

HERAIZ-BEKKIS Daphné & FARJAS Emilie

Robocologie

Manuel Utilisateur

Paris
9 Mai 2016

Table des matières

I- Spécifications matérielles et logicielles	
II- Etablissement des connexions	
III- Lancement du programme	
IV- Extinction d'un Robot	
V- Génération automatique de tags	
ANNEXE 1 : Installation d'OpenCV 2.4.9	
ANNEXE 2 : Tests simples du Thymio-2	
ANNEXE 3 : Arborescence du projet	
ANNEXE 4 - Kit de survie	

I- Spécifications matérielles et logicielles

1- Matériel

Le Robot est constitué de deux parties :

- Une Raspberry Pi
- Un Thymio 2

La Raspberry Pi est un nano-ordinateur à alimentation micro-USB, de dimension 65mm x 30 mm x 5 mm (taille d'une carte de crédit).

La Thymio 2 est un petit robot mobile pédagogique permettant l'apprentissage de la programmation graphique et/ou textuelle. Nous l'utilisons ici en exploitant ses capacités sans surcouches, par une programmation de haut niveau - Python. Thymio peut être programmé grâce à Aseba (ensemble d'outils permettant de programmer le robot).

Le choix de Thymio est arbitraire du fait de son coût, malgré le peu de documentation en ligne.

Ce système embarqué possède plusieurs ports USB sur lesquels nous pouvons connecter une Thymio-2, une carte wi-fi, un clavier et une souris. Nous pouvons également y connecter un moniteur en HDMI. La caméra est connectée à la Raspberry Pi via un slot dédié.

Nous utilisons la plateforme Raspbian (système d'exploitation libre de Debian optimisé pour fonctionner sur Raspberry).

2- Logiciels

Le langage utilisé est le plus adapté pour des échanges Thymio - Raspberry : Python 2.7 .

Pour gérer la caméra ainsi que le traitement des images, nous aurons besoins de librairies spécifiques devant être installées sur les Raspberry Pi.

Librairies graphiques

- OpenCV 2.4.9:

Nous utilisons la librairie de traitement d'images **OpenCV 2.4.9** en python pour exploiter les images capturées par la caméra de la Raspberry Pi, cette dernière ne supportant que les versions OpenCV inférieures à la version 2.4.9 .

Attention :

Nous avons observé de fortes variations dans la signature des certaines fonctions de base d'une version à l'autre d'OpenCv .

L'installation de cette librairie est très délicate et varie d'un système d'exploitation à l'autre. Sur un ordinateur, une installation Linux sur les systèmes d'exploitation Ubuntu, Debian ou encore Mint est préférable il est, par ailleurs, conseillé de tester plusieurs installations, en veillant à n'oublier aucune dépendance.

Installation :

*Les étapes de l'installation sont décrites dans l'**ANNEXE 1**.*

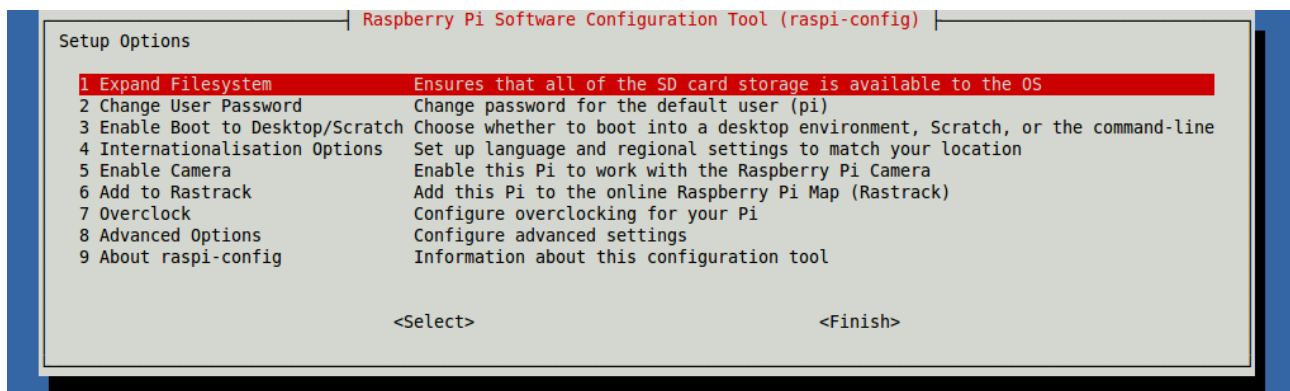
- **Librairie PiCamera :**

La caméra de la Raspberry Pi est capable de capturer des séquences d'images rapides.

La librairie PiCamera, en python, permet d'accéder aux fonctionnalités de gestion de la caméra.

