|  |  |
| --- | --- |
|  | Télécommande d’un robot Stinger avec un iPod Touch |
|  | http://bindapple.com/wp-content/uploads/2009/11/compare-and-contrast-iphone-and-ipod-touch.pnghttp://www.roboticsconnection.com/skins/skin_1/images/rc_robot_kits/StingerRobotKitLarge.png |
| 13/01/2011 | Alexandra HUILLET & Nicolas BEAUDROT – TX52 |
| http://blog.myprof.fr/wp-content/uploads/2009/01/utbm.jpeghttp://set.utbm.fr/img/logo.gif | Dans le cadre de la recherche sur la robotique mobile du laboratoire SET, les nouvelles approches portent sur la conduite à distance. Le but du travail proposé ici, est de mettre en œuvre un système permettant de contrôler un robot de type Stinger à distance avec un iPod Touch. |

Sommaire

[I. Cahier des charges 2](#_Toc282423879)

[1.1. L’interface Homme-Machine 2](#_Toc282423880)

[1.2. Le robot Stinger 2](#_Toc282423881)

[II. Analyse et conception du projet 3](#_Toc282423882)

[2.1. Etude du robot Stinger 3](#_Toc282423883)

[2.2. Analyse UML sur les fonctionnalités de l’interface 4](#_Toc282423884)

[2.3. Analyse sur les classes à développer 6](#_Toc282423885)

[III. Fonctionnement de l’application 7](#_Toc282423886)

[3.1 Communication iStinger/Robot 7](#_Toc282423887)

[3.2 Diriger le robot 7](#_Toc282423888)

[3.3 Présentation de l’interface 8](#_Toc282423889)

# Cahier des charges

L’objectif de ce projet est de mettre en œuvre une interface Homme-Machine pour un iPod Touch, dite « télécommande », visant à contrôler un robot de type Stinger à distance, à l’aide d’un réseau Wifi.

Le développement de l’application iPod doit être réalisé sur Mac OS avec le logiciel Xcode et l’API Cocoa. Cette API est utilisée pour réaliser l’architecture graphique de l’application et simuler le programme avec l’affichage d’un iPod Touch virtuel. Le langage de programmation qui procède à la réalisation de ce type d’interface est l’objective-C, un langage basé sur le C, mais orienté objet, appliquant la structure MVC et spécifique aux produits de Apple.

## L’interface Homme-Machine

L’interface Homme-Machine de l’application doit être simple, ergonomique et fonctionnelle. Elle doit disposer de boutons pour diriger le robot (avancer, tourner, reculer). Elle peut également utiliser l’accéléromètre de l’iPod Touch, afin d’entièrement le diriger comme-ci l’on disposait d’un volant (télécommande sans l’utilisation des boutons).

La gestion de la communication wifi et ainsi que les caractéristiques du robot devront être manipulés par l’intermédiaire d’une vue « options ». Dans celle-ci, on doit accéder à la modification de l’adresse IP et du port pour la communication wifi, et posséder un système pour gérer la vitesse du robot.

Une rubrique « aide » doit aussi être implémentée pour expliquer l’utilisation de l’interface et son paramétrage, ce qui facilitera le confort de l’utilisateur.

## Le robot Stinger

Un manuel est fourni pour expliquer les commandes du robot. Une commande de ce type, devra être envoyée par wifi au robot lorsque l’on interagi avec l’interface. Toute communication sera gérer par ce réseau à l’aide d’une carte wifi disposée sur le robot. Ainsi, l’iPod pourra se connecter comme un client sur le robot (serveur). Par l’intermédiaire de l’interface « options », on doit accéder à la modification de la vitesse du robot.

# Analyse et conception du projet

Avant de procéder au développement du projet, nous avons étudié :

* la documentation complète du robot afin de bien comprendre son fonctionnement et de définir les applications qui peuvent être réalisées en fonction des valeurs qu’il nous communique
* les fonctionnalités nécessaires au fonctionnement de l’application iPod en tant que télécommande
* l’architecture de l’interface, pour être la plus ergonomique possible pour l’utilisateur
* l’organisation des classes à développer en utilisant le modèle MVC afin de faciliter la réalisation de l’interface.

## Etude du robot Stinger

A FINIR

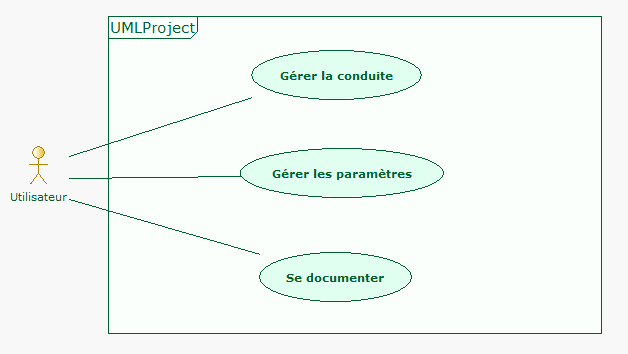
Fonctionnement du robot Stinger

* calcule de vitesse
* comment utiliser les commandes robots

## Analyse UML sur les fonctionnalités de l’interface

Afin de répertorier l’ensemble des fonctionnalités, que doit posséder l’interface de notre télécommande pour manipuler le robot, nous avons dans un premier temps procédé à une analyse UML. Cette analyse nous a permit d’organiser par la suite l’architecture de l’interface et les classes à créer pour le développement du projet.

