ISTIA – ANNEE 2013-2014

Projet Sphero

Contrôler le robot Sphero avec Node.js

Projet réalisé par :

Julien Tourneux
Alexandre Sambo
Rajesh Santhanam

Document conçu dans le cadre du projet de 4e année su cursus ingénieur option Automatique et Génie Informatique, à l'ISTIA, sous la responsabilité de M. Medhi Lhommeau

Remerciements

Nous souhaitons remercier tout le personnel enseignant de l'ISTIA pour les connaissances qu'ils nous ont apporté, et particulièrement M. Medhi Lhommeau notre tuteur, dont l'aide nous a été précieuse tout au long de ce projet.
Nous remercions également nos familles et nos amis pour leurs soutiens quotidien, leur présence lors des coups durs mais aussi lors des moments joie.

Merci à tous.

Table des Matières

Table des illustrations

I) Introduction

Dans le cadre du projet d'EI4 AGI à l'ISTIA nous avons été amenés à travailler sur un petit robot nommé Sphero, produit de la société Orbotix. Ce projet a été proposé et supervisé par note tuteur M. Mehdi Lhommeau, enseignant-chercheur de l'ISTIA. Ce robot ce différencie de ses concurrent et affirme son originalité grâce à sa forme, qui comme son nom l'indique, est sphérique. Il peut être commandé par Bluetooth depuis un ordinateur ou un smartphone permettant à l'utilisateur de jouer avec le robot en le faisant rouler sur différentes surfaces et même sur l'eau ! Dans cette optique la société Orbotix a mis à disposition des possesseurs de Sphero plusieurs applications Smartphones permettant de contrôler la Sphero, l'utiliser à travers différents jeux, et même de l'utiliser en tant que contrôleur.
L'objectif de notre projet fut donc, à la manière de la société Orbotix, de créer une application permettant de d'utiliser la Sphero et ses différentes caractéristiques à travers une application orientée Web. À travers cette application, les utilisateurs devaient pouvoir contrôler la Sphero à distance grâce à un navigateur web. Pour ce faire nous avons utilisé la technologie Node.js, nous permettant d'utiliser le langage JavaScript directement sur le serveur, et ainsi d'utiliser ce langage en dehors du navigateur comme il est souvent le cas.
Ainsi, à travers ce rapport, nous étudieront dans un premier temps la Sphero et ses caractéristiques puis nous ferons un bref état de l'art des différentes librairies utilisant l'API de Sphero. Dans un second temps nous entreront dans le vif du sujet avec la présentation des solutions que nous avons développé ainsi que les outils que nous avons utilisé en parallèle de Node.js. Enfin, nous terminerons avec l'explication des contraintes et autres difficultés que nous avons rencontrées lors de ce projet et de l'expérience que nous avons acquise grâce à ce dernier.

II) Etat de l'art

1. Concurrences et autres concepts :

Respectivement (de gauche à droite) : Robot POB, AR Drone et Robomow

Dans le domaine des robots commandables par smartphone ou tablette, ils existent de plus en plus de produits. On peut citer de nombreux exemples tels que les drones avec caméra intégrée. (Par exemple le AR.Drone) Autre exemple le robot POB qui se veut robot à tout faire, en effet ce robot est personnalisable, il propose des kits comportant des plaques et des connecteurs pouvant accueillir capteurs, servomoteurs et autres.. Il existe aussi des tondeuses automatiques, telles que le Robomow pouvant être contrôlé et paramétrer via son smartphone. En bref, ils existent de nombreux robots pouvant se contrôler et/ou se régler via son smartphone mais leurs nombres ne cessent d'augmenter. On peut cependant retenir que les concepts sont nombreux et radicalement différent d'un robot à l'autre. Il existe d'ailleurs une large gamme de robot de ce genre avec des cibles différentes. Par exemple, on peut situer la Sphero dans le milieu de gamme au niveau des prix car par exemple le drone coûte dans les 300€ contre 130€ pour une Sphero. Les robots tondeuses peuvent couter très cher allant jusqu'à plus de 1000€, à contrario, le bas de gamme étant plus destiné au domaine du divertissement pour les plus jeunes telle que des voitures commandables par smartphone (~30€) ou autres..

2. La Sphero

Pour en revenir à la Sphero, celle-ci a été pensée afin d'être utilisée en tant qu'objet de divertissement utilisable par le grand public. Ainsi, lors de sa sortie la Sphero fut accompagnée de plusieurs applications Smartphones et tablettes allant du simple contrôleur au jeu intégrant de la réalité augmenté. Aujourd'hui ces applications sont plus d'une trentaine disponible sur l'App Store d' IOS et le Google Play d'Android. Nous avons eu donc l'occasion d'en tester certaines.

Durant la phase de test du robot, nous avons tout d'abord fait des recherches sur ce qu'était la sphero car aucun de nous ne connaissait ce robot en particulier, bien sûr nous savions tous l'existence de robot commandable par smartphone. Nous avons pu voir durant cette phase de nombreuse vidéo vantant les mérites de l'appareil. Nous avons donc commencé à chercher comment contrôler ce robot et ce fut assez simple depuis son smartphone, l'appareil est reconnu directement il suffit alors de le calibrer. Nous nous sommes aperçu assez vite qu'il n'était pas si simple de diriger le robot, même avec un peu d'entrainement maintenant la précision reste assez grossière depuis un appareil tactile. De plus il est nécessaire de recalibrer le robot régulièrement, notamment en cas de choc.

Durant notre projet nous avons utilisé la Sphero 2.0, il existe en effet plusieurs versions de la Sphero. La seconde version, celle que nous avons utilisée se veut plus rapide et plus précise que la première version mais reste identique esthétiquement. Une nouvelle version, la Sphero 2B, devrait sortir courant 2014, la forme ronde a été abandonnée pour cette version qui se veut cylindrique. Cette dernière devrait présenter une nouvelle manière de piloter la Sphero, être plus rapide, plus agile et plus personnalisable.

En plus de ces deux moteurs, l'actuel robot Sphero (la version 2.0) dispose de plusieurs périphériques tels qu'un gyroscope et un accéléromètre lui permettant de récupérer les informations nécessaires afin de stabiliser sa direction et ainsi effectuer des déplacements rapides et précis. De plus, le robot peut se déplacer sur l'eau et se montre résistant au choc grâce à sa coque en polycarbonate. La forme sphérique du Sphero n'est pas son seul aspect innovant, celui est sans boutons apparents, sans ports ou autres connectiques visibles, ce qui lui confère un physique épurée très appréciable. Mais sans toutes ces interfaces, comment interagit-on avec la Sphero ?
La Sphero se veut très simple d'utilisation. Ainsi, pour l'allumer il suffira à l'utilisateur de secouer le robot afin que celui se mette à clignoter notifiant qu'il est bien en état de marche. Il se recharge par induction grâce au dock fourni avec le robot et il suffira de poser le robot sur le dock pour que celui-ci se mette en charge puis passe en mode veille avant de finalement s'éteindre après un laps de temps. Enfin, le robot embarque une antenne Bluetooth lui permettant de se connecter à un ordinateur, une tablette ou un smartphone distant d'environ 50m.

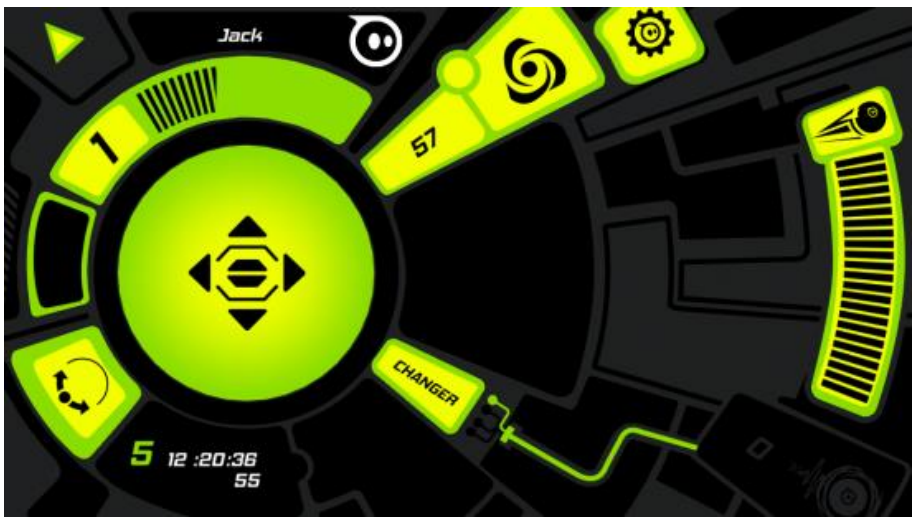
Sur l'image ci-dessous vous pouvez observer les différents composants de la Sphero :

- (A) Le dock de rechargement
- (B) Les roues contrôlées par moteur
- (C) Le pallier permettant de garder les roues de la Sphero fermement en contact avec la coque
- (D) Le circuit imprimé auquel sont intégrés l'accéléromètre et le gyroscope.
- (E) L'antenne Bluetooth
- (F) La LED.