Activation :

`$ sudo raspi-config` → *enable camera* → reboot



II- Etablissement des connexions

La méthode ci-décrite est spécifique à la configuration de l'ordinateur de l'équipe AMAC, ordinateur situé en 65-66 au 3ème étage. En effet, sur cet ordinateur, une configuration propre à la centralisation des commandes et de la gestion des Robots.

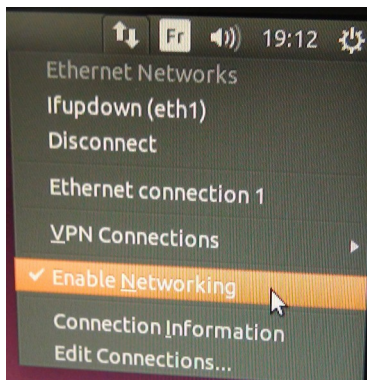
1- Connexion au routeur

Il est bien entendu nécessaire l'allumer l'ordinateur, ainsi que le routeur, émetteur de la connexion, et de connecter le câble ethernet du routeur à l'unité centrale.

Depuis l'ordinateur, la session par défaut de lancement est la session pi, de mot de passe pi.

a- Connecter le routeur

Une fois la session lancée, deux possibilités sont possibles :



- > à la souris : dans le coin supérieur droit de l'écran, cliquer sur l'icône du wifi → cocher [*Enable Network*](#)
- > Au clavier : ouvrir un terminal, effectuer la suite d'instructions suivantes :
 - [*\\$ sudo ifdown eth1*](#) (lance la connexion sur eth1)
 - [*\\$ sudo ifup eth1*](#) (stoppe la connexion sur eth1)
 - [*\\$ ifconfig*](#) (donne l'adresse IP de l'ordinateur)

A la suite de ces exécutions, les commandes de terminal [*ifconfig*](#) et [*hostname -I*](#) doivent afficher la même adresse IP réseau pour l'ordinateur.

b- Lancer le serveur DHCP

Redémarrer le serveur DHCP:

[*\\$ sudo /etc/init.d/isc-dhcp-server restart*](#)

Ainsi, dans le cas où celui-ci a été précédemment lancé, il est arrêté puis relancé.

```
pi@Ephemeride:~$ sudo /etc/init.d/isc-dhcp-server restart
* Stopping ISC DHCP server dhcpd [ OK ]
* Starting ISC DHCP server dhcpd [ OK ]
```

Sinon, il est tout simplement lancé pour la première fois à l'aide de la commande précédente.

Nous sommes à présent connectés au serveur dhcp.

2- Connexion à la Raspberry Pi

a- Allumage des composants du robots

Tout d'abord, pour une autonomie totale, la Raspberry Pi doit être connectée à une batterie externe, PowerBank, via son port d'alimentation micro-USB, pour être alimentée tout au long du lancement du programme.

De plus, le Thymio 2 doit être connecté à la Raspberry Pi, via un de ses ports USB, pour bénéficier lui aussi de cette source d'alimentation et être synchronisé.

Une fois connectée, cette batterie doit être allumée, à l'aide du bouton prévu à cet effet, pour mettre la Raspberry en route. Cette action lancera le démarrage automatique du Thymio 2 et permettra de l'alimenter durant la période où ce dernier sera allumé.



b- Connexion ordinateur - Raspberry Pi

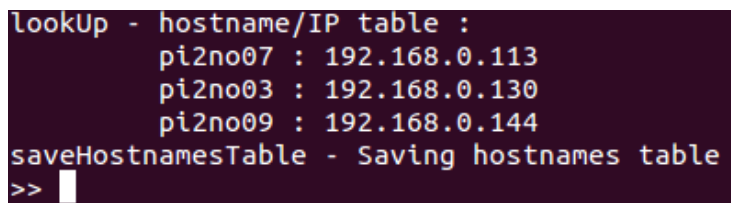
À partir de là, la connexion au robot par l'intermédiaire de la Raspberry peut être réalisée. Il faut tout d'abord récupérer l'adresse IP sur le réseau de la Raspberry en question.

Une adresse IP se présente sous la forme : X.X.X.X où X représente un nombre de un à trois chiffres allant chacun de 0 à 255.

Il est possible, et préférable, d'utiliser le script Python, écrit pour la gestion de l'application, en l'exécutant de la manière suivante :

```
$ python ThymioPYPI/OctoPY/OctoPY.py  
>> look -s
```

Le script OctoPY.py attend de l'utilisateur des commandes spécifiques. Ici, *look -s* permet de récupérer les adresses IP des Raspberry Pi, ainsi que leurs noms associés. Par exemple la Raspberry Pi numéro 3 aura comme nom pi2no03 .



```
lookUp - hostname/IP table :  
pi2no07 : 192.168.0.113  
pi2no03 : 192.168.0.130  
pi2no09 : 192.168.0.144  
saveHostnamesTable - Saving hostnames table  
>> █
```

La commande :

```
$ nmap -sn
```

permet d'obtenir les adresses IP sur le réseau mais nécessite l'installation de la commande nmap à l'aide de la ligne suivante :

```
$ sudo apt-get install nmap
```

Pour Ubuntu, Mint ou Debian

La commande suivante :

```
$ arp -a
```

permet elle aussi, le cas échéant, de récupérer les adresses IP présentes sur le même réseau que la machine initiatrice de la commande.

