Robotik - exercise 2

Team GIR : Evangelia Koumartzi , Julia Schuch

WiSe 2021/22

Assignment 2-1: RViz

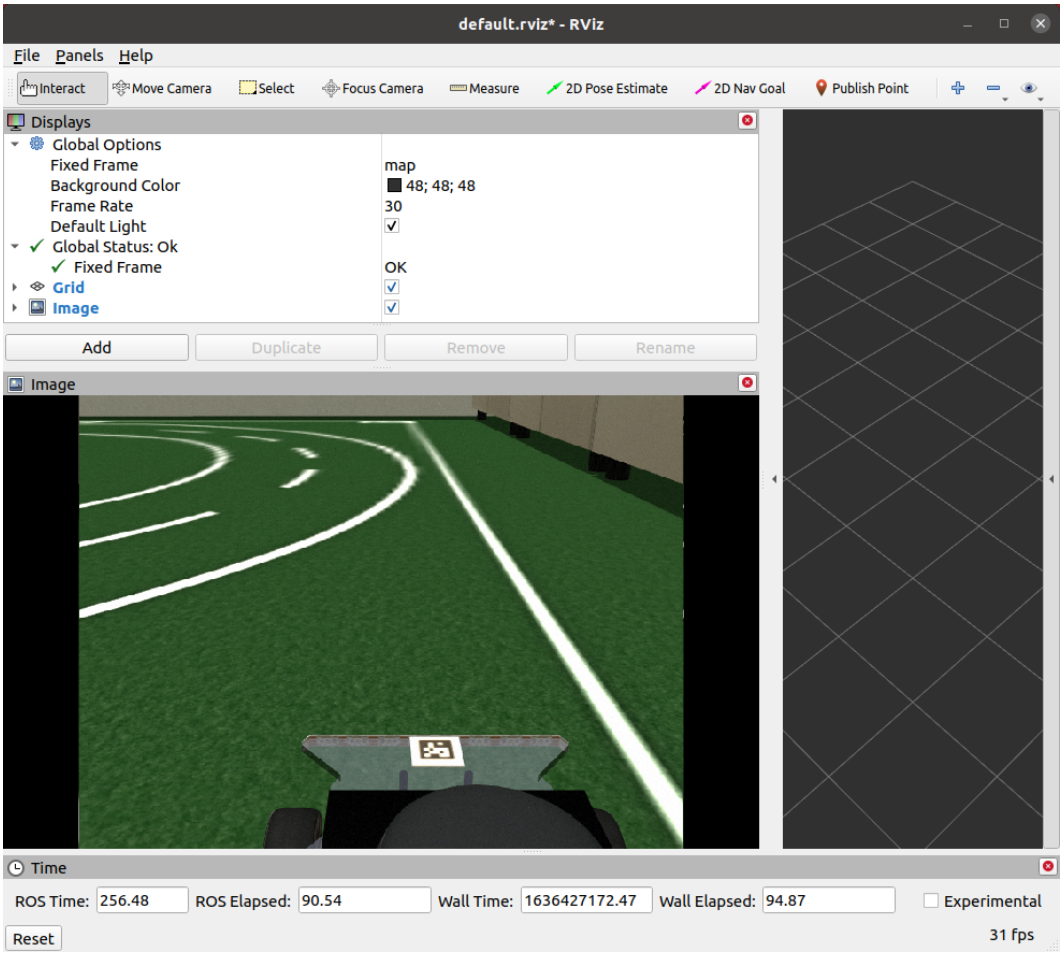
Run rviz. Record a camera image of the robot model car, as the image is seen in the robot data visualization tool `rviz`.