Figure 1: Composition du robot Sphero (Source : technologyreview.com)

3. Test d'applications pour la Sphero

Dans un premier temps nous avons pu test l'application de contrôle de la sphero proposé par Orbotix, elle se prénomme Sphero sur le store (google & apple) et elle est gratuite. Un tutoriel nous guide à travers les étapes de calibration du gadget afin de pouvoir en profiter au maximum. Le contrôle de la balle, bien qu'assez simple, prend du temps avant d'être assimiler. L'application dispose même d'actions spéciales, plus on utilise la sphero plus elle gagnera de l'expérience et plus on aura accès à de nouvelles compétences. Par exemple on trouve une propulsion rapide vers l'avant ou des sauts en rotation de la sphero. L'application permet aussi de changer la couleur de la sphéro, sa vitesse et aussi de recalibrer la sphero quand besoin est.



Nous avons aussi testé d'autres applications mais n'en avons choisi qu'une autre à titre d'exemple : Golf Sphero, cette application nous a plu et marche plutôt bien. Il s'agit d'un jeu de golf qui nous permet d'utiliser le smartphone comme club, à titre de comparaison on peut reprendre le principe du golf sur WII. Le principe est le même on ne touche pas la balle au final mais on la voit quand même bougé, on peut au final comparé cela à un mixte entre la version WII et la réalité. La particularité de cette application est qu'elle met bien en avant la précision de la sphero qui dans l'application de base ne se fait pas suffisamment ressentir. Le principe est simple, on peut calibrer la sphero avant le tir, régler la puissance de tire et le type de club. Il suffit dès lors de tenir son smartphone tel un club de golf, le tout demande un peu d'entrainement mais le coup de main vient assez vite. Le petit plus est qu'on est complètement libre de choisir son parcours, il suffit simplement d'un peu de créativité.

III) Présentation du projet

1. Objectifs

L'objectif de ce projet fut de développer une application orientée Web permettant de contrôler la Sphero à distance depuis une machine serveur et d'exploiter ses différentes fonctionnalités. Afin, d'exploiter ces différentes fonctionnalités nous avons décidé de mettre en œuvre le robot Sphero dans trois cas d'utilisation :

- Le pilotage classique (déplacer le robot, régler sa vitesse, changer sa couleur, etc…) grâce aux interfaces habituelles utilisées sur ordinateur (clavier, souris)
- Le pilotage grâce à un dispositif original (ex : la Leap motion)
- L'utilisation de la Sphero à travers un jeu vidéo.

Pour nous permettre réaliser ces différentes fonctionnalités nous avions besoin d'un langage web nous permettant de communiquer avec la Sphero depuis une machine serveur. Cette machine recevrait ensuite des requêtes provenant d'une machine cliente connectée à application web afin contrôler le robot Sphero à distance. Il nous a ainsi été conseillé d'utiliser Node.js.

2. Organisation

Au cours du premier semestre nous avons eu un cours en parallèle des séances de projet, ce cours avait pour objectif de planifié le projet. Lors de ce cours nous avons été amenés à créer un diagramme de Gantt que vous pouvez trouver en annexe (Annexe n°1). Cependant il faut noter que le Gantt a été effectué en début de projet et les temps ne correspondent pas forcément suite aux différentes aux différentes difficultés que nous avons rencontrées lors des projets. Prenons l'exemple de la tâche_41 qui nous a pris plusieurs semaines et qui n'était au départ prévu que sur environ 2 séances. Pendant cette étape nous étions tous les trois sur le même problème et étions tous concentré à le résoudre car nous ne pouvions pas nous divisé sans prendre du retard par la suite car cette étape est indispensable.
L'organisation s'est ensuite faite au fur et à mesure mais le projet étant planifié il nous a été plus facile de se répartir les tâches en fonctions des besoins, même si nous étions tous dans un processus de recherche initialement. Nous avons créé des camemberts de répartition de tâche pour illustrer la répartition au cours du projet de la répartition que nous avons adopté. (Disponible en annexe : Annexe n°2) Chaque camembert représente la proportion de travail pour chaque membre et accentue bien une répartition hétérogène.
Au final l'organisation adoptée a permis d'établir un diagramme de proportions des activités (Annexe n°3). Sur cette annexe, on peut voir distinctement la proportion que nous a pris la recherche de solution pour connecter la Sphero via Bluetooth à un pc. Le découpage nous donne environ 37.8% du temps que nous avons consacré aux prémices du projet, environ 24.2% pour la rédaction du rapport, la création du site de présentation et les résumés de séance. Le reste a été consacré au développement, notamment à l'application web de contrôle de la Sphero, on constate alors que seulement 37,8% du temps de projet n'a pu être alloué au développement de ce dernier.

3. Présentation des technologies utilisées

Dans ce chapitre nous allons présenter les différentes technologies que nous avons utilisées lors de ce projet. Nous n'allons pas toutes les énumérée car la liste serait bien trop longue et fastidieuse car il serait inutile de présenter des langages tels que l'HTML, le JavaScript ou des bibliothèques tel que JQuery, etc. Nous parlerons par contre des modules JavaScript principaux qui nous ont permis de répondre à nos besoins lors du développement de l'application.

A. Node.js

Du code JavaScript peut être intégré directement au sein des pages web, pour y être exécuté sur le poste client. C'est alors le navigateur Web qui prend en charge l'exécution de ces programmes appelés scripts. Ce langage est orienté objet et est basé sur les évènements, il permet entre autre de manipuler la page HTML afin de la dynamisée. A l'instar du PHP, Node.js offre un environnement coté serveur qui nous permet aussi d'utiliser le langage JavaScript pour générer des pages web.

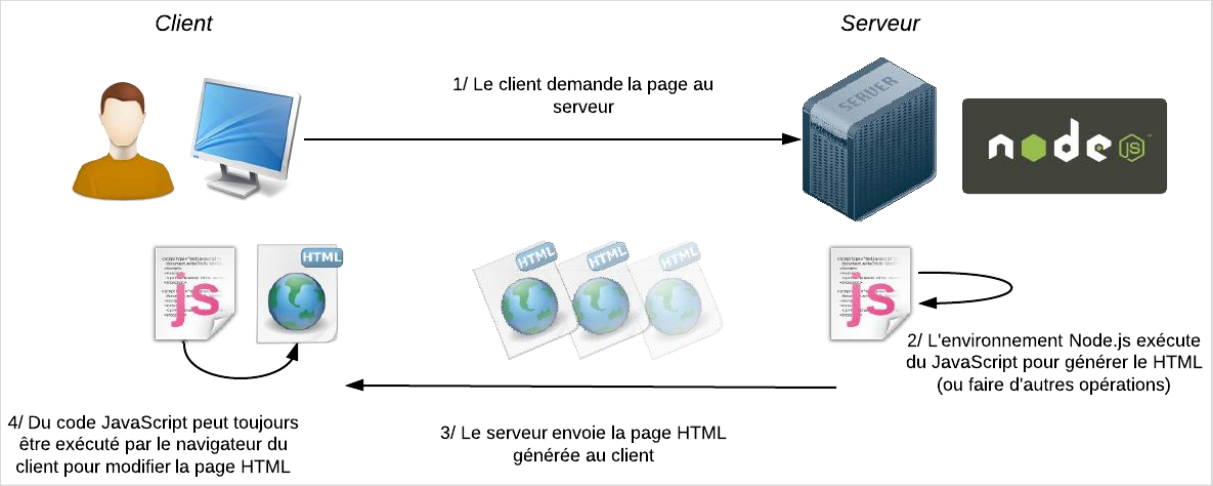


Figure 2: Description illustrée du fonctionnement de Node.Js (Source : Openclassrooms.com)

L'avantage avec Node.Js est que celui-ci est beaucoup plus rapide que ses « concurrents » tel que CommomJS qui est bien moins récent. Cette rapidité est due à l'utilisation du moteur d'exécution V8 développé par Google dans lequel les événements sont gérés de manière non bloquante. Cette architecture nous permet d'utiliser des fonctions dites de « callback » qui seront exécutées à la fin d'un évènement sans pour autant bloquer le reste de l'exécution du programme. Il est ainsi possible de paralléliser les opérations effectuées par la machine et ainsi réduire le temps d'exécution de plusieurs opérations. De par sa rapidité, la surcouche Node.js est de plus en plus choisie pour expérimenter des problématiques de temps réel ou quasi-temps réel…

Le noyau de node.js n'est pas réellement volumineux, cependant Node.js jouit d'une grande extensibilité et d'une communauté très active grâce à laquelle ces fonctionnalités ne cessent d'évoluer. Il est possible d'ajouter des modules à une application Node.js grâce au Node Package Manager (dit NPM) sur lequel les utilisateurs peuvent publier leur module mais aussi installer ceux des autres utilisateurs grâce à des commandes simples : npm publish et npm install. De plus, si ces modules sont dépendent d'autres modules, il est possible de définir un fichier de configuration JSON (JavaScript Object Notation) permettant d'installer et de mettre à jour tous les modules auxquels notre projet dépend. Cela est très intéressant car ça nous évite d'installer chacun des modules dont on dépend et gagner énormément de temps surtout si les dépendances sont nombreuses.

A. Express

Le noyau de node.js ne permettant pas de gérer les URLs de manière simple (code lourd et répétitif), nous avons eu recours au module Express.js qui nous permet de gérer aisément et proprement les URL ou routes accessibles via notre site web et même gérer les erreurs 404. Express.js permet de plus de gérer les routes dites dynamique, c'est-à-dire des URLs dont une portion de l'adresse sera considérée comme une variable nous évitant de passer par le suffixe ("?variable=valeur") et de dynamiser la navigation via ces URLs. Bien gérer les URLs d'un site web est important, surtout lorsque celui-ci grossit, mais dans notre cas nous aurons recours qu'a très peu d'URLs, une pour chacune des applications mises en œuvres.