L'adresse IP récupérée, une connexion à distance peut être établie.

Pour se faire, une connexion SSH (Secure Shell, session à distance sécurisée) est réalisée, en tapant dans un terminal la ligne suivante :

`$ ssh -Y pi@adresseIP` ou `$ ssh -X pi@adresseIP`

où *adresseIP* est l'adresse IP de la Raspberry Pi à connecter et obtenue précédemment.

Par exemple :

```
pi@Ephemeride:~$ ssh -Y 192.168.0.113
```

Cela engendre la demande de mot de passe de la Raspberry Pi en question, qui ici est : **pi** .

Lorsque le mot de passe est entré, il n'est pas visible sur le terminal, il suffit cependant de le valider.

Attention : L'option **-Y** (ou **-X**) est très importante. Cette option permet d'inclure un module nécessaire à l'utilisation combinée du Thymio 2 et de la Raspberry Pi, qui sera détaillé dans le manuel technique.

Attention : Il faut toujours vérifier qu'il n'y a pas eu de problème à la connexion avec l'option **-Y** . C'est-à-dire que la troisième ligne de l'image suivante ne s'affiche pas :

```
pi@Ephemeride:~$ ssh -Y 192.168.0.130
pi@192.168.0.130's password:
X11 forwarding request failed on channel 0
Linux pi2no03 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7l
```

Suite à cette connexion réussie, le terminal utilisé affiche alors un prompt propre à la Raspberry PI à laquelle celui-ci est connecté.

```
pi@pi2no07 ~ $
```


3- Liaison Raspberry Pi -- Thymio-2

La Raspberry Pi et le Thymio 2 sont deux entités séparées. Il est donc nécessaire de les relier.

Cette liaison s'effectue grâce à un module du Thymio 2 à installer sur la Raspberry Pi de la façon suivante :

```
$ sudo apt-get install aseba
```

Cette installation ne doit être réalisée qu'une seule fois par Raspberry Pi.

A partir de là, il est possible de définir pour la Raspberry Pi le port utilisé pour brancher le Thymio 2. Cette étape nécessite de taper la commande suivante dans un terminal :

```
$ asebamedulla ser:device=/dev/ttyACM0 &
```

***Attention :** L'esperluette (&) en fin de commande est nécessaire à l'exécution à venir du programme. (cf **ANNEXE 4-Kit de Survie**)*

III- Lancement du programme

1- Valeurs paramétrées

De tous les fichiers de l'application, le fichier default_cst.py est celui contenant toutes les variables paramétrables par l'utilisateur.

a. Paramètres de la camera

Il lui est alors possible de modifier, parmi les valeurs par défaut de la caméra, les paramètres suivants:

- ❖ **Brightness (fr: luminosité) :** le système étant autonome, il gère lui-même au fur et à mesure l'adaptation à la lumière ambiante pour une meilleure lecture des images. Cependant, pour obtenir un bon taux de réussite dès le début, il est nécessaire de fournir au programme une luminosité initiale satisfaisante, d'où la variable INIT_BRIGHTNESS, initialisée par défaut à 60.
- ❖ **Resolution (fr: résolution) :** dans le cas de tests du code fourni en faisant varier la précision des images, la résolution peut être modifiée. Elle est initialement paramétrée à sa valeur minimale : 640 x 480, avec les variables SIZE_X (nombre de pixels par ligne pour une image prise) et SIZE_Y (nombre de pixels par colonne pour une image prise) .

b. Paramètres du tag

L'utilisateur peut aussi paramétrer les valeurs liées au tag, telles que :

- ❖ **Dimensions du tag** : le programme est, pour s'exécuter correctement, dépendant du design du tag. En effet, la hauteur, TAG_HEIGHT et la largeur, TAG_WIDTH correspondent aux deux dimensions, en centimètres, principales d'un tag en général. Beaucoup de calculs s'appuient sur ces valeurs.
- ❖ **Architecture interne au tag** : comme il sera précisé dans un chapitre ultérieur, le tag est aussi constitué d'une bordure, que nous appellerons *bordure anti-occlusion*, OCCLUSION_MARGIN. La dimension de cette bordure, en centimètres, est tout aussi importante pour le bon déroulement de l'application que les dimensions du tag.
- ❖ **Agencement des informations dans le tag** : il est nécessaire de fournir à l'application des détails concernant la disposition des informations que contient le tag. La zone d'identifiant d'un Robot doit être décrite au programme, ce dernier doit connaître le nombre de lignes NB_LIGN_CASE et de colonnes NB_COL_CASE utilisées par cette zone du tag.

