente.

Le SDK* Leap Motion définit quatre classes pour les mouvements : `CircleGesture`* (mouvement circulaire), `SwipeGesture`* (mouvement rectiligne), `KeyTapGesture`* (mouvement du doigt comme si on frappait une touche) et `ScreenTapGesture`* (mouvement du doigt en avant, comme si on touchait un écran). Il est possible de créer ses propres mouvements en utilisant les informations des différents objets représentant les mains et/ou outil.

À chaque rafraîchissement du Leap Motion, une nouvelle `Frame` est créée au sein du contrôleur. Cet objet possède la liste des mains, des outils et des mouvements à un instant t. Les *Frames* possèdent d'autres données mais celles-ci ne seront pas utilisées dans notre programme.

3.1.2 Choix du langage final

Le SDK* [1] est disponible en plusieurs langages : JavaScript, C#, C++, Java, Python, Objective-C. Il peut aussi être utilisé avec les moteurs Unity ou Unreal. Dans tous les langages, l'architecture est très similaire et donc facilement adaptable.

Pour le client, le Leap Motion est connecté à un ordinateur. Pour que notre programme puisse être utilisable par un maximum de monde, il faut que celui-ci puisse être exécuté sur les trois principaux systèmes d'exploitation (Windows, OS X et Linux). Le choix du langage s'est donc porté sur le C++ car c'est un langage multiplateforme, performant, fortement utilisé et que nous connaissons bien.

1. Leap Motion, 2015. *Installer & SDK* Getting Started*, [online] <https ://developer.leapmotion.com/getting-started>[consulté le 18 janvier 2015]

3.2 Gladys - Partie serveur

3.2.1 Prérequis matériels

Au niveau matériel, il est recommandé de se procurer un Raspberry Pi version 2 type B, notamment pour le gain de performance qu'apporte le processeur Quad-Core et la mémoire vive supplémentaire, qui peuvent potentiellement permettre le support de modules et d'application plus lourd dans le futur (ainsi qu'à des utilisations dans d'autres projets). Par ailleurs, cette version est commercialisée au même prix que la version 1 et tendra à la remplacer ainsi que ses accessoires. Le reste du matériel nécessaire au fonctionnement de Gladys se limite à de simples haut-parleurs (avec connecteur composite ou 3.5 mm).

3.2.2 Paquets nécessaires et installation

Le programme d'installation de Gladys est disponible sur le site du développeur et est à décompresser dans le dossier */var/www/* du Raspberry Pi. Le programme est majoritairement composé de fichier PHP / JavaScript, mais on peut aussi trouver quelques scripts python servant principalement à la communication avec des modules externes (notamment Arduino pour les capteurs de mouvements). Le programme propose une interface Web utilisant le framework bootstrap, le rendant accessible aussi bien sur ordinateur que tablette ou smartphone. Il est recommandé d'utiliser un système d'exploitation Raspbian pour Raspberry Pi à jour, notamment en raison de l'accessibilité et de la disponibilité des différents paquets spécifiques au Raspberry Pi, qui peuvent parfois différer d'autres distributions Linux en raison d'une communauté plus présente et d'une architecture CPU différente (ARM-7).

Pour fonctionner, ce logiciel nécessite les paquets et prérequis suivants :

- NodeJS et Apache afin d'exécuter le code côté serveur.
- Éventuellement WiringPi et rcswitch-Pi pour contrôler les GPIOs*.
- Le lecteur MOC pour la musique
- Les modules npm « xmlhttprequest » et« googlemaps » (AJAX et géolocalisation)
- L'utilisateur *www-data* ajouté au groupe audio
- L'utilisateur *www-data* ajouté aux sudoers (notamment pour les fonctionnalités de l'interface web)

Tout ceci peut être directement réalisé en exécutant avec les droits sudoer le script d'installation « install.sh » contenu dans le répertoire racine de Gladys.

Toutefois, quelques configurations supplémentaires sont à réaliser :

- Premièrement, il est parfois nécessaire de créer un dossier cache pour le serveur de musique de MOC et de lui donner des droits de lecture et écriture suffisants, ceci afin de permettre l'exécution du serveur de musique et la lecture des playlists. Pour cela, il suffit d'effectuer les commandes suivantes dans un terminal : *cd /var/www*
  sudo mkdir .moc
  sudo chmod -R 755 .moc

— Un autre problème rencontré avec MOC provient du code source. En effet, une erreur arithmétique a été commise par un développeur dans les valeurs de réétalonnage du mixeur audio, résultant en des valeurs numériques différentes dans MOC et l'alsamixer. L'erreur a été rapportée mais la correction n'a pas encore été ajoutée à la branche de distribution du paquet. Afin de contourner ce problème et de permettre d'augmenter le volume sonore du Raspberry Pi dans d'autres applications, nous avons décidé d'inclure un script dans le répertoire */usr/local/bin/*. Ce fichier, nommé « vol », permet d'augmenter le volume général de l'alsamixer via de très simples commandes terminales telles que vol +, vol – ou vol 60. Le code de celui-ci est visible dans l'annexe.

3.2.3 Modifications des classes/contrôleurs

Afin d'offrir un maximum d'interactions avec le Leap Motion, quelques méthodes ont été ajoutées aux contrôleurs et classes de Gladys. Ces commandes sont appelées via requêtes GET par le client Java et leurs codes sont disponibles en Annexes, à raison d'un contrôleur et d'une classe par module.

Dans le cas du contrôle des lampes et de la gestion de la météo, de nouvelles classes et contrôleurs ont été implémentés. Le contrôle des lampes fait appel à l'API Phue dont l'implémentation est détaillée dans la partie suivante. Le service de météo fait quant à lui, appel à l'API json d'Open Weather Map, une application web permettant de générer des bulletins météo pour les coordonnées fournies sur une période paramétrable.

3.2.4 Phillips Hue et choix d'API

Les lampes connectées Phillips Hue* offrent diverses APIs dans une multitude de langages afin de pouvoir contrôler l'intensité et la couleur des lampes. Afin de s'intégrer naturellement dans le projet Gladys, nous avons choisi la librairie Phue [2], en raison de la richesse offerte par cette dernière.

