



École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
3^e année
2011 - 2012

Rapport de projet d'Algorithme et de Langage C

Simulation du trafic urbain

Encadrants

Néron EMMANUEL
emmanuel.neron@univ-tours.fr

Université François-Rabelais, Tours

Étudiants

Lei SHANG
lei.shang@etu.univ-tours.fr
Yanxiao HU
yanxiao.hu@etu.univ-tours.fr

DI3 2011 - 2012

Version du 15 juin 2012

Table des matières

1	Introduction	5
2	IDM (Intelligent Driver Model)	6
2.1	Définition	6
2.1.1	Caractéristique	7
2.1.2	Cadre de l'application	7
2.1.3	Bibliothèques	7
2.1.4	Phase d'exécution Android	7
2.1.5	Noyau Linux	8
2.2	Fonctionnement des applications Android	8
2.3	Environnement de développement	8
3	Algorithme de l'application	12
3.1	Introduction	12
3.2	Utilisation	12
4	OpenGL	16
4.1	Introduction	16
4.2	Format des commandes	17
5	Mise en oeuvre	20
6	Conclusion	22

Table des figures

2.1	Architecture du système Android ¹	6
2.2	Installation de ADT - Add repository	9
2.3	Preferences Android	10
2.4	Android SDK Manager	10
2.5	Création d'un AVD	11
3.1	La carte AndroPOD	13
3.2	AndroPOD Interface Controller	14
3.3	Schéma de la carte AndroPOD	15
4.1	Robot Lynxmotion	16
4.2	Terminal de contrôle : LynxTerm	17
4.3	Mouvement des servomoteurs du robot	18
4.4	Schéma de la carte SSC32 intégrée avec le robot	19
5.1	L'interface de l'application	20
5.2	L'oeuvre finale	21

Introduction

Ce projet est développé dans le cadre de la formation de l'algorithme et le langage C en troisième année du département informatique. Le but du projet est de pratiquer les connaissances que nous avons acquises durant les séances.

Pour cela, nous sommes demandés à développer une application avec langage C pour simuler le trafic urbain ,une file de voiture avec 2 feux de circulation. Donc il nous faut étudier à la fois :

- Le modèle mathématique du mouvement des voitures sur la route
- L'algorithme pour visualiser le modèle ci-dessus
- La réalisation langage C de l'algorithme ci-dessus (Nous avons choisi le technique OpenGL pour le réaliser)

Notre travail, ainsi que ce rapport, est organisé selon ces trois parties.

IDM (Intelligent Driver Model)

IDM est un modèle de la simulation du trafic urbain. Il est développé par Treiber, Hennecke and Helbing en 2000 pour améliorer les modèles existants qui ont moins de propriétés réelles.

2.1 Définition

$$\frac{dv}{dt} = a \left[1 - \left(\frac{v}{v_0} \right)^\delta - \left(\frac{s}{s^*} \right)^2 \right]$$
$$s^* = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}}$$

L'architecture d'Android se compose de 4 couches. Elles sont la couche d'application, le cadre de l'application, les bibliothèques et le noyau Linux. La figure 2.1 montre les principaux composants du système d'exploitation Android.

2.1.1 Caractéristique

Android fournit un ensemble d'applications de bases, y compris un client email, une application SMS, un calendrier, une carte, un navigateur, etc.

2.1.2 Cadre de l'application

En fournissant une plateforme de développement ouvert, Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique, d'obtenir les informations de la position géographique, d'exécuter les services d'arrière-plan, d'ajouter des notifications à la barre d'état, etc.

Les développeurs ont un accès complet aux APIs utilisés par les applications de noyau. L'architecture d'application est conçue pour la réutilisation des composants.


