



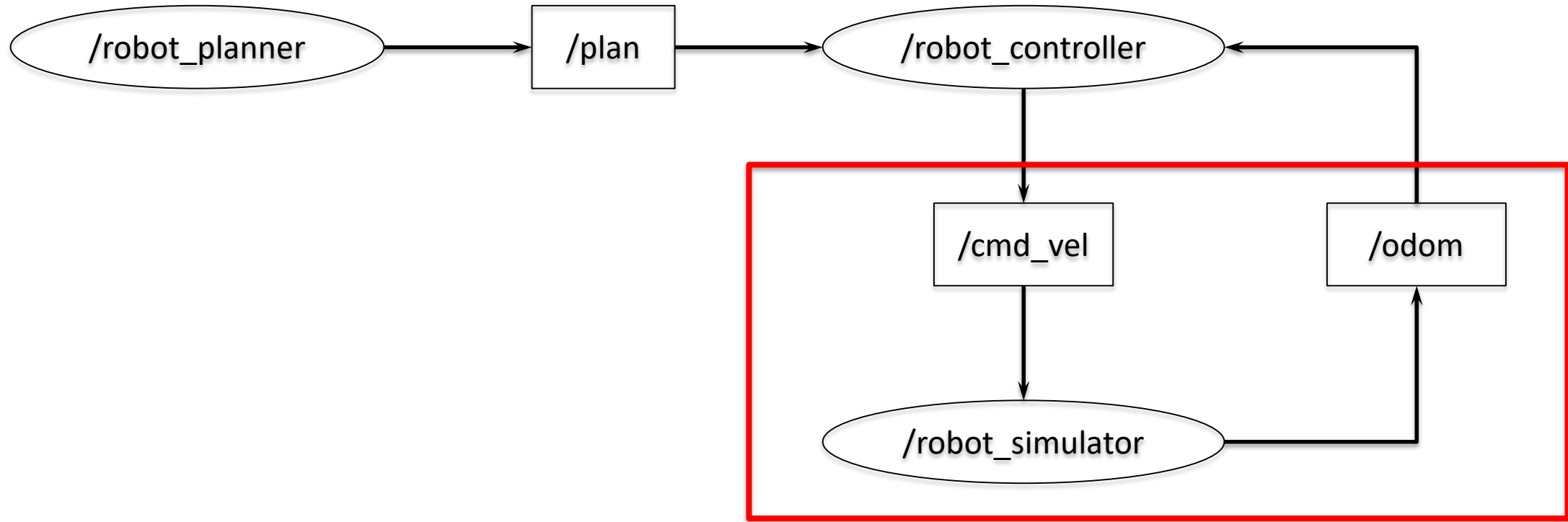
Control of Mobile Robots

A mobile robot simulator

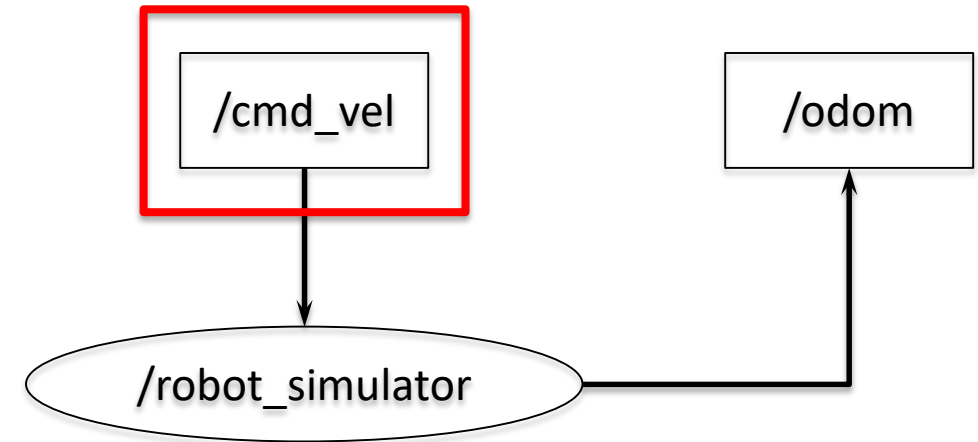
Prof. Luca Bascetta (luca.bascetta@polimi.it)

Politecnico di Milano – Dipartimento di Elettronica, Informazione e Bioingegneria

- To validate our planning and control techniques we need a simulator
 - We can consider many different software tools to design a simulator, each one with its own peculiarity
 - Matlab/Simulink
 - Dymola/Modelica
 - Gazebo
 - We select
 - Matlab/Simulink for a preliminary validation of the control system
 - Gazebo for realistic environment and sensory simulation
- and ROS to have a simulator whose interfaces are the same as the ones of the real robot



