BotCleaner

Documentationtechnique

**Enzo ROY**

CFPT - Informatique

TPI - 2019

Table des matières

[1 Introduction 2](#_Toc8048165)

[2 Rappel du cahier des charge 3](#_Toc8048166)

[2.1 Objectifs 3](#_Toc8048167)

[2.2 Spécifications 3](#_Toc8048168)

[2.3 Restrictions **Erreur ! Signet non défini.**](#_Toc8048169)

[2.4 Environnement 3](#_Toc8048170)

[2.5 Organisation 3](#_Toc8048171)

[2.6 Livrables 3](#_Toc8048172)

[3 Analyse fonctionnelle 4](#_Toc8048173)

[3.1 Fonctionnalités 4](#_Toc8048174)

[3.2 Interfaces 4](#_Toc8048175)

[4 Analyse organique 5](#_Toc8048176)

[4.1 Architecture du code 5](#_Toc8048177)

[4.1.1 Arborescence de fichier 5](#_Toc8048178)

[4.1.2 Classe 5](#_Toc8048179)

[4.2 Outils 5](#_Toc8048180)

[5 Tests 6](#_Toc8048181)

[6 Conclusion 6](#_Toc8048182)

[7 Sources 6](#_Toc8048183)

[7.1 Codes repris 6](#_Toc8048184)

[7.2 Sites utilisés 6](#_Toc8048185)

[7.3 Aides reçues 7](#_Toc8048186)

[8 Planning 7](#_Toc8048187)

[8.1 Planning prévisionnel 7](#_Toc8048188)

[8.2 Planning Effectif 7](#_Toc8048189)

[9 Table des illustrations 7](#_Toc8048190)

[9.1 Figures 7](#_Toc8048191)

[9.2 Tableaux 7](#_Toc8048192)

[10 Annexes 7](#_Toc8048193)

# Introduction

Cette documentation a pour but de détailler les étapes nécessaires à la réalisation de l’application python/web « BotCleaner » sur Raspberry pi 3, réalisé dans le cadre du Travail Pratique Individuel (TPI).

L’application que je dois réaliser permet à un robot (AlphaBot2) de chercher les intrus sur la zone qui se trouve autour de lui via la caméra. Depuis un navigateur, un utilisateur peut visualiser en temps réel les essais du robot, de suivre son analyse d’image et les décisions qui en découlent. De plus, l’application permet de télécommander le robot tout en ayant un traitement d’image (cadre vert autour des intrus).

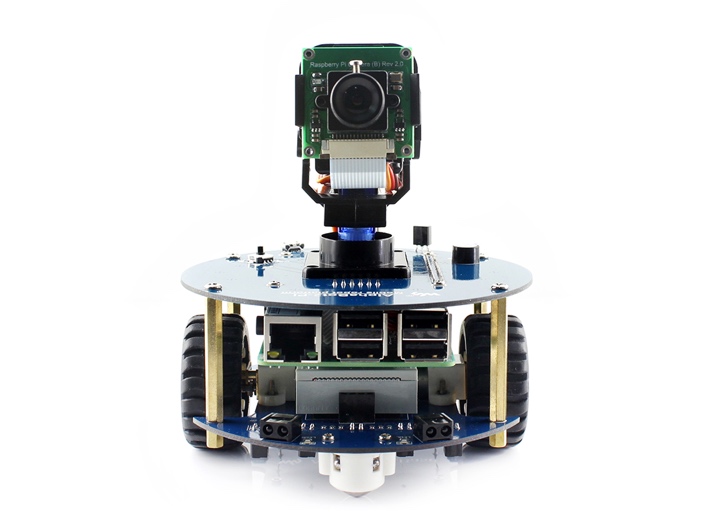


Figure 1 AlphaBot2 de waveshare

# Rappel du cahier des charges

## Organisation

Élève : Enzo Roy [enzo.r@eduge.ch](mailto:enzo.r@eduge.ch)  
Maître : Pascal Bonvin [pascal.bonvin@edu.ge.ch](mailto:pascal.bonvin@edu.ge.ch)  
Experts: Arnold Rullo [arnold.rullo@etat.ge.ch](mailto:arnold.rullo@etat.ge.ch)  
 Jean Sottas jean.sottas@etat.ge.ch

## Objectifs

L’objectif de ce projet est de réaliser en 11 jours une application python/web et ses documentations.