Toutes les applications sont une collection des services, y compris :

- Une collection riche de *View* qui peuvent être utilisée pour construire une application, y compris des listes, des grilles, des zones de texte, des boutons et même un navigateur web à embarquer.
- Des *Content Providers* qui permettent aux applications d'accéder aux données provenant d'autres applications (tels que les *Contacts*), ou de partager leurs données.
- Un *Resource Manager* qui offre l'accès aux ressources comme les chaînes de caractères localisées, les graphiques, et les fichiers de mise en page.
- Un *Notification Manager* qui permet toutes les applications à afficher des alertes personnalisées dans la barre d'état.
- Un *Activity Manager* qui gère le cycle de vie d'application et fournit une pile de navigation.

2.1.3 Bibliothèques

Android comprend une collection de bibliothèque de C ou C++ utilisé par les composants différents. Ces capacités sont fournies par le cadre d'application Android. Certaines de ces bibliothèques de noyau sont montrées ci-dessous :

1. <http://developer.android.com/guide/basics/what-is-android.html>



pics/ch1_architecture.png

FIGURE 2.1 – Architecture du système Android¹

- System C library - une implémentation de la bibliothèque C standard du système (libc).
- Media Libraries - basé sur OpenCORE de PacketVideo. Des bibliothèques permettent l'enregistrement audio et vidéo, ainsi que des fichiers d'images, y compris MPEG4, H.264, MP3, AAC, AMR, JPG et PNG.
- Surface Manager - gère l'accès au sous-système d'affichage
- LibWebCore - c'est un moteur de navigateur moderne qui alimente le navigateur Android et une Web View embarqué.
- SGL - le moteur graphique 2D.
- 3D Libraries - une implémentation basée sur OpenGL ES 1.0 APIs ; les bibliothèques utilisent l'accélération matérielle 3D (si disponible) ou le logiciel 3D optimisé hautement.
- SQLite - un moteur de base de données relationnelles puissant et léger à la disposition pour toutes les applications.

2.1.4 Phase d'exécution Android

Android comprend une collection des bibliothèques de noyau qui offrent la plupart des fonctionnalités disponibles du langage Java.

Chaque application Android s'exécute dans son propre processus, avec sa propre machine virtuelle

Dalvik. Dalvik est conçue pour qu'un matériel puisse exécuter efficacement plusieurs machines virtuelles en même temps. Le VM Dalvik exécute les fichiers de format exécutable Dalvik (.Dex), qui est optimisé pour la minimale d'occupation mémoire. Le VM Dalvik dépend le noyau Linux pour les fonctionnalités de bas-niveau comme le management de la mémoire.

2.1.5 Noyau Linux

Android est basé sur le noyau Linux 2.6 pour les services du système noyau, tels que la sécurité, la gestion de la mémoire, la gestion des processus, etc. Le noyau est regardé comme une couche d'abstraction entre le matériel et le reste de la pile logicielle.

2.2 Fonctionnement des applications Android

Toutes les applications Android sont écrites en langage Java. Les outils du SDK Android compile le code — avec toutes les données et les fichiers de ressources — dans un package Android, un fichier d'archive avec un suffixe *.apk*. Tout le code dans un seul fichier *.apk* est considéré comme une seule application.

Quand installé sur un matériel, l'application Android vit dans sa propre boîte de sécurité :

- Le système d'exploitation Android est un système multi-utilisateurs, sous lequel chaque application est un utilisateur différent.
- Par défaut, le système attribue à chaque application un ID utilisateur Linux qui est utilisé uniquement par le système et qui est inconnu pour l'application. Le système définit les autorisations de tous les fichiers de l'application, pour que seulement les utilisateurs qui ont le droit peuvent y accéder.
- Chaque processus a sa propre machine virtuelle (VM), donc l'application fonctionne séparément.
- Par défaut, chaque application s'exécute dans son propre processus Linux. Android démarre le processus lorsqu'un composant de l'application doit être exécuté, et arrête le processus quand il n'est plus nécessaire ou quand le système doit récupérer la mémoire pour d'autres applications.

Une application Android se compose d'une collection des composants. Il y a 4 composants principaux : *Activity*, *Service*, *ContentProvider*, *BroadCast receiver*.

2.3 Environnement de développement

Installation de JDK

On développe l'application Android avec le langage Java, donc le JDK est essentiel. Dans ce projet, nous utilisons le JDK version 6, qui peut être téléchargé du site officiel².

