POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

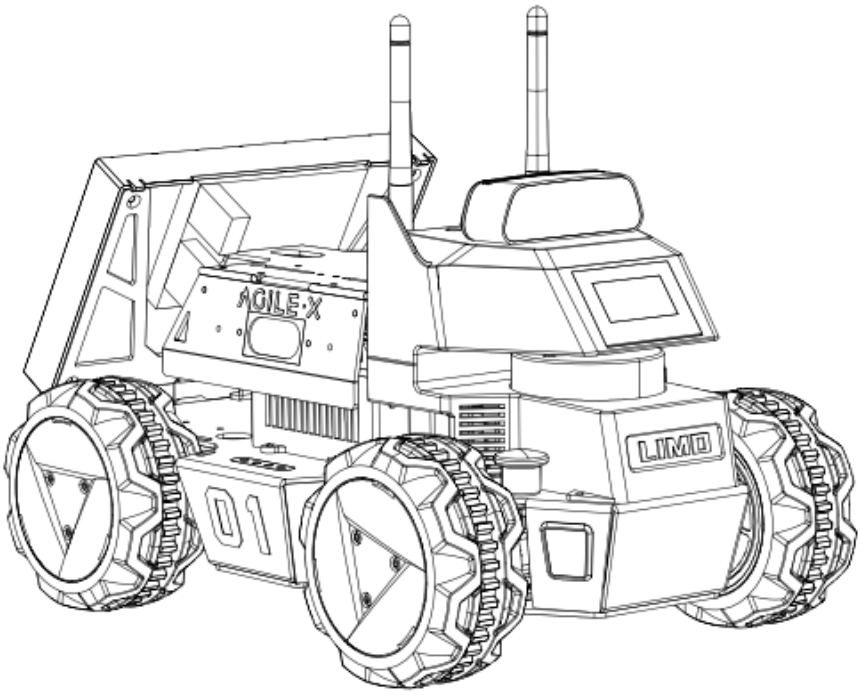
Département de génie informatique et génie logiciel

INF3995

Projet de conception d'un système informatique

Procédure de tests pour le projet CartoRovers

Équipe 103



Les composantes logicielles pour la station au sol ont été testées à l'aide de tests unitaires utilisant des librairies de tests telles que Jasmine et Sinon, ainsi que plusieurs essais et tests d'utilisabilité.
En ce qui concerne le reste des composantes (simulation et code embarqué), il n'a pas été possible de faire des tests unitaires. Une autre stratégie a été privilégiée, reposant principalement sur des tests d'intégration, de systèmes et de performance a été privilégiée.

1. Simulation

Tout d'abord, plusieurs tests de systèmes ont été faits en lançant la simulation indépendamment de la station au sol afin d'observer le comportement des robots dans l'environnement de simulation. Certains aspects pertinents à observer étaient la bonne génération de l'environnement et des robots, le déplacement autonome des robots, les données lues par les capteurs ainsi que l'exploration et la génération de la carte de l'environnement. Une fois le comportement et le déroulement général de la simulation validés, nous avons intégré la simulation à la station au sol permettant ainsi de pouvoir lancer une simulation à partir de l'interface. Il a donc fallu effectuer des tests d'intégration à ce niveau là du développement afin de nous assurer de la bonne intégration de la simulation avec la station au sol, donc que le lancement de la mission en simulation se fait au bon moment lorsque demandé, que les données s'envoient correctement des robots vers l'interface, que les logs de débogage sont reçus et que la missions cesse lorsque demandé par l'utilisateur. L'envoi de ces informations se fait à travers un Master node dont la procédure de tests sera détaillée plus bas. Pour la génération automatique de l'environnement, nous avons également donné des paramètres extrêmes (ex: création de 400 murs) et vérifié qu'en aucun cas les contraintes n'étaient pas respectées.

