

1.1 Le workspace contient un dossier src qui contient les packages, les packages contiennent le cmakefile et package.xml optionnellement un fichier launch et les codes sources qui contiennent les noeuds. (je n'ai pas compris ce que vous entendez par "articulation")

1.2

ros::ok() vérifie si la console est encore disponible, si on ne l'a pas arrêté avec un ctrl+c

srv.call() est le service côté client qui appelle la fonction côté serveur

loop.sleep() dit à la boucle de s'arrêter et de se synchroniser avec le master

2.2

home/workspace/src/exam

2.3

tourner.cpp s'abonne à /odom de type nav_msgs::Odometry dans le but d'avoir un angle initiale, s'abonne à /amplitude de type std_msgs::Float32& dans le but de recevoir une commande et savoir de combien de radiant il doit tourner, et publie dans /cmd_vel de type geometry_msgs::Twist une vitesse de rotation jusqu'à ce qu'il tourne à l'angle fournit par /amplitude.

detection_obstacles.cpp s'abonne à /scan de type sensor_msgs::LaserScan pour avoir le tableau de distances d'obstacles qu'il contient, et publie dans /mystere de type std_msgs::Float32 l'angle dans lequel l'obstacle est le plus proche.

2.4 tourner a besoin de savoir l'angle où il est par rapport à un repère fixe (ici celui d'odom, qui est celui à l'initial du robot) pour savoir si il a assez tourner ou non par rapport à son but.

2.5 Dans le noeud tourner, il publie tout le temps un message dans /cmd_vel, qui est ce qui permet de contrôler le robot.

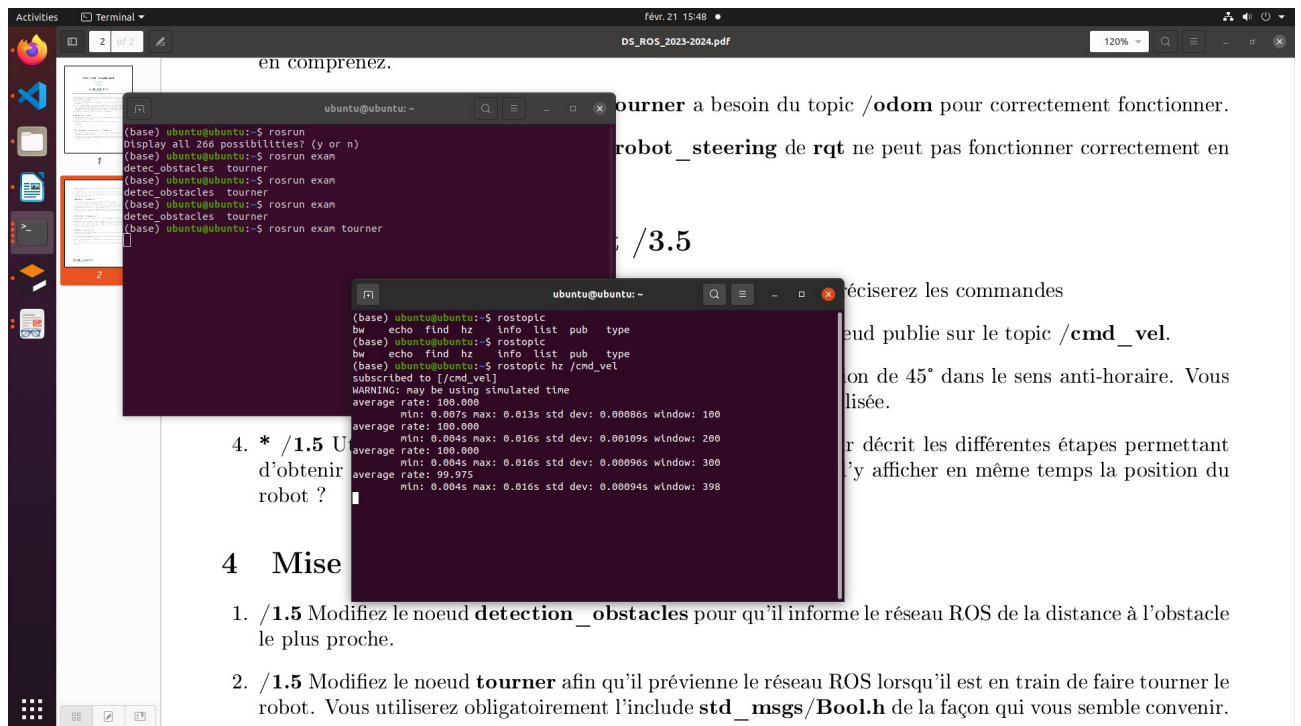
3.1

catkin_make

roslaunch roslaunch turtlebot3_gazebo turtlebot3_world.launch

roslaunch exam tourner

3.2



en comprenez.

robot a besoin du topic `/odom` pour correctement fonctionner.

robot_steering de rqt ne peut pas fonctionner correctement en

/3.5

Précisez les commandes

leud publie sur le topic `/cmd_vel`.

on de 45° dans le sens anti-horaire. Vous

lisée.