Avant d'intégrer cette librairie, il est nécessaire de résoudre les dépendances avec Composer puis d'utiliser les scripts présents dans le dossier *bin/* pour détecter les lampes et créer un utilisateur pour Gladys. Il sera ensuite possible d'utiliser cet utilisateur pour connecter Gladys aux lampes, récupérer les données relatives à l'intensité lumineuse, la couleur et l'état des lampes et ainsi agir sur celles-ci (voir Annexe 9).

3.3 Partie Client

3.3.1 Client C++

Pour écrire ce programme, nous nous sommes inspirés d'un projet écrit en Python permettant d'utiliser VLC à l'aide du Leap Motion [3]. Celui-ci proposait les mouvements suivants :

2. GitHub, 2013. *sqmk/Phue.*[online] <https ://github.com/sqmk/Phue>[consulté le 18 avril 2015]
3. GitHub, 2013. *michieldwitte/Leap Motion - VLCController.* [online] <https ://github.com/michieldwitte/LeapMotion-VLCController>[consulté le 5 février 2015]

— balayer vers la gauche pour passer à la musique suivante,

— balayer vers la droite pour revenir à la musique précédente,

— balayer vers le haut pour monter le volume,

— balayer vers le bas pour baisser le volume,

— « key-tap » pour mettre en pause.

Nous avons alors adapté le programme à notre projet pour pouvoir contrôler la musique avec le Leap Motion et avons ajouté deux mouvements :

— Faire des cercles dans le sens horaire pour avancer la lecture de 10 secondes,

— Faire des cercles dans le sens antihoraire pour revenir 10 secondes en arrière.

— « screen-tap » arrêterait la lecture.

Notre programme enverra des commandes au mocp car celui-ci est déjà présent sur Gladys. De plus, mocp est plus léger que VLC.

Le code source principal se situe dans la méthode `onFrame()` de `MyLeapListener`, une classe héritant de `Listener`. Celle-ci se contente simplement de vérifier si les mouvements précédemment cités sont reconnus et d'envoyer des commandes avec une pause d'une seconde après chaque envoi. La connexion au serveur est gérée par une classe `VLCController` renommée par la suite plus logiquement `MocpController`.

3.3.2 Client Java

De nombreux problèmes de compilation, de portabilité et de compatibilité entre librairies survenus lors du développement du programme sous Windows nous ont conduit à choisir un autre langage. Nous avons donc réécrit le programme en Java en raison de la portabilité de ce dernier et de sa syntaxe suffisamment proche du C++.

MyLeapListener : Transitions de C++ à Java

— La classe `MyLeapListener` possède une méthode `isConnected()` qui permet de savoir si le Leap Motion est connecté.

— `onInit()` initialise les différents attributs de classe :

— `alarm` qui permet de sauvegarder l'heure actuelle en milliseconde.

— `subjectAcquired` est initialisé à faux. Elle va permettre de savoir si le sujet de la phrase a été reconnu.

— `initModules()` va initialiser l'attribut `availableModules`, une `HashMap`, qui va associer la position à un objet de la classe à appeler.

— `setSession()` prend en paramètre un objet Session, initialise la variable session. Elle fait ensuite appel à la fonction `initModules()`.

— La méthode `setCurrentModule()` prend en paramètre une énumération de module. Ce nom de module va alors être recherché dans `availableModules` et cela va retourner l'objet créé lors de l'initialisation de la *hashMap*. Afin que les commandes soient envoyées au bon contrôleur, nous allons changer celui-ci.

— `isSubjectAcquired()` est une méthode qui retourne soit vrai ou faux suivant la valeur de l'attribut `subjectAcquired`.

— `setSubjectAcquired()` prend en paramètre un booléen qui va être affecté à l'attribut `subjectAcquired`.

— La fonction `onConnect()` prend en paramètre un `Controller`. Elle va associer des gestes reconnus au Controller et les configurer comme par exemple la vitesse du geste `Swipe` et de `KeyTap`. Elle va initialiser l'attribut `isConnect` à vrai.

— Les méthodes `onDisconnect()` et `onExit()` prennent en paramètre un `Controller`. La première affiche un message si le Leap Motion est déconnecté. Alors que pour la deuxième, l'affichage se fera si l'on quitte le programme.

— La méthode `sendEvent()` prend 2 paramètres, un message et une commande. Elle va affecter à alarm l'heure courante en lui ajoutant 600 millisecondes et afficher le mouvement reçu dans le terminal. La commande va être envoyée à Gladys. Nous mettons ensuite l'attribut `subjectAcquired` à faux.

Signes et système de phrases

Pour éviter que le programme ne reconnaisse des sujets par inadvertance, la classe `MyLeapListener` stockera les sujets reconnus sur les 12 dernières *frames* et considèrera qu'un sujet est effectué par l'utilisateur que s'il est reconnu sur au moins 10 *frames* (Les valeurs peuvent évidemment être modifiées).

— `moduleFrame` est le tableau qui stocke les sujets (`ModuleName`) reconnus aux dernières frames.

— `counter` est la variable qui indique la position courante dans le tableau (incrémenté à chaque *frame* et revient à 0 une fois arrivé à la fin du tableau.

— `nbFramePerModule` est une table de hachage qui indique pour chaque `ModuleName` combien il y a d'occurrence dans le tableau. Sa présence permet d'éviter de multiples boucles sur `moduleFrame`.

Le code `onFrame()` est maintenant séparé en quatre nouvelles méthodes :

— `subjectOfFrame()` : Récupère la frame courante et met à jour `moduleFrame`, counter et `nbFramePerMolule`.

— `detectSubject()` : Récupère la frame courante et appelle `subjectOfFrame()`, vérifie à l'aide de `nbFramePerModule` si un sujet est reconnu et met à jour les différentes variables (`subjectAcquired`, module courant et alarme) pour passer en mode « verbe » si oui.

— `timeElapsedSubject()` : Vérifie que le temps pour reconnaître un mouvement n'est pas dépassé et met à jour les variables pour repasser en mode « sujet » dans le cas contraire.