Installation de SDK Android

Nous avons téléchargé le SDK Android depuis le site développeur Android³. Puis, nous les avons extraits sous le répertoire *C:\Program Files\android-sdk-windows*.

Installation de Eclipse

Eclipse est un IDE gratuit très célèbre dans le domaine informatique. On peut le télécharger à partir du site Eclipse, la version *Eclipse IDE for Java Developers* est recommandée⁴.

Installation de ADT

Dans Eclipse, cliquer par séquence : *help* → *Install New software* → *Add*, puis remplir les champs en fonction de la figure 2.2 :

2. <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

3. <http://developer.android.com/sdk/index.html>

4. <http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/indigosr2>



FIGURE 2.2 – Installation de ADT - Add repository

Après cliquer sur "OK", Eclipse affiche les plugins disponibles. Nous sélectionnons le "Android DDM-S"(Android Dalvid Debug Monitor Server) et le "ADT"(Android Development Tools). Nous validons les étapes suivantes et nous redémarrons Eclipse pour finir l'installation des plugins. Maintenant on peut trouver les paramètre Android ici : Windows → Preferences → Android.

Configuration de SDK

Sous le menu Windows → Android SDK Manager, on peut sélectionner et installer les paquets nécessaires. Dans notre cas, API version 2.3 est suffisante. Ce qu'il faut faire attention, c'est que la version API doit être supporté par l'appareil ciblé.

Configuration de AVD

Pour tester notre application pendant le développement, on peut utiliser directement notre portable Android, ou bien la lancer dans un appareil virtuel(AVD). Pour ce dernier, on peut lancer le gestionnaire AVD sous le menu *Windows*, et créer un nouveau AVD en précisant les paramètre.



FIGURE 2.3 – Preferences Android



FIGURE 2.4 – Android SDK Manager



FIGURE 2.5 – Création d'un AVD

Algorithme de l'application

3.1 Introduction

L'Interface AndroPOD sert d'un port série pour les téléphones ou les tablettes Android. Il fonctionne avec l'appareil Android non modifié. Contrairement à toutes les autres solutions similaires sur le marché, l'interface AndroPOD permet les développeurs pour déboguer l'application de téléphone lorsque le téléphone est connecté à l'interface AndroPOD. Donc c'est un outil qui peut beaucoup faciliter le développement des applications Android.

Deux articles^{1 2} provenant du magazine *Elector* introduisent en détail la fonctionnalité de la carte avec un exemple réalisé. La carte était aussi commandée sur ce site.

L'interface AndroPOD est conçue pour être intégré dans votre propre projet. Nous pouvons l'utiliser dans plusieurs situations, par exemple pour les robots, les gadgets ou tous autres projets qu'on peut penser. Avec l'interface AndroPOD, on renforce son projet avec les fonctionnalités WIFI, 3G, 4G, Bluetooth, GPS etc. Si on a besoin un grand écran couleur avec écran tactile, AndroPOD peut aussi offre ses aides. Dans notre projet, nous l'utilisons pour que l'on puisse contrôler le robot Lynxmotion avec notre appareil³ Android.

3.2 Utilisation

Sur ce site⁴, nous avons trouvé un tutorial détaillé pour prendre en mains AndroPOD.

Pour connecter et communiquer avec la carte à partir du bout Android, il suffit d'établir un serveur TCP sur le port 1337.

```
try
{
    //Etablir un serveur sur le port 1337
    ServerSocket socket = new ServerSocket(1337);

    //Attendre la connexion de Andropod
    Socket andropodInterface = socket.accept();

    //Ouvrir IO flux
    PrintReader in = new BufferedReader(new InputStreamReader(andropodInterface.getInputStream()));
    PrintWriter out = new PrintWriter(andropodInterface.getOutputStream(), true);

    //Envoyer ce que vous voulez
    out.println("Hello Andropod!");

    //Recevoir les données
    String message = in.readLine();
}
```

