Robot Simulation

- Subsystems simulated
- Methods used for simulation
- Using Standard simulators
- Commonly used robots in simulations

Subsystems of a Robot

- **Actuation**
  - Mobile Bases
  - Manipulators
- **Sensing**
  - Binary Sensors
  - Scalar Sensors
  - *'Rich'* Sensors
- **Computing**
  - On-board
  - Remote

These are simulated, along with a simulated environment on various softwares.

The robots and environments can also be visualized in the simulation software

Actuation

Wheeled Robots: Differential drive, Omni Wheel drive, Ackerman Drive, etc.

Legged Robots

Aerial Robots: Quadcopter, Hexacopters, etc.

Manipulators: Lateral joints, Rotational joints, Twisting joints, Revolving Joints

Actuators as well as their geometry need to be simulated

Sensing

Single reading sensors- Binary reading/ Scalar reading

Depth Sensors - Visual cameras (monocular & stereo), IR Cameras,

Laser Scanners - 2D, 3D

Encoders - Hall effect, Optical, etc.

Simulated sensors need to interact with the model world to produce the data similar to the actual sensor

Messages Used

Messages used for actuation and sensing should be common for all robots, as far as possible

Position, orientation, frame and velocity

Geometry & drive of the robot should not be a concern of the high-level controller

geometry_msgs/Twist
geometry_msgs/Pose

sensor_msgs/JointState

geometry_msgs/Transform

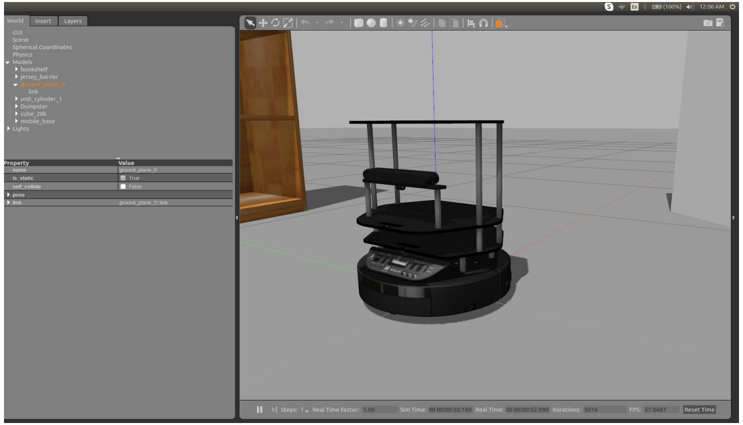
Camera outputs and depth maps

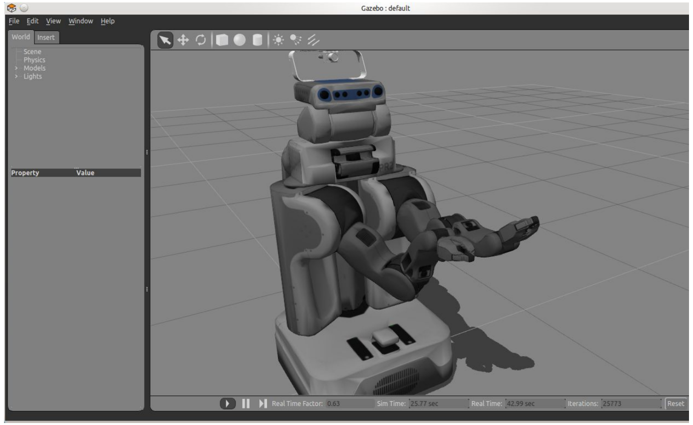
Output formats from sensor drivers/simulators must not be sensor-specific, but purpose-specific

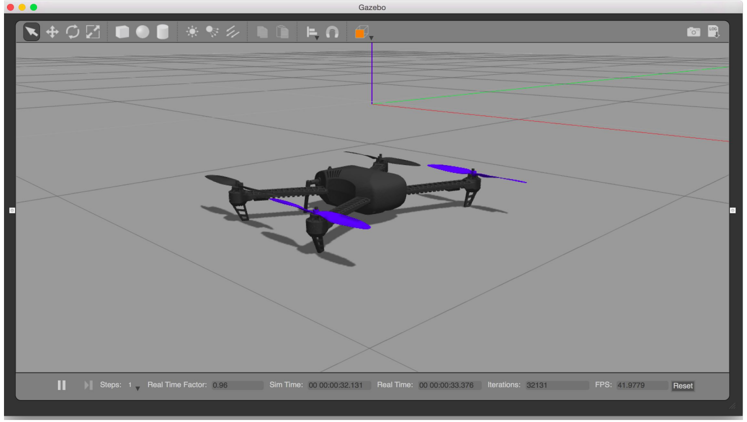
sensor_msgs/Image
sensor_msgs/CameraInfo