c. Paramètres d'exécution

L'application possède une gestion automatisée de la luminosité. Cette gestion est cependant régie par quelques paramètres :

- ❖ **Variation** : l'évolution de la luminosité, selon ce qui a été vu par la caméra, se fait progressivement, par palier de BRIGHTNESS_STEP en BRIGHTNESS_STEP.
- ❖ **Comparaison** : l'ajustement de la luminosité s'effectue par comparaison d'images, par groupe de BRIGHTNESS_COMP images.
- ❖ **Seuil d'acceptation** : une marge d'erreur est définie pour la comparaison, au plus (BRIGHT_MIN_WRONG_IMG-1) images peuvent être hors champs de validité pour la gestion de la luminosité.
- ❖ **Ecart de comparaisons** : pour ne pas alourdir l'application, un test ponctuel est effectué toutes les BRIGHTNESS_CHECK images, vérifiant la validité des BRIGHTNESS_COMP dernières images.

Des paramètres généraux sont nécessaires pour réguler plusieurs parties du programme:

- ❖ **Régulateur de flux** : certaines parties optionnelles du programme, telles que la classification d'images ou les tests de luminosité ne nécessitent pas des séquences d'images trop rapprochées. Suivant cette idée, un temps d'attente, `WAITING_TIME`, en secondes, limite le flux d'images prises pour ces parties.
- ❖ **Régulateur en nombre** : pour les parties optionnelles, la quantité d'images prises est bornée par `ITERATIONS`.

D'autres paramètres permettent à l'utilisateur de choisir son mode d'interaction :

- ❖ **Visualisation** : il est possible de choisir de visualiser sur l'écran ce que le Robot perçoit tout au long de l'exécution du programme en plaçant la variable `DEMO` à `True`.

2- valeurs non paramétrées

- ❖ **Framerate (fr: fréquence d'images par seconde)** : par défaut, une Raspberry Pi prend, théoriquement, 35 images/sec (35 fps). Or si un traitement de l'image est réalisé juste après sa prise, l'image suivante ne sera capturée qu'après la fin de ce traitement. Le framerate est borné par le temps de traitement de chaque image. Il n'est donc pas nécessaire de lui préciser une valeur.

3- Transfert du programme à chaque Robot

Cela étant plus aisé et plus rapide de modifier le programme depuis l'ordinateur central, un transfert de ce même programme de l'ordinateur vers chaque Robot est alors nécessaire. Ce transfert peut être réalisé de deux manières différentes.

a. Par le réseau local

Pour tout partage par réseau local, les deux appareils doivent y être connectés (*cf. partie II.1 et 2*) .

Un transfert se fait alors en utilisant le terminal de l'appareil expéditeur, c'est un transfert par SSH, en y tapant la ligne suivante :

```
$ scp PC_ArborescenceDuProgramme/pandroide/src/*  
pi@AdresseIP_RASPBERRY:RASP_ArborescenceDuProgramme/pandroid  
e/src/
```

Où :

PC_ArborescenceDuProgramme correspond à l'emplacement du programme sur l'appareil expéditeur

RASP_ArborescenceDuProgramme correspond à l'emplacement voulu du programme sur l'appareil destinataire

AdresseIP_RASPBERRY correspond à l'adresse IP sur le réseau local, récupérée en partie II.2.b .

Ci-dessous un exemple de transfert de l'ensemble des sources du programme sur la Raspberry Pi n°7 en utilisant l'ordinateur et le routeur présents en 65-66 3ème étage de l'Université Pierre et Marie Curie.

```
$ scp ~/dev/thymioPYPI/pandroide/src/*  
pi@192.168.0.113:~/dev/thymioPYPI/pandroide/src/
```

A la suite de cela, une authentification à la Raspberry Pi est nécessaire : il faut entrer le mot de passe: **pi** .

Pour transférer tout le programme, il suffit de taper :

```
$ scp -r PC_ArborescenceDuProgramme/pandroide  
pi@AdresseIP_RASPBERRY:RASP_ARBORESCENCE_DU_PROGRAMME/  
pandroide
```

Cette dernière ligne de commande est plutôt longue à l'exécution, car elle copie aussi toutes les images sauvegardées et les fichiers de log.

b. Par Internet

Pour tout partage par Internet, les deux appareils doivent y être connectés. Le transfert se fait alors en utilisant le terminal de l'appareil destinataire. Les transferts réalisés avec Internet se font, ici, par l'intermédiaire de Github. Cette méthode n'est donc possible que pour les contributeurs Github de l'application.

Github (.git) est actuellement présent depuis le répertoire *~/dev/thymioPYPI/*
Et le programme est, lui, placé dans *~/dev/thymioPYPI/pandroide* .
(pour plus de détails, voir l' **ANNEXE 3: Arborescence de l'application**)

Depuis le destinataire :

Il faut donc se placer dans le dossier contenant le .git :

```
$ cd ~/dev/thymioPYPI/
```

Suite à cela, il faut effectuer l'instruction de mise à jour:

```
$ git pull
```

4- lancement des tests

L'application comme décrite précédemment peut dépendre d'une gestion du Thymio 2 ou simplement des traitements obtenus de la Raspberry Pi.