— `verb()` : Récupère la frame courante, elle fait appel à `sendEvent()` et repasse en mode « sujet » si elle reconnaît un mouvement défini pour le module courant.

— La méthode `onFrame()` fera d'abord appel à `timeElapsedSubject()` si on est en mode « verbe » puis à `detectSubject()` ou `verb()` selon le mode.

Sessions : connexion et émission

Le client Java doit, pour communiquer avec Gladys, se connecter via le protocole HTTP afin d'envoyer les instructions reçues par le Leap Motion. Pour cela, il récupère dans

un premier temps les données nécessaires à la connexion, se connecte à Gladys et ensuite émet des requêtes. C'est la classe `Session` qui remplit ce rôle. Elle utilise la bibliothèque Apache *HttpComponents*[4] pour gérer le protocole HTTP.

Elle se construit avec trois paramètres : L'IP du serveur Gladys (*ip*), le pseudo de l'utilisateur sur Gladys (*pseudo*) et son mot de passe (*mdp*). Elle dispose de deux attributs supplémentaires, un `CookieStore cookies` qui sera rempli lors de la connexion à Gladys, et un `String` destination qui permettra, lors des requêtes au serveur, de spécifier l'adresse de destination de la requête.

Après construction, outre les *getter/setter*, la classe permet l'accès à deux méthodes :

`connect()` qui établit, grâce au paramètres de construction, une connexion HTTP au serveur Gladys. Une fois le lien établi, les données de connexion fournies sont sauvegardées via un cookie, en l'occurrence l'attribut `cookies` de la classe `Session`. En effet, lorsque l'authentification d'un utilisateur est réussie, Gladys fournit des informations de connexions supplémentaires qui seront alors nécessaires à chaque requête.

`sendGet(String action)` qui permet l'envoi d'un message action au serveur Gladys. C'est ce message qui sera ensuite interprété par Gladys et déclenchera les actions voulues. Pour cela, la méthode construit une requête HTTP GET, qui sera envoyée à l'adresse de destination prédéfinie (attribut destination). A noter que la requête n'est possible que si le client s'est déjà authentifié auparavant via la méthode `connect()`. La vérification de la connexion se fait par l'intermédiaire du cookie stocké dans l'instance de `Session` (`cookies`).

Modularité

La classe abstraite Module permet de gérer et d'ajouter plus facilement des modules. Elle possède l'adresse correspondant au module, une `HashMap` associant un mouvement (défini par l'énumération `Movement`) avec une commande (sous forme de `String`) et fournit les méthodes permettant de récupérer les différentes commandes ou vérifier leurs existences. Lors de la création d'un nouveau module, il faut créer une nouvelle classe héritant de `Module` et écrire sa méthode abstraite `subject()` recevant une frame et retournant un booléen selon que le signe correspondant au module soit reconnu ou non. L'adresse et les commandes sont définies en appelant le constructeur de `Module`. Il ne faut pas oublier d'ajouter son module dans l'énumération `ModuleName` et dans `availableModules` de `LeapListener`.

4. Apache, 2005-2015. *The Apache HttpComponents^{TM} project*, [online] <https ://hc.apache.org/>[consulté le 8 mars 2015]

4 Manuel d'utilisation

4.1 Pré-requis

La partie cliente de Leadys doit être installée sur une machine disposant au préalable d'une installation de la machine virtuelle Java et du logiciel Leap Motion. Lors de l'utilisation, vérifier que les périphériques audio et le Leap Motion Controller sont bien branchés et fonctionnels.

Les appareils domotique doivent être connectés sur la machine (Rasberry Pi) disposant de la partie serveur. Pour installer et configurer le serveur, veuillez consulter la documentation de Gladys en suivant ce lien : http ://gladysproject.com/developpeur.

4.2 Musique

Sujet

Pour le module `Music`, l'utilisateur doit réaliser un signe qui permet de commencer une phrase. Celui-ci correspond au sujet *Musique* qui est équivalant au mot « Musique » dans la langue des signes française. Le signe correspond à l'index tendu sur la main gauche ou droite .

Verbes

Les verbes sont :

— `Swipe Gesture [vers le haut]` : augmenter le son
— `Swipe Gesture [vers le bas]` : diminuer le son
— `Swipe Gesture [vers la gauche]` : passer à la chanson suivante
— `Swipe Gesture [vers la droite]` : passer à la chanson précédente
— `Key tap` : play/pause
— `Circle Gesture [dans le sens horaire]` : avancer
— `Circle Gesture [dans le sens antihoraire]` : rembobiner

4.3 Lumière

Sujet

Pour débuter une phrase pour le module `Light`, l'utilisateur doit effectuer un signe correspondant au sujet *Lumière* : ce signe correspond au signe du mot « Lampe » dans la langue des signes française. Ce signe s'effectue en étendant les cinq doigts de la main, paume orientée vers le périphérique.

Verbes

Les verbes disponibles pour ce module sont au nombre de cinq :

— `Swipe Gesture [vers le haut]` : augmente l'intensité de la lumière
— `Swipe Gesture [vers le bas]` : diminue l'intensité de la lumière
— `Swipe Gesture [vers la gauche/droite]` : change la couleur des ampoules
— `key tap` : allume/éteint la lumière

4.4 Météo

Sujet

Pour le sujet du module `Weather`, l'utilisateur doit effectuer le signe correspondant au sujet *Météo* : ce signe s'inspire du mot « Vent » dans la langue des signes française. Il s'effectue en étendant l'index et le majeur en V, paume orientée vers le périphérique.

Verbes

Les verbes disponibles pour ce module sont au nombre de quatre :

— `Swipe Gesture [vers le haut]` : augmenter le son
— `Swipe Gesture [vers le bas]` : diminuer le son
— `Swipe Gesture [vers la droite]` : donne la météo du lendemain
— `key tap` : donne la météo du jour

Conclusion

L'explosion du marché des objets connectés a permis l'émergence de nouvelles technologies, de nouveaux moyens de communications, aussi nous permet-elle de penser à de nouvelles approches dans le milieu des interfaces homme-machine.