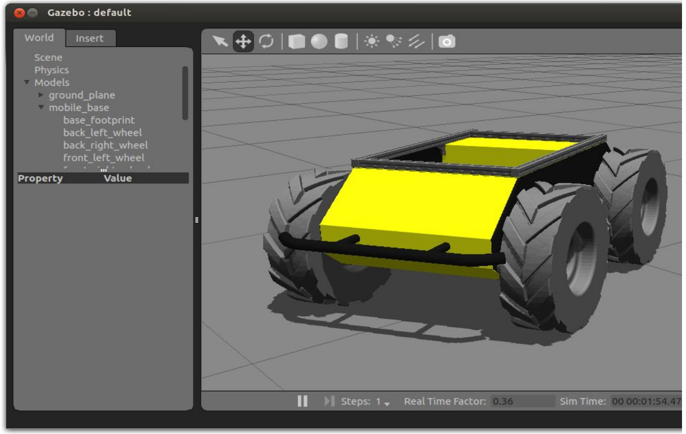
sensor_msgs/LaserScan
sensor_msgs/PointCloud2

Commonly simulated Robots









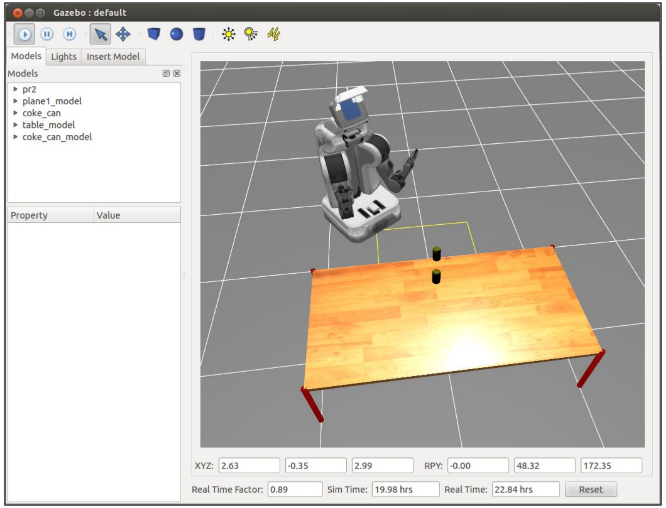
How Simulators Work

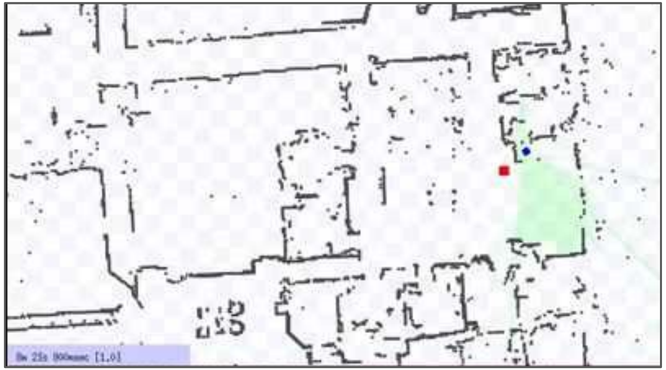
- Simulations contain a **virtual model** of the **environment** and of the **robot**
- The model of the robot includes the **geometry, constraints, behaviour**, etc. of the robot and simulations of the **sensors** on the robot.
- The simulator **subscribes** to the same topics that are used by the actual robot.
  It uses messages on these topics to simulate the actuation & the required action.
- **Sensors** on the robot model interact with the virtual world and **publish** the data on the same topics as actual sensor drivers
- For the high level software, it doesn't matter whether a simulation or an actual robot is being used.

