Projet Dédale 2015-2016 Documentation

Version 0.0.1a

Team Dédale 2015-2016

mai 31, 2016

Table des matières

Cette documentation a pour but de rendre compte des connaissances acquises lors de l'année 2015-2016 autour du projet Dédale, et de fournir un condensé des informations et ressources utiles à la continuation du projet.

Table des matières :

CHAPITRE 1

Ardrone 2.0

1.1 Description générale

Le projet utilise un drone construit par Parrot. Cette année nous avons travaillé avec l'Ardrone 2.0. Il embarque 2 caméra (devant et dessous).

1.2 Mise en place de la connexion

Ce modèle de drone créer un réseau wifi ouvert de type ad-hoc. La première personne connectée dessus peut contrôler le drone. Le réseau est accéssible très facilement puisque ouvert.

1.3 Commandes AT

Le drone est piloté par des commandes AT. Un descriptif de ces commandes à été réalisé et est disponible dans le dossier `ardrone` sur le dépot git Ici

1.4 Informations utiles

Une seule des deux caméra peut fonctionner à la fois. Pour couper les moteurs du drone en urgence, il suffit de le mettre la tête en bas.

1.5 Liens externes

Site Web d'un ancien ayant travaillé sur le drone : Upsilon Audio Plusieurs programmes et tuto sur le drone y sont disponibles.

CHAPITRE 2

Raspberry Pi

2.1 Utilité

La raspberry est l'ordinateur embarqué sur le drone. Elle s'y connecte et intéragit avec permettant le pilotage du drone par un programme de façon autonome (sans station de base au sol). Elle est aussi utiliser pour étendre la portée de connexion au drone. En effet, la raspberry pi est connectée sur le wifi du drone et sur le wifi du campus grâce à ses deux interfaces, et donc accessible par un opérateur depuis le wifi du campus. Concrètement, il est possible d'agir sur le drone dès lors que l'on est connecté au wifi du campus par le biais de la raspberry pi.

2.2 Installation

Le système de la raspberry est contenu sur une carte SD. Cette dernière est branchée dans le slot prévu à cet effet sur la rapsberry pi. Les principaux systèmes disponibles sont : `Raspbian` ; `ArchLinux-ARM` ; `Ubuntu-ARM` (uniquement raspberry 2 et +). Le framework ROS utilisé pour le projet est disponible pour les trois précédents systèmes, mais les paquets binaires uniquement sous `Ubuntu-ARM`. Ainsi, dans le cas de `Raspbian` et `ArchLinux-ARM`, le framework ROS et tous les paquets associés doivent être compilés depuis les sources (et ce n'est pas une bonne idée, les sources de beaucoup de paquets ROS ne sont plus maintenues après la release du paquet binaire sur Ubuntu). En bref, le seule système officiellement supportée par ROS est Ubuntu, vouloir en utiliser un autre n'implique qu'une perte de temps. (Il est même plus rapide de mettre en place une machine virtuelle sous Ubuntu et d'y installer ROS, que de vouloir l'installer sur un autre système où il n'est pas nativement supporté). Pour en revenir à l'installation d'un système sur raspberry pi, une fois en possession d'une carte SD et d'une image système pour le modèle de raspberry pi, il suffit de faire un :

```
$ dd if="/chemin/vers/fichier/image" of="/dev/sdX"
```

Avec `/dev/sdX` le chemin vers la carte SD. Pour obtenir ce chemin, la commande `lsblk` est utile. Pour plus d'informations, voir les pages `man` des différentes commandes utilisées, et la documentation officielle des raspberry pi : Installer un système sur sa raspberry pi

2.3 Usage

Pour accéder à la raspberry pi depuis un autre ordinateur, l'interface la plus utilisé est le `ssh`. Elle est accessible dès lors que la raspberry et l'ordinateur sont en réseau, et ouvre un terminal distant sur la raspberry depuis l'ordinateur, permettant pleinement de s'en servir. Il suffit, depuis un terminal, de lancer la commande :

```
$ ssh [user]@[hostnameORIPadress]
```

Dans le cas de la raspberry pi, l'utilisateur par défaut est `pi`, le nom d'hôte `raspberrypi` et l'addresse IP varie selon le sous réseau dans lequel les deux appareils sont. Le mot de passe par défaut pour l'utilisateur `pi` est `raspberry`. Un exemple concret serait (si la raspberry a 192.168.1.2 comme addresse IP) :

```
$ ssh pi@192.168.1.2
```

Pour plus d'informations, `man ssh`.

2.4 Configuration

Un certain nombre de configuration ont été faites sur la raspberry pi afin de faciliter son utilisation.