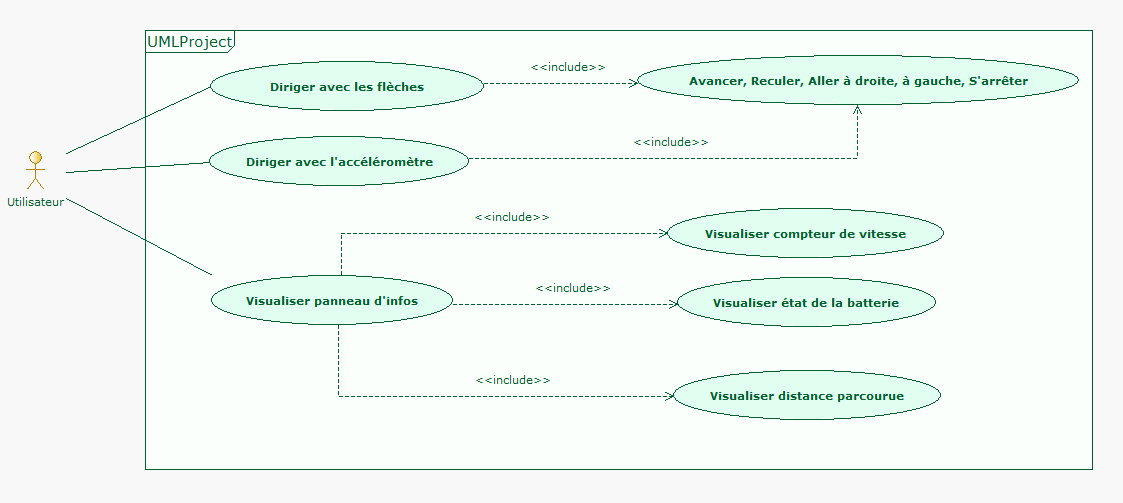
**Conception générale de l’interface**



*Fig. Conception générale de l’interface*

De manière générale, l’utilisateur doit avoir à disposition trois principales fonctionnalités dans l’interface (cf. schéma ci-dessus) :

* « Gérer la conduite » afin qu’il puisse diriger le robot où il le souhaite
* « Gérer les paramètres », une fonctionnalité indispensable pour le bon fonctionnement du robot
* « Se documenter », correspondant à une documentation sur l’utilisation de cette application iPod. Cette documentation rendra l’application plus ergonomique pour l’utilisateur, c'est-à-dire, elle facilitera sa compréhension sur le bon fonctionnement de l’interface.

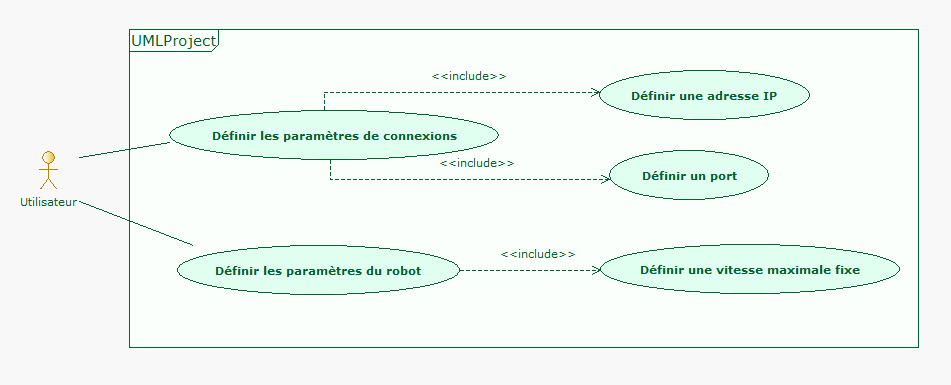
**Analyse sur « Gérer la conduite »** 

*Fig. Analyse sur « Gérer la conduite »*

« Gérer la conduite » (cf. figure ci-dessus) consiste à manipuler le robot de deux façons.

La première est de conduire le robot avec les flèches des boutons de l’interface et la seconde de le conduire en manipulant l’iPod à des degrés différents comme-ci c’était un volant. Ainsi on peut faire avancer, reculer et tourner le robot. Nous avons également rajouté la commande « arrêt » car la documentation du robot nous informe qu’il ne s’arrête pas de lui-même, il faut donc lui envoyer une commande « stop ». Comme le robot nous retourne des informations sur sa batterie et sa vitesse, nous avons donc pensé à rajouter dans les fonctionnalités de l’interface, un panneau d’informations. Dans celui-ci nous avons inclus un compteur de vitesse, un indicateur sur l’état courant de la batterie, et un compteur sur la distance parcourue (calcul obtenu à partir de la vitesse).