Simulators

- **Gazebo**
  - 3D simulator
  - Can simulate robots, obstacles, etc.
  - Uses a physics engine
  - Computationally heavy

- **Stage**
  - 2D Simulator
  - Designed for multi-agent systems
  - Minimal, computationally cheap





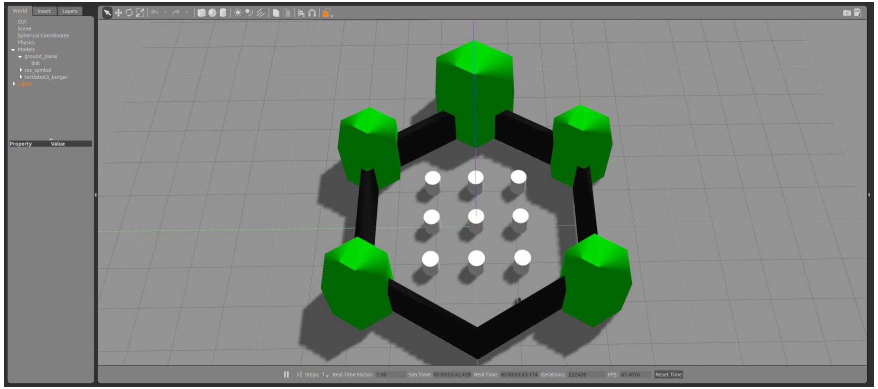
Worlds, URDF and Launchfiles

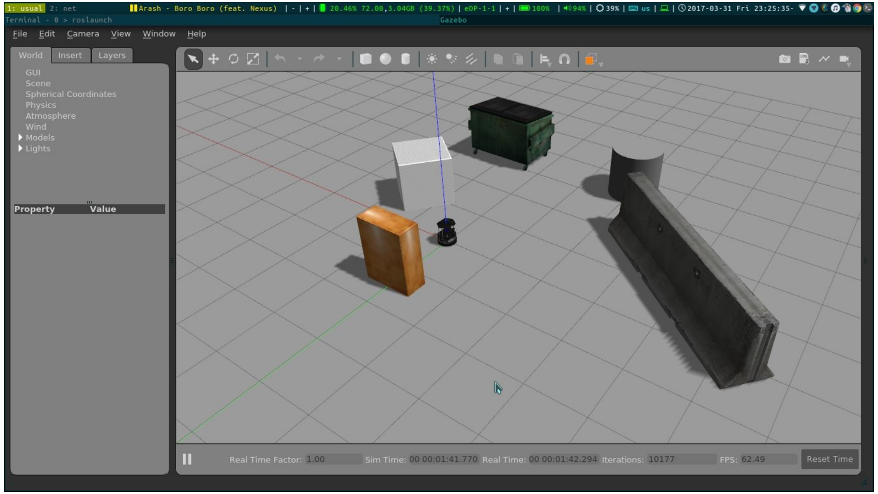
- **Worlds** are virtual environments that can be launched in a simulator.

- **URDF** (Universal Robot Description Format) is used to describe the geometry, constraints, etc. of the robot model

- **Xacro** is a modified form of URDF that is commonly used to define robot models

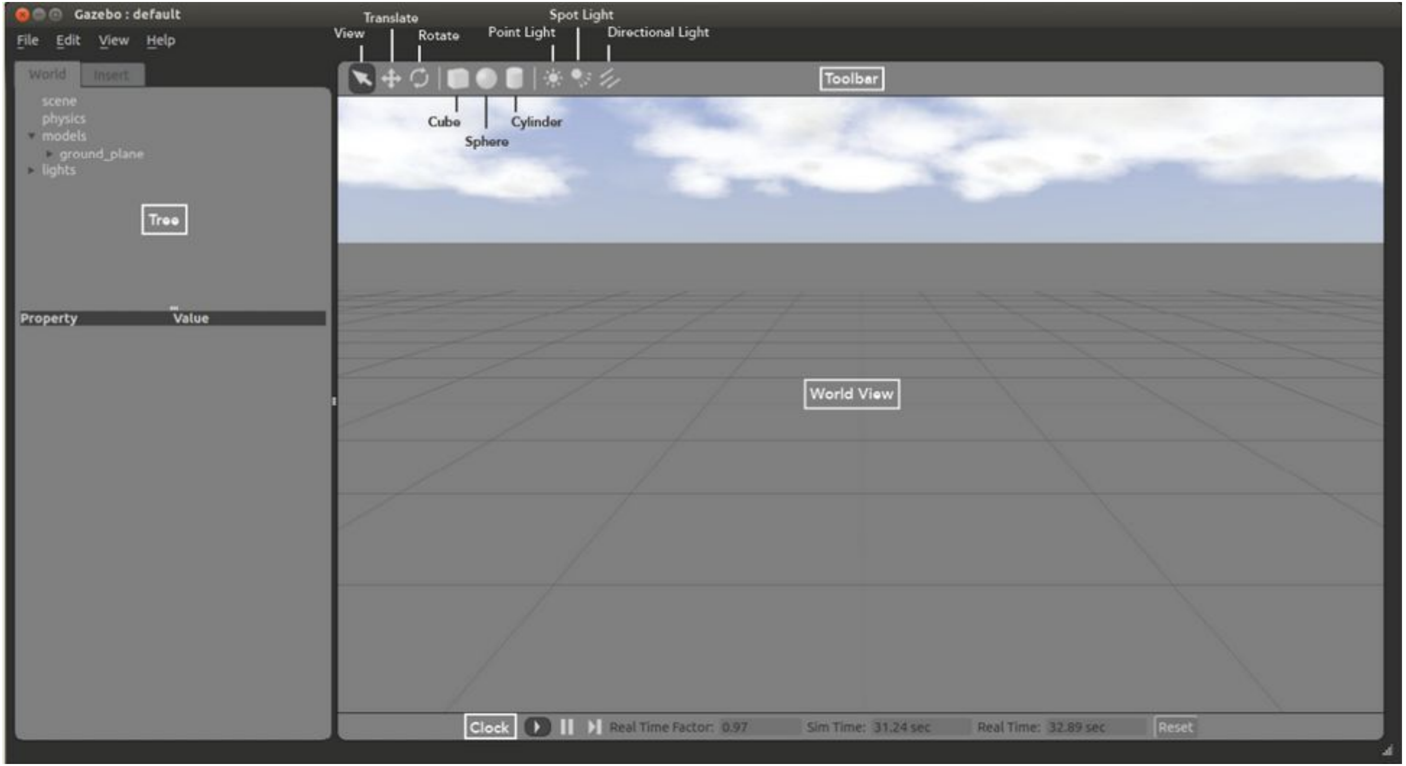
- **Launchfiles** are XML files that allow us to launch multiple rosnodes together, along with the roscore and allow us to set ROS parameters while launching.





