BotCleaner

Documentationtechnique

**Enzo ROY**

CFPT - Informatique

TPI - 2019

Table des matières

[1 Introduction 3](#_Toc9198496)

[2 Rappel du cahier des charges 4](#_Toc9198497)

[2.1 Organisation 4](#_Toc9198498)

[2.2 Objectifs 4](#_Toc9198499)

[2.3 Spécifications 4](#_Toc9198500)

[2.4 Restrictions 4](#_Toc9198501)

[2.5 Configuration matériel 4](#_Toc9198502)

[2.6 Configuration logiciel 4](#_Toc9198503)

[2.7 Livrables 5](#_Toc9198504)

[3 Analyse fonctionnelle 5](#_Toc9198505)

[3.1 Fonctionnalités 5](#_Toc9198506)

[3.1.1 Serveur web 5](#_Toc9198507)

[3.1.2 Flux vidéo en temps réelle 5](#_Toc9198508)

[3.1.3 Analyseur d’images 5](#_Toc9198509)

[3.1.4 Manette de contrôle 5](#_Toc9198510)

[3.1.5 Contrôleur moteur 6](#_Toc9198511)

[3.1.6 Contrôleur robot 6](#_Toc9198512)

[3.2 Interfaces 6](#_Toc9198513)

[3.2.1 Mode automatique 6](#_Toc9198514)

[3.2.2 Mode manuel 6](#_Toc9198515)

[4 Analyse organique 7](#_Toc9198516)

[4.1 Architecture du projet 7](#_Toc9198517)

[4.2 Architecture technique 7](#_Toc9198518)

[4.2.1 Page web 8](#_Toc9198519)

[4.2.2 Classe 9](#_Toc9198520)

[4.3 Arborescence de fichiers 12](#_Toc9198521)

[4.4 Outils 12](#_Toc9198522)

[5 Problèmes rencontrés 12](#_Toc9198523)

[5.1 Alimentation robot insuffisante 12](#_Toc9198524)

[5.1.1 Situation 12](#_Toc9198525)

[5.1.2 Source du problème 12](#_Toc9198526)

[5.1.3 Solution 12](#_Toc9198527)

[5.2 Pi camera : out of ressources 13](#_Toc9198528)

[5.2.1 Situation 13](#_Toc9198529)

[5.2.2 Source du problème 14](#_Toc9198530)

[5.2.3 Solution 15](#_Toc9198531)

[6 Tests 16](#_Toc9198532)

[7 Conclusion 16](#_Toc9198533)

[7.1 Bilan personnel 16](#_Toc9198534)

[8 Glossaire 16](#_Toc9198535)

[9 Sources 16](#_Toc9198536)

[9.1 Codes repris 16](#_Toc9198537)

[9.2 Sites utilisés 16](#_Toc9198538)

[9.3 Aides reçues 16](#_Toc9198539)

[10 Planning 16](#_Toc9198540)

[10.1 Planning prévisionnel 17](#_Toc9198541)

[10.2 Planning effectif 17](#_Toc9198542)

[11 Table des illustrations 17](#_Toc9198543)

[11.1 Figures 17](#_Toc9198544)

[11.2 Tableaux 17](#_Toc9198545)

[12 Annexes 17](#_Toc9198546)

# 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).

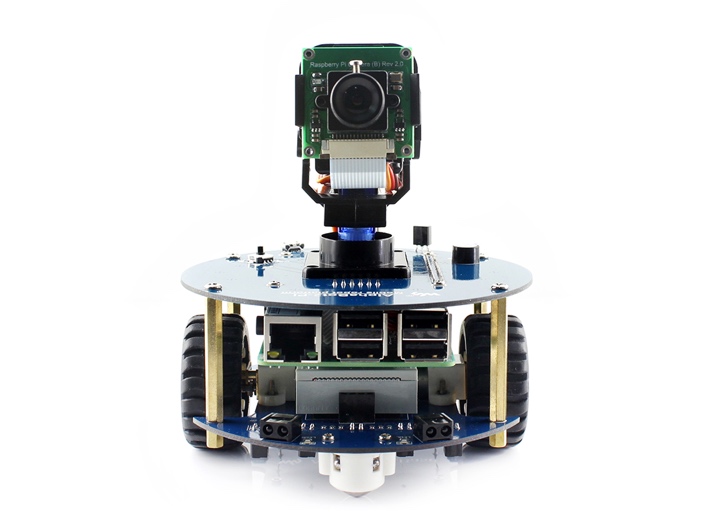


Figure 1 AlphaBot2 de waveshare

# 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 capable de nettoyer tous objets parasites de sa zone de travail (uniformément colorée) 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 matériel ainsi que de conception :