1. [http://www.elector.fr/magazines/2012/fevrier/andropod-\(1\).2057762.lynkx](http://www.elector.fr/magazines/2012/fevrier/andropod-(1).2057762.lynkx)

2. [http://www.elector.fr/magazines/2012/mars/andropod-\(2\).2085663.lynkx](http://www.elector.fr/magazines/2012/mars/andropod-(2).2085663.lynkx)

3. Samsung Galaxy Nexus, fourni par l'école

4. <http://www.xdevelop.at>

```
}  
catch (IOException e)  
{  
    //En cas d'exception  
}
```

D'autre part, pour relier AndroPOD et le mobile avec le câble usb-microusb fourni, il suffit de faire après ces 5 étapes :

1. Configuration de l'interface Andropod avec le PC

Installer le logiciel de configuration Andropod sur le PC, changer l'interface Andropod en "Configure mode" (voir figure 3.1) et la connecter au PC en utilisant un USB câble. Ensuite, on peut éventuellement configurer les autres paramètres avec *AndroPOD Interface Controller* (figure 3.2).



FIGURE 3.1 – La carte AndroPOD

2. Connecter-la au robot

Utiliser quatre fils simples, dont deux pour le voltage +5 V, GND et les deux autres pour RX et TX. L'interface AndroPOD a besoin de 5V vers 550mA afin de charger le téléphone Android (si on n'arrive pas à charger le téléphone lorsqu'il est connecté à l'USB, l'énergie sera consommée très vite. C'est notre cas.). Maintenant une LED sur l'interface Andropod doit s'allumer.



FIGURE 3.2 – AndroPOD Interface Controller

3. Activer l'option "Débogage USB" sur le mobile Android

4. Connectez l'appareil Android à Andropod

Maintenant la deuxième LED sur l'interface AndroPOD devrait commencer à clignoter, et le téléphone devrait nous dire que le débogage USB est connecté.

5. Démarrez l'application sur le téléphone Android

Maintenant la deuxième LED arrête de clignoter et rester allumée.

Après ces 5 étapes, nous avons réussi de connecter l'interface AndroPOD.