L’application permet de télécommander un robot à distance. De plus le robot doit être capables de nettoyer tous objets parasites de sa zone de travail en utilisant uniquement sa caméra embarquée.

## Spécifications

Les points principaux de l’application sont les suivants :

* Le flux vidéo de la caméra est accessible par un connecteur TCP/IP
* L’analyseur d’image trace un cadre vert autour des intrus sur le flux vidéo. Il doit aussi donner la distance du cadre par rapport au centre de l’image.
* Une page web permet de télécommander le robot (formulaire web + script python)
* Une page web permet de visualiser le travail du robot

## Restrictions

Pour ce projet des restrictions de matériels ainsi que de conceptions :

* Les robots sont des Raspberry pi 3 monté sur une structure AlphaBot2 version Pi
* L’application doit être réalisée en OOP selon le pattern MVC.

## Configuration matériel

Pour le TPI, j’utilise deux ordinateurs, un pour programmer et l’autre installer et consulter les différentes pages WEB sur le robot :

* Un ordinateur personnel
* Un Raspberry Pi 3 B+
* Un routeur Wi-Fi (pour connecter les deux machines sur le même réseau)

Ensuite le matériel pour compléter le robot :

* Alphabot2 kit version Pi (AlphaBot2-Base, AlphaBot2-Pi)
* Raspberry Pi Camera (B) v2.0

## Configuration logiciel

* PC : macOS Mojave (10.14.4)
* Raspberry pi 3 : raspbian (9.9)
* IDE :
  + Python 3 : PyCharm (2018.3)
  + Web : PhpStorm (2018.3)
* Git client : GitKraken (5.0.4)
* Bureautique : Office 360 (2019)
* Autre : Pencil (3.0.5)

## Livrables

A la fin du TPI, les éléments suivants sont rendus au experts et formateur au format électronique :

* La documentation technique et utilisateur contenant le résumé du rapport
* Le code source format imprimable

Le formateur recevra les éléments suivant en plus :

* Le carnet de bord
* Le code source

Le projet est disponible à l’adresse suivante : https://github.com/RoyEnzo/BotCleaner

# Analyse fonctionnelle

## Fonctionnalités

### Serveur web

Cette fonctionnalité permet de faire un lien entre les interfaces utilisateurs et les scripts présents sur le Raspberry pi 3 embarqué.

En se connectant à différentes URL, il sera possible d’exécuter des fonctions disponibles sur le robot. De plus grâce au serveur, les utilisateurs ont accès aux différentes interfaces WEB.

### Flux vidéo en temps réelle

Cette fonctionnalité permet aux utilisateurs d’avoir accès au flux vidéo de la caméra du robot.

Sur le flux vidéo, un carré verre est affiché autour des intrus présents dans la zone du robot visible par la caméra. Celle-ci sera accessible par un connecteur TCP/IP depuis un poste de travail se trouvant sur le même réseau. Grâce au serveur web que j’utilise, il me sera possible d’intégrer se flux dans mes différentes pages web.

### Analyseur d’images

Cette fonctionnalité permet aux robots de déterminé à partir du flux vidéo reçu les possibles intrus positionnés dans sa zone d’action.

Le résultat de cette analyse retournera la position de l’intrus sur l’image ainsi qu’une approximation de sa taille.

Pour des raisons de vitesse de calcul, l’analyse d’image pourra aussi être fait depuis un ordinateur distant possédant une plus grande puissance de calcul. Mais devra être connecté sur le même réseau que le robot.

### Manette de contrôle

Cette fonctionnalité permet aux utilisateurs de contrôler les actions du robot depuis une formulaire web.