Use the command `rostopic list`, you should see a list of topics, coming from the car now.

```
pecco@fg7:~$ rostopic list
/actuators/speed
/actuators/speed_normalized
/actuators/speed_pwm
/actuators/steering
/actuators/steering_normalized
/actuators/steering_pwm
/autonomics/emergency_stop/parameter_descriptions
/autonomics/emergency_stop/parameter_updates
/autonomics/emergency_stop/wanted_speed
/clicked_point
/clock
/communication/gps/999
/core_nodelet_manager/bond
/diagnostics
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/initialpose
/lab/joint_states
/model_car/car_cont/gains/frontwheel_left/parameter_descriptions
/model_car/car_cont/gains/frontwheel_left/parameter_updates
/model_car/car_cont/gains/frontwheel_right/parameter_descriptions
/model_car/car_cont/gains/frontwheel_right/parameter_updates
/model_car/car_cont/gains/frontwheel_steering_left/parameter_descriptions
/model_car/car_cont/gains/frontwheel_steering_left/parameter_updates
/model_car/car_cont/gains/frontwheel_steering_right/parameter_descriptions
/model_car/car_cont/gains/frontwheel_steering_right/parameter_updates
/model_car/car_cont/gains/rearwheel_left/parameter_descriptions
/model_car/car_cont/gains/rearwheel_left/parameter_updates
/model_car/car_cont/gains/rearwheel_right/parameter_descriptions
/model_car/car_cont/gains/rearwheel_right/parameter_updates
/model_car/joint_states
/move_base_simple/goal
/rosout
/rosout_agg
/sensors/arduino/steering_angle
/sensors/arduino/ticks
/sensors/arduino/voltage
/sensors/camera/color/camera_info
/sensors/camera/color/image_rect_color
/sensors/camera/color/image_rect_color/compressed
/sensors/camera/color/image_rect_color/compressed/parameter_descriptions
/sensors/camera/color/image_rect_color/compressed/parameter_updates
/sensors/camera/color/image_rect_color/compressedDepth
/sensors/camera/color/image_rect_color/compressedDepth/parameter_descriptions
/sensors/camera/color/image_rect_color/compressedDepth/parameter_updates
/sensors/camera/color/image_rect_color/theora
/sensors/camera/color/image_rect_color/theora/parameter_descriptions
/sensors/camera/color/image_rect_color/theora/parameter_updates
/sensors/camera/color/parameter_descriptions
/sensors/camera/color/parameter_updates
/sensors/camera/depth/camera_info
/sensors/camera/depth/image_rect_raw
/sensors/camera/depth/points
/sensors/camera/depth_aligned_to_color/camera_info
```

Start rviz to visualize the car camera data as shown in the tutorial.

- Run `rviz` (type `rviz` in terminal)
- Click add button (bottom left)
- Click on `By topic` tab
- Select `/sensors/camera/color/image_raw_rect/Image`



Assignment 2-2: Basic Subscriber / Publisher Node

The goal of this task is to write a two python nodes. The first node should should subscribe to the car's speed. The second node should let the car drive in a circle. This task is similar to the talker / chatter example from ROS and you might find some useful information there:

http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28python%29 *Create a new catkin package* `assignment2_publisher_subscriber` *in the* `src` *folder of the* `catkin_ws_TEAMNAME` *workspace on your machine. It should contain the catkin dependencies* `rospy` *and* `autominy_msgs`*. For creating a package, you can use the command:* `catkin create pkg`*. See http://wiki.ros.org/ROS/Tutorials/catkin/CreatingPackage for more information and the documentation from catkin tools at https://catkin-tools.readthedocs.io/en/latest/verbs/catkin_create.html.*

In the terminal navigate to `assignment2_publisher_subscriber/src` *directory. Create an empty python script file for the subscriber and publisher and mark it as executable with:*

```
 touch subscriber.py
chmod +x subscriber.py
touch publisher.py
chmod +x publisher.py
```

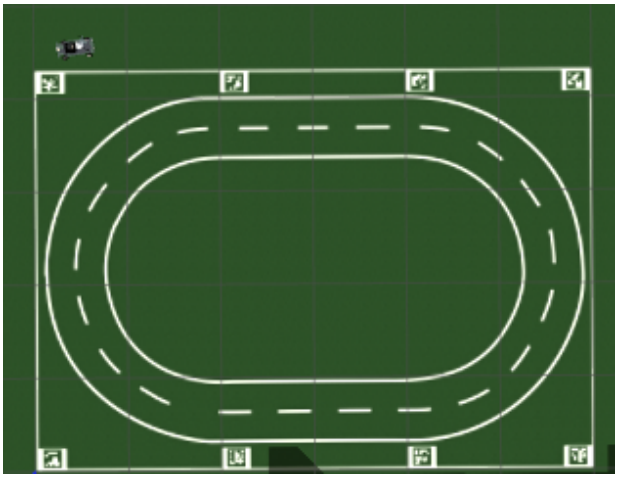
Write a simple node which subscribes to the topic: `/sensors/speed` *The data type of this topic is* `autominy_msgs/Speed` *and it contains information about the speed from the motor. You can get the type on the current running ROS system with:* `rostopic type /topic name`.