B. Socket.io

La bibliothèque Socket.io est une bibliothèque permettant un échange bilatéral synchrone entre le client et le serveur, autrement dit elle permet la communication temps réel entre ces deux machines. Plus encore socket.io permet de communiquer en Broadcast, c'est-à-dire émettre vers toutes les machines cliente connectée à cette socket.
Ce socket a été primordial lors de ce projet, car il nous a permis de transmettre les requêtes du client, tel que changer la couleur du robot Sphero, vers le serveur. Puis le serveur se chargera d'exécuter cette action grâce à la communication Bluetooth établie avec le robot Sphero (Cf. illustration).

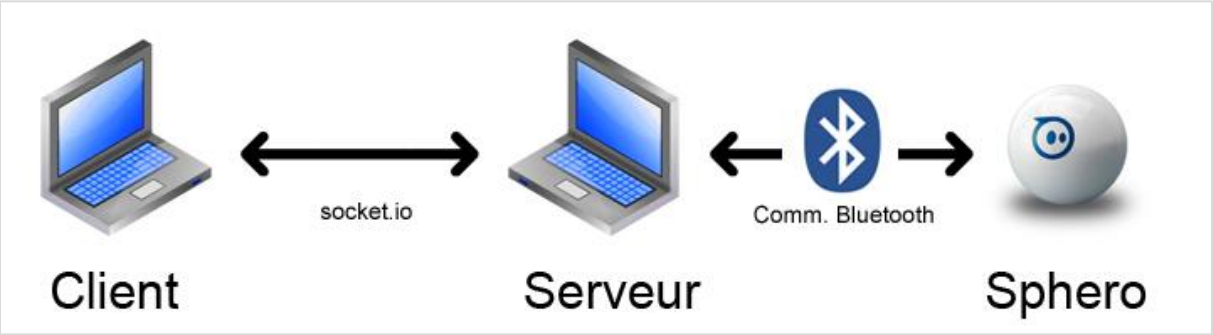


Figure 3: Illustration du protocole de communication

C. Spheron

Spheron est le module Node.js nous permettant de communiquer avec la Sphero. Ce dernier est disponible sur NPM et contient toutes les fonctionnalités présentes sur l'API officiel de la Sphero bien que celle-ci n'est pas été développée officiellement par l'équipe de développeur d'Orbotix. De plus, Spheron permet la communication Bluetooth grâce à un autre module nommé serial-port. Ce module permet, de définir sur quel port la Sphero est connectée et ainsi établir la communication. Spheron, quant à lui, se charge de convertir les informations que l'on désire transmettre à la Sphero (rouler, changer de direction, etc.) en trames compréhensibles par ce dernier, et nous permet aussi de récupérer les informations envoyés par la Sphero tels que les données du gyroscope, de l'accéléromètre ou encore le niveau de charge de la batterie. Le seul point négatif de ce module est son manque de documentation, cependant nous avons pu nous aider des commentaires laissés par le développeur dans le code source ainsi que de la documentation de l'API Sphero afin de comprendre la structure des paquets que le Sphero souhaite en entré et de ceux qu'ils nous envoi.

IV) Développement

La phase de développement s'est vue décomposée en plusieurs sous tâches bien distinctes. Tout d'abord la mise en place du serveur abritant l'application Node.js qui va nous permettre de communiquer avec la Sphero. Puis dans un second temps, le développement d'un test de connexion afin de vérifier comment se déroule une connexion avec Sphero.
La connexion étant établie et le serveur mis en place, l'étape suivant a été de créer l'interface graphique de l'application afin de permettre à l'utilisateur d'avoir un retour sur les informations envoyés par la Sphero mais aussi des différentes interactions qu'il pourrait

établie avec le robot. Dans cette partie, l'ergonomie est un aspect assez important car nous essayons de faire en sorte que le contrôle soit très simple de prise en main afin que le contrôle soit plus agréable mais aussi afin qu'il soit le plus intuitif que possible. L'étape qui suivi fut de développer les différentes façons de contrôler la Sphero grâce à différents périphériques tel que le clavier ou la souris pour les plus classiques ou la Leap Motion pour les plus originaux. Puis nous en viendrons à exploiter les informations de la Sphero à travers un jeu dans lequel cette dernière nous servira de contrôleur !

1. Création du serveur

Le serveur est la machine sur laquelle l'application web sera abritée et à laquelle la connexion à la Sphero sera établie. Dans les langages plus classique tel que le PHP, le langage s'associe avec un serveur http tel qu'Apache. Le serveur gère les demandes de connexion HTTP au serveur émises par le client. Quant au langage, il exécute les code sources et renvoie les résultats au serveur qui se charge de le renvoyé au client à travers son navigateur. Dans Node.js, rien de tout ça n'est d'actualité car ce dernier n'utilise pas de serveur HTTP mais nous permet dans créer un !

Pour ce faire, nous faisons appel à la bibliothèque « http ». Cette bibliothèque Node.js implémente les fonctions qui nous permettent de créer un serveur web et nous permet de mettre en place un serveur très facilement grâce à quelque ligne de code très simple placée dans un script:

```
//Utilisation de la bibliotheque « http »
var http = require('http');

//Création d'un objet server
var server = http.createServer(function(req, res) {
  //fonction de callback affichant un message sur la page web
  res.writeHead(200);
  res.end('Bienvenue sur notre serveur Node.js !');
});

//on ecoute sur le port 8080
server.listen(8080);
```

Lancer le serveur est encore plus simple, il suffit de lancer la commande :

```
Node monServer.js
```

2. La connexion avec Sphero

Afin de pouvoir contrôler la Sphero, il fallait tout d'abord pouvoir établir une connexion entre le serveur et cette dernière. Comme nous l'avons dit précédemment, le robot Sphéro communique grâce à une connexion Bluetooth. Il ne nous restait plus qu'à appairer la Sphero et l'ordinateur grâce au gestionnaire de connexion Bluetooth disponible sur la plupart de système d'exploitation. Lors de cette étape nous avons eu malgré nous plusieurs obstacles qui nous ont poussés à utiliser le système d'exploitation Linux (distribution Ubuntu 12.0) plutôt qu'un autre système, mais nous verrons cela plus loin dans ce rapport. Ainsi afin d'établir la connexion Bluetooth sous Ubuntu nous devions procéder comme suit :

Récupérer l'adresse MAC du Sphero :

```
hcitool scan
```

```
Scanning ...
    00:06:66:4A:3F:4B   Sphero-PRB
```

Ensuite, pour appairer notre robot au système :

```
sudo rfcomm bind hci0 00:06:66:4A:3F:4B
```

Cette commande aura pour effet d'enregistrer le dispositif dans le fichier etc/rfcomm0.
Une fois, le robot sphero appairé arrive la phase de connexion avec notre application. Pour se faire nous créons un script Node.js permettant de gérer la connexion avec notre robot et contenant le code suivant :

```
//Utilisation du module spheron
var spheron = require('spheron');
var sphero = spheron.sphero();

//Le port auquel Sphero est appairé
var spheroPort = '/dev/rfcomm0';

//Gestion de l'événement « open »
sphero.on('open', function() {
  console.log(Connexion réussie !');
});

//Gestion de l'événement « error »
sphero.on('error', function(error) {
  console.log('Erreur lors de la connexion !');
});

//Connexion à la sphero
sphero.open(spheroPort);
```

Ce script nous permet de faire un test de connexion avec la Sphero grâce au module spheron. On peut entre autre observer que nous faisons appel à ce module grâce à la commande `require`, puis nous récupérons un objet de type « sphero » issu de ce module. Cet objet contient toutes les fonctions implémentées dans l'API officielle de Sphero et c'est celui-ci que nous utiliserons dans la suite du projet.
La fonction « open » établie la connexion entre Sphero et le serveur et déclenche l'évènement du même nom. Si la connexion à bien été établie alors la console affichera « Connexion Réussie ! » dans le cas contraire, c'est l'événement « error » qui sera appelé et la console affichera « Erreur lors de la connexion !». Enfin pour démarrer script sur le serveur il suffira de lancer la commande ainsi :

```
node MonScript.js
```

3. Architecture du projet

Le projet à une architecture bien spécifique due à l'utilisation de la bibliothèque Express. Ainsi, il existe 3 dossiers et un fichier essentiels au bon fonctionnement de notre application :

- Le dossier « node_modules » : Ce dossier est le dossier dans lequel seront installés tous les modules Node.js suite à l'appel de la commande `npm install`. Dans notre cas il contiendra les modules « http », « ejs », « express », « socket », « spheron » et « leap ». Nous verrons dans la suite de ce rapport l'utilité de chacun des modules que nous n'avons pas encore détaillé.
- Le dossier « public » : Ce dossier contient les différents éléments externes au projet, tels que les images, les scripts JavaScript ou encore les feuilles de styles. Afin qu'Express reconnaisse l'utilisation de ce dossier en tant que dossier contenant des fichiers statiques nous utilisons une ligne de code spécifique dans notre script serveur :

```
// utilisation de la variable global __dirname contenant le chemin
// d'accés du dossier.
app.use(express.static(__dirname + '/public'));
```

- Le dossier « views » : Il contient toute les pages web de notre application. Ces pages ne sont d'ailleurs pas au format HTML mais au format EJS (Embedded Javascript) qui est un format template à l'instar du PHP.
- Le script du serveur.