Tout lancement du programme se fait à partir d'un terminal, après avoir effectué les connexions et transfert du programme vers chacun des Robots.

Une ligne instruction préliminaire étant à taper dans le terminal :

```
$ cd ~/dev/thymioPYPI/pandroide/
```

a. Vérification de la vision Robot

Les Robots étant coiffés d'un chapeau haut de forme, avec une ouverture étroite au niveau de la caméra. Il est nécessaire de vérifier que ceux-ci ont bien été positionnés et n'obstruent pas la vision de la caméra.

Cette vérification peut être faite manuellement par l'utilisateur :

```
$ python src/realite.py
```

Il lui suffit alors de regarder la réalité du Robot et de réajuster le chapeau de ce dernier en conséquence. Pour quitter le flux d'images, il faut se positionner dans la fenêtre affichant les images et taper "q" au clavier.

b. Tests avec Thymio 2

Pour lancer un test simple avec gestion du Thymio 2, ce qui lui permettra alors de faire des sons selon ce qui passe dans son champs de vision et ce qu'il en a déduit. Une sauvegarde des images est effectuée la ligne attendue est la suivante:

```
$ python src/mainPANDROIDE.py
```

Pour un lancement avec affichage des images vues et traitées par le Robot, ainsi que ce que celui-ci y a vu, tels que les possibles tags qui n'en étaient pas ou simplement les tags reconnus:

```
$ python src/mainPANDROIDE.py -d
```

Cependant l'affichage de la vision du Robot se fait en différée, car les échanges de flux se font par le réseau wifi. Pour quitter le flux d'images, il faut se positionner dans la fenêtre affichant les images et taper "q" au clavier.

c. Tests de traitement

Pour lancer un test simple sans gestion du Thymio 2, traitant les images selon ce qui passe dans son champs de vision et ce qu'il en a déduit, la ligne attendue est la suivante:

```
$ python src/zytest.py
```

Pour un lancement avec affichage des images vues et traitées par le Robot, ainsi que ce que celui-ci y a vu, tels que les possibles tags qui n'en étaient pas ou simplement les tags reconnus:

```
$ python src/zytest.py -d
```

Cependant l'affichage de la vision du Robot se fait en différée, car les échanges de flux se font par le réseau wifi. Pour quitter le flux d'images, il faut se positionner dans la fenêtre affichant les images et taper "q" au clavier.

Pour par la même occasion, l'option -s est nécessaire :

```
$ python src/zytest.py -s
```

Dans ce cas-ci, les options sont cumulables :

```
$ python src/zytest.py -ds
```

D'autres tests de traitement des images ou de vérification d'optimalité du programme sont présents.

\$ python src/zytest.py -b

La ligne de commande ci-dessus permet de tester l'adaptation à la luminosité et donc la gestion de celle-ci pour optimiser l'initialisation par l'utilisateur des valeurs paramétrables.

Cette option est cumulable avec l'option de sauvegarde.

Pour tester la performance de reconnaissance du programme, il est nécessaire d'effectuer une comparaison des résultats obtenus par le programme par rapport à la réalité.

\$ python src/zytest.py -k

Pour ce faire, il faut tout d'abord capturer des images grâce à l'instruction précédente, puis les classer manuellement dans le fichier *classification.py*.

Il est alors possible, après tout cela, de tester le taux de réussite du programme à l'aide de la commande suivante :

\$ python src/zytest.py -l

5- Récupération du programme d'un Robot

Cela étant plus aisé et plus rapide de modifier le programme depuis l'ordinateur central, un transfert de ce même programme d'un Robot vers l'ordinateur est alors nécessaire.

Ce transfert peut être réalisé de deux manières différentes .

a. Par le réseau local

Pour tout partage par réseau local, les deux appareils doivent y être connectés (*cf. partie II.1 et 2*) . Un transfert se fait alors en utilisant le terminal de l'appareil expéditeur, c'est un transfert par SSH, en y tapant la ligne suivante :

```
$ scp  
pi@AdresseIP_Raspberry:RASP_ArborescenceDuProgramme/pandroide/src  
/ PC_ArborescenceDuProgramme/pandroide/src/*
```

Où :

PC_ArborescenceDuProgramme correspond à l'emplacement du programme sur l'appareil expéditeur

RASP_ArborescenceDuProgramme correspond à l'emplacement voulu du programme sur l'appareil destinataire

AdresseIP_RASPBERRY correspond à l'adresse IP sur le réseau local, récupérée en partie II.2.b .