A**nalyse sur « Gérer les paramètres »**



*Fig. Analyse sur « Gérer les paramètres»*

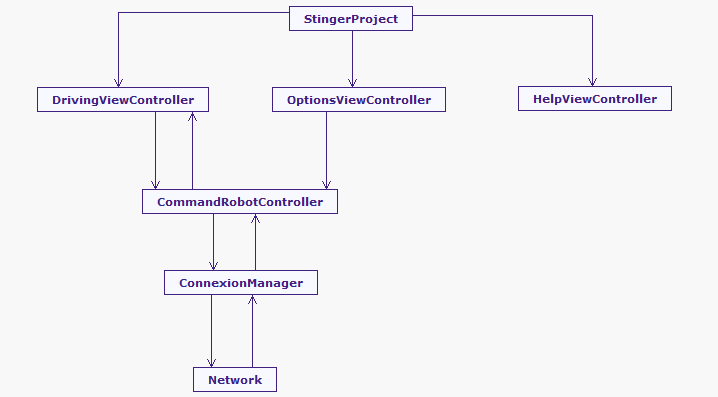
Certains paramètres nécessites d’apparaître dans l’interface pour faire fonctionner la communication entre l’application iPod et le robot. Comme leur communication se fait en wifi sous la forme d’un serveur-client, il faut donc définir des paramètres de connexion, où l’on renseigne l’adresse IP et le port utilisé pour envoyer des commandes au serveur.

D’autres paramètres interviennent dans l’interface, plus propres au robot, comme la modification de sa vitesse maximale. Si par la suite le robot possèdera de nouveaux équipements (une caméra, des capteurs frontaux...), leurs paramètres pourront apparaître ici.

## Analyse sur les classes à développer

Le langage de programmation Objective-C est un langage orienté-objet qui utilise le modèle MVC.

Nous avons donc réfléchi à l’organisation des classes à développer tout en respectant ce modèle. Ainsi des classes devront représenter les « vues » de l’interface et d’autres manipuleront les commandes du robot et interagiront avec l’interface pour afficher des résultats. Nous devons aussi implémenter des classes pour la gestion d’un serveur client.



*Fig. Analyse sur les classes à développer*

Dans un premier temps, nous avons une classe « générale » qui gère l’ensemble des autres classes. Ensuite, nous avons trois classes qui gèrent les « vues » de l’interface (DrivingViewController, OptionsViewController, HelpViewController) c’est-à-dire la partie visuelle manipulée par l’utilisateur (affichage, boutons…).

La classe **HelpViewController** correspond à la vue qui affichera la documentation utilisateur, elle n’a pas besoin d’interagir avec les autres classes, il s’agit juste d’un affichage.

La classe **OptionViewController** gère la vue sur les paramètres de connexions avec le serveur et du robot. Les données modifiées par l’utilisateur sont récupérées dans cette classe et par la suite envoyées vers :

* la classe **CommandRobotController** quand il s’agit des paramètres du robot
* la classe **ConnexionManager** quand il s’agit de modifier les paramètres de connexion

La classe **DrivingViewController** permet de gérer la conduite du robot, c’est-à-dire les boutons de directions et l’affichage du panneau d’informations. Elle envoie ses données venant de l’utilisateur et reçoit aussi des informations de la classe **CommandRobotController** qu’elle affiche sur l’interface.

La classe **CommandRobotController** est de type « controller » dans le MVC, c’est-à-dire qu’elle manipule les données qui ont été envoyées par les « vues » précédentes. Elle permet d’envoyer les commandes étudiées dans la documentation du robot, au serveur. Certaines commandes que l’on envoie au robot retourne un résultat, c’est pour cela qu’elle peut recevoir des informations du serveur et ainsi diffuser ce résultat à une vue, pour que celle-ci l’affiche sur l’écran de l’iPod.

La classe **ConnexionManager** définie la configuration du serveur en fonction de ce qu’elle a reçu de la classe **OptionsViewController** (ip, port) et lance la connexion du serveur avec la classe **Network.** Elle transmet également les commandes du robot qu’elle a reçu, à la classe **Network.** Le serveur recevant aussi les résultats des commandes du robot, les retransmet à **ConnexionManager** qui les fait suivre à **CommandRobotController.**

*Exemple d’un scénario : Si l’utilisateur appuie sur un bouton de direction du robot, la fonction qui gère ce bouton dans DrivingViewController, appelle une fonction dans CommandRobotController. Cette dernière envoie au serveur la commande du robot qui correspond à la demande.*

# Fonctionnement de l’application

## 3.1 Communication iStinger/Robot

TODO : schéma, socket, exemple

## 3.2 Diriger le robot

TODO : schéma sur le fonctionnement de l’accéléromètre, communication entre les classes

## 3.3 Présentation de l’interface

TODO : imprim écran de l’application et explication des fonctionnalités