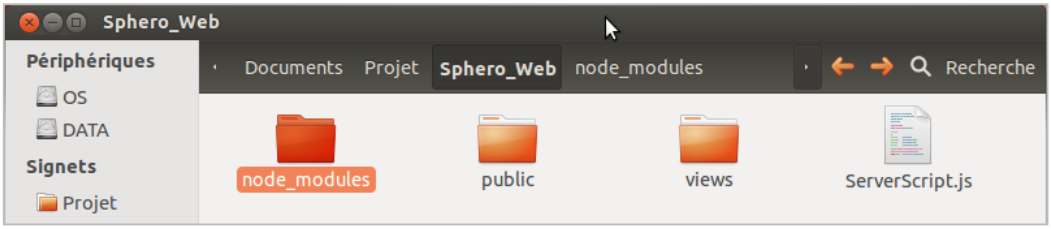


Figure 4: Illustration de l'architecture du projet

A noter que les fichiers de template nous permettent de produire du code HTML et d'insérer au milieu du contenu variable. Ces derniers gèrent l'essentiel des instructions de programmations, à savoir : les variables, les conditions, les boucles, etc. Comme avec le PHP, les instructions seront contenues dans des balises spécifiques. Le module Express nous permet ainsi de transmettre des variables stockés sur le fichier serveur vers le client. Pour ce faire, nous appelons le template de notre choix depuis notre fichier JavaScript, en lui transmettant les variables dont il a besoin pour construire la page client.
Observons ci-dessous un exemple d'utilisation d'un fichier EJS coté serveur via notre script JavaScript.

```
//Lorsque l'utilisateur souhaite accéder à la page « pageTest »…
app.get('/pageTest', function(req, res) {
    /*
    * …nous envoyons le code HTML contenu dans le fichier maPage.ejs
    * ainsi que la variable nbVisiteur qui sera utilisée sur la page
    * client en tant que numvisiteur.
    */
```

```
    res.render('maPage.ejs', {numvisiteur: nbVisiteur});
});
```

Du côté de la page client, l'utilisation des variables récupérée se fait ainsi :

```
// La variable numvisiteur à été envoyé depuis le serveur
<p>Vous êtes le visiteur n°<%= echo numvisiteur; %></p>
```

4. Design de l'application

Avant de réaliser le développement de l'application nous voulions tout d'abord nous mettre d'accord sur l'aspect de que celle-ci abordera ainsi que les différents outils que nous utiliserons. Ainsi nous avons mis en place un prototype simple de ce à quoi ressemblerai l'application afin de mieux cerné nos besoins. Ce dernier contient un « color picker » (sélecteur de couleur) permettant de définir la couleur de la LED. Ensuite nous hésitions entre intégrer un joystick virtuel ou une interface représentant les touches de notre clavier. Afin de régler la vitesse nous pensions utiliser un slider permettant de régler la vitesse avec laquelle la Sphero se déplacera. Enfin, nous pensions à mettre à disposition de l'utilisateur une petite « toolbox » lui permettant de savoir comment s'utilise chacun des outils sur la page, cette aide s'activera lorsque l'utilisateur passera sa souris sur l'un des outils.

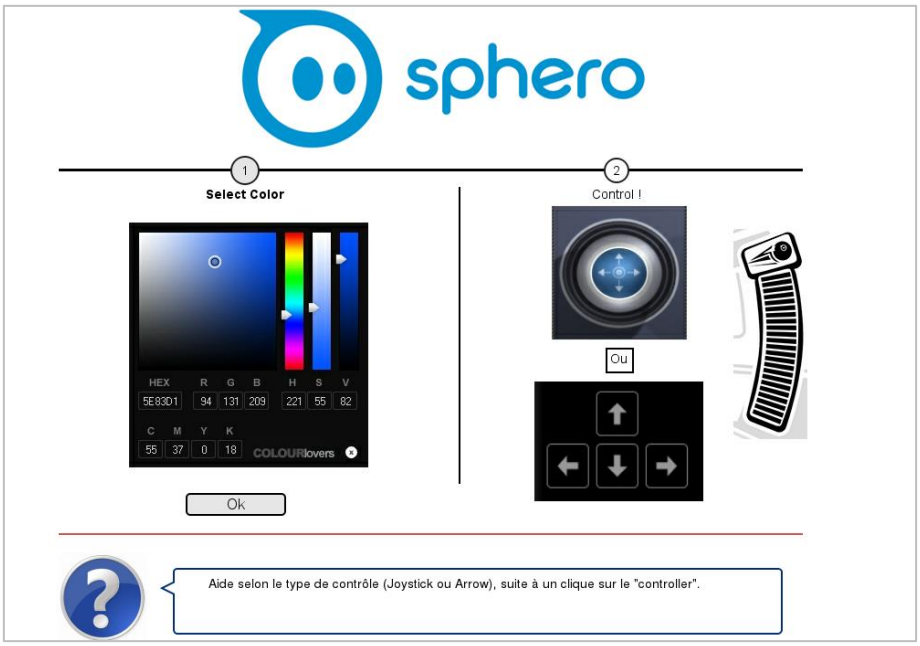


Figure 5: Illustration du prototype de l'application

Grâce à ce prototype nous avons pu développer une interface assez rapidement. Cette interface présente les mêmes outils que nous avons présentés sur le prototype à l'exception de la « toolbox ». En plus de ces outils, nous avons ajouté à l'interface un bouton de connexion permettant d'établir la connexion avec la Sphero et d'une icône permettant à l'utilisateur de distinguer le niveau de charge de la batterie. Par manque de temps, les outils que nous avons développés arborent un design simple mais clairement suffisant quant à l'utilisation que nous allons en faire. Nous pouvons observer ci-dessous l'actuelle interface de notre application :

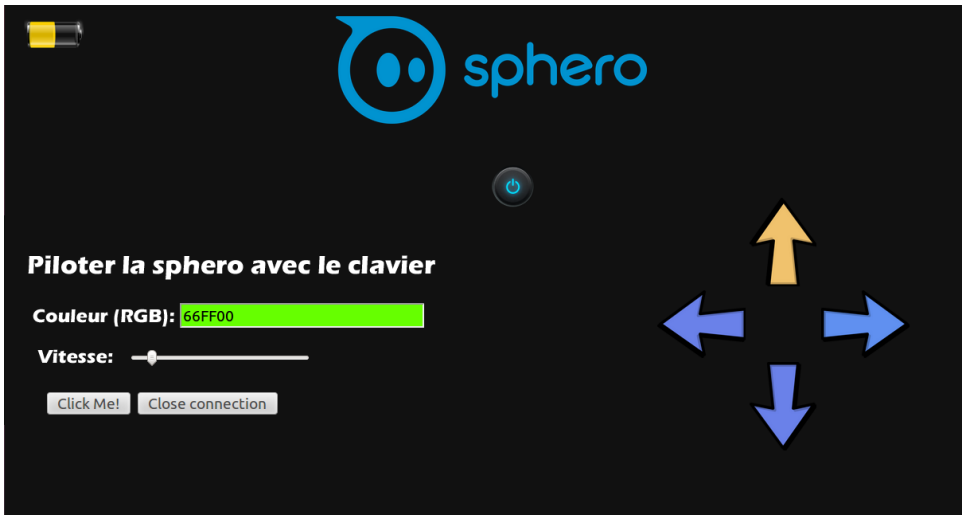


Figure 6: Capture d'écran de l'application

5. Présentation des solutions

A. Communiquer avec la Sphero

1. La connexion

Les différents boutons et autre outils de l'interface vont permettre à l'utilisateur de communiquer avec la Sphero via le server. Afin de gérer les interactions ces derniers et l'utilisateur nous avons utilisé la bibliothèque JQuery. Cette bibliothèque JavaScript est open source et porte sur l'interaction entre JavaScript et HTML. Elle nous servira entre autre à gérer les événements lors du clique sur les différent buttons de l'application. Exemple, pour connecter la Sphero nous mettons à disposition de l'utilisateur un bouton « Power ». Lorsque le bouton sera pressé, une requête de connexion sera envoyée au serveur et celui-ci se chargera de se connecter à la Sphero afin de permettre la communication. Une fois la connexion établie le serveur renvoie l'information de connexion au client : si la connexion s'est bien déroulé alors le bouton « Power » s'allumera, dans le contraire ce dernier restera éteint. Illustrons les différentes étapes :