2. Fonctionnement des robots

Le bon fonctionnement des robots est testé de façon très similaire à la simulation. Après avoir testé le bon fonctionnement de l'exploration en simulation (comme décrit plus haut), nous pouvons effectuer des tests sur les robots physiques en suivant la même procédure. On effectue d'abord des tests de systèmes et de performance en faisant une simple connexion manuelle par SSH et en relançant des missions à plusieurs reprises, tout en observant le comportement dans un environnement réel, ce qui nous permet d'identifier d'éventuels bogues ou soucis de fonctionnement. Ceci nous permet aussi de calibrer les paramètres des robots afin que l'exploration autonome respecte les requis. Entre autres, cela voulait dire vérifier manuellement les algorithmes de « path planning » pour voir si le chemin emprunté correspondait à celui déterminé par l'équipe comme optimal et s'assurer que les paramètres concordent avec le robot physique utilisé. Les tests incluent également une part de vérification des contraintes temporelles. Entre autres, l'odométrie est vérifiée en faisant approcher le robot d'objets statiques et en vérifiant que l'odométrie correspond au déplacement réel ainsi qu'en vérifiant avec le lidar que la vitesse de déplacement est logique. Les "costmap" utilisées par move_base sont aussi superposés aux différentes mesures et à la carte générée par cartographer pour vérifier qu'aucune variation majeure

n’est présente. Finalement, pour la génération de la carte, l’équipe déplace manuellement le robot avec des mouvements brusques ou des vas-et-viens et on vérifie que la carte converge tout de même assez rapidement. Encore une fois, on peut par la suite intégrer la logique de lancement de missions sur des robots physiques à la station au sol en effectuant la connexion directement via la station et en passant par le Master node pour l’envoi d’informations.

3. Ros nodes

Les nodes sont au cœur des communications entre les robots et la station au sol. ROS offre une suite de tests permettant de tester les communications entre les nodes. Cependant, la mise en place de cette suite est chronophage pour les besoins de notre projet. Nous avons opté pour une vérification procédurale à l’aide de tests d’intégrations. D’abord, nous connaissons quelles entités interagissent avec quelles nodes. Nous savons alors à quelles entités sont associées les nodes. ROS offre les programmes «rostopic» et «rosnode» qui permettent respectivement de lister les nodes connectées au master node et qui sont les publishers/subscribers. Une à une, on teste les entités (Master, Server et Limos) et on s’assure que les bonnes nodes sont apparues dans le réseau de nodes. On teste finalement avec toute la solution en place si l’ensemble des publishers/subscribers sont abonnés correctement.

4. Tests utilisateurs

La suite de tests utilisateur a comme objectif de permettre la vérification du bon fonctionnement des différentes parties du code ROS et des scripts Python compris dans le node «master».

Du côté de Python, on compte: controller, identify, map-remap et spawn-walls.

Du côté des fichiers ROS, on a les «.launch» suivants: cartographer, limo_control, limo_real, limo_sim, master, real et sim.

4.0 Mise en place

Afin de tester le fonctionnement des différents fichiers, il faut mettre en place la solution adéquatement.

4.0.1 Tester en simulation

Pour tester adéquatement les fonctionnalités en simulation, assurez-vous de faire «make» dans le répertoire principal de la solution. Une fois la solution prête, vous pouvez partir une simulation (bascule à l'état de simulation et appuyez sur «Lancer la mission».

Note: Vous n'avez pas besoin d'avoir de Limos à votre disposition.

4.0.2 Tester avec les Limos

Afin de tester avec les Limos, vous devez d'abord vous assurez qu'ils sont allumés et qu'ils contiennent l'image du conteneur «docker» de la solution. Autrement, assurez-vous de compléter les étapes du «README.md» disponible dans le répertoire principal de la solution.

Une fois les limos prêts, vous n'aurez qu'à faire «make» dans le répertoire principal de la solution. Une fois la solution prête, vous pouvez partir une mission réelle (bascule à l'état de réelle et appuyez sur «Lancer la mission».

4.0.2.1 Vérifier la connexion des Limos

Si jamais vous êtes en train de tester et que les Limos ne semblent pas répondre correctement, ils sont peut-être déconnectés.

Afin de vérifier la connexion, vous pouvez utiliser la commande suivante pour chacun des robots:

```
ping <ip du robot>
```

4.0.3 Commandes utiles

Certaines commandes seront utilisées afin de vérifier différents aspects. Assurez-vous que le «master» roule, sinon rien ne sera détecté.

```
1. roswtf # Vérifier les configurations ROS

2. rostopic list # Faire la liste des topics disponibles

3. rosnode list # Faire la liste des nodes actifs

4. rosnode info <nom du node> # Informations sur les publishers et
   subscribers du node
5. rostopic echo <nom du node>

6. rostopic pub /limoX/limo_status <tab> <tab>
```

4.1 Master

À partir du répertoire principal, effectuez les commandes suivantes:

```
cd master
make
```

Le «master» devrait partir sans erreur. Si une erreur est générée, elle sera dans un des fichiers de lancement ou un des scripts Python.

