A51 - 326 – Robotique

Librairie pour les robots Parrot

Section EMF Informatique

Joseph Gremaud

Projet du 15.12.2015 au 22.01.2016

Table des matières

[1 Introduction 1](#_Toc450034625)

[2 Analyse 1](#_Toc450034626)

[2.1 Comprendre comment se connecter à un drone Parrot WIFI 1](#_Toc450034627)

[2.1.1 Connexion TCP/IP plus en détail (Wireshark) 1](#_Toc450034628)

[2.2 Connexion UDP et commandes 3](#_Toc450034629)

[2.2.1 Comprendre les commandes 3](#_Toc450034630)

[2.2.2 Détail sur la construction d’une commande 4](#_Toc450034631)

[2.2.3 Ping Pong UDP 4](#_Toc450034632)

[2.3 Structure de la librairie 5](#_Toc450034633)

[2.4 Gestion des commandes sur les autres drones 5](#_Toc450034634)

[3 Utilisation 6](#_Toc450034635)

[3.1 Tester le robot 6](#_Toc450034636)

[3.2 Coder pour utiliser le robot 6](#_Toc450034637)

[4 Bibliographie 6](#_Toc450034638)

[5 Autoévaluation 7](#_Toc450034639)

[6 Conclusion 7](#_Toc450034640)

# Introduction

Le but de mon projet est de standardiser, les commandes qu’on peut envoyer aux robots de Parrot, en effet tous les robots on des commandes communes et des similitudes pour connecter une application aux différents robots. Donc je vais accentuer mes efforts à faire un standard sous la forme d’une librairie en JAVA.

# Analyse

Pour pouvoir me connecter au robot et envoyer des commandes à celui-ci voici ma procédure d’analyse.

## Comprendre comment se connecter à un drone Parrot WIFI

Après quelque recherche infructueuse sur internet, j’ai téléchargé l’application FreeFlight 3 sur le Windows Store, ensuite j’ai lancé Wireshark et sniffer la carte Wifi de mon portable (Windows 10).

Ensuite j’ai démarré le Jumping Sumo, et connecter mon portable à celui-ci.

Mon constat après avoir lancé des commandes depuis l’application officielle :

1. La connexion s’effectue en TCP/IP
2. Les commandes sont envoyées en UDP
3. Apparemment il serait possible de téléchargé des vidéos avec FTP sur les drones.

### Connexion TCP/IP plus en détail (Wireshark)

Lorsque je me suis connecté, j’ai pu constater que l’application envoyait un JSON(String) au drone qui lui répond aussi par un JSON(String).

Pour voir ces trames TCP/IP au format texte, j’ai simplement analysé le flux TCP/IP sur Wireshark.

Ce que FreeFlight 3 envoie :

{ "d2c port":43210, "controller\_type":"Phone", "controller\_name":"com.example.arsdkapp" }

Explication :

|  |  |  |
| --- | --- | --- |
| Clé | Obligatoire | Description |
| d2c\_port | Oui | Le port UDP qui sera utilisé pour lire les données. |
| controller\_type | Oui | Le type d’appareil (Téléphone, tablette, ordinateur, …) |
| controller\_name | Oui | Le nom de l’appareil. |
| Device\_id | Non | Numéro de série du drone. |

Ce que le drone répond :

{ "status":0, "c2d\_port":54321, "arstream\_fragment\_size":65000, "arstream\_fragment\_maximum\_number":4, "arstream\_max\_ack\_interval":-1, "c2d\_update\_port":51, "c2d\_user\_port":61 }

|  |  |  |
| --- | --- | --- |
| Clé | Obligatoire | Description |
| status | Oui | Si différent de 0, la connexion est refusée |
| c2d\_port | Oui | Le port UDP qui sera utilisé pour envoyer des données |
| arstream\_fragment\_size | Non | La taille des frames envoyés par le drone |
| arstream\_fragment\_maximum\_number | Non | Le nombre maximum de fragments envoyé par frame vidéo |
| arstream\_max\_ack\_interval | Non | Le temps maximum entre les ACKs des paquets |
| c2d\_update\_port | Oui | Le port FTP pour mettre à jour le drone |
| c2d\_user\_port | Non | Un autre port FTP pour d’autre usage. |
| skycontroller\_version | SC Oui | Seulement dans le cas d’une utilisation d’un Skycontroller |

On appelle cette procédure de connexion « Handshake ». Le Handshake nous donne toutes les informations nécessaires à la connexion UDP.

## Connexion UDP et commandes

Contrairement à la connexion TCP/IP, l’échange d’information en UDP ce fait au moyen de tableau de bytes.

### Comprendre les commandes

Le schéma suivant explique comment est construit le header d’un paquet UDP. Suivi de la commande.



|  |  |  |
| --- | --- | --- |
| Type de données | |  |
| 1 ACK | Paquet sans données qui souhaite un acquittement | |
| 2 DATA | Paquet avec données sans acquittement | |
| 3 DATA LOW LATENCY | Paquet avec données pour la vidéo | |
| 4 DATA WITH ACK | Paquet avec données qui retourne un acquittement même si la séquence n’est pas acceptée. | |

Le buffer ID est un nombre qui doit être entre 2 et 127 compris. Il a peu d’importance apparemment.