- We first concentrate on the simulator interfaces
- `/cmd_vel` is a `geometry_msgs/Twist`
`geometry_msgs/Vector3` linear
`geometry_msgs/Vector3` angular
where `geometry_msgs/Vector3`
`float64` x
`float64` y
`float64` z
- Let's consider an example, `unicycle_cmd` is a `geometry_msgs/Twist` variable
`unicycle_cmd.linear.x = v;`
`unicycle_cmd.angular.z = omega;`



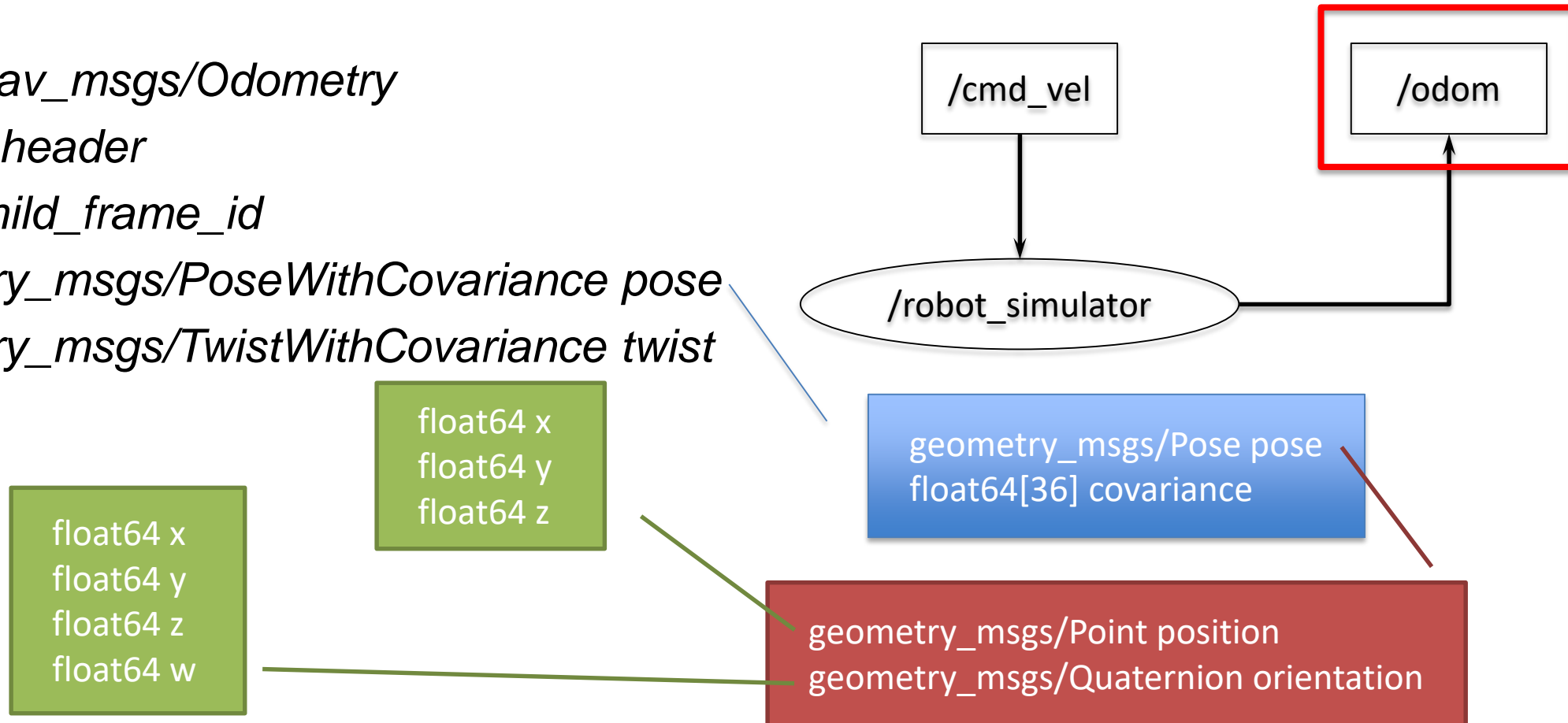
- */odom* is a *nav_msgs/Odometry*

Header header

string child_frame_id

geometry_msgs/PoseWithCovariance pose

geometry_msgs/TwistWithCovariance twist



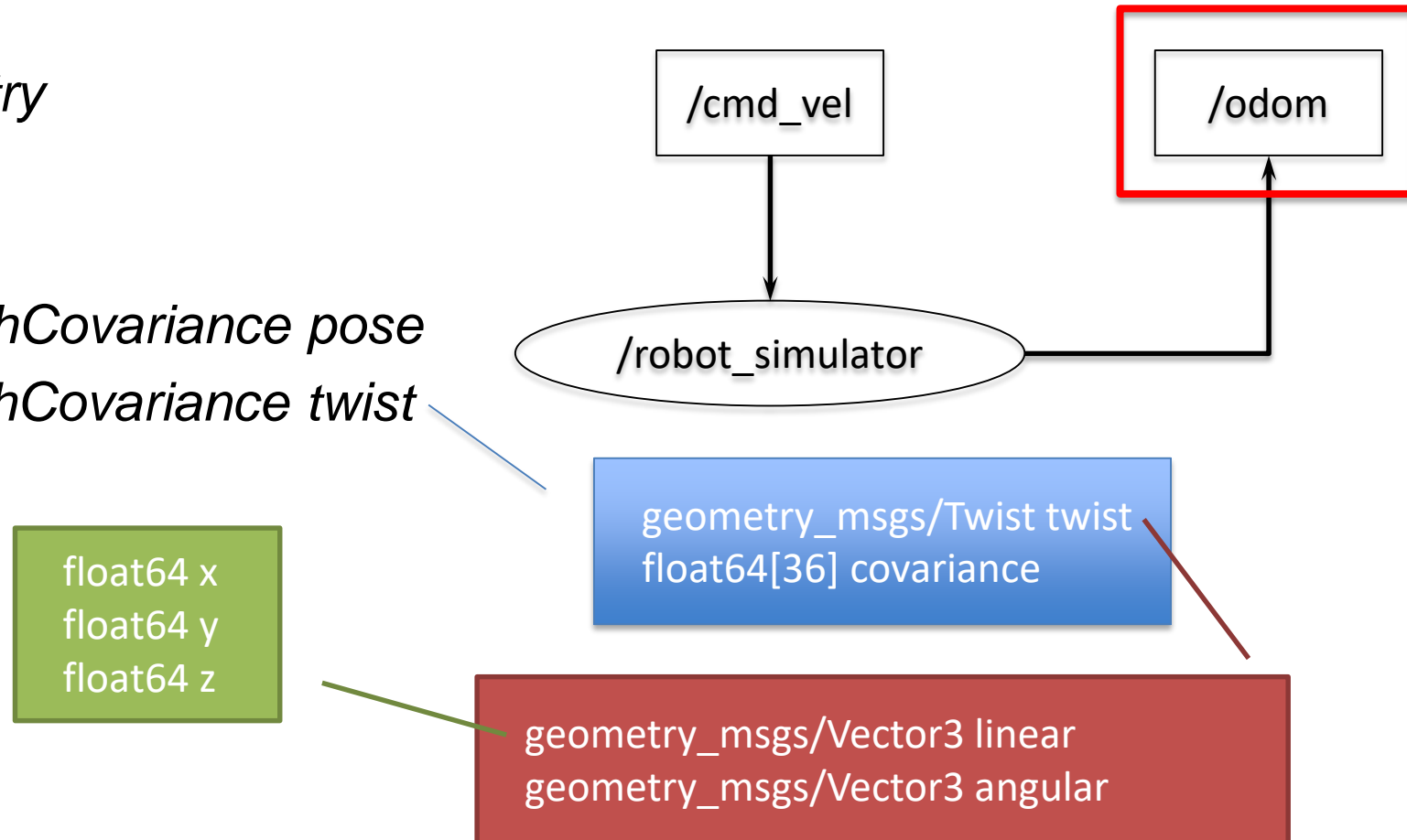
- */odom* is a *nav_msgs/Odometry*

Header header

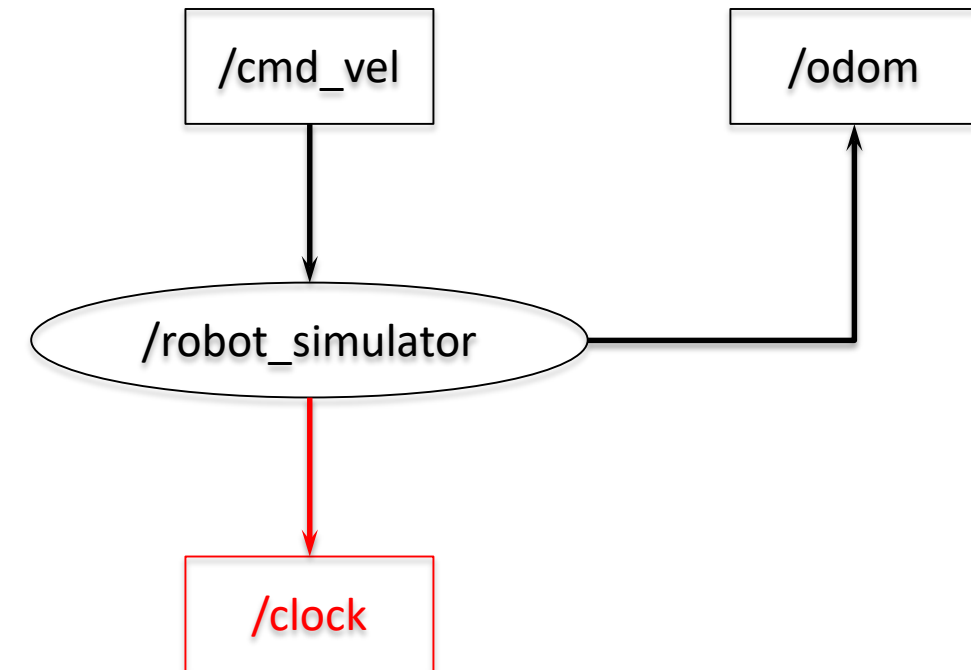
string child_frame_id

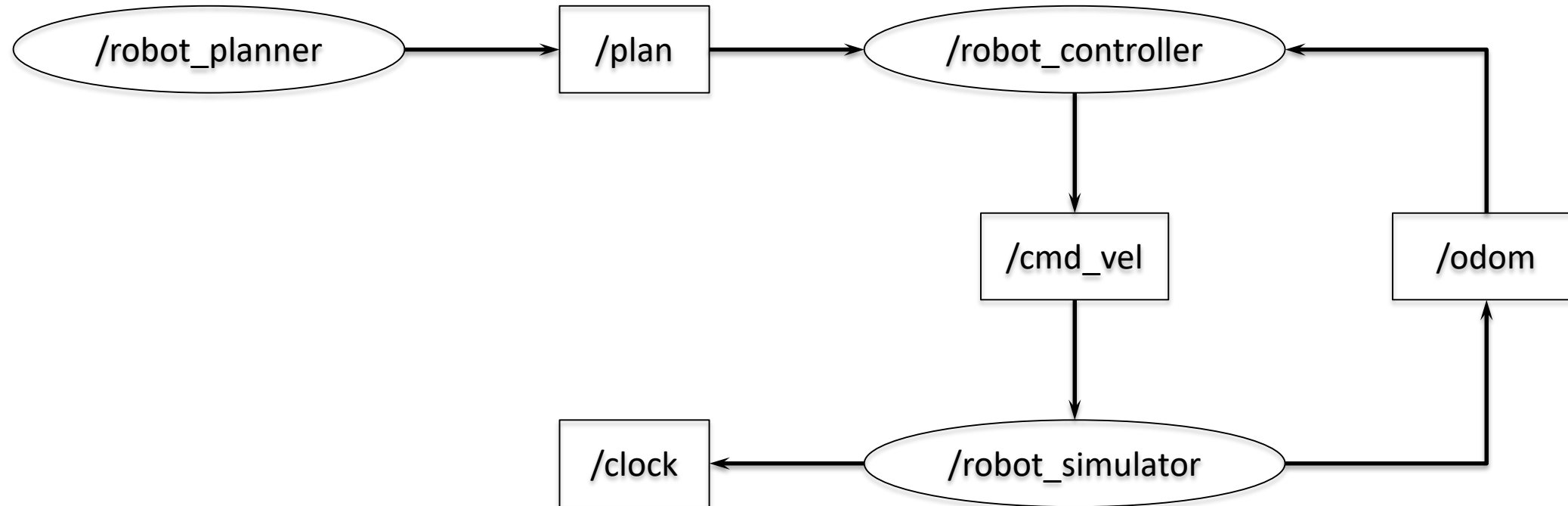
geometry_msgs/PoseWithCovariance pose

geometry_msgs/TwistWithCovariance twist