Le robot peut être déplacer grâce à des flèches directionnelles. Lorsque l’utilisateur reste appuyé sur un bouton de direction, le robot se déplace tant que l’utilisateur n’a pas relâché le bouton. Il est aussi possible de paramétrer la vitesse de déplacement que celle de la rotation de l’AlphaBot2.

Il est aussi possible de définir la couleur de la surface à nettoyer afin que l’analyseur d’image différencie la couleur de la surface et des objets intrusifs.

### Contrôleur moteur

Cette fonctionnalité permet de contrôler les moteurs installer sur le AlphaBot2.

Chaque roue est indépendante et peuvent tourner dans le sens horaire et antihoraire. La vitesse peut aussi être choisi.

### Contrôleur robot

Cette fonctionnalité permet au robot de récupérer les résultats de l’analyse d’image et de décider les déplacements du robot.

Lorsqu’un objet est placé dans la zone d’action du robot, le contrôleur robot indique les déplacements que le robot doit effectuer pour pousser l’objet hors de la zone. Sinon il indique au robot de se déplacer pour avoir un autre angle de vue.

## Interfaces

### Mode automatique

L’interface du mode automatique ne nous permet pas de contrôler le robot. Il permet uniquement de voir en temps réel les essais du robot, de suivre son analyse d’image et les décisions qui en découlent.



Figure 2 Page web - Mode automatique

### Mode manuel

L’interface manuel permet de télécommander le robot à distance.

Une image contenant capture d’écran

Description générée automatiquement

Figure 3 Page Web - Mode manuel

# Analyse organique

## Architecture du projet

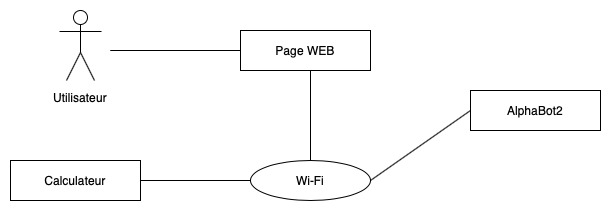


Figure 4 Architecture technique

Tous les scripts du projet sont sur le Alphabot2, le serveur est donc hébergé sur le Raspberry Pi.

Un utilisateur peut avoir accès aux différentes pages WEB en se connectant par TCP/IP via le même Wi-Fi utilisé par le robot.

### Arborescence de fichiers

### Classe

## Outils

# Problèmes rencontrés

## Alimentation robot insuffisante

### Situation

Le robot utilisé pour ce projet, le AlphaBot2 avec Raspberry Pi 3 B+. Celui-ci est munis de deux batteries. Lorsque le robot est mis en tension uniquement avec celles-ci, le Raspberry Pi ne fonctionne pas correctement et redémarre lors de l’utilisation de tous les moteurs.

### Source du problème

Le problème est que les moteurs consomment trop d’énergie et/ou que les batteries n’en délivrent pas assez. Le Raspberry Pi n’est donc pas assez alimenté.

### Solution

Pour que le robot fonctionne correctement, il faut ajouter une alimentation externe sur un des input micro-sd.