* Le robot est un Raspberry pi 3 monté sur une structure AlphaBot2 version Pi
* L’application doit être réalisée en OOP selon le pattern MVC.
* Le robot se trouve sur une surface uniformément colorée

## Configuration matériel

Pour le TPI, j’utilise deux ordinateurs, un pour le robot et un pour programmer et consulter les différents page WEB :

* Un ordinateur personnel
* Un Raspberry Pi 3 B+ (sur le robot)
* 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), Draw.io (application web)

## Livrables

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

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

Le formateur recevra, en plus, les éléments suivants :

* Le carnet de bord
* Le code source

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

# Analyse fonctionnelle

## Cas d’utilisation

AJOUTER ILLUSTRATION CAS UTILISATION( PERTINANT ?)

## 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 URLs, 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 deux points, le point en haut à gauche et le point en bas à droite de l’objet présent sur l’image. Avec ces deux points, il est possible de déterminer le centre de l’objet ainsi que sa taille approximative.

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 déplacé 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 rotation des moteurs.

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 installés sur le AlphaBot2.

Chaque roue est indépendante et peut avancer ou reculer (sens horaire ou anti horaire). La vitesse peut aussi être paramétrée.

### 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 tourner sur lui-même pour avoir un autre angle de vue.

## Interfaces

### Mode automatique



Figure 2 Page web - Mode automatique

### Mode manuel

Une image contenant capture d’écran

Description générée automatiquement

Figure 3 Page Web - Mode manuel

# Analyse organique

## Architecture du projet

### Général

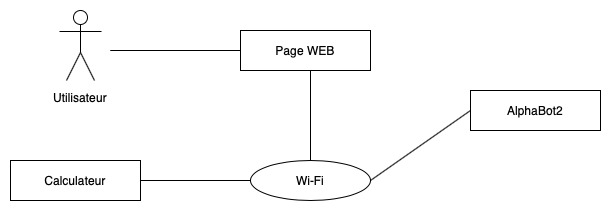


Figure 4 Architecture générale du projet

### Détaillée

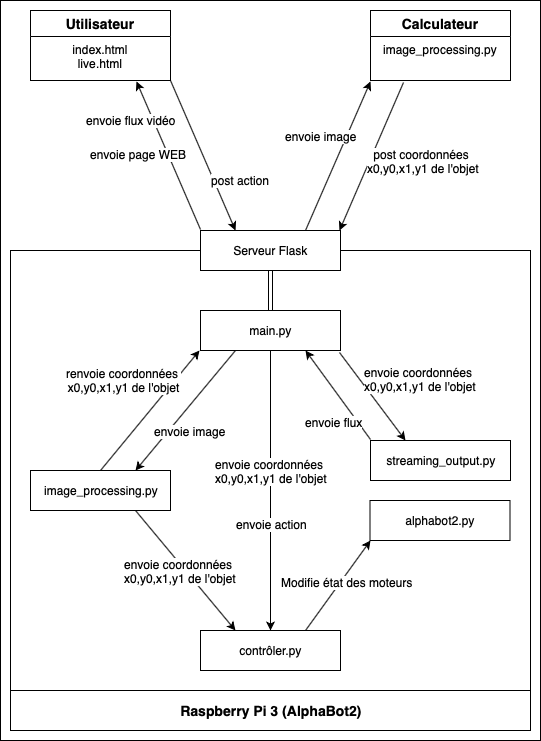


Figure 5 Architecture détaillée du projet

## Architecture technique

### Page web

Pour afficher les différentes pages WEB, j’utilise le moteur de modèle Jinja2 puisque Flask est livré avec. Grâce à la fonction **render\_template()**, il suffit de fournir le nom du modèle en paramètre :