- What are the main functions a robot simulator should provide?
- Given the input commands compute the outputs integrating a kinematic/dynamic model
- Provide exteroceptive sensor measurements
- Synchronize all the other blocks in the system





- Develop a Simulink diagram to simulate a robot...
 - you can start considering the kinematic model of a unicycle...
 - ...and go on considering its dynamic model, or the kinematic model of a bicycle
- Generate the C++ code of the corresponding ROS nodes
- For each simulator develop a node that generates appropriate input commands to test it and do some tests (record a bag and analyze the results)
- In order to have a preliminary and immediate way to check the results you can visualize the robot trajectory using Rviz
- To start Rviz add the following line to your launch file

```
<node type="rviz" name="rviz" pkg="rviz" args="-d $(find unicycle_robot)/config/unicycle-kin-config.rviz" />
```

- We complete this part introducing Gazebo, the most used simulator inside the ROS community
- We will use Gazebo in the next labs to test our control system in a realistic environment using the model of an existing mobile robot called Turtlebot3 Burger
 - For further information on Turtlebot3 robots
emanual.robotis.com/docs/en/platform/turtlebot3/overview/
- To start familiarizing with Gazebo we create a launch file that starts an empty world with a Burger robot and runs the same command generator node we have used for our kinematic model
- Perform some tests, considering different inputs, and analyze the results
- When you analyze the results, consider that Gazebo model is less ideal than our Simulink kinematic model