Le numéro d’ACK, est le nombre d’acquittement qui permet de garder une trace chronologique des pour les paquets UDP envoyé et reçu.

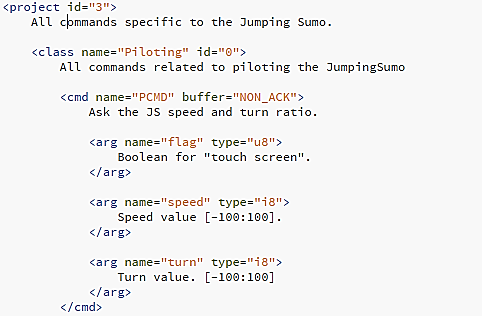
La taille totale est égale au nombre de byte du tableau.

### Détail sur la construction d’une commande

Chaque drone possède un fichier XML qui permet de créer les commandes.



La commande en dessus se réfère à la liste du SUMO.



Seul point ennuyeux, il faut compter à partir de 0 dans une classe pour savoir l’ID d’une commande.

### Ping Pong UDP

Pour éviter le timeout du robot, il faut constamment envoyer des paquets pour dire que le controller est toujours présent, mon choix envoyer un Pong tous les 4 Ping.

Un Ping ses reconnait par une construction de ce type :

1. Premier byte de type DATA
2. Second byte = 0 pour un Ping et 1 pour un Pong

## Structure de la librairie

Cette structure est hypothétique et peut changer.



Explication :

La classe « DeviceController » est mère des classes de drones. Ici j’ai pris la classe Sumo comme exemple pour la classe fille.

Les classes suffixées de « Command » sont les classes qui contiennent les commandes.

La classe Handshaker permet à l’utilisateur de se connecter au Sumo.

La classe ControllerUDP permet à l’utilisateur d’envoyer les commandes et recevoir du drone.

## Gestion des commandes sur les autres drones

Tous les drones Parrot ont une liste de commande dans un fichier XML de projet, les commandes sont toutes construites de la même manière dans ces fichiers, voici un lien vers cette liste : <https://github.com/Parrot-Developers/libARCommands/tree/master/Xml>

Les choses importantes à savoir pour construire les commandes :

1. 1 byte pour le numéro du projet
2. 1 byte pour la classe de la commande
3. 2 bytes le premier pour la commande le second a 0
4. 1 byte par argument
5. Si les arguments comportent des « enums », il faut compter les « enums » et mettre autant de bytes à 0 à la fin du tableau.

Pour simplifier ce travail fastidieux, j’ai écrit un fichier XSL qui interprète les fichiers XML de commandes en lambda pour JAVA, pour rendre l’utilisation plus simple.

Le fichier : <https://www.dropbox.com/s/dyskt9fq6mtdy46/sumoCommand.xsl?dl=0>

Attention il faut ouvrir le fichier et changer « CommandCommand » au même nom que la classe de commande java !

# Utilisation

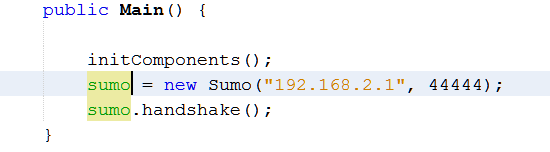
## Tester le robot

Pour tester le robot, il faut se connecter en Wifi au robot et ensuite simplement lancer la librairie.

## Coder pour utiliser le robot

Pour commencer il faut lier la librairie au projet, ensuite pour créer un sumo il faut :

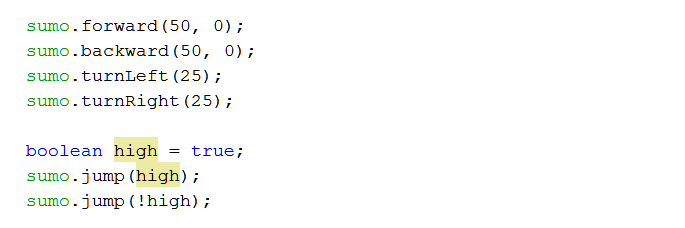
1. Créer un sumo
2. Faire un handshake pour négocier les ports UDP.



Ensuite il est de bon usage d’envoyer cette commande au Sumo avant les autres :



Finalement on peut faire bouger le robot avec les méthodes suivantes :



# Bibliographie

|  |  |
| --- | --- |
| Lien | Utilité |
| <http://developer.parrot.com//docs/bebop/ARSDK_Protocols.pdf> | Base pour comprendre le fonctionnement de la connexion au drone. |
| <https://github.com/Zepheus/ardrone3-pcap> | Source pour les explications de construction d’une commande en byte |
| <http://forum.developer.parrot.com/t/jumping-sumo-udp-packets/301> | Décomposition d’un paquet Parrot pour un drone. |

# Autoévaluation

Je pense qu’avec tous les problèmes que j’ai rencontrés pour faire fonctionner la librairie, déjà rien qu’au niveau de l’analyse de fonctionnement, de construction d’une commande était très compliqué. La documentation fournie par Parrot est très petite et imprécise. Mais je pense que l’autonomie que j’ai démontrée lors du module mérite d’être récompensé.

# Conclusion

J’ai beaucoup apprécié pouvoir faire un travail comme celui-ci, c’était un défi à relever et je pense avoir beaucoup acquis au cours du module.