2.4.1 Démarrage

L'utilisation de scripts de démarrage a été très utile dans notre projet, notamment un script permettant au démarrage du système, d'envoyer un email contenant l'adresse IP de la raspberry pi (l'adresse pouvant changer à chaque redémarrages). Il est possible de lancer un script au démarrage via la crontab. Elle est éditable en utilisant la commande :

```
$ crontab -e
```

et voici un exemple d'entrée dans la crontab :

```
@reboot /home/pi/Dedale201516_RPI_Setup/startup.sh
```

qui lance le script `startup.sh` ayant pour code :

```
#!/bin/sh
wget --spider www.google.fr
while [ "$?" != 0 ] ; do
        sleep 2
        wget --spider www.google.fr
done
while [ `cat /sys/class/net/wlan1/operstate` != "up" ] ; do
        sleep 2
done
ip addr show wlan1 | mail -s "RaspberryIP" votreadresse@mail.com
```

Ce script essaye simplement de se connneter à google, et dès qu'il réussie, vérifie que l'interface wifi est bien disponible, puis envoie par email sa configuration wifi vers l'adresse `votreadresse@mail.com`

2.4.2 Réseau

La configuration réseau est certainement la plus importante, elle permet à la raspberry de communiquer avec le drone et de se connecter au wifi du campus. Elle se fait via le fichier `/etc/network/interfaces`. Nous utilisions ce fichier :

```
# Please note that this file is written to be used with dhcpcd.
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'.

auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
```

```
    iface eth0 inet static
        address 169.254.0.2
        netmask 255.255.255.0
        network 169.254.0.0
        broadcast 169.254.0.255

    auto wlan0
    allow-hotplug wlan0
    #iface wlan0 inet dhcp
    #wireless-essid ardrone2_090332

    auto wlan1
    allow-hotplug wlan1
    iface wlan1 inet dhcp
        pre-up wpa_supplicant -B -Dwext -i wlan1 -c /etc/wpa_supplicant/wpa_supplicant.conf
        post-down killall -q wpa_supplicant
```

L'interface eth0 est configurée pour une connexion ethernet statique, le wlan0 pour se connecter au drone, et le wlan1 pour se connecter au wifi du campus grâce au fichier de configuration `wpa_supplicant` que voici :

A noter qu'à des fins de test, il est aussi possible de se connecter en réseau avec la raspberry pi en liaison ethernet directe sans même avoir besoin de changer les fichiers de configuration présentés ci-dessus. C'est très utile lors de la première configuration notamment. Pour celà, une fois une image système copiée sur la carte SD, il suffit de monter la partition de la carte SD de 50mo ayant un système de fichier FAT/FAT32 (automatique sous Windows), puis d'éditer le fichier `cmdline.txt` et d'ajouter à la fin de l'unique ligne du fichier : `ip=169.254.0.2`. Ensuite, lors du prochain démarrage, la raspberry se configurera en adressage statique sur son interface ethernet, il suffit alors avec son ordinateur de se mettre en adressage statique avec l'adresse `169.254.0.3` par exemple. Il est ensuite possible de se connecter à la raspberry en ssh via

```
    $ ssh pi@169.254.0.2
```

2.4.3 Mails

Dans la section démarrage, il était question d'envoyer un mail au démarrage de la raspberry pi contenant son adresse IP. Les scripts disponible précédemment utilise la commande `mail`. On détail ici sa configuration. Tout d'abord, il faut installer les paquets nécessaires :

```
    $ sudo apt-get install mailutils mpack
    $ sudo apt-get install ssmtp
```

Puis d'ajouter la configuration de `ssmtp` dans le fichier `/etc/ssmtp/ssmtp.conf`. Voici le fichier utilisé cette année :

```
    root=postmaster
    mailhub=mail
    hostname=raspberrypi

    root=ddedale2016@gmail.com
    mailhub=smtp.gmail.com:587
    hostname=srvweb
    AuthUser=ddedale2016@gmail.com
    AuthPass=*******
    FromLineOverride=YES
    UseSTARTTLS=YES
```

Il est ensuite possible d'envoyer un email de la façon suivante :

```
$ echo "Contenu du mail" | mail -s "Titre du mail" destinataire@gmail.com
```

2.5 Liens externes

Une majorité des scripts inscrits dans cette page est disponible dans le dossier `raspberrypi` du dépot git : Ici

CHAPITRE 3

ROS

3.1 Introduction

ROS est un framework dédié à la robotique. Le choix d'utiliser ROS pour notre projet vient du fait qu'ayant du repartir de 0 au mois de Mars, il nous fallait gagner du temps sur la réalisation de notre solution. L'intéret réside dans le fait que pour ce framework existe un nombre important de paquets, notamment un driver pour l'ardrone et un paquet de reconnaissance de tags. Nous n'avions donc plus à coder l'interface avec le matériel, et la reconnaissance de formes, mais à les faire intéragir entre eux.

