ROS TurtleBot

AdresseIP PC =192.168.0.10
AdresseIP Turtle = 192.168.0.202

Sur notre PC lancer un terminal :
ssh turtulbot@192.168.0.202 (en filaire)
mdp : turtlebot

To do on PC and turtlebot :
cp ~/.bashrc ~/.bashrc.sav

On turtlebot :
echo export ROS_MASTER_URI=http://192.168.0.202:11311 >> ~/.bashrc
echo export ROS_HOSTNAME=192.168.0.202 >> ~/.bashrc
source ~/.bashrc (pour mettre à jour)

On PC :
echo export ROS_MASTER_URI=http://192.168.0.202:11311 >> ~/.bashrc
echo export ROS_HOSTNAME=192.168.0.10 >> ~/.bashrc
source ~/.bashrc

Ensuite faire “roscd” sur le PC puis lancer “roscore” sur turtlebot et sur le pc “rostopic list”
Si problème de ROS Master unreachable faire echo $ROS_MASTER_URI et vérifier que c’est bien l’adresse du turtulebot. Sinon vérifier dans le ~/.bashrc

Sur turtleBot :
cd ~/turtlebot2i
source devel/setup.bash (vérifier qu’il n’y a pas de export avec master_uri si probleme d’ip)
roslaunch turtlebot2i_bringup minimal.launch

Toujours sur turtulebot dans un autre terminal:
rostopic list
rostopic echo diagnostics

=> on obtient le retour du turtlebot qui envoie des donnée mais les capteurs ne semble pas lancé => c’est normal on a fait un minimal.launch, il faut faire un 3dsensor.launch
roslaunch turtlebot2i_bringup 3dsensor.launch

Création d’un workspace ROS

Sur le PC Exterieur sur le `Bureau` (`/home/student/Bureau`)

```
cd /home/student/Bureau
mkdir - p battandier_bonucci_ws/src
cd attandier_bonucci_ws/src
catkin_init_workspace
git clone https://github.com/yujinrobot/kobuki_msgs.git
catkin_make
```

```
catkin_create_pkg collision std_msgs rospy roscpp kabuki_msgs
```

Creer le fichier collision_warning.py avec les droit d'execution
Copier ce code dans collision_warning.py dans le dossier src du package collision:
https://answers.ros.org/question/244837/reading-bump-sensors-on-simulated-turtlebot/
ne pas oublier de rajouter au début :
```
#!/usr/bin/env python
```

ensuite faire un catkin_make sur le pc dans notre ws
puis lancer : rosrun collision src/collision_warning.py

On veut accéder au publisher des info du Scan du Laser, pour cela en faisant un 'rostopic list' on voit bien qu'il y a un topic "/scan" et si l'on fait un 'rostopic type /scan' on voit que son type est un sensor_msg 'sensor_msgs/LaserScan'

NAVIGATION

Odométrie : Technique permettant d'estimer la position d'un robot en mouvement. On utilise les capteurs de mouvement pour estimer un changement de position en temps réel par rapport à un point de référence (point de départ).
L'odométrie n'est pas suffisante pour naviguer car elle ne permet pas de déterminer la position exacte mais de l'estimer. De plus, elle ne permet pas de détecter les objets avoisinants pouvant obstruer son trajet. La localisation est le processus qui détermine où se trouve le robot dans une carte connue, afin de se repérer sur cette carte.
Un robot peut se localiser en utilisant l'odométrie ET les données du Scan laser.

Qu'est ce qu'AMCl ? (Adaptative Monte Carlo Localisation) Que permet-il de réaliser ?
La localisation de Monte Carlo est un algo permettant au robot de se localiser à l'aide de l'odométrie et du scan combiné. Il permet de réaliser une carte des localisation possible du robot grâce aux positions des particules (nuage de points) sur la carte

Pour lancer le turtlebot2i en mode navigation :
sur le turtlebot arrêter les launch en cours (si nécessaire)
ensuite faire se placer dans "cd ~turtlebot2i",
ne pas oublier "source devel/setup.bash"
puis : roslaunch turtlebot2i_bringup turtlebot2i1.launch

Il faut ensuite lancer la visualisation sur le PC externe :
roslaunch turtlebot2i_bringup remote_view.launch

Enfin si l'on veut déplacer le robot pour la map, ouvrir un terminal sur le turtlebot :
cd ~/turtlebot2i
source devel/setup.bash
roslaunch turtlebot_teleop keyboard_teleop.launch