Ci-dessous un exemple de transfert de l'ensemble des sources du programme sur la Raspberry Pi n°7 en utilisant l'ordinateur et le routeur présents en 65-66 3ème étage de l'Université Pierre et Marie Curie.

```
$ scp pi@192.168.0.113:~/dev/thymioPYPI/pandroide/src/  
~/dev/thymioPYPI/pandroide/src/*
```

À la suite de cela, une authentification à la Raspberry Pi est nécessaire, il faut entrer le mot de passe: **pi** .

Pour transférer tout le programme, il suffit de taper :

```
$ scp -r  
pi@AdresseIP_RASPBERRY:RASP_ArborescenceDuProgramme/pandroide  
PC_ArborescenceDuProgramme/pandroide
```

Cette dernière ligne de commande est plutôt longue à l'exécution, car elle copie aussi toutes les images sauvegardées et les fichiers de log.

b. Par Internet

Pour tout partage par Internet, les deux appareils doivent y être connectés.

Le transfert se fait alors en utilisant le terminal de l'appareil expéditeur et celui de l'appareil destinataire. Les transferts réalisés avec Internet se font, ici, par l'intermédiaire de Github. Cette méthode n'est donc possible que pour les contributeurs Github de l'application.

Github (.git) est actuellement présent depuis le répertoire *~/dev/thymioPYPI/*
Et le programme est, lui, placé dans *~/dev/thymioPYPI/pandroide* .
(pour plus de détails, voir l' **ANNEXE 3: Arborescence de l'application**)

Le robot expéditeur doit alors mettre à jour l'application contenue sur Github en exécutant les instructions suivantes :

```
$ cd ~/dev/thymioPYPI/
```

Suite à cela, il lui faut effectuer l'ensemble d'instructions ci-dessous:

```
$ git add pandroide
```

```
$ git commit -m "Application à jour"
```

```
$ git push
```

À partir de là, les identifiants et mot de passe github sont nécessaires.

Ensuite, le destinataire doit simplement mettre à jour l'application qu'il stocke:

```
$ cd ~/dev/thymioPYPI/
```

```
$ git pull
```

IV- Extinction d'un Robot

L'extinction d'un Robot peut être partiellement réalisée à distance.

En effet, la Raspberry Pi peut être éteinte à partir du terminal par lequel la connexion a été effectuée, en y saisissant la commande qui suit :

```
$ sudo shutdown -h now
```

Quant au Thymio 2, celui-ci ne peut être éteint que localement. Pour se faire, il suffit d'appuyer sur son bouton central (présent sur sa face supérieure) pendant 3 secondes, après avoir déconnecté son câble de la Raspberry Pi.

Suite à cela, le terminal utilisé pour la connexion à la Raspberry Pi affiche alors un prompt par défaut de l'ordinateur.

V- Génération automatique de tags

1. Définition d'un tag

Le tag transporte deux types d'informations :

- **l'identifiant du robot** (id : numéro de 0 à $2^{((NB_LIGN_CASE * NB_COL_CASE) - 1)}$)
- **la face du robot sur laquelle est le tag** (direction : avant, arrière, droite ou gauche)

De plus, ce tag permet de gérer les occlusions avec une bordure externe particulière.

Le tag possède donc trois zones principales :

- **un contour extérieur** avec un pattern particulier, utile pour reconnaître le tag, et gérer l'occlusion de celui-ci, c'est-à-dire qu'il ne peut être reconnu si il n'est pas entièrement visible.
- **une zone centrale haute** correspondant à l'identifiant du Robot
- **une zone centrale basse** correspondant à la face visible du Robot (celle sur laquelle le tag est lu).

Taille d'un tag de référence : 7.8 cm x 10.4 cm

2. Génération de tags

Une image ([pandroide/data/templates/tag_template.png](#)), dont l'id et la direction ne sont pas remplies, est utilisé comme modèle pour générer automatiquement les tags particuliers nécessaire à l'utilisateur par la commande suivante :

```
$python src/tag_generator.py --tag=X
```

ou

```
$python src/tag_generator.py -t X
```

Où X correspond au numéro du Robot dont le tag est à générer.

L'image du tag généré est alors sauvegardée dans :

[pandroide/data/tags/](#)

Sous le nom : *tag_X_Y.png*

Où X reste le numéro du Robot et Y les différentes directions de tag (avant, arrière, droite, gauche)

ANNEXE 1 : Installation d'OpenCV 2.4.9

Installation opencv 2.4.9

```
$ sudo apt-get install -y cmake libgtk2.0-dev pkg-config libavcodec-dev  
libavformat-dev libswscale-dev libgphoto2-dev libdc1394-22-dev gstreamer-  
tools unzip opencv-2.4.9.zip -d /usr/local; mkdir /usr/local/opencv-  
2.4.9/release
```

```
$ cd /usr/local/opencv-2.4.9/release
```

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE -D  
CMAKE_INSTALL_PREFIX=/usr/local .. make -j4 make install
```

OpenCV2.4.9 ne semble pas fonctionner avec Fedora

ANNEXE 2 : Tests simples du Thymio-2

Lancement des commandes :

```
$ cd
```

```
$ python dev/thymioPYPI/OctoPY/OctoPY.py
```

→ Lancement d'un programme qui écoute (attendre avant de lancer une autre commande que le prompt suivant s'affiche)

```
>>
```

```
>> send 0
```

→ Connexion automatique de l'ordinateur vers toutes les Raspberry Pi détectées

→ Lancement automatique de la commande *asebamedulla*

```
>> set braintenger.cfg
```

→ Commmande à effectuer pour lancer une simulation.

