DSM ROS - Février 2024

Romain Marie

2023-2024

3iL INGENIEURS
</les sciences informatiques>

Dans cet examen, vous allez être évalué sur votre capacité à élaborer, analyser, compléter et utiliser un écosystème ROS. Votre note sera basée sur deux éléments que vous devez absolument rendre sur Moodle sous la forme d'une archive en fin d'examen :

- Le répertoire du package dans lequel vous allez travailler, incluant l'ensemble des fichiers sources et de configuration,
- Un compte-rendu word dans lequel vous recenserez les différentes réponses et captures d'écran réclamées au fil de l'examen. Vous prendrez à chaque fois le soin de préciser la question à laquelle vous répondrez. **Les questions nécessitant une action dans le compte-rendu sont marquées d'une étoile *. Il s'agira souvent d'y transposer les commandes utilisées pour répondre à la question**.

Le barême est donné à titre indicatif, et est susceptible de changer.

1 Questions de cours /3

1. * **/1.5** Expliquez en quelques lignes l'articulation entre workspace, package, noeuds, fichiers launch et codes sources dans un écosystème ROS.
2. * **/1.5** Donnez une description succincte du rôle de chacune des trois fonctions suivantes :
   - ros::ok();
   - srv.call();
   - loop.sleep();

2 Compréhension et mise en place de l'existant /2.5

1. Dans un terminal, lancez l'environnement de simulation en utilisant la commande **roslaunch turtlebot3_gazebo turtlebot3_world.launch**
2. * **/0.5** Débrouillez-vous pour incorporer à votre workspace les fichiers récupérés sur Moodle **en respectant les conventions ROS**. Vous incluerez à votre compte-rendu un descriptif de la structuration que vous avez choisi.

3. * /**1** Pour chacun des deux noeuds fournis dans l'archive, expliquez en quelques mots/lignes ce que vous en comprenez.
4. * /**0.5** Expliquez pourquoi le noeud **tourner** a besoin du topic /**odom** pour correctement fonctionner.
5. * /**0.5** Expliquez pourquoi le plugin **robot_steering** de **rqt** ne peut pas fonctionner correctement en même temps que le noeud **tourner**

3 Utilisation de l'existant /3.5

1. * /**0.5** Après avoir compilé, lancez le noeud **tourner**. Vous préciserez les commandes
2. * /**0.5** Dans un terminal, affichez la fréquence à laquelle le noeud publie sur le topic /**cmd_vel**.
3. * /**1** Débrouillez-vous pour que votre robot effectue une rotation de 45° dans le sens anti-horaire. Vous inclurez dans le compte-rendu un descriptif de la stratégie utilisée.
4. * /**1.5** Utilisez rviz pour visualiser la nappe laser. Après avoir décrit les différentes étapes permettant d'obtenir ce résultat, expliquez pourquoi il n'est pas possible d'y afficher en même temps la position du robot ?

4 Mise à jour de l'existant /4.5

1. /**1.5** Modifiez le noeud **detection_obstacles** pour qu'il informe le réseau ROS de la distance à l'obstacle le plus proche.
2. /**1.5** Modifiez le noeud **tourner** afin qu'il prévienne le réseau ROS lorsqu'il est en train de faire tourner le robot. Vous utiliserez obligatoirement l'include **std_msgs/Bool.h** de la façon qui vous semble convenir.
3. /**1.5** Débrouillez-vous pour que le noeud **detection_obstacles** publie le topic **mystere** uniquement lorsque le robot n'est pas en train de tourner.

5 Création d'un noeud /7.5

1. /**1** Mettez en place un nouveau noeud **gestion_rotation** en le dotant d'un code ROS minimal, et en mettant à jour le fichier CMakeLists.txt
2. * /**1.5** Incorporez à votre package un fichier launch permettant de lancer les 3 noeuds en même temps, et utilisez-le (à incorporer dans votre CR).
3. /**2.5** Complétez votre noeud afin qu'il demande au noeud **tourner** d'orienter le robot dans la direction de l'obstacle le plus proche.
4. /**2.5** Débrouillez-vous pour que, une fois la rotation finie le robot recule jusqu'à être à 1m de l'obstacle derrière lui.