Leadys s'inscrit ainsi dans cette démarche et offre une nouvelle manière, plus naturelle, d'interagir avec ces nouveaux objets du quotidien, grâce à des mouvements inspirés de la langue des signes française. Il est aussi à noter que le projet peut tout aussi bien être adapté, en adoptant la même approche et stratégie, avec d'autres objets de réalité virtuelle, réalité augmentée ou d'interface plus précise.

L'utilisation du Leap Motion permet une interaction simple et naturelle avec Gladys, la rendant accessible à tous, tant par des programmeurs chevronnés que par des utilisateurs ingénus lors de présentations ou journées portes ouvertes. D'un point de vue plus éthique, Leadys rend aussi ces technologies de contrôles accessibles aux personnes sourdes, malentendantes et muettes. Ce qui est impossible avec des commandes vocales dont le développement est toujours en cours mais dont la fiabilité n'est pas encore optimale.

Enfin, si les difficultés techniques rencontrées avec le Leap Motion nous ont contraints à choisir le confort d'utilisation via des gestes simples au détriment de mouvements plus variées, complexes et précis, cette même simplicité rend désormais le programme plus facile d'accès à d'autres développeurs. En effet, grâce au principe de modularité développé, des développeurs du projet Gladys peuvent désormais participer à l'expansion des capacités de Leadys, en leur permettant de rapidement développer de nouveaux modules contrôlant encore plus d'objets connectés. C'est d'ailleurs dans cette optique que le projet Leadys sera partagé sur le site de la communauté du projet Gladys.

Sitographie

Site	Type de site	Information recherchée
http ://naturalpad.fr/leap-motion/	Site Natural Pad	Applications thérapeutiques possibles
http ://goo.gl/62gU98	Magazine de vulgarisation scientifique	Effet « Gorilla arm »
http ://goo.gl/9azy3p	Site du projet Gladys	Projet Gladys
https ://ifttt.com/wtf	Site officiel IFTTT	Interaction entre appareils connectés
http ://goo.gl/gjYo8C	Site officiel Leap Motion	SDK & documentation Leap Motion
http ://goo.gl/D5v4Tx	GitHub du projet Leap Motion - VLCController	Code source
http ://goo.gl/kAknxn	GitHub du projet Phue	Code source
https ://hc.apache.org/	Site officiel du projet Apache HttpComponents™	Librairie & documentation

