UV5.8 - Gazebo Arena

Joris TILLET - Rémi RIGAL

Décembre 2019

1 Introduction

L'objectif de ce TP est de mettre en place une architecture multi-machines avec ROS sous la forme d'une course de robots en aveugle. Un serveur va héberger une instance de Gazebo et faire la simulation pour tous les robots qui seront ensemble sur le circuit. L'objectif de ce TP est de configurer ROS pour communiquer avec le serveur, faire apparaître son robot dans la simulation et suivre la piste jusqu'à la ligne d'arrivée en un temps minimum.

2 Spécifications

Afin que la simulation de plusieurs robots en même temps fonctionne bien il faut s'assurer que chaque robot peut être contrôlé indépendamment des autres et que les informations de ses capteurs sont bien publiées sur des *topics* indépendants.

Très important: le cahier des charges qui suit doit être intégralement respecté avant d'envoyer son robot dans le simulateur.

2.1 La notion de *namespace*

ROS offre la possibilité d'utiliser un *namespace* dans les *topics* en utilisant des "/" comme dans une arborescence de fichiers. Par exemple, le topic */cmd_vel* devient */ensta/cmd_vel* lorsque l'on utilise le *namespace* **ensta**.

Pour ce TP, vous devez utiliser comme *namespace* votre identifiant de connexion ENSTA (généralement composé des premières lettres de votre nom suivi des premières lettres de votre prénom) qui a l'avantage d'être unique et composé de caractères alpha-numériques.

2.2 Le robot

Le robot doit vérifier les critères suivants:

- Il doit rentrer dans une boîte de dimension 70cm x 1m x 1m (largeur, longueur, hauteur).
- Le robot ne doit pas pouvoir voler.
- L'axe des X positifs doit correspondre à la direction d'avancement du robot.
- Le robot doit posséder un Lidar à 360° avec au plus 360 *samples* et une fréquence maximale de 5Hz.
- Tout autre capteur est interdit.
- Le robot doit posséder un *link* racine de telle sorte que lorsque ce dernier est positionné à (0, 0, 0) le robot est "posé" sur le sol et centré.

Si le robot utilise le plugin *libgazebo_ros_diff_drive.so* il est nécessaire de le remplacer par le plugin *libgazebo_ros_ensta_diff_drive.so* qui se trouve dans le package *gazebo_arena_client* disponible sur Moodle, une simple compilation de ce package suffit pour pouvoir l'utiliser.

2.3 Configuration réseau

Le serveur de simulation sera mis à disposition, il ne faut donc pas lancer une instance de Gazebo mais simplement vous y connecter. Pour cela, vous devrez être sur le même réseau que le serveur (celui de l'école), et il est nécessaire de bien configurer ROS et notamment les deux variables d'environnement suivantes:

- *ROS_MASTER_URI*: HTTP://172.20.12.78:11311
- *ROS_HOSTNAME*: MY_IP_ADDRESS (pas localhost, en 172.20.X.X)

2.4 Checklist

Une fois que les spécifications précédentes ont été validées, vérifiez un par un les points suivants:

- Le topic de commande du robot est */ns/cmd_vel*
- Le robot publie les données du Lidar dans le topic */ns/scan*
- Le robot s'affiche correctement sur Rviz

Gloabalement, tout ce qui concerne directement votre robot soit être sous le bon *namespace*.

3 Utiliser son robot dans la simulation

Pour utiliser son robot dans la simulation il faut créer un fichier launch avec les nœuds suivants:

- Robot State Publisher
- Model Spawner (du package *gazebo_arena_client* à télécharger sur Moodle)
- Rviz (Optionnel)

Tous ces noeuds doivent être inclus dans un élément *group* ayant l'attribut *ns* assigné, vous pouvez vous inspirer du fichier launch ci-après:

```
<launch>

    <!-- Remplacer 'ensta' par votre namespace -->
    <arg name="ns" default="ensta"/>

    <group ns="$(arg ns)">
        <!-- Param indispensable pour conserver l'intégrité du
        ↪  robot avec des namespaces -->
        <param name="tf_prefix" value="$(arg ns)"/>

        <!-- TODO: Attribuer le param 'robot_description' à la
        ↪  description URDF/Xacro de votre robot -->
        <param name="robot_description" command="TODO"/>

        <node name="$(arg ns)_robot_state_publisher"
        ↪  pkg="robot_state_publisher"
        ↪  type="robot_state_publisher" />

        <node name="$(arg ns)_spawn_urdf"
        ↪  pkg="gazebo_arena_client" type="spawn_model.py"
        ↪  output="screen" args="-urdf -model $(arg ns)_robot
        ↪  -param robot_description -ns $(arg ns)" />
    </group>

</launch>
```

4 La course

Le tracé exact du circuit n'étant pas connu, vous pouvez compter uniquement sur votre Lidar pour terminer la course. L'objectif ici est de mettre en place un noeud ROS pour contrôler automatiquement le robot et le faire terminer la course le plus rapidement possible.

Les seules contraintes pour le noeud de contrôle sont les *topics* d'entrée et de sortie:
Entrée: */ns/scan* (sensor_msgs/LaserScan)
Sortie: */ns/cmd_vel* (geometry_msgs/Twist)
L'utilisation de package(s) de mapping, déplacement etc... est interdite. En revanche il n'y a aucune contrainte sur la stratégie de guidage à adopter.

Let's race !!!!!