@app.route('/')

def index():

return render\_template('index.html')

Dans cette exemple la page *index.html* est à afficher. Mais pour cela Flask cherche les modèles dans le dossier *templates* devant être dans le même répertoire que le serveur (*main.py*)*:*

/main.py

/templates

/index.html

#### Mannette de contrôle

La manette de contrôle comporte un flux vidéo ainsi que quatre boutons directionnels. Le flux vidéo est récupéré sur le serveur Flask grâce à la balise suivante :

<img src="{{ url\_for('video\_feed') }}" alt="stream">

La fonction **url\_for()** permet de récupérer l’url d’une fonction grâce à son nom passé en paramètre. La fonction retourne un générateur comportant un objet *StreamingOutput.* La source est un flux d’image JPEG à environ 5-10 images secondes. J’utilise le peu d’image par seconde au profit de la latence, une seconde de décalage est visible sur le flux selon la connexion réseau.

Pour le contrôle du robot à distance, j’utilise des boutons avec chacun un id correspondant à l’une des actions du robot (forward, backward, turn\_left, turn\_right):

<button id="forward" class="btn"></button>

Une partie de code jQuery permet d’ajouter un évènement à tous les boutons. Ils exécutent une fonction lorsqu’ils sont appuyés. Pour des soucis d’appareil avec un écran tactile le *mousedown* ne fonctionne pas sur tous les appareils. Il m’a donc fallu ajouter *touchstart* :

$(function() {

$(".btn").on('touchstart mousedown' , function() {

postCmd(this.id);

});

});

La fonction exécutée envoie avec la méthode POST l’id du bouton, correspondant à une action. J’utilise pour ça la fonction **$.post()** proposé par jQuery. Du côté du serveur je récupère l’information grâce à *Response* une fonction proposée par Flask :

command = request.form[index]

Pour la suite, le script *web\_controller.py* s’occupe de recevoir le string de l’action et de modifier l’état du robot.

Pour que le robot n’effectue que l’action lorsque l’on reste appuyé sur l’un des boutons, il faut ajouter un évènement lorsque le bouton est relâché. La commande choisit à envoyer en POST est *stop* :

$(".btn").on(‘touchend mouseup’, function() {

postCmd(‘stop’);

### Classe

AJOUTER DIAGRAMME DES CLASSES

#### Main

Le script *main* est le plus important puisqu’il effectue le lien entre le robot et les utilisateurs. Pour cela j’ai besoin d’un serveur. Pour des raisons de temps et de faciliter, j’utilise un Framework Web Python. Il y a beaucoup de choix, j’ai donc décidé d’utiliser un des plus connus : Flask, Django, Tornado, Diesel, Pecan et Falcon. Ayant déjà utilisé Flask et très peu Django, mon choix ne se limite qu’aux deux :

Flask[[1]](#footnote-1) est un framework très jeune (2010). J’ai souvent lu qu’il était conseillé pour les petits projet en raison de sa facilité à mettre en place d’une application.

Les points positifs que j’ai pu observer durant mes précédentes utilisations :

* Simple à utiliser
* Le routage des URL est facile

Par contre, dû à sa jeunesse, la documentation n’est pas très fournie et la recherche d’information est rapidement très longue

Django[[2]](#footnote-2) est un autre framework que j’ai utilisé auparavant mais de manières très brèves. La documentation est nettement plus importante et les forums regorges de ressources. La majorité de personnes le conseille. Malheureusement je ne connais pas assez celui-ci pour me sentir à l’aise durant la réalisation du projet

Pour des questions de temps et de facilité, c’est Flask que je compte utiliser.

#### StreamingOutput

La classe streaming output est une classe trouver dans la documentation de la Pi Camera (doc. 4.10). Elle permet de faire un objet sur lequel la camera pourra écrire les images. L’objet sera ensuite envoyé en retour comme générateur python.