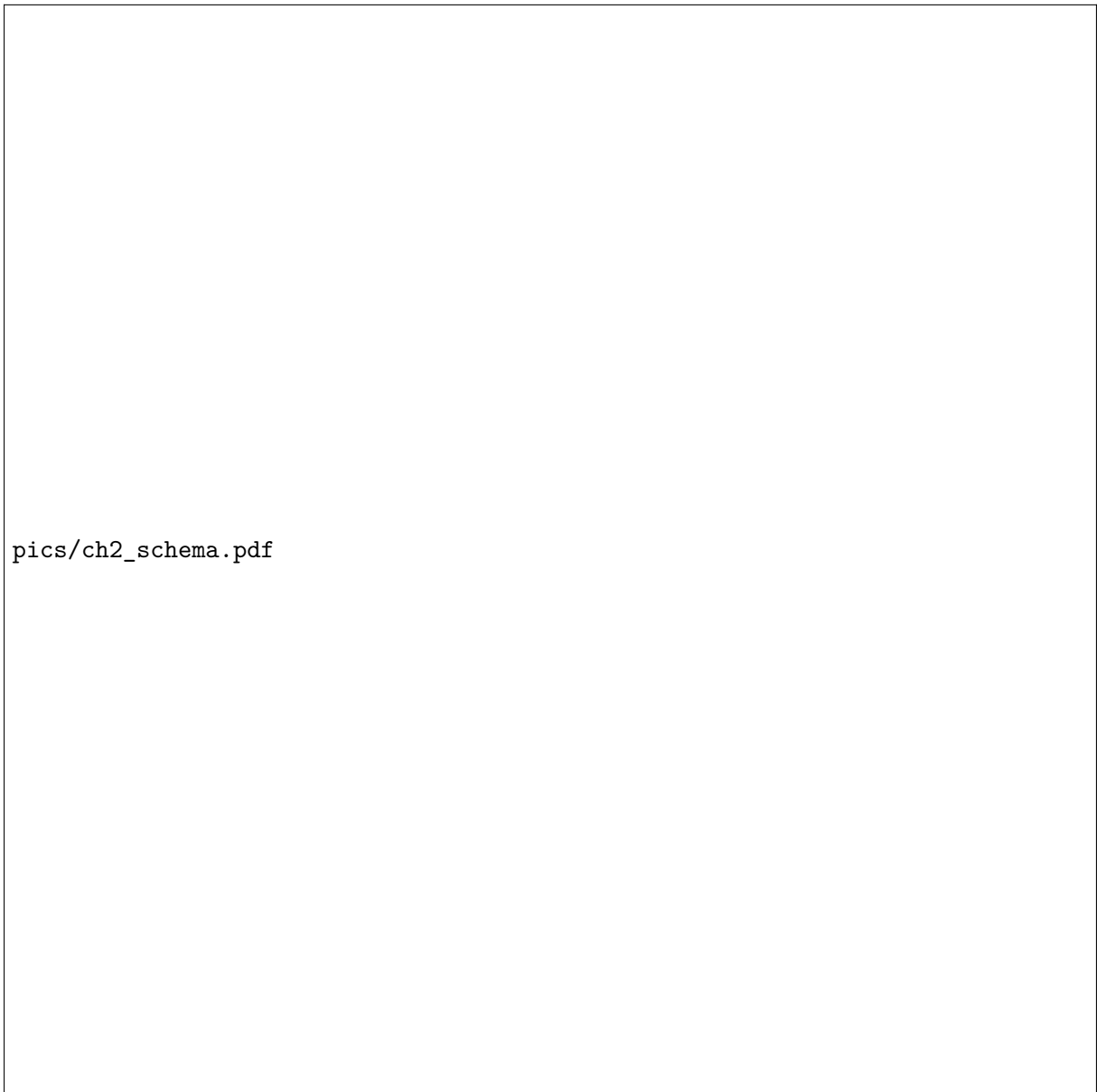


FIGURE 3.3 – Schéma de la carte AndroPOD

OpenGL

4.1 Introduction

Nous choisissons le robot Lynxmotion comme le matériel à contrôler par AndroPOD, parce qu'il n'existe actuellement pas beaucoup d'équipement qui se munit encore une interface série dont AndroPOD a besoin pour le contrôler. D'ailleurs, contrôler un petit robot avec un mobile est une façon assez directe pour montrer la force de la carte AndroPOD.



FIGURE 4.1 – Robot Lynxmotion

Le robot est un bras robotisé dont les documentations sont disponibles sur le site de Lynxmotion ¹. Il a été utilisé avant dans un notre projet, alors tout d'abord nous essayons de vérifier son état de fonctionne-

1. <http://www.lynxmotion.com/driver.aspx?Topic=assem01#l6>

ment. Nous avons installé son logiciel de contrôle nommé *LynxTerm* qui a une interface comme figure 4.2. Pour faire fonctionner ce logiciel, il faut que l'on installe un pilote USB-to-Serial², pour avoir un port série virtuel.