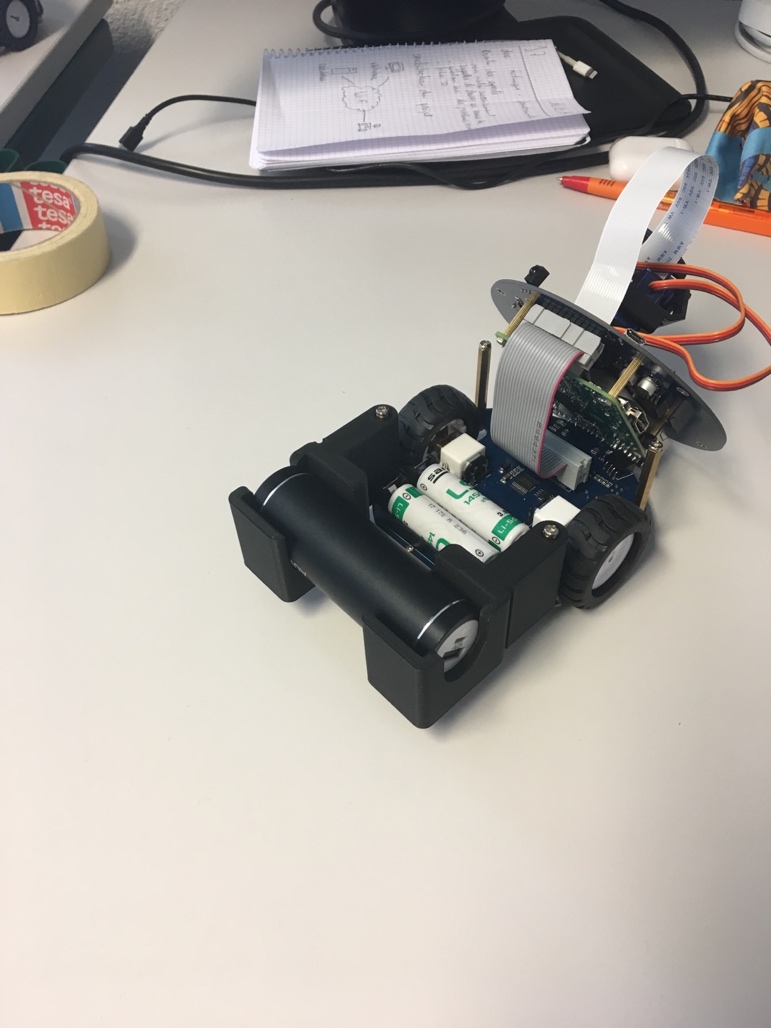


Figure 5 Support batterie supplémentaire

Mon formateur m’a fourni une petite batterie portable pour remédier au problème. Mon collègue, travaillant sur le même robot, et moi avons fait modéliser et imprimer en 3D des supports par un contact extérieur. Le modèle 3D (conçu pour maintenir Aukey PB-N37) est disponible à l’adresse suivant :

**adresse….**

## Pi camera : out of ressources

### Situation

Pour la gestion du projet, j’utilise PyCharm, sur mon poste de travail, puisqu’il propose de nombreux outils utiles. Le programme me permet de faire un déploiement SFTP automatique sur mon Raspberry pi 3 avec la configuration suivante :