#### Robot - Wheel

La classe robot permet de contrôler le AlphaBot2-Pi. Celui-ci possède une multitude de capteurs et des moteurs. Mais pour ce projet seul les deux moteurs à brosses permettant le déplacement du robot m’intéressent. Grâce au manuelle utilisateur du AlphaBot2[[3]](#footnote-3), je sais que les différents GPIO pour les contrôler sont les suivant :

|  |  |  |
| --- | --- | --- |
| Action | GPIO (mode BCM) | Type GPIO |
| Roue gauche avance | 20 | GPIO.OUT |
| Roue gauche recule | 21 | GPIO.OUT |
| Roue gauche vitesse | 26 | GPIO.PWM |
| Roue droite avance | 13 | GPIO.OUT |
| Roue droite recule | 12 | GPIO.OUT |
| Roue droite vitesse | 6 | GPIO.PWM |

Tableau 1 AlphaBot2-base, GPIO (mode BCM) pour contrôler les moteurs

Lorsque l’on souhaite faire avancer une roue, le pin avancer doit être actif et le pin reculer doit être inactif. Si l’on souhaite faire reculer une roue l’état des pins sont inversés. Un exemple pour faire avancer une des deux roues :

GPIO.output(self.IN\_1, GPIO.HIGH)

GPIO.output(self.IN\_2, GPIO.LOW)

pwm.ChangeDutyCycle(duty\_cycle)

Le robot étant constitué de deux roues parallèles, celui-ci avance comme un char :

* Tourner à droite : la roue gauche avance
* Tourner à gauche : la roue droite avance

Pour éviter que le robot ne se déplace pas trop lorsqu’il tourne, je fais aussi reculer la roue opposée.

#### ImageProcessing

La classe ImageProcessing regroupe tous les outils pour analyser et traiter les images provenant du flux vidéo de la Pi Camera. Plusieurs méthodes sont possibles pour détecter les objets dans la surface du robot.

Je choisis d’utiliser OpenCV, une librairie extrêmement complète et varié pour le traitement d’image et flux vidéo. Ne sachant pas l’utiliser complètement je décide de rester sur de l’analyse par image. La librairie Numpy me sera aussi nécessaire pour convertir les images en tableaux.

Pour faciliter la conception de mon algorithme, je décide d’utiliser une image de référence :



Figure 6 Image de référence pour analyse d'image

Dans cette image, la partie orange est considérée comme la surface à analyser. Le cube noir représente l’objet à détecter. Le reste ne doit pas être pris en compte dans l’analyse. Je prendrai comme couleur à détecter l’orange RGB (185,80,40), la couleur de la surface visible sur l’image (Figure 5).

La première étape consiste à différencier la surface et le reste de l’image. Je commence par créer un masque pour obtenir uniquement les pixels d’une certaine échelle de couleurs RGB. En faisant mes recherches, j’ai rapidement compris qu’il fallait utiliser un autre espace colorimétrique : le HSV[[4]](#footnote-4). Je converti donc la couleur choisie en HSV[17°,75%,75%]. OpenCV utilise cette espace de couleurs sous une autre échelle qui est [H 0-179, S 0-255, V 0-255]. Une fois la bonne échelle et arrondis les valeurs, j’obtiens la couleur HSV[10°, 190, 190].

La fonction ***inRange()*** de CV2 nécessite deux couleurs : la couleur la plus claire et la plus foncée pour faire un masque ou seuls les pixels dans cette échelles de couleurs soit visible. Le conseil le plus donné sur *stack overflow* [[5]](#footnote-5) est de faire HSV[H-10, 100, 100] et HSV[H+10, 100, 100] comme limite inférieur et supérieur.



Figure 7 Échelle de couleurs de la surface du robot

Ne nécessitant plus les couleurs, j’utiliserai le masque pour la suite de l’analyse. La qualité étant aussi trop élever pour mon utilité, je la réduis à 3% (si je réduis d’avantage la qualité, certains objets ne serait plus visible). En faisant cette réduction de qualité, j’évite ainsi d’avoir trop de artefacts :



Figure 8 Masque après application du filtre et réduction de la qualité à 3%

Nous pouvons voir sur l’image (Figure) que seul la surface est désormais visible. Ce qui nous permet de différencier aussi l’arrière-plan, hors de la surface, et l’intrus, dans la surface.