Si le Master part sans erreur, vous pouvez passer aux prochaines sections de test.

Exécutez la commande 1 pour vérifier l'existence d'anomalie dans l'infrastructure ROS.

4.2 Scripts Python

4.2.1 controller

Le controller doit se connecter à l'ensemble des topics suivants:

- /mission_cmd
- /robot_position
- /robot_state
- /security_zone
- /low_battery
- /limoX/identify
- /limoX/explorelite
- /limoX/move_base
- /limoX/base_link
- /limoX/odom
- /limoX/limo_status

Afin de vérifier les connexions, effectuez la commande no.2.

Pour chacun des topics, on doit utiliser la commande 5 et effectuer la commande. Observer ce qui transige sur le topic. Vous devriez voir de l'information apparaître:

- /mission_cmd: Lancer une mission
- /robot_position: Lancer une mission
- /robot_state: Lancer une mission
- /security_zone: Lancer une mission
- /low_battery: Envoyer un pourcentage sous 30% à /limoX/limo_status
- /limoX/identify: Envoyer "identify" à /mission_cmd
- /limoX/explorelite: Lancer une mission
- /limoX/move_base: Lancer une mission et faites la commande «Retour à la base»
- /limoX/base_link: Lancer une mission
- /limoX/odom: Lancer une mission
- /limoX/limo_status: Lancer une mission

Notes: Si vous n'avez pas de Limo, les topics /limoX/* n'apparaîtront pas.

4.2.2 identify

Note: Assurez-vous de tester ceci avec une simulation et une mission réelle

Le fichier «identify» est exécuté directement sur les limos, mais est tout de même fonctionnel en simulation.

1. Vérifiez l'existence du node avec la commande 3:
    - /limoX/identify

2. Vérifier à l'aide de la commande 4 que *identify* s'abonne au topic suivant:
    - /limoX//identify

node: /limoX/identify
subscribe: /limoX/identify

4.2.3 map-remap

Note: Assurez-vous de tester ceci avec une simulation et une mission réelle

1. Toujours avec la commande 2, vérifiez l'existence des topics suivants:
    - /map
    - /map_updates
    - /remap_updates

2. Vérifiez l'existence du node avec la commande 3:
    - map_remap_node

3. Vérifiez avec la commande 4 que *map_remap_node* publie sur le topic suivant:
    - /remap_updates

4. Vérifiez avec la commande 4 que *map_remap_node* s'abonne aux topics suivants:
    - /map
    - /map_updates
    - /security_zone

5. Une fois l'ensemble des connexions vérifiées, essayer de faire une zone de sécurité à partir de l'interface client.

Espionnez à l'aide de la commande 5 les topics suivants:

- /map
- /map_updates
- /security_zone

Vous devriez avoir des informations qui sont publiées sur ces topics.

4.2.4 spawn-walls

Ce module n'est utile qu'en simulation. Il communique directement avec Gazebo. Vous devez partir d'une simulation.

1. Vérifiez l'existence du node suivant:
    - spawn_walls

4.3 Fichiers de lancement «.launch»

Les fichiers de lancement sont appelés lors d'événements spécifiques et génèrent certains nodes. Il faudra tester chacun de ces événements.

4.3.1 cartographer

Le module *cartographer* est appelé à chaque lancement de mission. Il faut alors tester pour une simulation et une mission réelle:

1. Vérifiez l'existence des nodes suivants avec la commande 3:
    - cartographer_server
    - cartographer_occupancy_grid_node

4.3.2 limo_control

Le module *limo_control* est appelé à chaque lancement de mission. Il faut alors tester pour une simulation et une mission réelle:

1. Vérifiez l'existence des nodes suivants avec la commande 3:
    - cartographer_node
    - robot_pose_ekf
    - movebase
    - map_remap
    - explorelite

4.3.3 limo_real

Le module *limo_real* est appelé à chaque lancement de mission réelle. Il faut alors tester pour une mission réelle:

1. Vérifiez l'existence des nodes suivants avec la commande 3:
    - identify
    - ydlidar_lidar_publisher
    - base_link_to_laser_link
    - base_link_to_imu_link

2. Vérifier l'existence des topics suivants à l'aide de la commande:
    - cmd_vel
    - limo_status
    - imu
    - odom

4.3.4 limo_sim

Le module *limo_sim* est appelé à chaque lancement d'une simulation. Il faut alors tester pour une simulation:

1. Vérifiez l'existence des nodes suivants avec la commande 3:
    - spawn_limo_model
    - controller_spawner
    - robot_state_publisher
    - identify