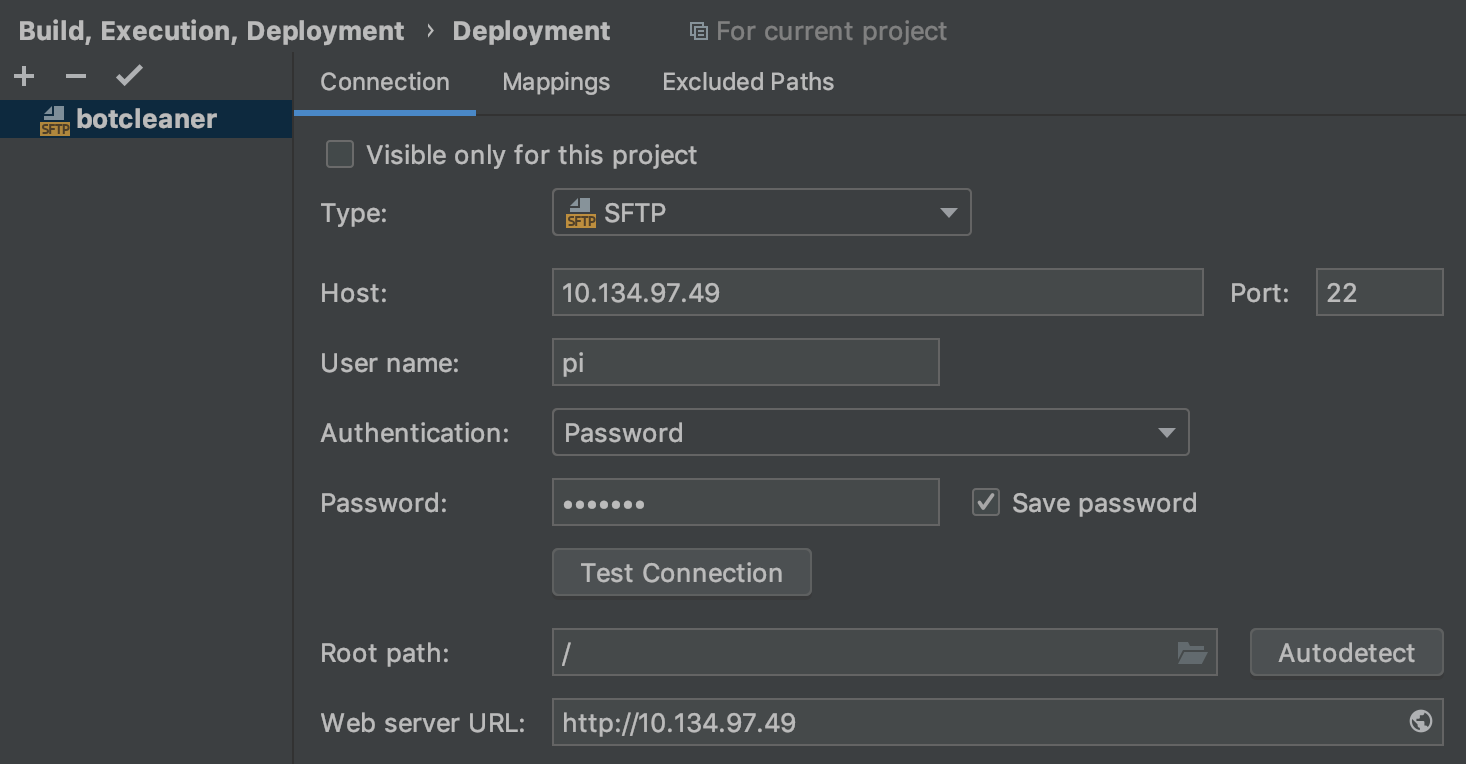


Figure 6 Configuration déploiement SFTP – Connection

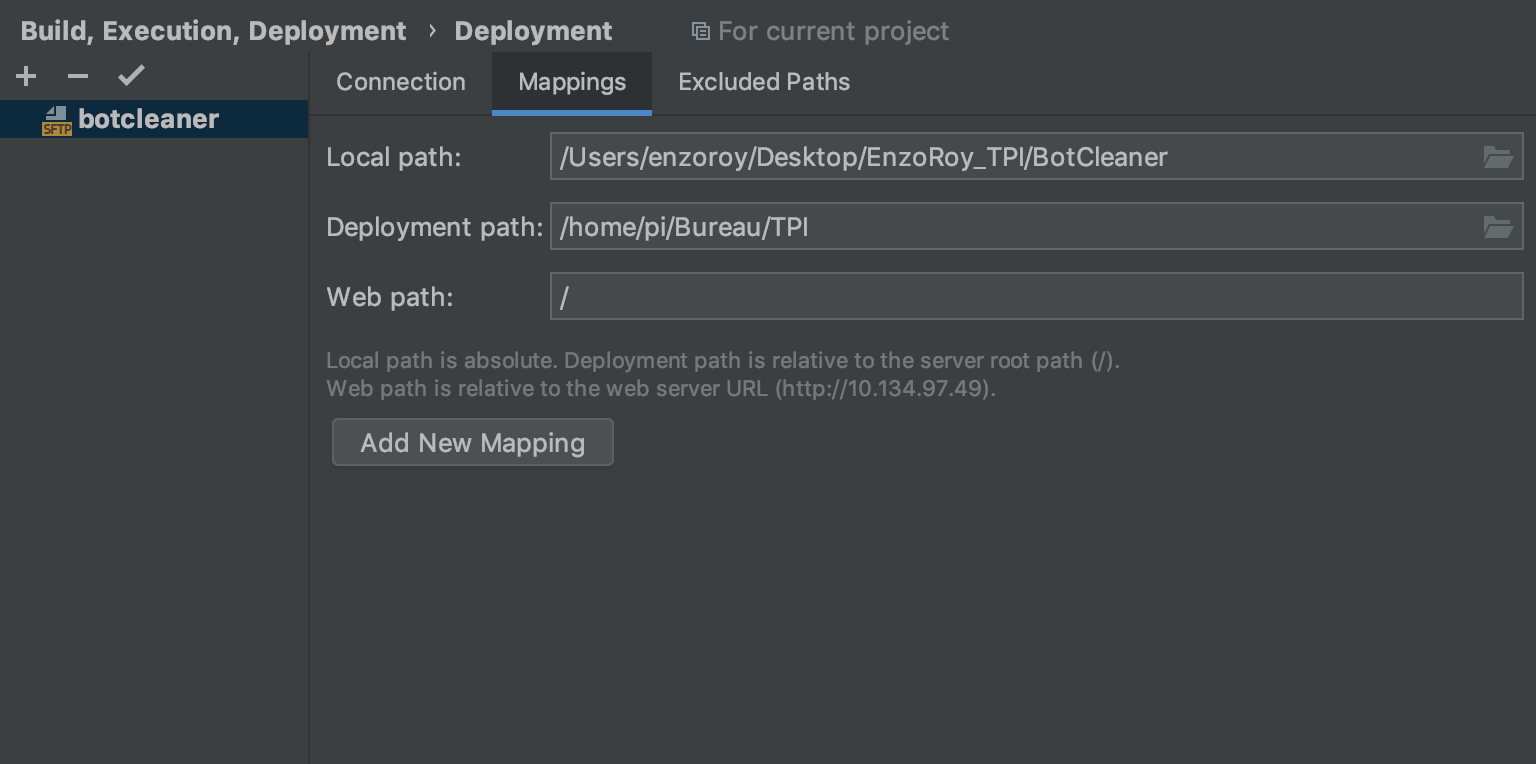


Figure 7 Configuration déploiement SFTP – Fichiers de déploiement

Le Raspberry a comme adresse IPv4 : 10.134.97.49

De plus j’utilise un interpréteur python 3 à distance (celui qui se trouve sur le Raspberry) avec la configuration suivante :