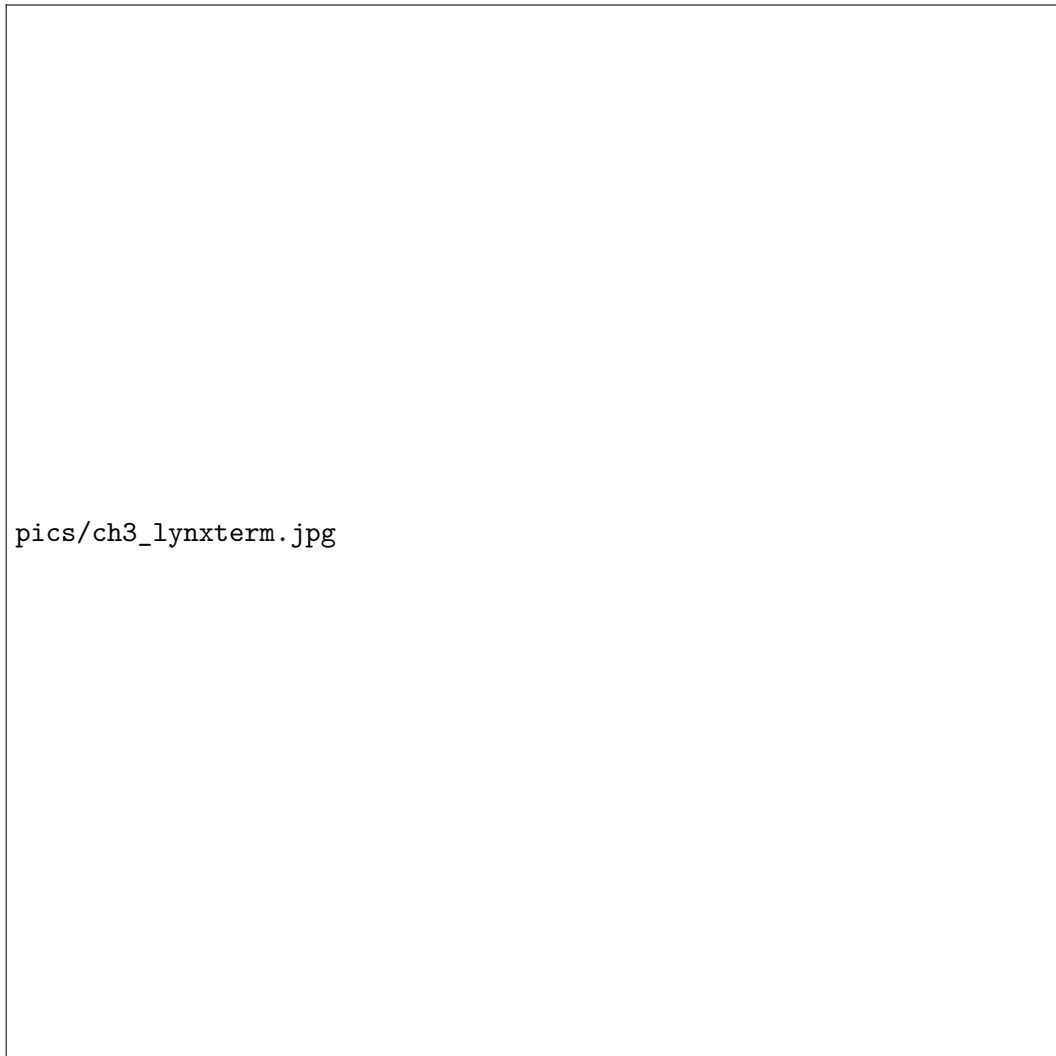


FIGURE 4.2 – Terminal de contrôle : LynxTerm

Avec ce logiciel de contrôle, nous avons trouvé qu'il y a 5 servomoteurs au total dont 2 ne fonctionnent pas, mais nous ne les avons pas puisque ça ne nous empêche pas de montrer la fonctionnalité de l'AndroPOD. Alors maintenant, ce qui nous intéresse, c'est le format de commandes pour contrôler ce robot, c'est-à-dire les informations transmises entre LynxTerm et le robot.

4.2 Format des commandes

Pour trouver la structure des commandes, nous avons étudié le manuel de référence de la carte SSC-32³ qui est intégré avec le robot, et qui reçoit et traite les commandes. Finalement, nous constatons que le format de commandes est de forme "`# <ch> P <pw> S <spd> T<time> <cr>`", dont les explications sont :

2. <http://www.prolific.com.tw/eng/downloads.asp?id=31>

3. Il est attaché à la fin du rapport. On peut le trouver aussi ici : <http://www.lynxmotion.com/images/data/ssc-32.pdf>

- `<ch>` : Le numéro de canal, forme décimal, 0-31. Il correspond au servomoteur qu'on veut contrôler ;
- `<pw>` : La largeur de pulsation en microseconde, 500-2500 ;
- `<spd>` (facultatif) : La vitesse de mouvement en microseconde ;
- `<time>` (facultatif) : La durée en milliseconde désignée pour un mouvement complet, valide pour tous les canaux ;
- `<cr>` : Retour de chariot, ASCII 13, obligatoire pour finir une commande.