Annexes

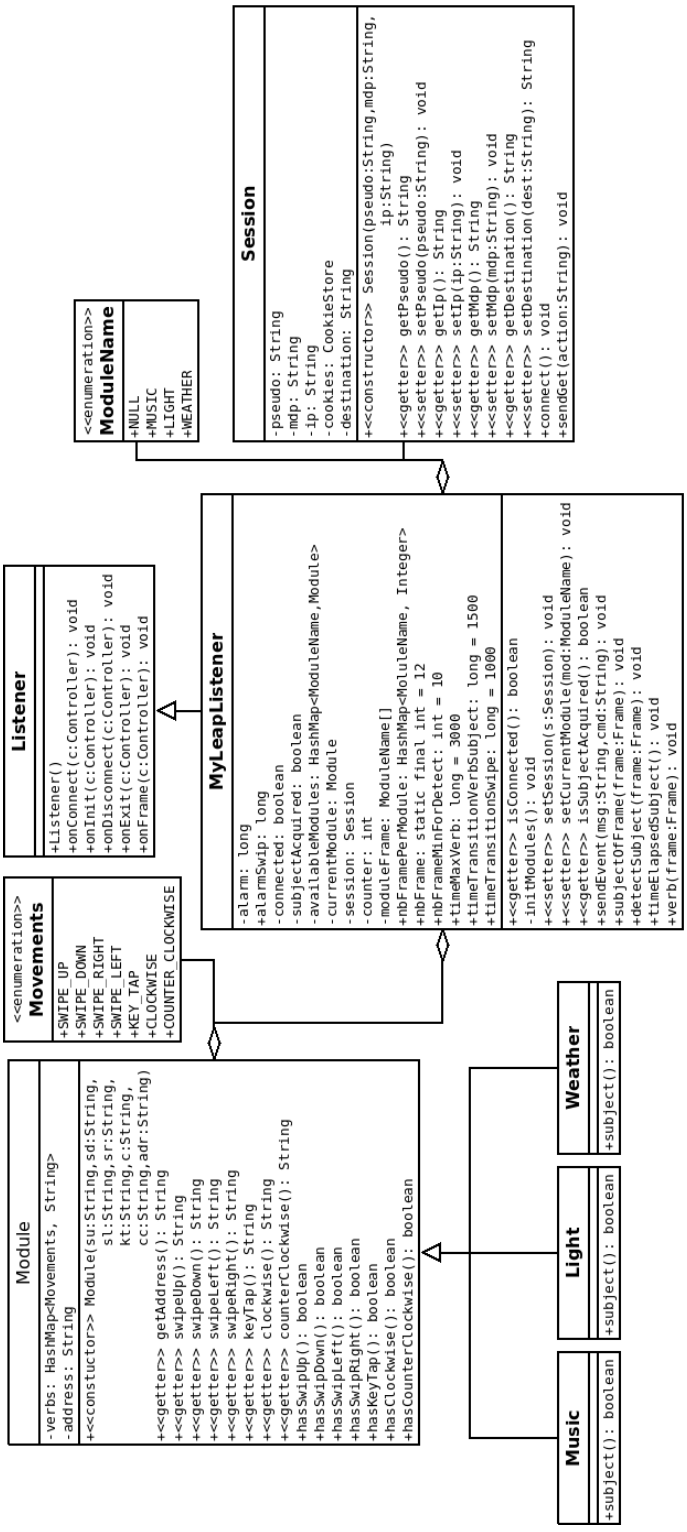


FIGURE 1 – Diagramme de classes

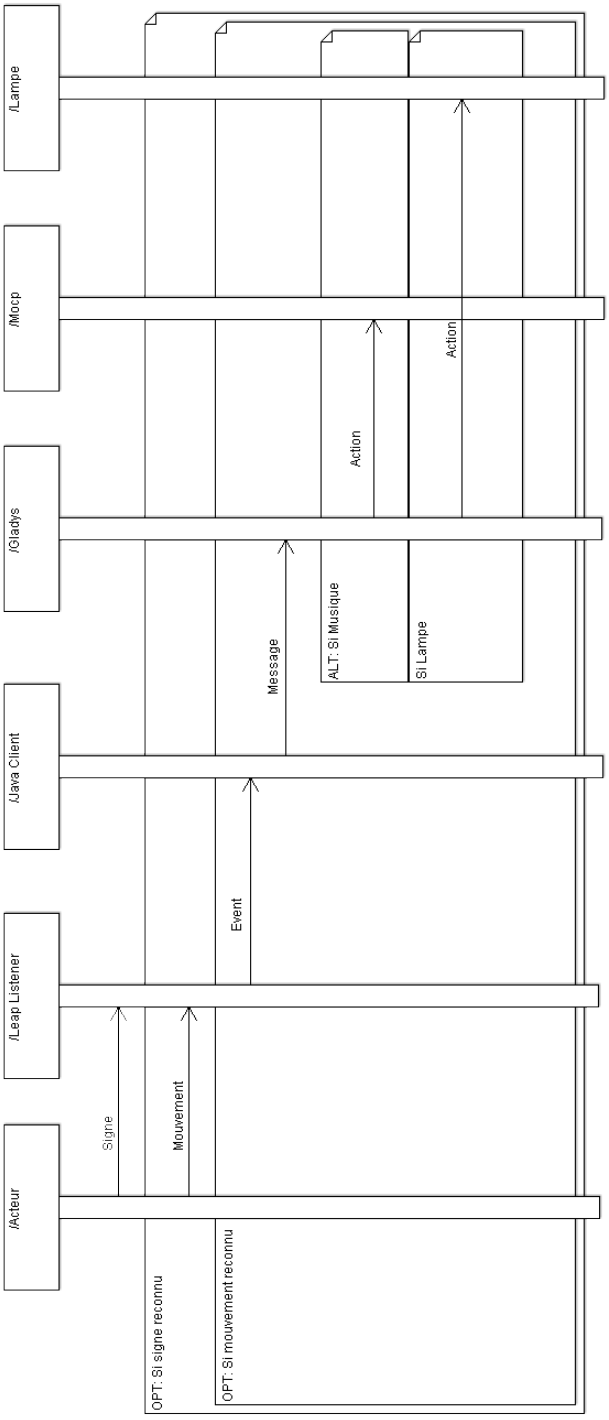


FIGURE 2 – Diagramme de séquence

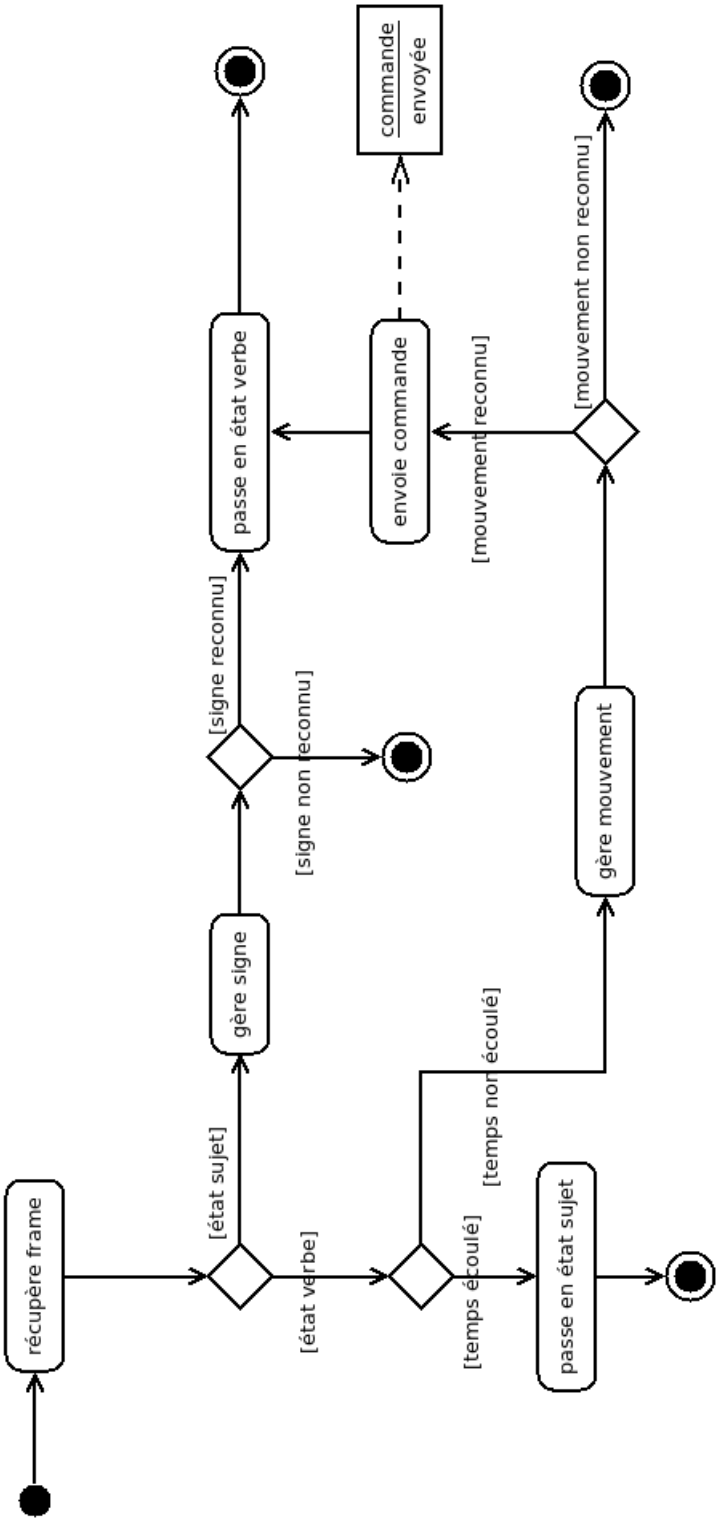


FIGURE 3 – Diagramme d'activité

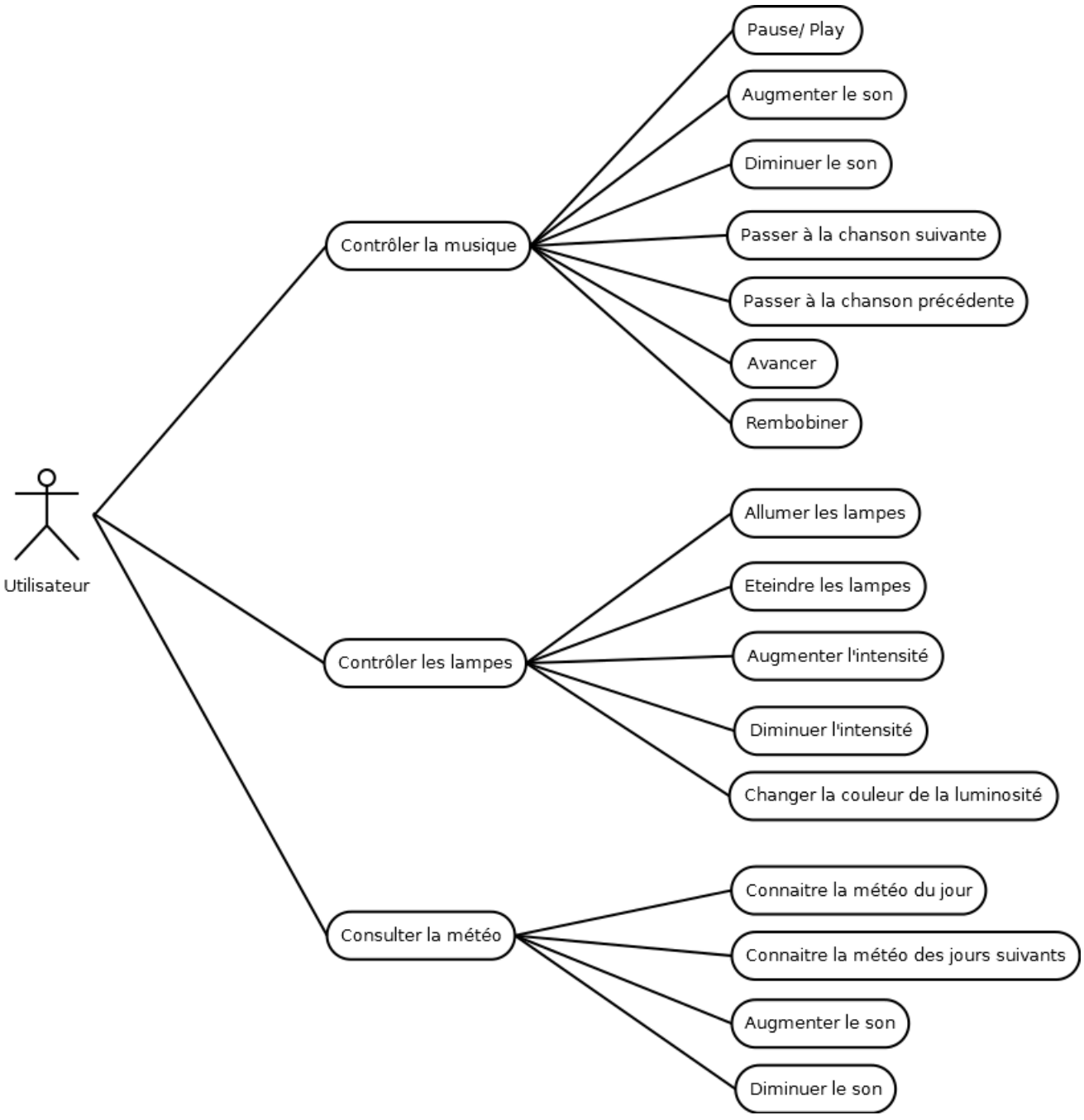


FIGURE 4 – Diagramme des cas d'utilisation

```
#VLCController.py
def on_frame(self, controller):
    frame = controller.frame()

    if not frame.hands.empty:
        for gesture in frame.gestures():
            if gesture.type == Leap.Gesture.TYPE_SWIPE:
                swipe = SwipeGesture(gesture)

                if swipe.direction[0] > 0.5 and self.alarm < time.time():
                    self.alarm = time.time() + 1.00
                    print "Right: next song"
                    self.vlcController.send("next")
                if swipe.direction[0] < -0.5 and self.alarm < time.time():
                    self.alarm = time.time() + 1.00
                    print "Left: previous song"
                    self.vlcController.send("prev")
                if swipe.direction[1] > 0.5 and self.alarm < time.time():
                    self.alarm = time.time() + 1.00
                    print "Up: volume up"
                    self.vlcController.send("volup 2")
                if swipe.direction[1] < -0.5 and self.alarm < time.time():
                    self.alarm = time.time() + 1.00
                    print "Down: volume down"
                    self.vlcController.send("voldown 2")

            if gesture.type == Leap.Gesture.TYPE_KEY_TAP and self.alarm < time.time():
                self.alarm = time.time() + 1.00
                self.vlcController.send("pause")
                print "Key Tap: toggle pause"

            if gesture.type == Leap.Gesture.TYPE_SCREEN_TAP and self.alarm < time.time():
                self.alarm = time.time() + 1.00
                self.vlcController.send("pause")
                print "Screen Tap: toggle pause"
```

FIGURE 5 – VLCController.py

```php
<?php // /var/www/gladys/PHP/class/Music.class.php

    public function volume_up(){
        exec('vol +');
    }

    public function volume_down(){
        exec('vol -');
    }

    public function forward(){
        exec('mocp -k +10');
    }

    public function backward(){
        exec('mocp -k -10');
    }