```
>> send 1
```

→ Lancement du programme de marche aléatoire du Thymio-2 .

```
>> send 4
```

→ Mise en pause du déplacement du Thymio-2

→ Relancement avec la commande *send 2* .

```
>> send 5
```

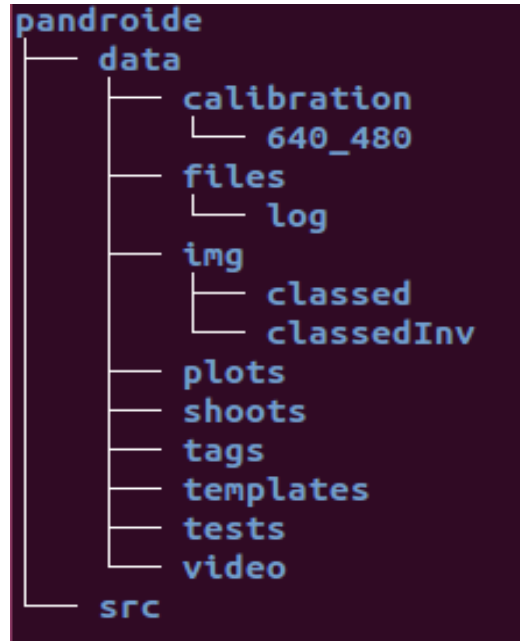
→ Arrêt total du programme

→ Relancement avec la commande *send 1* .

```
>> send 6
```

→ Extinction des Raspberry Pi auxquelles l'ordinateur s'est connecté avec la commande *send 0* .

ANNEXE 3 : Arborescence du projet



Où :

src contient les fichiers sources (.py)

data/calibration contient, pour chaque résolution, les images d'un tag reconnu à certaines distances

data/files/log contient les fichiers de log accessibles en temps réel lors d'une exécution

data/img/classed[Inv] contient les images classifiées dans le fichier *classification.py*

data/img contient les images prises lors des tests de luminosité

data/img/plots contient les graphes et courbes représentant les performances de l'algorithme

data/shoots contient toutes les prises de la dernière séquence

data/tags contient les tags générés à la demande de l'utilisateur par *tag_generator.py*

data/templates contient les modèles de tag pour en généré manuellement ou automatiquement

data/tests contient des images de test

data/video contient les vidéos des Robots en action

ANNEXE 4 - Kit de survie

Problèmes liés à Asebamedulla ou au DBUS :

ATTENTION !!! Lancement d'*asebamedulla* sur le même terminal que l'application.

ATTENTION !!! Un et un seul *asebamedulla* peut être lancé pour un même Robot. Sinon l'erreur suivante survient :

```
pi@pi2no07 ~/dev/thymioPYPI $ asebamedulla ser:device=/dev/ttyACM0 &
[1] 6364
pi@pi2no07 ~/dev/thymioPYPI $ terminate called after throwing an instance of
f 'Dashel::DashelException'
what(): Cannot bind socket to port, probably the port is already in use.

[1]+  Aborted                  asebamedulla ser:device=/dev/ttyACM0
pi@pi2no07 ~/dev/thymioPYPI $
```

ATTENTION !!! *asebamedulla* ne peut se lancer que si l'option **-Y** (ou **-X**) est ajoutée lors de la connexion.

```
pi@Ephemeride:~$ ssh -Y 192.168.0.130
```

ATTENTION !!! Même avec l'option **-Y** (ou **-X**) *asebamedulla* n'est pas forcément bien lancé. Il se peut donc que des bugs apparaissent au lancement de l'application.

```
pi@Ephemeride:~$ ssh -Y 192.168.0.130
pi@192.168.0.130's password:
X11 forwarding request failed on channel 0
Linux pi2no03 3.18.7-v7+ #755 SMP PREEMPT Thu Feb 12 17:20:48 GMT 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Aug 14 20:42:11 2015 from 192.168.0.210
pi@pi2no03 ~ $ cd dev/thymioPYPI/
```

Il faut alors réinitialiser la connexion.