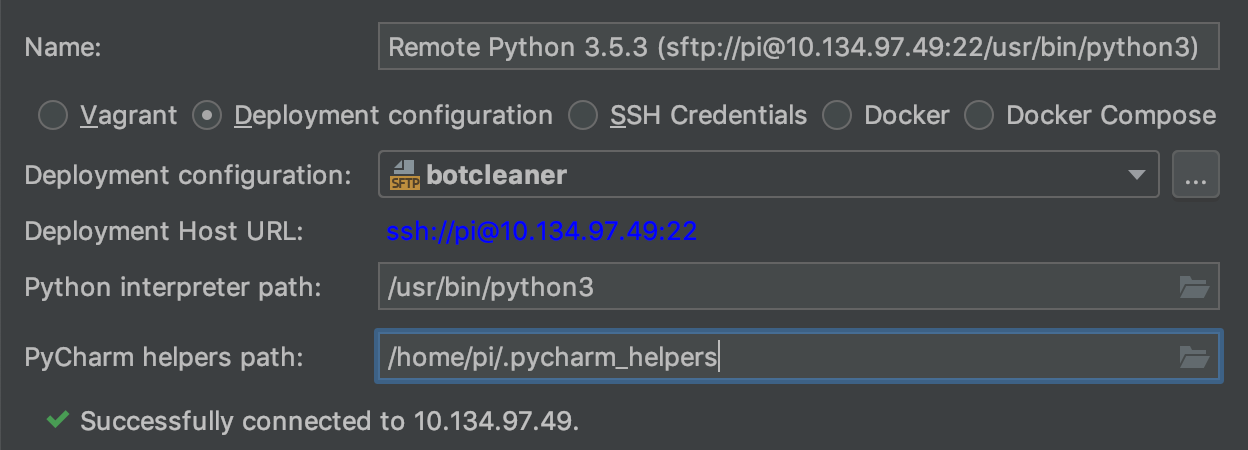


Figure 8 Configuration interpréteur python à distance

Cela me permet de programmer sur mon ordinateur personnel et de gérer à distance le Raspberry.

Nous avons donc le script *main.py* enregistré sur le Raspberry pi 3 à l’emplacement : */home/pi/Bureau/TPI*. Celui-ci contient l’initialisation de la camera (PiCamera V2) ainsi que le serveur Flask.

Pour exécuter *main.py*, j’utilise interpréteur python à distance cité précédemment. Le programme exécute la commande suivante :

ssh://pi@10.134.97.49 :22/usr/bin/python3 -u /home/pi/Bureau/TPI/main.py

Mais l’erreur suivante s’affiche et le processus se termine :

mmal: mmal\_vc\_port\_enable: failed to enable port vc.null\_sink:in:0(OPQV): ENOSPC

mmal: mmal\_port\_enable: failed to enable connected port (vc.null\_sink:in:0(OPQV))0x8af4f0 (ENOSPC)

mmal: mmal\_connection\_enable: output port couldn't be enabled

Traceback (most recent call last):

File "main.py", line 35, in <module>

camera = picamera.PiCamera(resolution=CAM\_RESOLUTION)

File "/usr/lib/python3/dist-packages/picamera/camera.py", line 433, in \_\_init\_\_

self.\_init\_preview()

File "/usr/lib/python3/dist-packages/picamera/camera.py", line 513, in \_init\_preview

self, self.\_camera.outputs[self.CAMERA\_PREVIEW\_PORT])

File "/usr/lib/python3/dist-packages/picamera/renderers.py", line 558, in \_\_init\_\_

self.renderer.inputs[0].connect(source).enable()

File "/usr/lib/python3/dist-packages/picamera/mmalobj.py", line 2212, in enable

prefix="Failed to enable connection")

File "/usr/lib/python3/dist-packages/picamera/exc.py", line 184, in mmal\_check

raise PiCameraMMALError(status, prefix)

picamera.exc.PiCameraMMALError: Failed to enable connection: Out of resources

### Source du problème

Le problème est lisible à la dernière ligne d’erreurs : la picamera n’est pas disponible. Plusieurs raisons sont possibles[[1]](#footnote-1) :

* La camera est désactiver dans raspi-config
* Camera défectueuse / mal branchée
* Un processus utilise déjà la camera
* Un problème de compatibilité

Tout d’abord, j’ai vérifié que la caméra était bien activée. Nous pouvons l’activer de la manière suivante dans le raspi-config :

sudo raspi-config

Ensuite pour tester le bon fonctionnement de la camera :

raspistill -o test.jpg

J’ai vérifié que l’image prise était correcte : la camera est bien active et fonctionnelle. Cela nous a enlevé en même temps le problème de la caméra défectueuse ou mal branchée. Nous avons cherché si un processus faisait tourner un script python utilisant déjà la caméra :

ps aux | grep python3

Résultat :

pi 1860 0.0 0.2 4588 2556 pts/1 Ss+ 15:46 0:00 bash -c cd /home/pi/Bureau/TPI; env "PYCHARM\_HOSTED"="1" "PYTHONUNBUFFERED"="1" "PYTHONIOENCODING"="UTF-8" "PYCHARM\_MATPLOTLIB\_PORT"="56770" "JETBRAINS\_REMOTE\_RUN"="1" "PYTHONPATH"="/home/pi/Bureau/TPI:/home/pi/.pycharm\_helpers/pycharm\_matplotlib\_backend" '/usr/bin/python3' '-u' '/home/pi/Bureau/TPI/main.py'