Pour faire la distinction entre l’arrière-plan et l’objet, je commence par chercher l’index de la première ligne comportant du blanc (ligne rouge). Puis je parcours chaque colonne à partir de cet index et enregistre le premier pixel comportant du blanc. Cela me permet d’obtenir une liste comportant tous les pixels du bord supérieur de la surface (points verts):

Une image contenant texte

Description générée automatiquement

## Arborescence de fichiers

## Outils

# Problèmes rencontrés

## Alimentation robot insuffisante

### Situation

Le robot utilisé pour ce projet, le AlphaBot2 avec Raspberry Pi 3 B+. Celui-ci est muni 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.

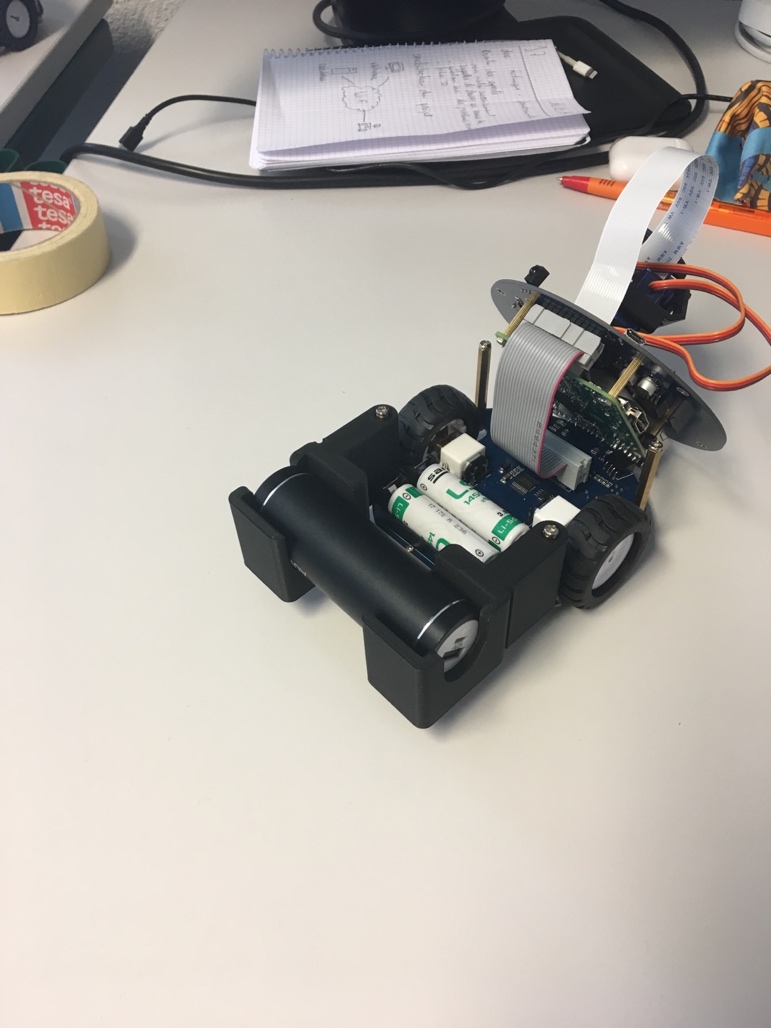


Figure 9 Support batterie supplémentaire

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 suivante :