4.3.5 master

Le module *master* est appelé à chaque lancement de mission. Il faut alors tester pour une simulation et une mission réelle:

1. Vérifiez l'existence du node suivant avec la commande 3:
    - master

4.3.6 real

Le module *real* est appelé à chaque lancement de mission réelle. Il faut alors tester pour une mission réelle:

1. Vérifiez l'existence du node suivant avec la commande 3:
    - rviz

4.3.7 sim

Le module *sim* est appelé à chaque lancement d'une simulation. Il faut alors tester pour une simulation:

1. Vérifiez l'existence du node suivant avec la commande 3:
    - rviz
    - spawn_walls

5. Tests de comportement

Une fois l'ensemble des relations vérifiées avec ROS, on peut procéder à la vérification des fonctionnalités. Ces vérifications se divisent en mission réelle et en simulation.

5.1 Simulation

Pour les tests suivants, partez une simulation à partir de l'interface web.

5.1.1 Lancement de la mission

Une fois que vous avez appuyé sur «Lancer la mission»:

1. Gazebo et Rviz devraient partir immédiatement.
2. Après quelques minutes, Gazebo devrait être prêt.
3. Un message devrait apparaître dans les logs.
4. Vous devez appuyer sur la touche "Espace" dans Gazebo pour amorcer la mission.

5.1.2 Arrêt de la mission

Après avoir appuyé sur "Terminer la mission":

1. Gazebo et Rviz devraient rapidement fermer.
2. La mission devrait être disponible dans l'historique.
3. Les différentes interfaces clients ouvertes devraient toutes afficher que la mission est terminée.
4. Un message devrait apparaître dans les logs.

5.1.3 Identifier

Après avoir lancé une mission, vous appuyez sur *identifier* pour un des robots:

1. Le robot en question devrait jouer une mélodie pour indiquer que vous souhaitez l'identifier.
2. Un message devrait apparaître dans les logs.

5.1.4 Ajout de la zone de sécurité

Après avoir lancé une mission, vous ajoutez une zone de sécurité:

1. Le robot se dirige vers la zone de sécurité spécifiée.
2. Le robot explore uniquement la zone de sécurité sans en sortir.

5.1.5 Retrait de la zone de sécurité

Après avoir lancé une mission et ajouté une zone de sécurité, vous la retirez:

1. La zone est retirée et le robot retourne à l'exploration habituelle.

5.1.6 Retour à la base

Après avoir lancé une mission, vous appuyer sur «Retour à la base»:

1. Le robot se redirige vers sa position d'origine.
2. Le robot s'arrête à une distance de moins de 0.5 mètres de sa position d'origine.
3. Un message devrait apparaître dans les logs.

5.1.7 Évitement des obstacles et exploration autonome

Après avoir lancé une mission:

1. Le ou les robots devraient explorer de façon autonome leur environnement.
2. Les robots devrait s'éviter entre eux, ainsi qu'éviter les obstacles

5.1.8 Position du robot dans la carte

Après avoir lancé une mission:

1. La position des robots devrait être la même dans Rviz quand dans la carte de l'interface client.
2. Les positions devraient être affichées à chaque seconde dans les logs de la mission.

5.1.9 Logs de mission

Après avoir lancé une mission:

1. L'onglet des logs devrait être fermé par défaut.

2. En cliquant sur l'onglet, celui-ci devrait s'ouvrir et afficher les logs envoyés depuis l'ouverture de l'onglet mission de la mission en cours.

3. Les logs de positions devraient être affichés à chaque seconde et les logs d'événements devraient être affichés pour chaque événement (Lancer, Terminer, Retour à la base, Batterie faible, Zone de sécurité).

Après avoir terminé une mission:

5.1.10 Historique des missions

Après avoir ouvert l'interface client, vous pouvez ouvrir l'onglet d'historique des missions:

1. L'ensemble des missions contenues dans la base de données devraient être affichées.

2. En cliquant sur une mission, une boîte devrait s'ouvrir et afficher les logs de la mission en question ainsi que la carte. Les logs devraient être contenus dans une boîte avec défilement.

Lors de l'exécution d'une mission:

1. La mission en cours ne devrait pas être affichée.

Après avoir terminé une mission:

1. La mission qui vient d'être terminée devrait être disponible sur l'ensemble des interfaces client. Le type de mission devrait afficher "Simulation".

