Utiliser les services du robot:

Après s'être connecté sur le robot

rosservice list | grep play_motion permet d'afficher les services de mouvements pré-enregistrés

Création d'un fichier de message

cd beginner_tutorials
mkdir msg
cd msg
echo "int64 num" > ./Num.msg

Les messages sont des fichiers texte avec un type de champ et un nom de champ par ligne

Dans src/beginner_tutorial/package.xml décommenter
<build_depend>message_generation</build_depend>
<run_depend>message_runtime</run_depend>

ajouter dans src/beginner_tutorials/CMakeLists.txt (cf. pdf)

04/10/2022 On ne modifie jamais le CMakeLists de src !

Ordinateur 192 Vérifier si l'environnement est correct
↳ on ouvre le .bashrc contenu dans le root et tout commenter sauf le chemin correspondant à notre workspace

- Crée un dossier srv dans beginner_tutorials
Taper roscp rospy_tutorials AddTwoInts.srv srv/AddTwoInts.srv

Dans le CMakeList.txt de src ajouter
add_service_files

(FILES
  AddTwoInts.srv
)

Faire catkin_make à la racine du workspace
⚠ build, devel et install sont toujours à la racine du workspace

Ecrire des nodes

(regarder nous même pour c++ dans le pdf)
- pour ROS 2

Python : Créer dans beginner_tutorials un dossier :

```
mkdir scripts
cd scripts
```

Télécharger le script talker.py :

```
wget https://raw.github.com/ros/ros_tutorials/kinetic-devel/rospy_tutorials/001_talker_listener/talker.py
```

Le rendre exécutable : `chmod +x talker.py`

Dans le code source :

```
import rospy   permet d'accéder au module python pour ROS
pub = rospy.Publisher("chatter", String, queue_size=10)
```

- "chatter" : nom du message
- String : type du topic
- queue_size=10 : taille de la file d'attente

```
pub.publish(hello_str)   publie l'objet pub
```

Télécharger le script listener.py et le rendre exécutable
↳ Vérifier avec `ls -al` pour afficher les droits.

```
rospy.init_node('listener', anonymous=True)
```

crée le node listener et anonymous ajoute un chiffre random pour éviter les nodes de même nom

Lancer le roscore

Lancer `rosrun beginner_tutorial talker.py`

- beginner_tutorial : nom du package
- talker.py : nom du node

Distribution des messages :

1. Se connecter sur le réseau du robot
2. export ROS_MASTER_URI = http://10.68.0.1:11311
3. /sbin/ifconfig pour noter inet 10.68.0.134
4. export ROS_IP=10.68.0.134 (le réseau physique de notre ordinateur)

On peut envoyer sur le réseau et être entendu :
ros run beginner_tutorial talker.py

Dans un autre terminal refaire les étapes avec listener pour écouter
Ici on utilise le roscore du robot (celui dans le ROS_MASTER_URI)
Pour faire communiquer 2 robots avec des roscores différents il faut un bridge se connectant aux deux réseaux ou on s'en affranchit avec ROS 2

Stack de navigation robot : (dans le handbook du robot)

Pour récupérer le handbook sur le réseau du robot

scp pal@10.68.0.138 : /Desktop/TIAGO*ferrum.pdf .

- scp : copie ssh
- pal@10.68.0.138 : utilisateur pal à l'adresse 10.68.0.138
- /Desktop/TIAGO*ferrum.pdf : emplacement à copier ; finissant par ferrum
- . : à l'emplacement actuel où copier

Aller sur le chapitre 30 Navigation

30.3 Navigation à relire

30.4 SLAM and path planning à relire

1. Connexion au robot (ROS_MASTER_URI et ROS_IP)
2. rosservice call pal_navigation_sm "input_map"
3. Naviguer avec le joystick rostopic echo joystick (priority data doit être à True appuyer sur le start du joystick sinon)

④ rosrun map_server map_saver pour sauvegarder la map
⑤ Enregistre la map en local rosservice call /pal_map_manager/save_map " directory: '' "
⑥ Déverrouiller la localisation rosservice call /pal_navigation_sm " input: 'LOC' " et désactiver le joystick (bouton start)
⑦ Grâce à rviz rosrun rviz rviz -d `rospack find tiago_2d_nav`/config/rviz/navigation.rviz "
et placer un 2D Nav Goal sur l'interface logicielle
=> le robot se déplace vers cette position

Navigation dans une simulation (en local) :

① Se connecter en local et sourcer avec source /opt/pal/ferrum/setup.bash
② Ne pas lancer de roscore sur la machine car nous sommes connecté au robot virtuel
③ Lancer la simulation roslaunch tiago_0_gazebo tiago_mapping.launch
④ rosrun key_teleop key_teleop.py
⑤ Naviguer avec les flèches
⑥ Sauvegarder la map rosrun map_server map_saver
⑦ Enregistrer la carte en local rosservice call
⑧ Arrêter la localisation rosservice call /pal_navigation_sm "input: 'la'"
⑨ Placer le goal sur rviz pour déplacer le robot