```

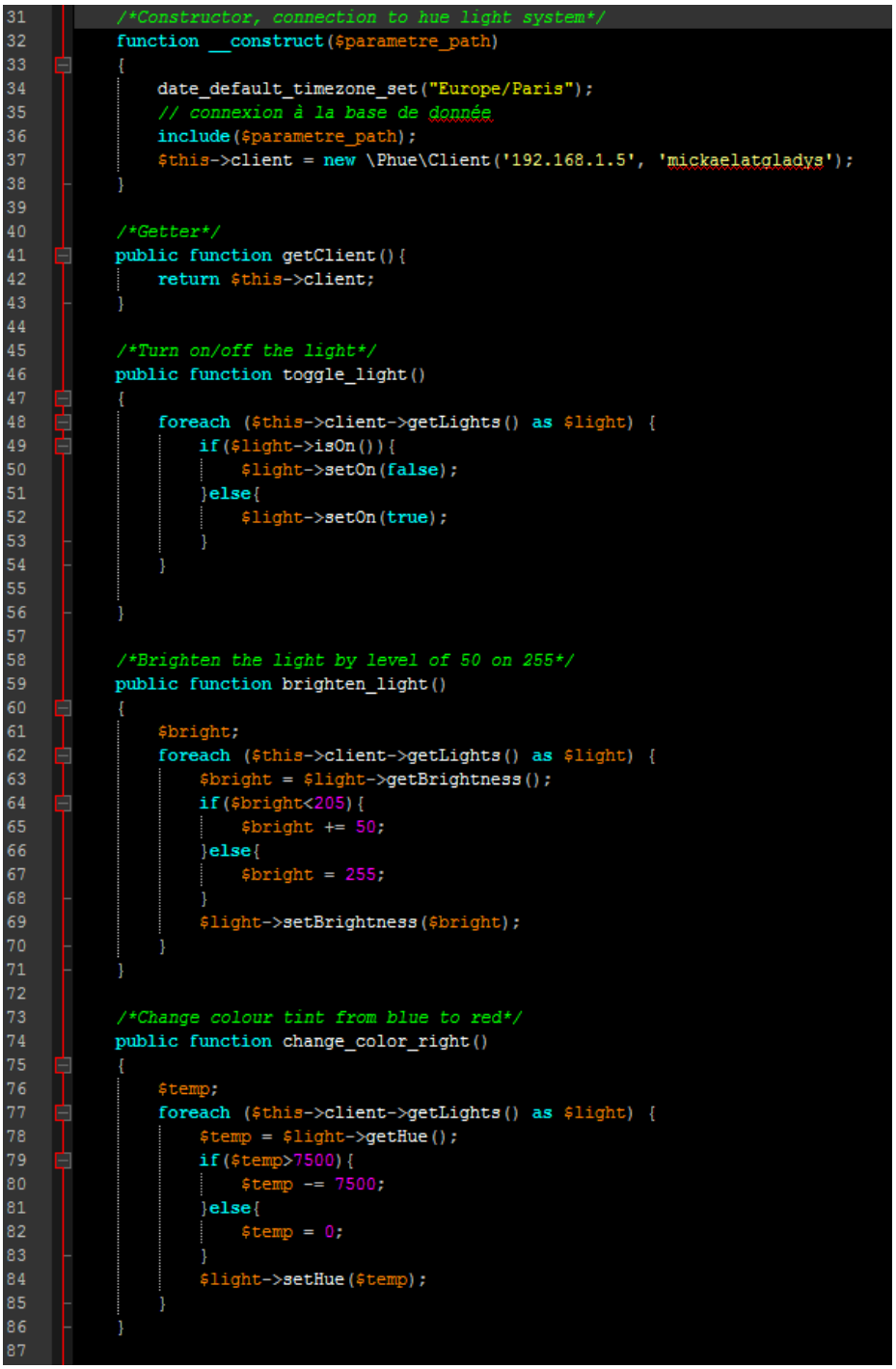
FIGURE 6 – Music.class.php

```php
<?php // /var/www/gladys/PHP/controller/music.controller.php

    //...
    if( (isset($_COOKIE['user']) && $user->check_cookies($_COOKIE['user'])))
    {
        switch($_GET['action'])
        {
        //...
            case 'toggle_pause':
                $music->toggle_pause();
            break;
            case 'volume_up':
                $music->volume_up();
            break;
            case 'volume_down':
                $music->volume_down();
            break;
            case 'forward':
                $music->forward();
            break;
            case 'backward':
                $music->backward();
            break;
        }
    }

```

FIGURE 7 – music.controller.php



```php
/*Constructor, connection to hue light system*/
function __construct($parametre_path)
{
    date_default_timezone_set("Europe/Paris");
    // connexion à la base de donnée
    include($parametre_path);
    $this->client = new \Phue\Client('192.168.1.5', 'mickaelatgladys');
}

/*Getter*/
public function getClient(){
    return $this->client;
}

/*Turn on/off the light*/
public function toggle_light()
{
    foreach ($this->client->getLights() as $light) {
        if($light->isOn()){
            $light->setOn(false);
        }else{
            $light->setOn(true);
        }
    }

}

/*Brighten the light by level of 50 on 255*/
public function brighten_light()
{
    $bright;
    foreach ($this->client->getLights() as $light) {
        $bright = $light->getBrightness();
        if($bright<205){
            $bright += 50;
        }else{
            $bright = 255;
        }
        $light->setBrightness($bright);
    }
}

/*Change colour tint from blue to red*/
public function change_color_right()
{
    $temp;
    foreach ($this->client->getLights() as $light) {
        $temp = $light->getHue();
        if($temp>7500){
            $temp -= 7500;
        }else{
            $temp = 0;
        }
        $light->setHue($temp);
    }
}
```