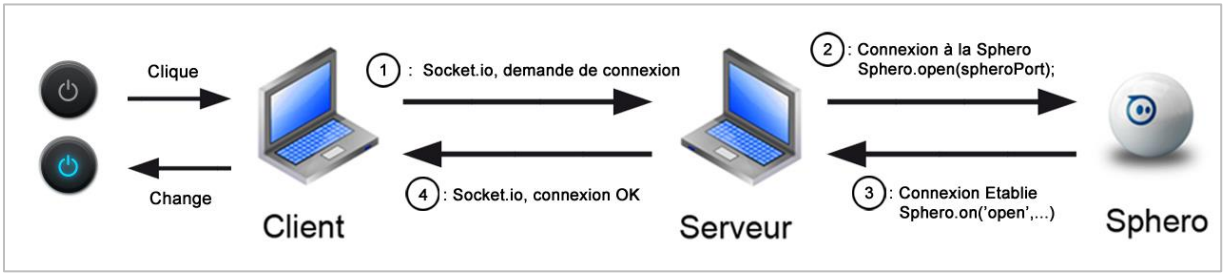


Figure 7: Illustration de la demande de connexion Client-Sphero

Extrait de la page HTML du client:

```
<!--  Le button de connexion  -->
<img src="images/power_btn_normal.png" alt="power button" id="btn_power"/>
<!– Appel a JQuery -->
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
```

```
<script src="/socket.io/socket.io.js"></script>
<script>
     //On récupere la socket grace au port sur lequel on écoute
     var socket = io.connect('http://localhost:8080');
     //Gestion de l'événement isConnect avec en parameter la variable
     //représentant la connexion
     socket.on('isConnect', function(valueConnect) {
          if(valueConnect){
             doc.getElementById("btn_power").src="power_btn_actif.png";
          }
          else{
             doc.getElementById("btn_power").src="power_btn_normal.png";
          }
     })
     //JQuery
     //Gestion de l'évenement lorsque l'on clique sur le bouton "Power"
     $('#btn_power').click(function () {
             //Envoi la demande de connexion à la socket
              socket.emit('connect');
     })
```

Extrait du script serveur :

```
//Variables internes
var isConnected = false;

// Lorsque la socket reçoit la demande de connexion
socket.on('connect', function (message) {
     console.log('Connexion sphero...');
     sphero.open(spheroPort); // Connecter sphéro
});
// Lorsque la sphero reçoit l'instruction de se connecter
sphero.on('open', function() {
    console.log('Connexion réussie !');
    //Changer la couleur de Sphero
    sphero.setRGB('0x0000FF', false);
    isConnected = true;
    // Envoyer au client la variable de connexion
    io.sockets.emit('isConnect', isConnected);
});
```

On constate que la phase de connexion se déroule de manière asynchrone, nous attendons que le serveur est réussi sa connexion avant d'avertir l'utilisateur du succès de l'opération via le changement de couleur du bouton « Power ». A partir de ce point tout le reste des requêtes lancées par le client se verront effectués directement, sans l'attente d'une réponse du serveur car les résultats seront directement visibles via la Sphero (ex : changement de couleur, déplacements, etc). Nous, avons tout de même sécurisé ce processus en acceptant les requêtes si et seulement si la connexion à la Sphero est toujours effectives.

2. *Les paquets*

L'une des fonctions présente dans la plupart des applications est l'affichage de la batterie restante. Ainsi, nous voulions nous aussi disposer d'une telle fonction. Or, nous ne savions pas comment nous y prendre. La bibliothèque Spheron nous permettait d'envoyer des instructions mais nous ne savions sous quel format étaient envoyées ces dernières et encore

moins comment récupérer les informations émises par la Sphero. La documentation de la bibliothèque Spheron étant assez pauvre, nous sommes directement allés voir dans le code source qui est très bien commenté.
A travers ce code source, nous avons compris que Spheron envoyait les données sous forme de paquet interprétable par la Sphero. Ces paquets sont construits dans le format suivant :

```
COMMANDE CLIENT
{
  SOP1          (Début du paquet, valeur toujours égale à 0xFF)
  SOP2          (De 0xF8 à 0xFF selon les options, cf. API Sphero)
  DID           (Identifiant du périphérique auquel le paquet est destiné)
  CID           (Identifiant de la commande)
  SEQ           (Identifiant séquence transmis pendant la requête)
  DLEN          (Taille des données)
  DATA          (Buffer contenant les données)
  CHK           (Checksum du paquet)
}
```

De la même manière, Spheron nous permettait de récupérer ces paquets. Il existe deux types de paquet envoyés par la Sphero : Les messages et la notification représentés ainsi :

```
MESSAGE
{
  SOP1          (Début du paquet, valeur toujours égale à 0xFF)
  SOP2          (Description du paquet: 0xFF pour les messages)
  MRSP          (Message Response: code représentant le type de réponse)
  SEQ           (Identifiant séquence transmis pendant la requête)
  DLEN          (Taille des données)
  DATA          (Buffer contenant les données)
  CHK           (Checksum du paquet)
}
```

```
NOTIFICATION
{
  SOP1          (début du paquet, valeur toujours égale à 0xFF),
  SOP2          (Description du paquet: 0xFE pour les notifications)
  ID_CODE       (identifiant du type de notification qui est envoyés)
  DLEN          (Taille des données)
  DATA          (Buffer contenant les données)
  CHK           (Checksum du paquet)
}
```

La bibliothèque Spheron nous permet de récupérer chacun des paquets que Sphero envoie en gérant ces envois en tant qu'événements. Ainsi pour récupérer la valeur de charge de la batterie, nous nous somme renseigné sur l'API afin de connaitre l'identifiant du paquet nous permettant d'obtenir ces infos et générer le code suivant :

```
//Gestion de l'événement à la reception d'une notification
sphero.on('notification', function(notification) {
  //0x01 : identifiant du paquet contenant les informations de batterie
  if(notification.ID_CODE === 0x01)
  {
      //La première donnée du paquet contient l'état de charge de la
```

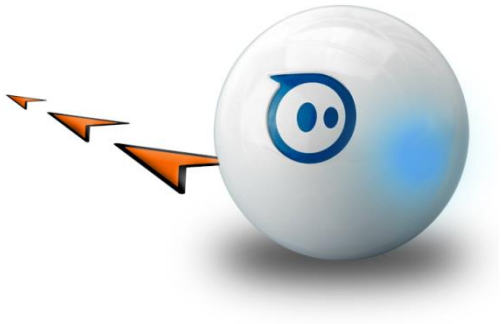
```
        //batterie. Il y a 4 état possible :
        switch(notification.DATA[0])
        {
            case 0x01:
                    console.log('Batterie en charge');
            break;
            case 0x02:
                    console.log('Batterie OK');
            break;
            case 0x03:
                    console.log('Batterie faible');
            break;
            case 0x04:
                    console.log('Batterie critique');
            break;
        }
        //On envoie les infos au client
        io.sockets.emit('BatteryInfo', notification.DATA[0]);
    }
});
```

B. Contrôler la Sphero

1. *Contrôle avec clavier/souris*

Pour la première phase de contrôle nous avions décidé de partir sur des bases simples. Les périphériques les plus utilisé sur un ordinateur étant le clavier et la souris, nous avons décidé d'utiliser ces derniers pour cette première étape… Ainsi, afin d'avoir un retour sur ce que nous demandons à la Sphero, nous avons ajouté à notre interface quatre flèches directionnelles, chacune correspondant aux flèches directionnelles présentes sur le clavier. Lorsque l'utilisateur appuie sur une des flèches du clavier, celles correspondantes sur l'interface changeront de couleur afin de d'informer l'utilisateur quelle touche est pressée. Afin de gérer l'événement d'appuie sur une touche nous avons utilisé un script nommée keyboard.js qui s'appuie sur les événements Javascript et permet de géré toutes les touches du clavier. Nous avons préféré utilisé cette solution plutôt que d'utiliser JQuery car celle-ci nous faisait gagner plus de temps car le code source de keyboard.js était déjà tout fait. Ainsi dans le code de keyboard.js il nous a suffi de rajouter des appels à socket.io dans chaque écouteur d'évènement nous permettant d'envoyer la requête de déplacement au serveur chaque fois qu'une touche est pressée.

Presser la flèche du haut permet de faire avancer la sphero vers l'avant, une pression sur la flèche de droite permet de changer son orientation de 10 degrés vers la droite et inversement pour la flèche de gauche. Une pression sur la flèche du bas permet de faire demi-tour. Nous avons de plus, ajouté d'autres fonctionnalités : une pression sur la touche « I » permet de faire apparaitre/disparaitre la petite LED permettant de déterminer l'orientation de la Sphero et la touche « A » permettant à la Sphero de se calibrer en tournant sur elle-même tant que la touche n'est pas relâchée.



La commande permettant de faire rouler la Sphero est la suivante:

```
// vitesse : de 0 à 255
// direction : angle de 0 à 360 degrés
// délais : durée pendant laquelle la Sphero continue de rouler
sphero.roll(vitesse, direction, délais);
```

Utiliser le clavier pour contrôler la Sphero fut la phase la plus simple, nous avons pu la mettre en œuvre très rapidement et disposer d'un contrôle assez basique mais qui cependant nécessitait plusieurs pressions à répétitions sur les touches restreignant notre temps de réaction et nous faisant manquer de précision dans les virages.

2. *Contrôle avec un joystick virtuel*

L'utilisation du clavier pour contrôler la Sphero fut un succès cependant, nous ne retrouvions pas les même sensations que sur les applications smartphone. Nous avions donc pensé à utiliser un joystick virtuel comme c'est le cas dans les applications pour mobiles à la différence que nous n'utiliserons pas nos doigts pour interagir avec ce dernier mais la souris. Ainsi, avec l'aide d'un canvas HTML nous créons un script Javascript permettant d'envoyer des instructions au robot Sphero via le serveur. En cliquant sur le canvas et en orientant le joystick vers la droite, vers la gauche, vers le haut ou vers le bas, nous récupérons l'angle de rotation ainsi que la vitesse que l'on souhaite attribuer à la Sphero.