er décrit les différentes étapes permettant

y afficher en même temps la position du

```
(base) ubuntu@ubuntu:~$ roslaunch
Display all 266 possibilities? (y or n)
(base) ubuntu@ubuntu:~$ roslaunch exam
detec_obstacles tourner
(base) ubuntu@ubuntu:~$ roslaunch exam
detec_obstacles tourner
(base) ubuntu@ubuntu:~$ roslaunch exam
detec_obstacles tourner
(base) ubuntu@ubuntu:~$ roslaunch exam tourner
(base) ubuntu@ubuntu:~$ rostopic pub type
bw echo find hz info list pub type
(base) ubuntu@ubuntu:~$ rostopic
bw echo find hz info list pub type
(base) ubuntu@ubuntu:~$ rostopic hz /cmd_vel
subscribed to [/cmd_vel]
WARNING: may be using simulated time
average rate: 100.000
min: 0.007s max: 0.013s std dev: 0.00086s window: 100
average rate: 100.000
min: 0.004s max: 0.016s std dev: 0.00109s window: 200
average rate: 100.000
min: 0.004s max: 0.016s std dev: 0.00096s window: 300
average rate: 99.975
min: 0.004s max: 0.016s std dev: 0.00094s window: 398
```

4. * /1.5 U d'obtenir robot ?

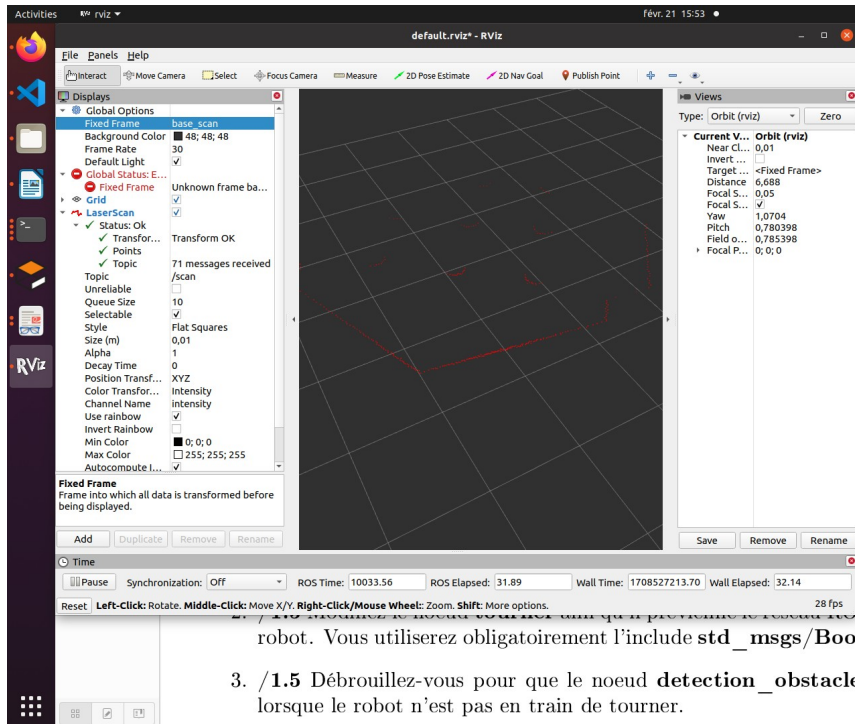
4 Mise

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.

3.3 rostopic pub /amplitude std_msgs/Float32 "0.785398163397448"

3.4 sur un terminal je lance rviz en faisant `rviz`
je clique sur add en bas à gauche
je vais dans l'obglet By topic
je vais sur le topic /scan et je sélectionne LaserScan

On ne peut pas visualiser la position du robot en même temps que le laser car ils n'utilisent pas le même Fixed Frame n'ont donc pas le même repère, l'un base_scan l'autre odom



The screenshot shows the RViz interface with the following details:

- Displays Panel:**
 - Global Options:** Fixed Frame: base_scan, Background Color: 48; 48; 48, Frame Rate: 30, Default Light: ✓.
 - Grid:** ✓
 - LaserScan:** Status: Ok, Transform OK, Points: 71 messages received /scan, Topic: /scan, Unreliable: -, Queue Size: 10, Selectable: ✓, Style: Flat Squares, Size (m): 0.01, Alpha: 1, Decay Time: 0, Position Transf...: XYZ, Color Transf...: Intensity, Channel Name: Use rainbow, Invert Rainbow: ☐, Min Color: 0; 0; 0, Max Color: 255; 255; 255, Autocompute I...: ✓.
- Views Panel:**
 - Type: Orbit (rviz)
 - Current View: Orbit (rviz)
 - Near CL...: 0.01
 - Invert...: ☐
 - Target...: <Fixed Frame>
 - Distance: 6.688
 - Focal S...: 0.05
 - Focal S...: ✓
 - Yaw: 1.0704
 - Pitch: 0.780398
 - Field o...: 0.785398
 - Focal P...: 0; 0; 0
- Time Panel:**
 - Pause: ☐ Synchronization: Off
 - ROS Time: 10033.56 ROS Elapsed: 31.89 Wall Time: 1708527213.70 Wall Elapsed: 32.14
 - Reset: Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

2. / 1.5 Déterminez le message et le type de données que publie 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.