**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 :

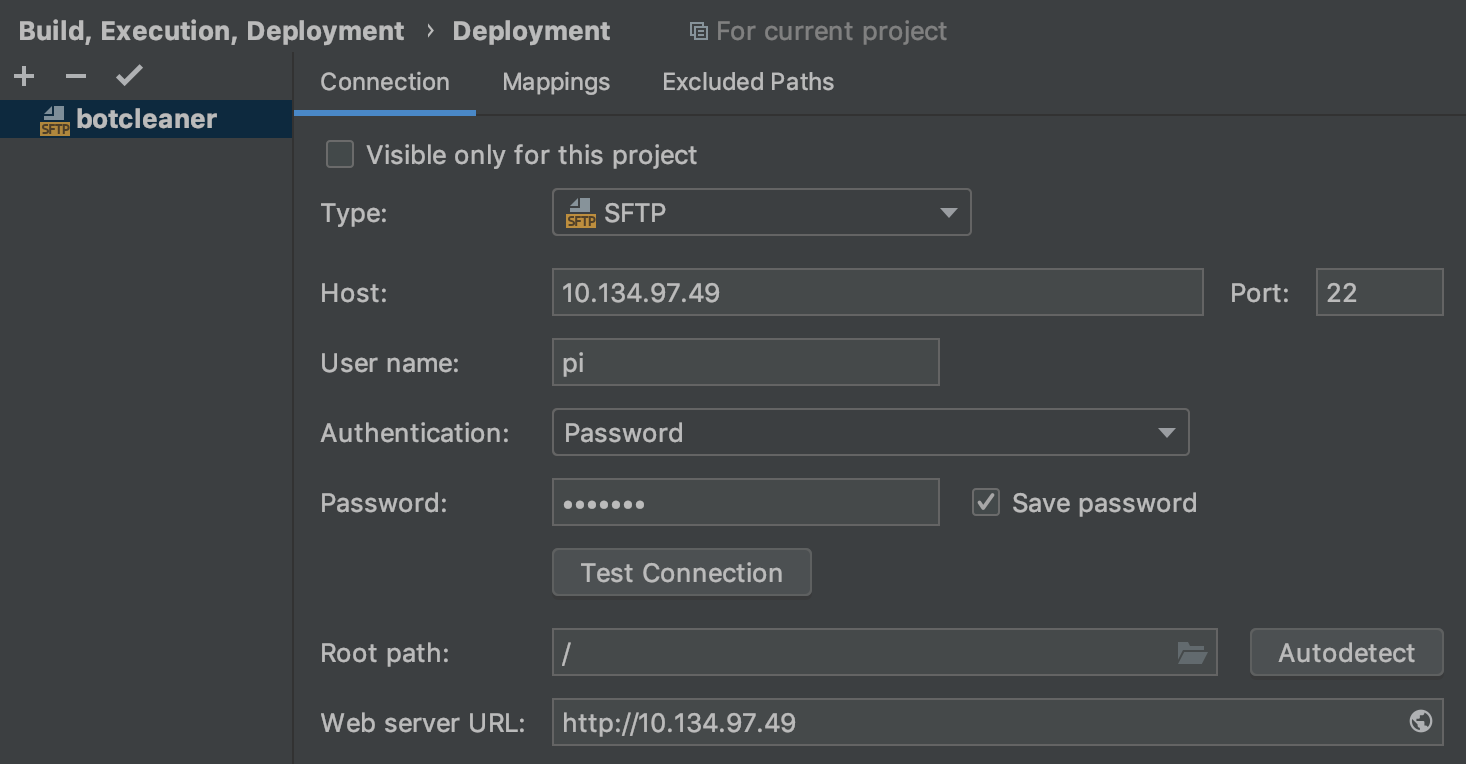


Figure 10 Configuration déploiement SFTP – Connexion

