ROB314 – Session 1 - Exo 2

Turtlesim

Configuration

You will use the *turtlesim* package, which should be already be installed with ROS.

You have to use 4 terminals. With *Terminator*, it is easier: you can split it into 4 terminals.

In each terminal, before using any ROS command, you have to execute the command :

```
> source /opt/ros/noetic/setup.bash
```

Or add this command to the end of the file *~/.bashrc*

In Terminal #1 – Starting the *roscore*

- Start a roscore with

```
> roscore
```

- Take a look at what's displayed.

In Terminal #2 – Starting the *turtlesim* node

- Run the turtlesim demo with

```
> rosrun turtlesim turtlesim_node
```

- This will start the *turtlesim_node* node of the *turtlesim* package.
- You should see the “TurtleSim” window
- Normally you should have **auto-completion** with ROS commands. For example,
  “rosrun tu” + TAB will give “rosrun turtle”
  “rosrun turtle” + TAB + TAB will give a list of possible packages
  “rosrun turtlesim” + TAB + TAB will give a list of possible nodes
  And so on...

In Terminal #3 – Analyze the *turtlesim* node

- See the list of active nodes

```
> rosnode list
```

- We find the *turtlesim* node in the list

- Show information about the *turtlesim* node

```
> rosnode info /turtlesim
```

- We see that the node /*turtlesim* has several possible publications, one subscription and several services

In Terminal #4 – Starting the *turtle_teleop_key* node

- Run the *turtle_teleop_key* demo node with:

```
> rosrun turtlesim turtle_teleop_key
```

- This node allows you to move and control the turtle in the window with the keyboard.
- You must be careful to click on the **terminal** (not on the “TurtleSim” window) before using the keyboard arrows !

In Terminal #3 – Analyze

- See the new *turtle_teleop_key* node with:

```
> rosnode list
```

- We have a new element */teleop_turtle*

- See the topics used here with:

```
> rostopic list
```

- When you launch *turtlesim*, the topics start with */turtle1* instead of */turtlesim* because each turtle in *turtlesim* is identified by a unique name. By default, the first turtle is named *turtle1*. This naming convention allows you to manage multiple turtles simultaneously, with each having its own set of topics for movement commands and position information.

- Show the connection of the nodes over the */turtle1/cmd_vel* topic (cmd_vel = command velocity) with:

```
> rostopic info /turtle1/cmd_vel
```

- We see the *publishers* of this topic: here the */teleop_turtle node*
- We see the *subscribers* of this topic: here the */turtlesim* node

- In ROS, the *cmd_vel* topic is commonly used to send movement commands to a robot. This topic accepts messages of type *geometry_msgs/Twist*, which contain information about the robot's linear and angular velocities.
  Here's a brief explanation of the components of *geometry_msgs/Twist*:

  Linear Velocity (linear):
  - x: Forward or backward speed.
  - y: Lateral speed (often unused for non-holonomic robots).
  - z: Vertical speed (rarely used for ground robots).

  Angular Velocity (angular):
  - x: Rotation around the x-axis.
  - y: Rotation around the y-axis.
  - z: Rotation around the z-axis (commonly used for turning left or right).

In Terminal #3 – rqt_graph

- The rqt_graph tool provides a visualization of the ROS computation graph. It is useful to understand what is happening in our ROS project.

```
> rqt_graph &
```

In Terminal #3 – Publish my own message from terminal

- For example, to make the turtle move forward at a speed of 0.2m/s, you can publish a *cmd_vel* message in the */turtle1/cmd_vel* topic:

```
> rostopic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 1.5}}'
```

- Check the result in the “TurtleSim” window.

- We can get the same result by specifying all the axis of velocity:

```
> rostopic pub /turtle1/cmd_vel geometry_msgs/Twist '{linear: {x: 1.5, y:
0, z: 0}, angular: {x: 0, y: 0, z: 0}}'
```

- Some of the messages like *cmd_vel* have a predefined timeout
- If you want to publish a message continuously use the **-r argument** with the loop rate in Hz

- For example, to make the turtle run continuously in a circle, type:

```
> rostopic pub /turtle1/cmd_vel -r 10 geometry_msgs/Twist '{linear: {x:
0.8}, angular: {z: 0.5}}'
```

In Terminal #3 – Spawn a new turtle

- You can spawn a new turtle by using a service */spawn* provided by the /*turtlesim* node: rosservice call /spawn [x] [y] [theta] [name]

```
> rosservice call /spawn 3.0 3.0 0.0 "Donatello"
```