5.2 Mission réelle

Pour les tests suivants, partez une mission réelle à partir de l'interface web. assurez-vous que les robots sont connectés au même réseau wifi que la solution.

5.2.1 Lancement de la mission

Une fois que vous avez appuyé sur «Lancer la mission»:

1. Rviz devrait partir immédiatement.

2. Un message devrait apparaître dans les logs.

3. Les robots devraient commencer leur exploration quelque temps après.

5.2.2 Arrêt de la mission

Après avoir appuyé sur "Terminer la mission":

1. Rviz devrait rapidement fermer.

2. La mission devrait être disponible dans l'historique.

3. Les différentes interfaces clients ouvertes devraient toutes afficher que la mission est terminée.

4. Un message devrait apparaître dans les logs.

5. Les robots s'arrêtent où ils sont.

5.2.3 Identifier

Après avoir lancé une mission, vous appuyez sur *identifier* pour un des robots:

1. Le robot en question devrait jouer une mélodie pour indiquer que vous souhaitez l'identifier.

2. Un message devrait apparaître dans les logs.

5.2.4 Ajout de la zone de sécurité

Après avoir lancé une mission, vous ajoutez une zone de sécurité:

1. Le robot se dirige vers la zone de sécurité spécifiée.

2. Le robot explore uniquement la zone de sécurité sans en sortir.

5.1.5 Retrait de la zone de sécurité

Après avoir lancé une mission et ajouté une zone de sécurité, vous la retirez:

1. La zone est retirée et le robot retourne à l'exploration habituelle.

5.2.6 Retour à la base

Après avoir lancé une mission, vous appuyer sur «Retour à la base»:

1. Le robot se redirige vers sa position d'origine.

2. Le robot s'arrête à une distance de moins de 0.5 mètres de sa position d'origine

3. Un message devrait apparaître dans les logs.

5.2.7 Évitement des obstacles et exploration autonome

Après avoir lancé une mission:

1. Le ou les robots devraient explorer de façon autonome leur environnement.

2. Les robots devrait s'éviter entre eux, ainsi qu'éviter les obstacles

5.2.8 Position du robot dans la carte

Après avoir lancé une mission:

1. La position affichée des robots devrait ressembler à ce qu'on observe en réalité.

2. Les positions devraient être affichées à chaque seconde dans les logs de la mission.

5.2.9 Piles

Après avoir lancé une mission:

1. Le niveau de pile de chaque robot devrait être affiché dans leur section respective.

2. Lorsque le niveau de pile est inférieur à 30%, le robot concerné devrait activer le retour à la base de façon autonome.

Note: le niveau de pile varie avec les déplacements. C'est normal de le voir fluctuer.

5.2.10 Logs de mission

Après avoir lancé une mission:

1. L'onglet des logs devrait être fermé par défaut.

2. En cliquant sur l'onglet, celui-ci devrait s'ouvrir et afficher les logs envoyés depuis l'ouverture de l'onglet mission de la mission en cours.

3. Les logs de positions devraient être affichés à chaque seconde et les logs d'événements devraient être affichés pour chaque événement (Lancer, Terminer, Retour à la base, Batterie faible, Zone de sécurité).

5.2.11 Historique des missions

Après avoir ouvert l'interface client, vous pouvez ouvrir l'onglet d'historique des missions:

3. L'ensemble des missions contenues dans la base de données devraient être affichées.

4. En cliquant sur une mission, une boîte devrait s'ouvrir et afficher les logs de la mission en question ainsi que la carte. Les logs devraient être contenus dans une boîte avec défilement.

Lors de l’exécution d’une mission:

2. La mission en cours ne devrait pas être affichée.

Après avoir terminé une mission:

2. La mission qui vient d’être terminée devrait être disponible sur l’ensemble des interfaces client. Le type de mission devrait afficher “Réel”.

5.3 Comportements généraux

5.3.1 Page d’accueil

Lors de l’ouverture du site:

1. Si aucune mission n’est en cours, la page d’accueil devrait avoir comme options de partir une nouvelle mission ou d’accéder à l’historique des missions.

2. Si une mission est en cours, elle devrait proposer de rejoindre la mission en cours ou d’accéder à l’historique des missions.

5.3.2 Barre de navigation

La barre de navigation devrait être disponible en tout temps sur le site.

1. Vous devriez être redirigés vers la page d’accueil lorsque vous cliquez sur «Accueil».

2. Vous devriez être redirigés vers la page d’historique de mission lorsque vous cliquez sur «Historique de mission».