FIGURE 8 – Light.class.php

```
#!/bin/bash
#Found on http://www.dronkert.net/rpi/vol.html
MIX=amixer
declare -i LO=0     # Minimum volume; try 10 to avoid complete silence
declare -i HI=100   # Maximum volume; try 95 to avoid distortion
declare -i ADJ=3    # Volume adjustment step size

usage ()
{
    echo "Usage: `basename $0` [ - | + | N ]" >&2
    echo "  where N is a whole number between $LO and $HI, inclusive." >&2
    exit 1
}

# Zero or one argument
[ $# -le 1 ] || usage

# Argument must be one of: empty, -, +, number
[[ $1 == ?(-|+|+([0-9])) ]] || usage

ARG="$1"

# Number argument
if [[ $ARG == +([0-9]) ]]; then
    # Strip leading zeros
    while [[ $ARG == 0+([0-9]) ]]; do
        ARG=${ARG#0}
    done
    # Must be between LO and HI
    (( ARG >= LO && ARG <= HI )) || usage
fi

EXE=$(which $MIX)
if [ -z "$EXE" ]; then
    echo "Error: $MIX not found. Try \"sudo apt-get install alsa-utils\" first." >&2
    exit 2
fi

GET=$($EXE cget numid=1)
declare -i MIN=$(echo $GET | /bin/grep -E -o -e ',min=[^,]+' | /bin/grep -E -o -e '[0-9-]+')
declare -i MAX=$(echo $GET | /bin/grep -E -o -e ',max=[^,]+' | /bin/grep -E -o -e '[0-9-]+')
declare -i VAL=$(echo $GET | /bin/grep -E -o -e ': values=[0-9+-]+' | /bin/grep -E -o -e '[0-9-]+')

if (( MIN >= MAX || VAL < MIN || VAL > MAX )); then
    echo "Error: could not get the right values from $MIX output." >&2
    exit 3
fi

declare -i LEN=0
(( LEN = MAX - MIN ))

declare -i ABS=0
(( ABS = VAL - MIN ))

declare -i PCT=0
(( PCT = 100 * ABS / LEN ))

if [ ! -z "$ARG" ]; then

    declare -i OLD=$PCT

    if [[ "$ARG" == "+" ]]; then
        (( PCT += ADJ ))
    elif [[ "$ARG" == "-" ]]; then
        (( PCT -= ADJ ))
    else
        PCT=$ARG
    fi

    if [[ "$PCT" != "$OLD" ]]; then
        (( ABS = PCT * LEN / 100 ))
        (( VAL = MIN + ABS ))
        $EXE -q cset numid=1 -- $VAL
    fi
fi

echo $PCT
exit 0
```

FIGURE 9 – vol (bash)

	Mode Tâche	Nom de la tâche	Durée	Début	Fin
1		Tests, recherches des avantages/inconvénients d'utilisation	7 jours	Jeu 22/01/15	Ven 30/01/15
2		2eme période	10 jours	Lun 02/02/15	Ven 13/02/15
3		Recherche des existants	5 jours	Lun 02/02/15	Ven 06/02/15
4		Nouvelles idées	4 jours	Jeu 05/02/15	Mar 10/02/15
5		Décision et schémas pour l'idée finale	3 jours	Mer 11/02/15	Ven 13/02/15
6		Période 3	17 jours	Lun 16/02/15	Mar 10/03/15
7		Installation et préparation de Gladys	6 jours	Lun 16/02/15	Dim 22/02/15
8		Définition du language et des mouvements	6 jours	Lun 16/02/15	Dim 22/02/15
9		Création d'un outil de contrôle d'un serveur de musique	5 jours	Lun 23/02/15	Ven 27/02/15
10		Tests mais retard : Leap Motion indisponible	6 jours	Lun 02/03/15	Lun 09/03/15
11		Période 4	6 jours	Mer 11/03/15	Mer 18/03/15
12		Passage en socket HTTP	2 jours	Mer 11/03/15	Jeu 12/03/15
13		Alliage avec Gladys	2 jours	Mer 11/03/15	Jeu 12/03/15
14		Création d'une phrase Leap Motion	4 jours	Ven 13/03/15	Mer 18/03/15
15		Période 5	6 jours	Jeu 19/03/15	Jeu 26/03/15
16		Création d'un module lumière pour Gladys	3 jours	Jeu 19/03/15	Sam 21/03/15
17		Création de mouvements de gestion de la lumière	3 jours	Jeu 19/03/15	Sam 21/03/15
18		Création d'une phrase Leap Motion	5 jours	Dim 22/03/15	Jeu 26/03/15
19		Période 6	6 jours	Ven 27/03/15	Ven 03/04/15
20		Création de mouvements pour la météo	2 jours	Ven 27/03/15	Lun 30/03/15
21		Création d'une phrase	4 jours	Mar 31/03/15	Ven 03/04/15
22		Période 7	16 jours	Lun 06/04/15	Dim 26/04/15
23		Peaufinage, nettoyage	11 jours	Lun 06/04/15	Dim 19/04/15

FIGURE 10 – Diagramme de Gantt

Résumé

Le projet Leadys vise à créer une interface homme-machine supplémentaire pour le projet Gladys, un assistant domotique intelligent, en adoptant une approche simple et naturelle inspirée de la langue des signes. Grâce à la technologie Leap Motion, le but est d'offrir une solution centralisée et innovante pour communiquer avec divers objets domestiques.

Une attention particulière a été accordée à l'étude des contraintes liées au périphérique en termes de langages et de performance, ainsi qu'à la place qu'il peut occuper dans le milieu florissant des appareils connectés.

Cette application a été développée en utilisant les langages Java, PHP, JavaScript, le SDK Leap Motion et la librairie Phue.

Mots clés : *Leap Motion, Gladys, Domotique, Java, PHP, Phillips Hue, Raspberry Pi, Objets connectés, langue des signes.*

Summary

The Leadys project aims to create an additional human/machine interface for the Gladys project, an intelligent home automation assistant, with a simple and natural approach inspired by sign language. Using Leap Motion technology, the goal is to provide a centralized and innovative solution to communicate with various connected objects.

Particular attention was paid to the device's constraint study in terms of language and performance, and also to the position that it can occupy on the growing market of connected equipment.

This application was developped using Java, PHP, JavaScript, Leap Motion SDK and the Phue Library.

Keywords : *Leap Motion, Gladys, house automation, Java, PHP, Phillips Hue, Raspberry Pi, Connected devices, Sign language.*