```
pi@pi2no03 ~/dev/thymioPYPI $ logout
```

ATTENTION !!! Si en lançant le programme, l'erreur ci-dessous apparaît :

```
pi@pi2no07 ~/dev/thymioPYPI/pandroide $ python src/mainPANDROIDE.py -s
Traceback (most recent call last):
  File "src/mainPANDROIDE.py", line 2, in <module>
    from SimulationControlThymio import * # thread controlling the thymio
  File "/home/pi/dev/thymioPYPI/pandroide/src/SimulationControlThymio.py",
line 7, in <module>
    from thymio_tools import *
  File "/home/pi/dev/thymioPYPI/pandroide/src/thymio_tools.py", line 2, in
<module>
    from ThymioFunctions import *
  File "/home/pi/dev/thymioPYPI/pandroide/src/ThymioFunctions.py", line 228
, in <module>
    network = dbus.Interface(bus.get_object('ch.epfl.mobots.Aseba', '/'), d
bus_interface='ch.epfl.mobots.AsebaNetwork')
  File "/usr/lib/python2.7/dist-packages/dbus/bus.py", line 241, in get_obj
ect
    follow_name_owner_changes=follow_name_owner_changes)
  File "/usr/lib/python2.7/dist-packages/dbus/proxies.py", line 248, in __i
nit__
    self._named_service = conn.activate_name_owner(bus_name)
  File "/usr/lib/python2.7/dist-packages/dbus/bus.py", line 180, in activat
e_name_owner
    self.start_service_by_name(bus_name)
  File "/usr/lib/python2.7/dist-packages/dbus/bus.py", line 278, in start_s
ervice_by_name
    'su', (bus_name, flags)))
  File "/usr/lib/python2.7/dist-packages/dbus/connection.py", line 651, in
call_blocking
    message, timeout)
dbus.exceptions.DBusException: org.freedesktop.DBus.Error.ServiceUnknown: T
he name ch.epfl.mobots.Aseba was not provided by any .service files
```

Alors, il faut :

1- Trouver le processus d'*asebamedulla* toujours en cours d'exécution :

```
pi@pi2no07 ~/dev/thymioPYPI/pandroide $ ps aux
```

pi	6010	0.0	0.5	6864	5276	pts/1	Ss+	07:59	0:01	-bash
root	6137	0.0	0.0	0	0	?	S<	08:02	0:00	[kworker/1
pi	6320	0.0	0.9	40532	8324	pts/0	Sl	08:06	0:00	asebamedul
pi	6326	0.0	0.1	3380	1768	pts/0	S	08:06	0:00	dbus-launc
pi	6327	0.0	0.1	3184	1412	?	Ss	08:06	0:00	/usr/bin/d
root	7033	0.0	0.0	0	0	?	S	08:23	0:00	[kworker/2

2- Tuer le processus d'*asebamedulla*

```
pi@pi2no07 ~/dev/thymioPYPI/pandroide $ kill -9 6320
```

3- Relancer *asebamedulla* :

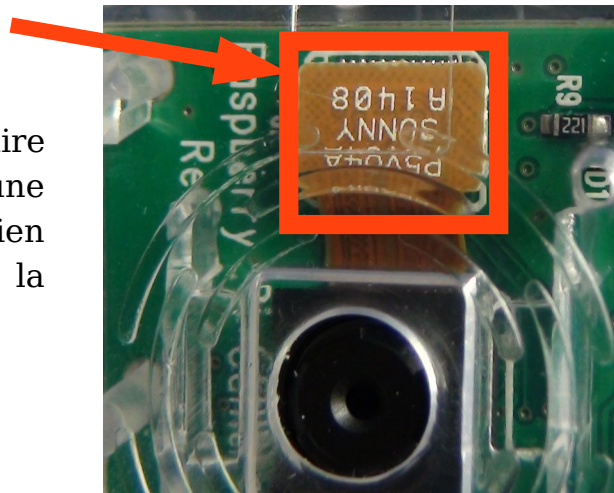
```
pi@pi2no07 ~/dev/thymioPYPI/pandroide $ asebamedulla ser:device=/dev/ttyACM0
0 &
[2] 8480
[1] Killed asebamedulla ser:device=/dev/ttyACM0 (wd: ~/dev/thymioPYPI)
```

4- Re-tester le programme

5- Dans le cas où l'erreur est récurrente, il faut alors redémarrer la Raspberry Pi

Problèmes liés à la caméra Raspberry Pi :

ATTENTION !!! Il faut aussi faire très attention à la petite puce jaune encadrée ci-contre qui doit être bien appuyée. Il faut alors redémarrer la Raspberry Pi.



ATTENTION !!! Il faut tout d'abord vérifier que la caméra a bien tous ses connecteurs branchés à la Raspberry Pi.

ATTENTION !!! Il faut vérifier que la caméra ne soit pas obstruée par le chapeau.

Problèmes lors de l'exécution du programme :

ATTENTION !!! ARRET URGENCE THYMIO :

La commande python

`$ python src/stop_thymio.py`

permet d'arrêter le Thymio-2 sans arrière-tâche.