PROGRAMMING FOR ROBOTS AND MANIPULATORS (VRM)

Faculty of Mechanical Engineering, Brno University of Technology
Institute of Automation and Computer Science
Technicka 2896/2, Brno 616 69, Czech Republic
Roman.Parak@vutbr.cz

ROS Melodic - Getting Started with Turtlesim

ROS (Robot Operating Systems):

ROS is an open-source, meta-operating system for your robot.It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building,writing, and running code across multiple computers.

Keywords:

A **node** is a process that performs computation. **Topics** are named buses over which nodes exchange **messages**. **Nodes** communicate with each other by publishing **messages** to **topics**. Request / reply is done via a **Service**, which is defined by a pair of **messages**: one for the request and one for the reply.

Link: http://wiki.ros.org/ROS

TurtleSim Example (Hello world in ROS):

```
$ roscore
$ rosrun turtlesim turtlesim_node

http://wiki.ros.org/turtlesim
```

Informations:

```
$ rosnode info /turtlesim
```

ROS Parameters:

```
$ rosparam
$ rosparam list
$ rosparam get turtlesim/background_b
$ rosparam set
```

ROS Service:

```
$ rosservice
$ rosservice list
$ rosservice call /turtle1/set_pen 255 0 0 5 0
$ rosservice type /turtle1/teleport_absolute
$ rosservice args /turtle1/teleport_absolute
$ rosservice call /turtle1/teleport_absolute 1 1 0
$ rosservice call /turtle1/teleport_relative 1 0
```

ROS Topic:

```
$ rostopic
$ rostopic list
$ rostopic info /turtle1/pose
$ rostopic type /turtle1/pose
$ rosmsg list
$ rosmsg show turtlesim/Pose
$ rostopic echo /turtle1/pose

Using the Linux shell we can combine two commands.
$ rostopic type /turtle1/cmd_vel | rosmsg show

This displays a verbose list of topics to publish to and subscribe to
and their type:
$ rostopic list -v

Make the Turtle move in a circle (Test Turtle Move):
$ rostopic type /turtle1/cmd_vel
$ rosmsg show geometry_msgs/Twist
$ rostopic type /turtle1/cmd_vel | rosmsg show
$ rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist --
'[2.0, 1.0, 0.0]' '[0.0, 0.0, 1.0]'

-r RATE, --rate=RATE  publishing rate (hz).  For -f and stdin input, this
defaults to 10. Otherwise it is not set. -1, --once publish one message and exit
```

Other:

```
Reset
$ rosservice call /reset
$ rostopic pub -r 10 /turtle1/cmd_vel geometry_msgs/Twist --
'[1.5, 0.0, 0.0]' '[0.0, 0.0, 1.57]'
$ rostopic hz /turtle1/pose
$ rostopic echo /turtle1/pose
$ rqt_plot /turtle1/pose/x /turtle1/pose/y

Publish the pose in another terminal:
$ rostopic echo /turtle1/cmd_vel ;

Clear:
$ rosservice call /clear

Teleoperation:
$ rosrun turtlesim turtle_teleop_key
```

Create ROS Workspace (Catkin):

```
Create Catkin:
$ source /opt/ros/melodic/setup.bash
$ mkdir -p ~/turtlesim_ws/src
$ cd ~/turtlesim_ws/
$ catkin_make
$ source devel/setup.bash
$ echo $ROS_PACKAGE_PATH

Create Package (Python Control):
$ catkin_create_pkg turtlesim_ctrl rospy
$ cd src/turtlesim_ctrl
$ mkdir launch
$ cd launch
$ touch start_turtlesim.launch
$ roslaunch turtlesim_ctrl start_turtlesim.launch
```