Dans les commandes au-dessus, il y a des paramètres en microseconde qui concernent à la conception au moteur servo. Puisque c'est un peu loin de notre but du projet, on ne va pas étudier profondément sur ce point, par contre il suffit de savoir la position en angle correspondante comme l'indiqué à la figure 4.3.



FIGURE 4.3 – Mouvement des servomoteurs du robot

Maintenant on connaît le format de commandes. Par exemple, la commande `#5 P1500 S100 <cr>` déplace le moteur 5 à la position centrale (2^e rangée dans la figure 4.3), ayant une vitesse de $100\mu\text{s}$ par seconde. Un mouvement de $1000\mu\text{s}$ correspond au 90° , donc " $100\mu\text{s}$ par seconde" signifie que ça coûte 10 secondes à tourner 90° .

pics/ch3_schema_ssc32.pdf

FIGURE 4.4 – Schéma de la carte SSC32 intégrée avec le robot

Mise en oeuvre

Sur le site ¹, il y a des tutoriaux assez détaillés pour commencer à programmer pour AndroPOD. Ils offrent même un petit projet d'exemple qui a une fonctionnalité élémentaire pour communiquer entre l'application et AndroPOD par la connexion TCP. Ayant auparavant des connaissances sur Java et la programmation des réseaux, nous avons étudié ce petit programme qui a facilité notre travail.

Notre application possède une interface ci-dessous 5.1 :



FIGURE 5.1 – L'interface de l'application

A la fin, nous avons relié toutes les trois parties — mobile, AndroPOD et ordinateur, et nous avons réussi de contrôler le robot avec notre mobile.

1. <http://www.xdevelop.at>



FIGURE 5.2 – L'oeuvre finale

Conclusion

Notre projet contient trois parties principales : le système Android, la carte AndroPOD et le petit robot. Nous avons fait beaucoup de recherche et d'étude pour connaître chacune partie, et à la fin de notre projet, nous avons réussi à contrôler le robot avec notre mobile Android, ayant AndroPOD comme le pont entre eux, et une application installée sur le mobile.

La programmation sous Android est en vogue pour le moment. Ce projet nous permet pas seulement d'avoir une connaissance fondamentale du développement sous Android, mais encore d'avoir une impression sur le contrôle de matériel. C'est aussi intéressant de faire un lien entre notre mobile Android et un petit robot. A la fin, nous voudrions remercier notre encadrant M. Pascal Makris, qui nous a accompagné tout au long du développement du projet, en nous guidant par les réunions fréquentes et en nous offrant les matériels requis.

Simulation du trafic urbain

Département Informatique
3^e année
2011 - 2012

Rapport de projet d'Algorithme et de Langage C

Résumé : Le *modèle de conducteur intelligent*(IDM) est un modèle qui représente l'action des conducteurs, c'est-à-dire le mouvement raisonnable des voitures dans un trafic urbain. Nous avons établi l'algorithme à partir de ce modèle pour simuler visuellement sur l'écran, une file de voiture sur la route avec 2 feux de circulation. L'application que nous avons réalisé peut être fortement configurable à l'aide du fichier de configuration.

Mots clefs : IDM, Simulation du trafic, Algorithme, OpenGL

Abstract: The *Intelligent driver model*(IDM) is a time-continuous car-following model for the simulation of freeway and urban traffic. According to this model, we have proposed an algorithm and then an application to simulate traffic on the screen. This application is developed with C programming language and can be highly customized with the help of the configuration file.

Keywords: IDM, Traffic simulation, Algorithm, OpenGL

Encadrants

Néron EMMANUEL
emmanuel.neron@univ-tours.fr

Université François-Rabelais, Tours

Étudiants

Lei SHANG
lei.shang@etu.univ-tours.fr

Yanxiao HU
yanxiao.hu@etu.univ-tours.fr

DI3 2011 - 2012