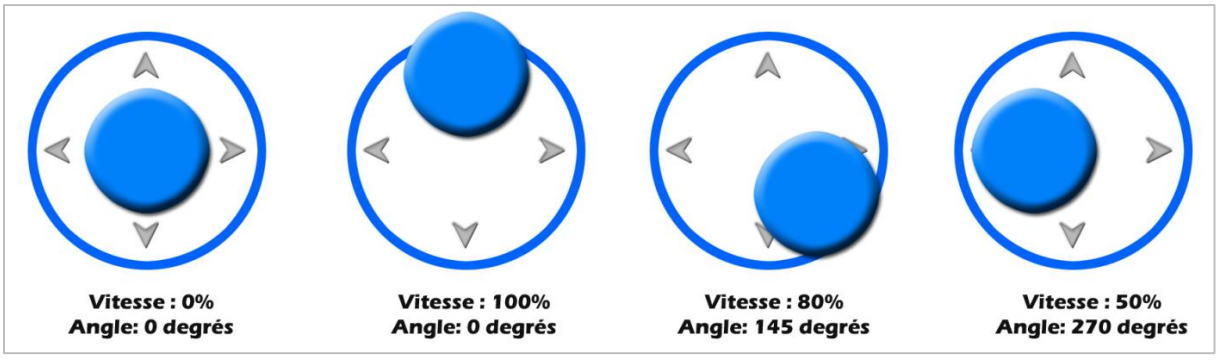


Figure 8: Relation entre la position du joystick et les données d'angle et de vitesse envoyés à la Sphero

Utiliser un joystick virtuel nous a permis de nous rapprocher de l'interface employé dans les applications mobiles d'Orbotix. Cependant, la précision donc nous faisons preuve avec nos doigts n'est pas la même que celle que nous avons avec une souris. Finalement, la solution du

joystick virtuel était intéressante mais celle-ci demande trop d'entrainement à l'utilisateur et perd tout son intérêt car la prise en main est trop compliqué pour être divertissante pour ce dernier.

C. Contrôler la Sphero avec la Leap Motion

Une fois la première étape de contrôle terminé, une Leap Motion nous a été confié afin d'avoir une alternative de contrôle pour la Sphero.

1. Qu'est-ce que la Leap Motion ?

La Leap Motion est une interface qui propose de s'affranchir de tout contact physique entre l'homme et la machine, plus précisément l'ordinateur, grâce au mouvement des mains, des doigts et de pattern de gestes simples (pointer, rotation des doigts, etc…) captés non pas sur une surface mais dans un volume en 3D devant notre écran. De la taille d'un gros briquet, celle-ci est souvent comparé à une version plus petite de la Kinect développée par Microsoft, la Leap motion peut toutefois distinguer individuellement les doigts et suivre les mouvements avec une précision de 1/100e de millimètre ("200 fois plus précis" que la Kinect selon ses concepteurs).

2. Utiliser la Leap Motion

L'utilisation que nous aurons de la Leap Motion se fera du coté client. Il faudra ainsi que l'utilisateur dispose d'une Leap Motion et télécharge le script Leap.js. Le script Leap.js est la version JavaScript de l'API de la Leap Motion est nous permettra de gérer toute les interactions que nous souhaitons réaliser avec le dispositif. La documentation de ce dernier est d'ailleurs bien étoffée et très claire, ainsi nous avons pu très rapidement mettre en place un prototype dans lequel nous contrôlons la Sphero grâce à la position de notre main et gérer des mouvements tel que le balayage avec la paume de la main ou encore la rotation des doigts. Nous allons grâce à la LeapMotion nous allons utiliser notre main comme un joystick en exploitant uniquement 2 dimensions de l'espace 3D perçues par la Leap : l'axe des X et l'axe des Z respectivement axe horizontale et axe de profondeur. La vitesse et la direction que la Sphero prendra sera déterminé par la position de la main par rapport à la Sphero (Cf. image ci-dessous). Plus la main de l'utilisateur sera éloigné du centre de détection de la leap motion plus la vitesse sera élevée, de la même façon si la main est positionnée à droite du centre de détection alors la direction de la Sphero sera égale à 90 degrés et inversement si la main est positionnée à gauche de centre de détection, si celle-ci est positionner au centre alors aucune instructions de déplacement n'est envoyées. Quant au pattern de mouvement, nous les avons exploités de la manière suivante : un geste de balayage aura comme un effet de coup de vent sur la Sphero, ainsi selon la direction du mouvement la Sphero roulera de manière rectiligne et à grande vitesse dans la direction du mouvement. Les patterns de rotations de doigts sont gérés de manière ce que tant qu'ils sont détectés, la Sphero se met à effecteur des tours sur elle-même afin de se recalibrer.

```
// Création de l'objet représentant la Leap Motion
var controller = new
Leap.Controller({frameEventName:'deviceFrame',enableGestures:true});
```

```
//Chaque frame est considéré comme un événement
controller.on('frame', function(frame) {
    //gestion des mouvements de type...
    if (frame.gestures.length)
    {
      var g = frame.gestures[0];
      //...balayage
      if (g.type == 'swipe')
      {
        console.log('mouvement de balayage détecté');
        //fonction gérant le mouvement de balayage
        GestionBalayage(g);
      }
      //...cercle
      if (g.type == 'circle')
      {
        console.log('mouvement circulaire détecté');
        //fonction gérant le mouvement de cercle
        GestionCercle(g);
      }
    }
});
```

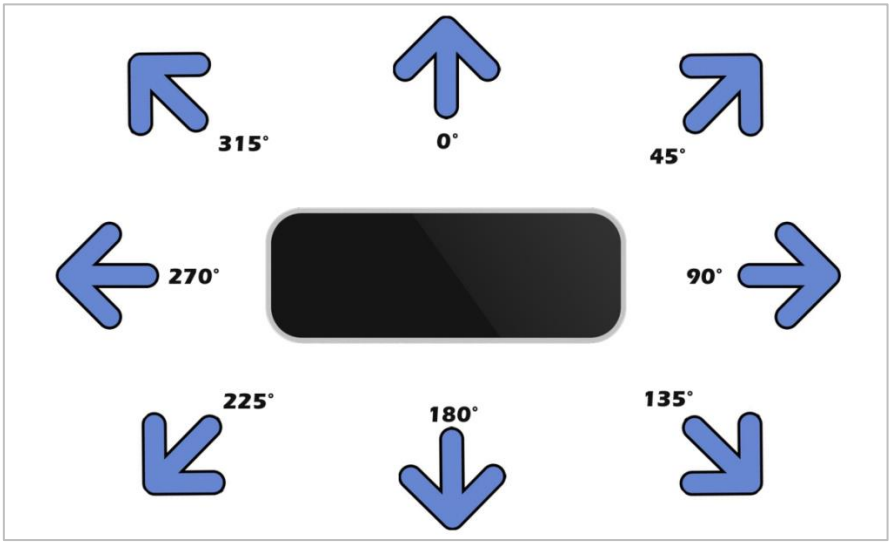


Figure 9: Relation entre la position de la main et la direction de la sphero

D. Créer un jeu utilisant la Sphero

Nous n'avons malheureusement pas eu le temps de terminer le jeu dans lequel nous pensions utiliser la Sphero en tant que Controller, cependant le concept est là et c'est ce que nous allons vous présenter dans cette partie. Nous savions que la Sphero était dotée d'un gyroscope ainsi que d'un accéléromètre et nous voulions utiliser les données de ces deux dispositifs afin de transformer la Sphero en « manette à 3 Dimensions ». Afin de gagner du temps, nous avions décidé de réaliser un jeu simple mais connu de tous: Pong. Les raquettes du jeu serait contrôlés grâces aux Spheros (vu que nous en avions deux, c'était parfait). Nous pensions utiliser un axe de rotation du robot afin de déplacer la raquette : une rotation dans le sens horaire ferait se déplacer la raquette vers la droite et inversement.