3.2 Installation

3.2.1 Installer ROS

ROS peut être compliqué à installer à première vue, notamment lorsqu'on se lance dans son installation depuis les sources. La méthode la plus simple et la plus sûr (perdre du temps sur cette étape n'est vraiment pas intéressant) consiste à l'installer depuis des paquets binaires disponible dans les dépots des distributions officiellement supportées par ROS. A ce jour, uniquement Ubuntu et ses variantes. D'où l'utilité d'avoir une raspberry 2 ou plus pour pouvoir y installer le système `Ubuntu-ARM`. ROS est disponible sous plusieurs versions. A ce jour, `ROS indigo` est la version avec le plus de compatibilité, et est disponible sur `Ubuntu 14.04 (Desktop)` au maximum. Lors du choix de la version de ROS, il faut faire attention à ce que tout les paquets ROS que l'on projette d'utiliser soit nativement compatibles avec cette versions (pour encore une fois éviter les installations depuis les sources).

Plus de détails sur l'installation de ROS sur la documentation officielle ici

3.2.2 Installer un paquet ROS depuis les dépots Ubuntu

Pour tous les paquets, la procédure d'installation est indiquée dans la documentation. Lorsque le paquet est disponible sur les dépots Ubuntu, il suffit pour l'installer de lancer :

```
$ sudo apt-get install ros-indigo-ardrone-autonomy
```

3.2.3 Créer un espace de travail

Un espace de travail ROS est nécéssaire au développement de paquets. Pour initialiser un espace de travail, la démarche est disponible sur la documentation officielle ici

3.2.4 Installer un paquet depuis les sources

Pour installer un paquet depuis les sources, il faut charger les sources dans le répertoire `catkin_ws/src/` puis la depuis le répertoire `catkin_ws` lancer les commande

```
$ catkin_make
$ source ~/catkin_ws/devel/setup.bash
```

3.3 Comprendre ROS

ROS utilise une structure particulière qui lui est propre. Un programme est constitué de `noeuds` qui s'échangent des informations `messages` à travers des `topics`. Un programme peut être lancé via un fichier de lancement `launch` ou directement en lançant le noeud (archaïque). Différents outils en ligne de commande permettent de manipuler ROS, notamment `roslaunch` qui permet de lancer un fichier `launch`, et `rostopic` qui permet d'intéragir directement avec les topics ouvert par un noeud (leur envoyer des messages ou recevoir ce qu'ils émettent). Pour plus d'informations, utilisez la commande `man`

3.3.1 Lancer un fichier de démarrage

Une fois un paquet installé, les fichiers de démmarage sont disponible dans son répertoire racine sous `/launch`. Pour executer une fichier de démarrage :

```
$ roslaunch [PackageName] [LaunchFile]
```

Par exemple :

```
$ roslaunch ardrone_autonomy ardrone_driver.launch
```

3.4 Utilisation du paquet ROS 2015-2016

Le paquet ROS écrit en 2015-2016 peut être installé et utilisé suivant la procédure (nécessite ROS indigo d'installé et un espace de travail catkin sous `~/catkin_ws/`

Installer les dépendances

Installer le paquet

Lancer le programme

3.5 Développement

Pour le développement et l'utilisation générale de ROS, la documentation officielle est très bien faite : ici. Une grande aide pour le développement est la consultation d'exemples dans les sources d'autres paquets.

- falkor_ardrone
- tum_ardrone
- arl_ardrone_examples
- AR Drone Tutorials
- tum_simulator

Enfin, la documentation du paquet `ardrone_autonomy` est très complète et disponible ici

3.6 Liens externes

Installer ROS sur Ubuntu : ici Initialiser un espace de travail : ici Tuto ROS : ici

CHAPITRE 4

Outils

4.1 Git

Les bases de git (utile pour le travail collaboratif) :

```
$ git add [Fichier|Dossier]
$ git commit -m [Message]
$ git push
```

La première commande sert à spécifier les fichiers ayant été modifiés, la deuxième à annoter leur modification, la troisième à envoyer les modifications sur les serveurs distants. Ces commandes doivent être effectués dans un dossier ayant été initialisé avec la commande `git clone`. Plus d'informations disponibles avec `man git`.

Liens externes

Dépots Git 2015-2016 Dépots Git 2014-2015 Upsilon Audio - blog d'un ancien regroupant les premiers travaux sur le drone.