

RAPPORT DE PROJET PYTHON

CODE ORIENTÉ OBJET

```
31
32 self.file = None
33 self.fingerprints = self()
34 self.logdups = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'request_
39     self.file.seek(0)
40     self.fingerprints.update(s.request)
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('SUPERFILTER_DEBUG')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
```

Introduction

L'objectif de ce projet est de réaliser une application avec une interface graphique (« GUI » = graphical user interface) qui permet de se connecter au robot. Une fois connecté, l'utilisateur peut visualiser les mouvements du robot et commander ses déplacements.

I. Répartition des Tâches

1) Thibaut Barottin :

- Implémentation de la réception des données du robot avec gazebo
- Implémentation des commandes de déplacement du robot
- Création des fonctions liant boutons et touches du clavier (Avancer, Reculer, Pivoter vers la gauche, Pivoter vers la droite)

2) Pierre Peaupardin :

- Création de l'interface graphique avec TKinter.
- Implémentation des différents écrans de visualisation
- Ajout des boutons

3) Vlad Tondeur :

- Utilisation de matplotlib.animation pour visualiser la position relative et la vitesse du robot.
- Création de la zone de visualisation des mouvements et des graphiques de vitesse.
- Documentation du code et des fonctionnalités de l'interface dans le fichier README.md.

4) Nicolas Notter :

- Mise en place des tests unitaires
- Ecriture du code main.py
- Amélioration du code de l'interface graphique

II. Fonctionnement de l'Application

Le programme principal **main.py** est le programme à exécuter. Ce dernier va faire appel aux autres programmes ainsi qu'aux packages nécessaires (décrites dans requirement.txt). Le programme **Gazebo_launch.py** va lancer Gazebo à chaque utilisation, afin de pouvoir simuler convenablement le TurtleBot que l'on souhaite commander. Les programmes

robot_controller.py et **command_frame.py** vont respectivement assurer la simulation correcte du robot et de ses déplacements à l'aide de Gazebo, ainsi que le contrôle dudit robot dans la simulation (déplacement linéaire dans une direction, rotation, arrêt, etc). Enfin, **movement_frame.py** est le programme qui va permettre de représenter sous forme de graphe dans l'interface graphique (GUI) les différentes grandeurs que mesure le TurtleBot.

III. Problèmes Rencontrés

La Carte OpenCR de notre TurtleBot étant défectueuse (refuse toute compilation de programme et tout transfert entre le TurtleBot et un quelconque ordinateur), nous ne pouvons hélas pas tester notre programme sur le robot "en réel", et devons donc utiliser Gazebo pour simuler son comportement. La conséquence de ce dysfonctionnement est la suppression de la partie "Connexion au robot" sur l'interface graphique car nous ne pourrions nous connecter au TurtleBot réel, cette partie est donc devenue superflue.

IV. Résultats Finaux

Nous avons au final un projet qui permet effectivement de simuler et de contrôler le Turtlebot dans l'environnement Gazebo, ainsi que de récupérer les grandeurs qu'il nous renvoie pour les afficher sous forme de Graphes. Cependant, en raison de la défaillance technique de notre TurtleBot au niveau de la carte OpenCR, nous ne pouvons plus utiliser efficacement ROS dans une optique de communication entre notre Robot et nos ordinateurs. Nous avons donc un projet opérationnel en théorie, mais que nous n'avons pas pu tester sur le projet réel.

V. Ressources Utilisées

Pour utiliser ROS :

- <https://www.ros.org/>
- <https://wiki.ros.org/rospy>
- <https://docs.ros.org/en/kinetic/api/rospy/html/index.html>
- <https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/>

Pour utiliser Gazebo :

- <https://gazebo.org/home>

Pour importer les packages nécessaires :

- Le Package SetupTools

Le dépôt Github dont nous nous sommes servis :

- <https://github.com/AsapNedaLNeB/Projet-Python-Turtlebot-Cormoran>

Pour setup ROS avec Gazebo :

- <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/>