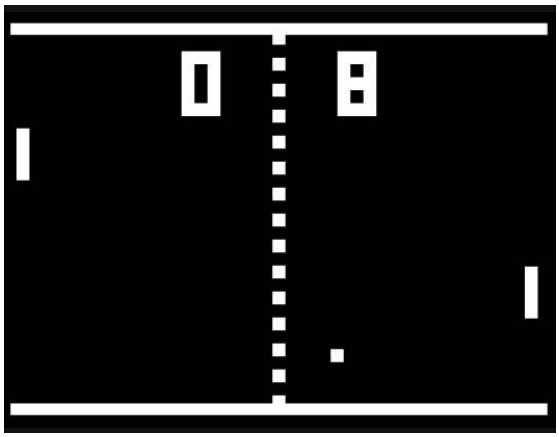


Figure 10: Illustration du jeu Pong

Pour récupérer les variables de la du gyroscope et de l'accéléromètre, 2 étapes sont nécessaire: stopper la stabilisation de la Sphero et activer le « Data Streaming ». Sans stopper la stabilisation nous n'aurions pu récupérer les variables du gyroscope de manière précise car la Sphero tente de se restabilisé automatiquement lorsque qu'elle n'est plus droite, c'est d'ailleurs pour cela que sa tenue de route est précise malgré sa forme sphérique. Sphero peut communiquer de manière asynchrone avec l'appareil auquel il est connecté. La commande « SetDataSteaming » permet d'activer cette communication et définir certaine option liée à celle-ci. Elle permet entre autre de choisir la fréquence d'échantillonnage interne, la taille des paquets et le masque de paramètre. Le masque de paramètre est un masque binaire permettant de sélectionner la source de données (gyroscope, accéléromètre, etc.) dont on souhaite récupérer les valeurs. Afin de récupérer les valeurs de ces sources de données nous avons procédé de la manière suivante :

```
//Lorsqu'on reçoit l'instrution de parameter la sphero pour le jeu
socket.on('ApplyGameOption', function () {
      if(isConnected){
            //Définition du masque permettant de récupéré les données
            //du gyroscope
            var maskX = 0x00040000, //Gyro axis X
                maskY = 0x00020000, //Gyro axis Y
                maskZ = 0x00010000, //Gyro axis Z
                mask = maskX | maskY | maskZ;
            //On désactive la stabilisation
            sphero.setStabalisation(false);
            //On le active le data streaming (fréquence: 40Hz)
            sphero.setDataStreaming(40, 1, mask, 0);
            console.log('Sphero est pret pour le jeu !');
      }
      else{console.log('Error: sphero is not connected !!!!');}
});

//Les données du gyroscope sont reçues sous forme de paquet
//Ainsi, lors de la reception d'un paquet :
sphero.on('packet', function(packet) {
      //si l'identifiant du paquet correspond au « Sensor data streaming »
      if (packet.ID_CODE === 0x03) {
            //On récupère les valeurs codées sur une base hexadecimal
            var roll = packet.DATA.readInt16BE(0);
```

```
            var pitch = packet.DATA.readInt16BE(2);
            var yaw = packet.DATA.readInt16BE(4);
            //On envoie les les valeurs au client
            io.sockets.emit('SendGyroValues', roll, pitch, yaw);
        }
});
```

6. Contraintes/Difficultés :

Durant le projet nous avons rencontré diverse difficultés notamment dues à notre inexpérience avec Node.js et le langage Javascript. Nous sommes souvent échoué durant nos phases de test, mais nous nous sommes relevés, et avons persévéré jusqu'à ce que nous puissions produire un code fonctionnel qui cependant reste modeste.

Aussi l'une des difficultés qui nous à poser le beaucoup de soucis ainsi qu'une perte de temps considérable a été la phase de connexion avec la Sphero. Nous avons constaté un gros souci de connexion Bluetooth avec le robot et les machines sous le système d'exploitation Mac OS X qui entrainait des déconnexions intempestives entre les deux appareils. Nous avons alors essayé de nous connecté avec une machine sous Windows. Tout semblait fonctionné, malgré quelque difficulté lors de la configuration d'installation du module Spheron, et la connexion Bluetooth se passait sans accroc. Le problème vint lors de l'exécution du code de test (celui permettant d'établir la communication avec Sphero, cf. p.14). Celui-ci ne fonctionnait pas… Nous avons bien entendu configuré le port de manière à ce qu'il réponde au critère de Windows (les connexions Bluetooth se faisant sur les ports COM) mais en vain… Voyant que nous perdions du temps à résoudre les problèmes de connexion sur ces deux OS, nous décidâmes de passer sur l'OS Ubuntu grâce à une machine virtuelle. De peur que cette solution ne fonctionne pas non plus, une partie de l'équipe à commencer à développer sur arduino afin de gérer la connexion Bluetooth à défaut de pouvoir se connecter via un pc. L'utilisation de la machine virtuelle fut un échec due cette fois ci aux erreurs liées aux dépendances des modules Node.js et plus exactement sur le module sensé géré la communication Bluetooth, le module « serialport ». Mais finalement, le problème fut résolu en utilisant pc nativement sous Ubuntu, et nous pûmes abandonnés l'idée de la connexion via arduino. Après avoir passé beaucoup de temps a essayé de se connecter à la Sphero, nous nous sommes retrouvé à devoir travaillé à grande cadence pour pouvoir rattraper le retard. C'est ainsi que le temps devint notre plus grosse contrainte.

Comme dit précédemment, la seconde difficulté fut notre inexpérience avec la technologie Node.js et le langage JavaScript. Faire appel à des script ce trouvant dans un autre dossier, animé un canvas et déterminé son framerate avec Javacript , etc… beaucoup de choses pourtant simples ont été pour nous le sujet de plusieurs heures de réflexion mais auxquels nous avons réussi à venir à bout.

V) Conclusion

Ce projet nous a permis de découvrir un nouveau langage que nous n'avions jamais utilisé, le JavaScript ainsi que l'environnement Node.js. Un des atouts de ce projet est que nous avons développé une application web pour mettre en œuvre le contrôle du robot. Ce projet nous a permis de mettre à l'épreuve nos connaissances en développement ainsi que d'autres notions liées à un tel projet tel que la gestion de projet et le travail en équipe. Comprendre l'architecture de trames envoyées par la Sphero mais afin d'en exploiter les informations fut très instructif et nous as beaucoup appris sur la communication entres 2 machines.
Bien que nous ayons manqué de temps, nous avons été assez satisfait des résultats que nous avons obtenus. Et nous avons encore beaucoup d'idée avec lesquels nous pourrions exploiter les capacités de Sphero. Il existe en effet plusieurs aspects de ce dernier qui peuvent être améliorés ou modifiés. On pourrait penser à une autre manière d'interagir avec la Sphero via la LeapMotion ou encore le développement d'un jeu vidéo complet utilisant les caractéristiques de la Sphero, voire d'utiliser plusieurs Sphero dans ce même jeu. Nous espérons ainsi que notre travail sera mené à évoluer et que notre retour d'expérience sera bénéfique à l'avancée des recherches et des prochains développements sur des sujets semblables.

Bibliographie/Webographie

Test de la sphero 1.0

- *http://www.humanoides.fr/2013/06/02/test-sphero-balle-robotique-smartphone/*

Test de la Sphero 2.0

- *http://belgium-iphone.lesoir.be/2013/11/06/test-de-la-sphero-2-0-une-balle-etonnante-controlable-depuis-son-appareil-sous-ios/*

Présentation des robot POB, AR Drone et Robomow

- *http://www.generation-nt.com/robot-pob-wifi-pilotage-iphone-fnac-actualite-1221611.html*
- *http://www.generationrobots.com/fr/400907-parrot-ar-drone-2-0-carene-externe-verte.html*
- *http://www.kelrobot.fr/2014/03/03/nouveau-robot-tondeuse-robomow-tuscania-tc/*

Page NPM du module spheron

- *https://www.npmjs.org/package/spheron*

Repository de spheron

- *https://github.com/alchemycs/spheron*

Documentation officiel de Leap Motion

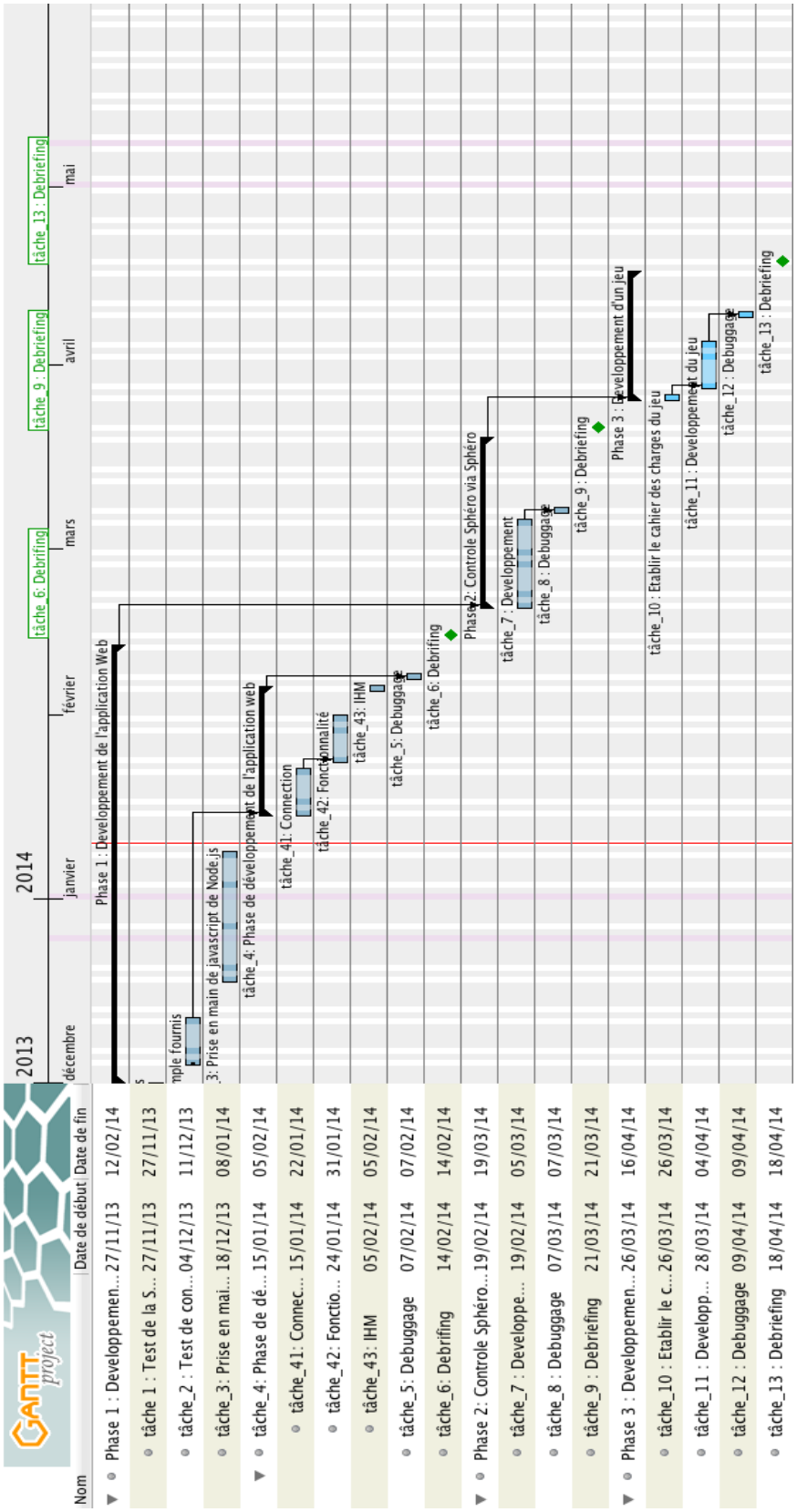
- *https://developer.leapmotion.com/documentation/javascript/index.html*