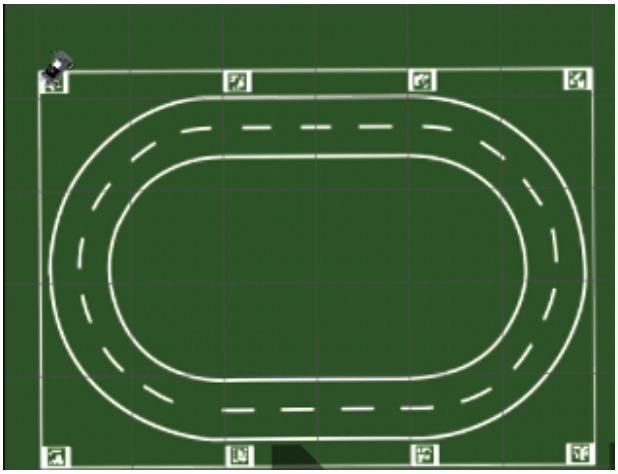
For subscribing you need to define a callback function which is called, when a new message is received. Inside this callback function simply print the value of the speed message to the console.

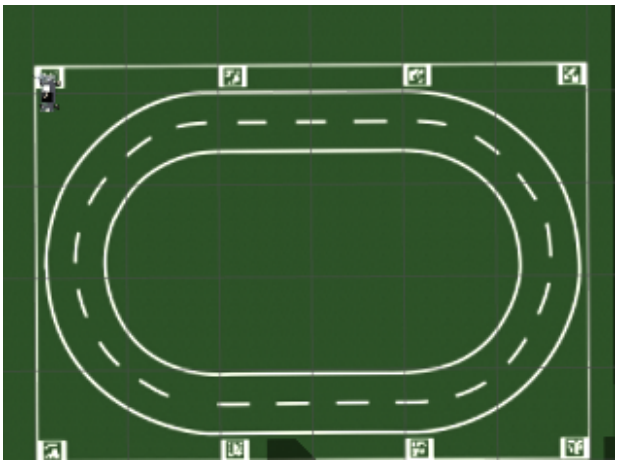
Navigate to your workspace directory and compile the package with: `catkin build assignment2_publisher_subscriber`

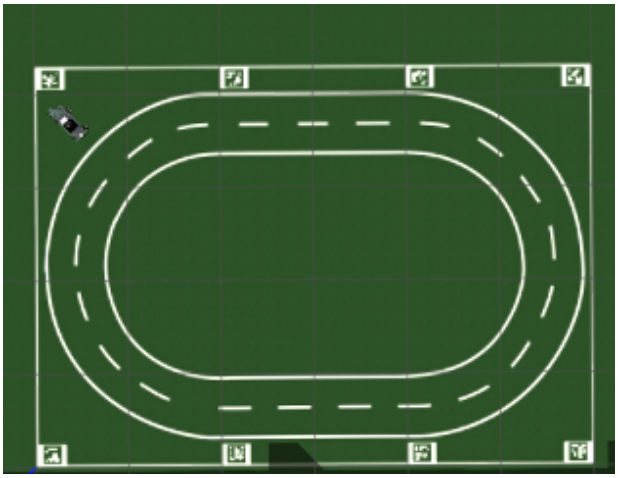
For the second node publish to the topic `/actuators/steering_normalized` *and publish an* `autominy_msgs/NormalizedSteeringCommand` *with a value of 1.0 to steer maximum to the left. In the same node add another publisher and publish to the topic* `/actuators/speed`. *This topic receives an* `autominy_msgs/SpeedCommand` *message for the speed motor. Using the publisher send a speed message to drive the car at low speed (0.3m/s). Publish these messages periodically.*

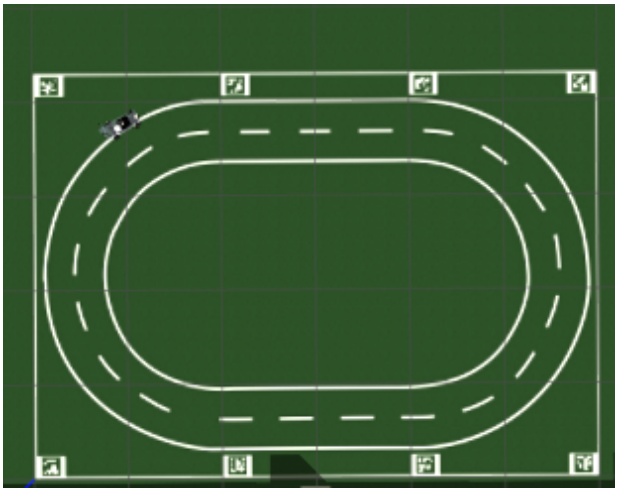
Here it can be seen how the publischer leads the car to circle:

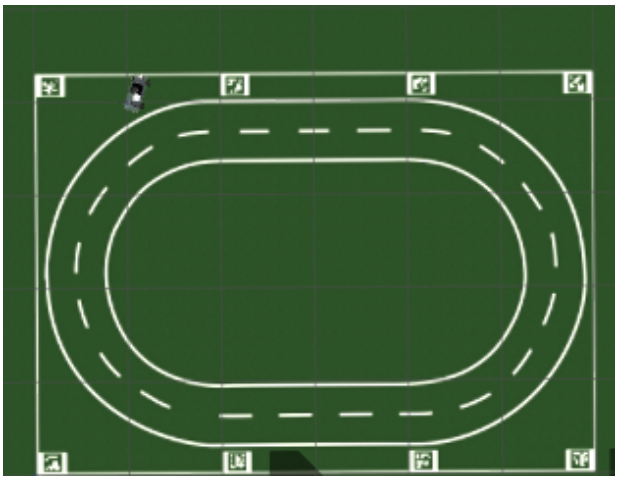




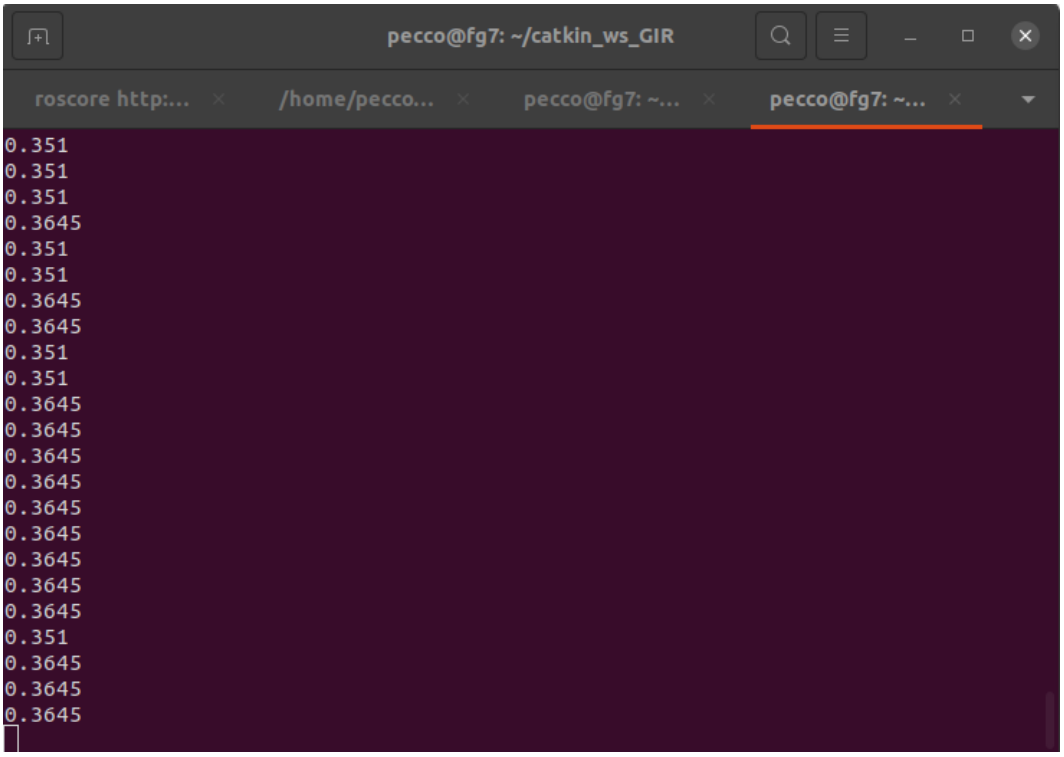








while the subscriber tracks the speed all along:



The sourcecode to eighter can be found at: https://github.com/evakoumartzi/catkin_ws_GIR