pi 1861 7.8 2.3 110548 21476 pts/1 Sl+ 15:46 0:06 /usr/bin/python3 -u /home/pi/Bureau/TPI/main.py

pi 1886 0.0 0.0 4368 524 pts/0 S+ 15:47 0:00 grep --color=auto python3

Il y avait déjà le script*main.py* du projet TPI qui était en exécution. Pour arrêter le processus (PID : 1860) et donc fermé le script :

kill -9 1860

Le problème avait l’air résolue. Mais lorsque je redémarrais le Raspberry, le script était de nouveau en exécution. J’ai donc regarder si je n’avais pas ajouter le projet dans le fichier *.bashrc* :

sudo nano /home/pi/.bashrc

Mais à la dernière ligne seul un script éteignant le haut-parleur du AlphaBot2 était inscrit. J’ai cherché plusieurs autres manières de lancer un script au démarrage de la machine[[2]](#footnote-2) mais aucun n’a été concluant…

Une fois ma machine de travail éteinte et le Raspberry redémarré pour réinstaller un OS, le script bloquant n’apparaissait plus dans les processus. Donc j’en ai conclu que le Raspberry n’avait pas de script, utilisant la caméra, qui se lance automatiquement au démarrage. Le problème venait de mon poste de travail.

Pour finir il s’avère que depuis PiCharm, j’avais dupliqué un *profil* dans l’outil utilisant les interpréteurs distants et que ce doublon se lançait en parallèle. C’est-à-dire que lorsque je lançais le script *main.py*, il se lançait deux fois mais le doublon dans une console non affiché, ce qui m’empêchait de savoir qu’il était en exécution…

Cette erreur a été trouvé après coup, une fois que j’ai réinitialisé les paramètres par défaut de PiCharm.

# Tests

# Conclusion

## Bilan personnel

# Glossaire

**Aucune entrée d'index n'a été trouvée.**

# Sources

## Codes repris

Pour mettre en place le flux vidéo, j’ai utilisé des parties de code présentent dans les projets suivants :

https://blog.miguelgrinberg.com/post/video-streaming-with-flask  
https://raw.githubusercontent.com/RuiSantosdotme/Random-Nerd-Tutorials/master/Projects/rpi\_camera\_surveillance\_system.py

## Sites utilisés

<https://sourceforge.net/projects/mjpg-streamer/>

<http://www.magdiblog.fr/divers/raspberry-pi-camera-5-facons-de-faire-du-streaming/>

<https://stackoverflow.com/questions/42601478/flask-calling-python-function-on-button-onclick-event>

<https://www.digitalocean.com/community/questions/running-mutliple-flask-application>

<https://openclassrooms.com/fr/courses/1654786-creez-vos-applications-web-avec-flask/1655538-tp-service-web-dupload-dimages>

<https://stackoverflow.com/questions/7391945/how-do-i-read-image-data-from-a-url-in-python>

https://realpython.com/documenting-python-code/

## Aides reçues

M. Bonvin Notamment lors de l’erreur avec la caméra

# Planning

## Planning prévisionnel



Tableau 1 Planning prévisionnel

## Planning effectif

# Table des illustrations

## Figures

[Figure 1 AlphaBot2 de waveshare 2](#_Toc8716832)

[Figure 2 Page web - Mode automatique 4](#_Toc8716833)

[Figure 3 Page Web - Mode manuel 5](#_Toc8716834)

## Tableaux

[Tableau 1 Planning prévisionnel 7](#_Toc8716843)

# Annexes

EnzoRoy\_Planning\_Taches.xlsx Planning prévisionnel/effectif, tâches effectuées  
EnzoRoy\_documentation\_utilisateur.docx Documentation pour la mise en place du robot

1. https://raspberrypi.stackexchange.com/questions/26829/picamera-not-working [↑](#footnote-ref-1)
2. https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/ [↑](#footnote-ref-2)