Documentation officiel de Sphero

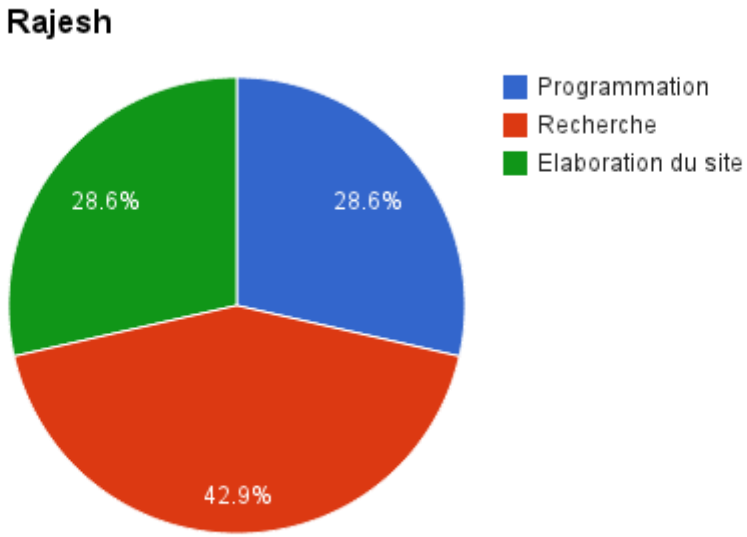
- *http://orbotixinc.github.io/Sphero-Docs/docs/sphero-api/index.html*

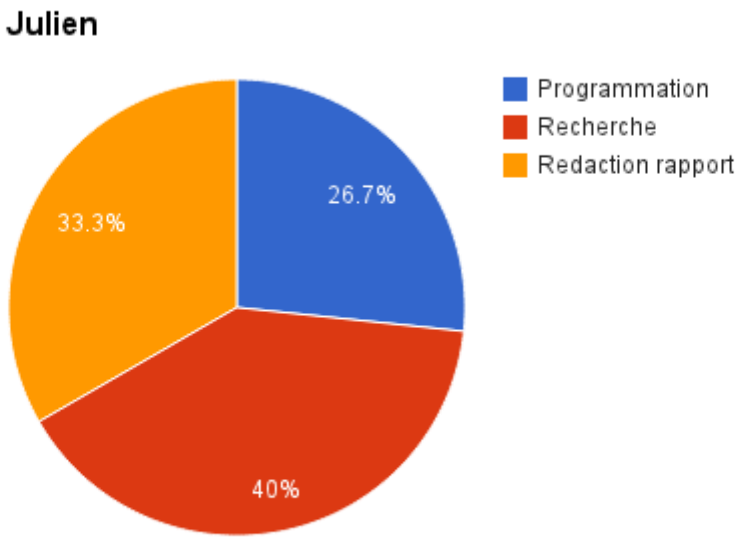
Annexes

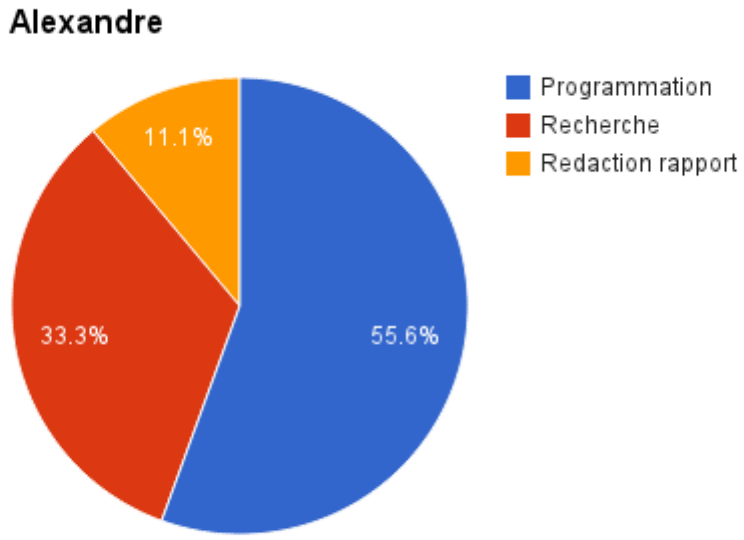
1. Diagramme de GANTT



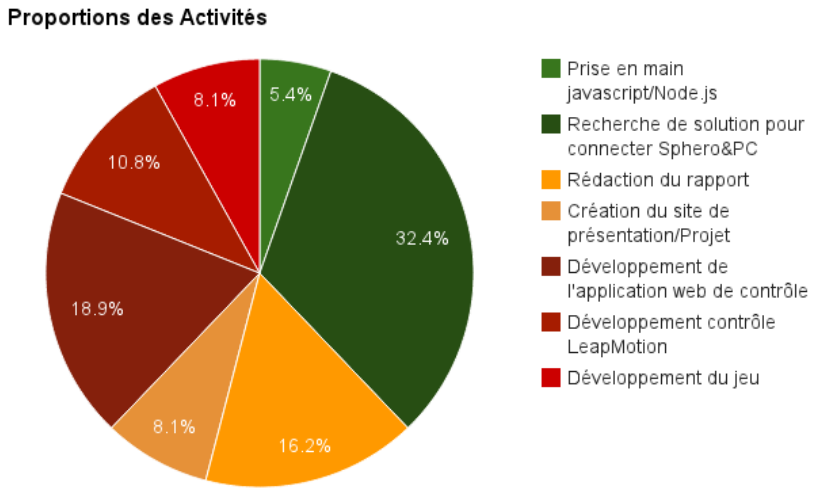
2. Répartition des tâches par personne







3. Répartition des différentes activités



(On distingue 3 grandes parties: une partie Initialisation&Recherche en vert[~37.8%], une partie Redaction&Rapport en jaune[~24.3] et une partie Développement en rouge[~37.8].)

Mots-Clés

Sphero, Node.js, JavaScript, Robot, Bluetooth, Smartphone, Jouet intelligent, Programmation, Application Web, Interface, Jeux, Télécommande.

Résumé

Dans le cadre du projet d'EI4 AGI à l'ISTIA nous avons été amenés à travailler sur un robot en forme de sphère nommé Sphero, ce produit a été développé par la société Orbotix. Ce robot est un robot commandable avec son smartphone grâce au Bluetooth.

L'objectif de notre projet fut de développer une application Web permettant de le contrôler à distance depuis un serveur grâce à la technologie Node.js. Tout au long de ce projet nous avons exploité les différentes fonctionnalités de la sphero afin de développer une application de pilotage clavier/souris puis nous l'avons amélioré afin de pouvoir le contrôler depuis une Leap Motion (Contrôle par gestuel de la main). Enfin, au terme de ce projet, nous voulions mettre en œuvre à travers un jeu dans lequel nous utiliserions cette dernière en tant que contrôleur. Pour ce faire nous utiliserons le gyroscope intégré permettant de détecter les rotations qui sont appliquées à la sphero quand on le bouge manuellement.

Durant ce projet nous avons travaillé avec différentes technologies que nous n'avions jamais utilisées avant telles que Node.js, Socket.io et Spheron. Tout au long de ce rapport nous allons vous montrez en quoi ces technologies nous ont aidé et comment elles marchent.

Summary

Under the project on EI4 AGI in ISTIA, we got the chance to work with a robot shaped sphere called Sphero, this product was developed by Orbotix. This robot can be controlled by Bluetooth by your smartphone.

The project aimed to develop a web application provides the after control from a server thank to Node.js. Along this project we took advantage from different function of the Sphero in order to develop an application of control with keyboard / mouse then we improved it in order to control it from a Leap Motion. (Steering with hand gesture) Finally, at the end of the project, we wanted to implement a game where we use a Sphero as a controller. To make this we can use the gyroscope inside Sphero that allows us to detect the Sphero's rotation when we move it manually.

During this project we worked with different technologies we never use before as Node.js, Socket.io and Spheron. Along that report we will show you how those technologies help us and how they work.