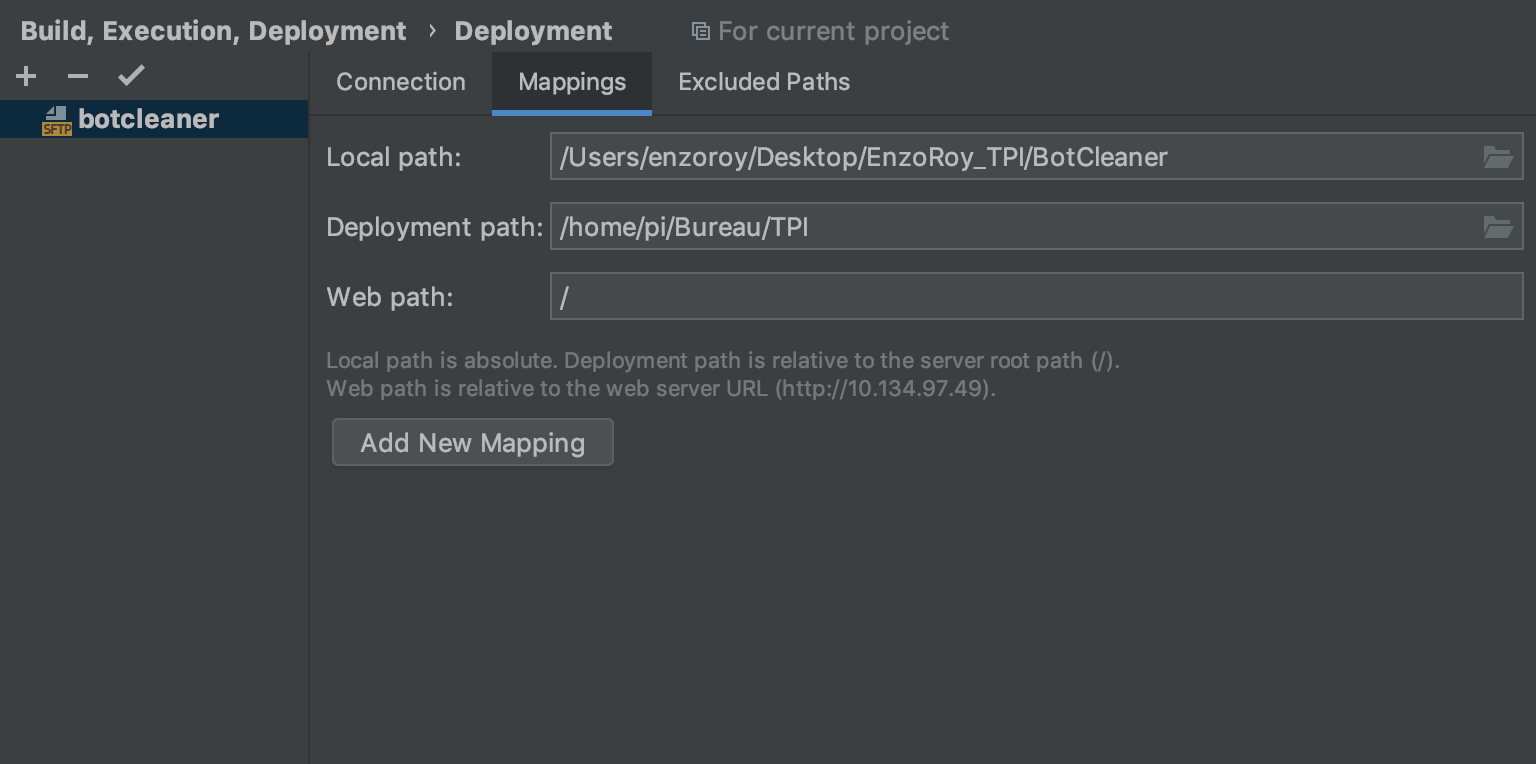


Figure 11 Configuration déploiement SFTP – Fichiers de déploiement

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 :

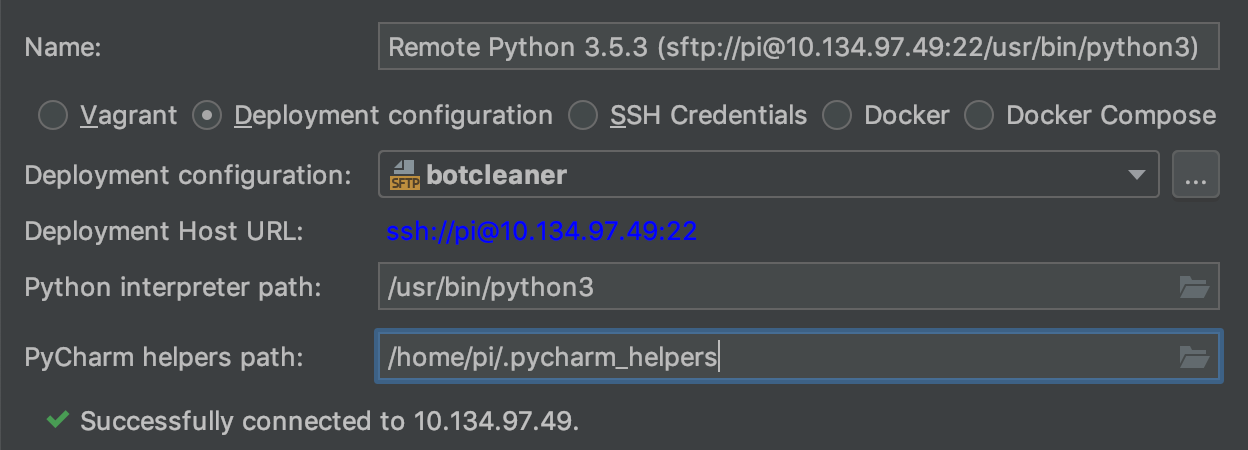


Figure 12 Configurations de l’interpréteur python à distance

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 caméra (PiCamera V2) ainsi que le serveur Flask.

Pour exécuter *main.py*, j’utilise l’interpréteur python distant 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[[6]](#footnote-6) :

* La caméra est désactivée dans raspi-config
* Camera défectueuse / mal branchée
* Un processus utilise déjà la caméra
* 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 caméra 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 fermer le script :

kill -9 1860

Le problème avait l’air résolu. Mais lorsque je redémarrais le Raspberry, le script était de nouveau en exécution. J’ai donc regardé si je n’avais pas ajouté 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[[7]](#footnote-7), 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.

### Solution

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 affichée, ce qui m’empêchait de savoir qu’il était en exécution…

Cette erreur a été trouvée 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-NerdTutorials/master/Projects/rpi_camera_surveillance_system.py>

## Sites utilisés

https://stackoverflow.com/

<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. Pascal Bonvin

M. Quentin Forestier

M. François Oscar

# Planning

## Planning prévisionnel



Tableau 2 Planning prévisionnel

## Planning effectif

# Table des illustrations

## Figures

[Figure 1 AlphaBot2 de waveshare 3](#_Toc9355653)

[Figure 2 Page web - Mode automatique 7](#_Toc9355654)

[Figure 3 Page Web - Mode manuel 7](#_Toc9355655)

[Figure 4 Architecture générale du projet 8](#_Toc9355656)

[Figure 5 Architecture détaillée du projet 8](#_Toc9355657)

[Figure 6 Image de référence pour analyse d'image 11](#_Toc9355658)

[Figure 7 Échelle de couleurs de la surface du robot 12](#_Toc9355659)

[Figure 8 Masque après application du filtre et réduction de la qualité à 3% 12](#_Toc9355660)

[Figure 9 Support batterie supplémentaire 13](file:////Users/enzoroy/Desktop/EnzoRoy_TPI/BotCleaner/Docs/EnzoRoy_documentation_technique.docx#_Toc9355661)

[Figure 10 Configuration déploiement SFTP – Connexion 13](#_Toc9355662)

[Figure 11 Configuration déploiement SFTP – Fichiers de déploiement 14](#_Toc9355663)

[Figure 12 Configurations de l’interpréteur python à distance 14](#_Toc9355664)

## Tableaux

[Tableau 1 AlphaBot2-base, GPIO (mode BCM) pour contrôler les moteurs 11](#_Toc9355648)

[Tableau 2 Planning prévisionnel 17](#_Toc9355649)

# Annexes

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

1. http://flask.pocoo.org [↑](#footnote-ref-1)
2. https://www.djangoproject.com [↑](#footnote-ref-2)
3. https://www.mouser.com/pdfdocs/Alphabot2-user-manual-en.pdf [↑](#footnote-ref-3)
4. https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/ [↑](#footnote-ref-4)
5. https://stackoverflow.com/questions/28539719/hsv-ranges-for-color-filter/28542584#28542584 [↑](#footnote-ref-5)
6. https://raspberrypi.stackexchange.com/questions/26829/picamera-not-working [↑](#footnote-ref-6)
7. https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/ [↑](#footnote-ref-7)