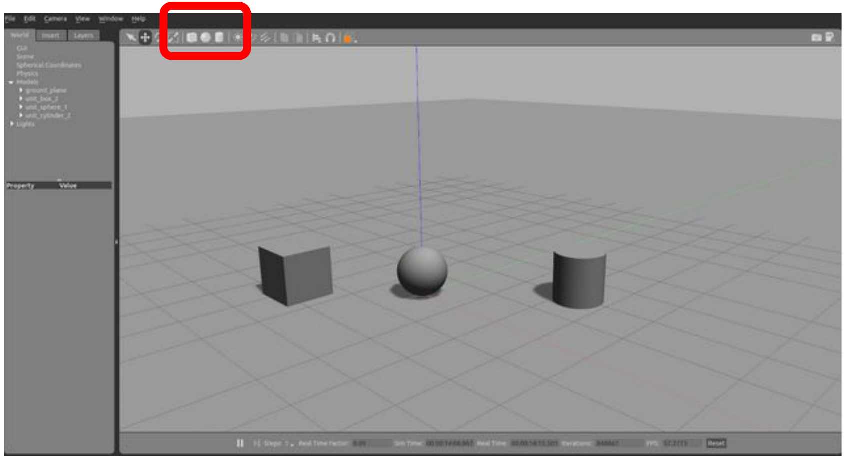
Gazebo GUI

`roslaunch gazebo_ros gazebo` *or* `gazebo <world_name>.sdf`

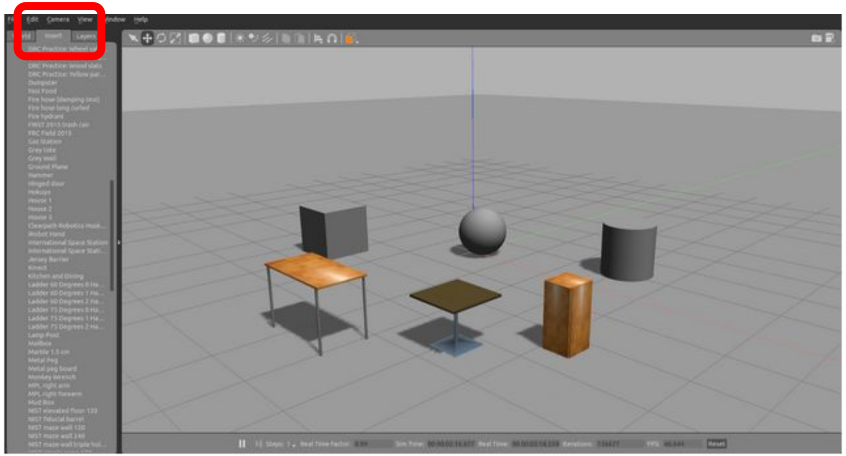


Gazebo - Creating a World

Inserting shapes



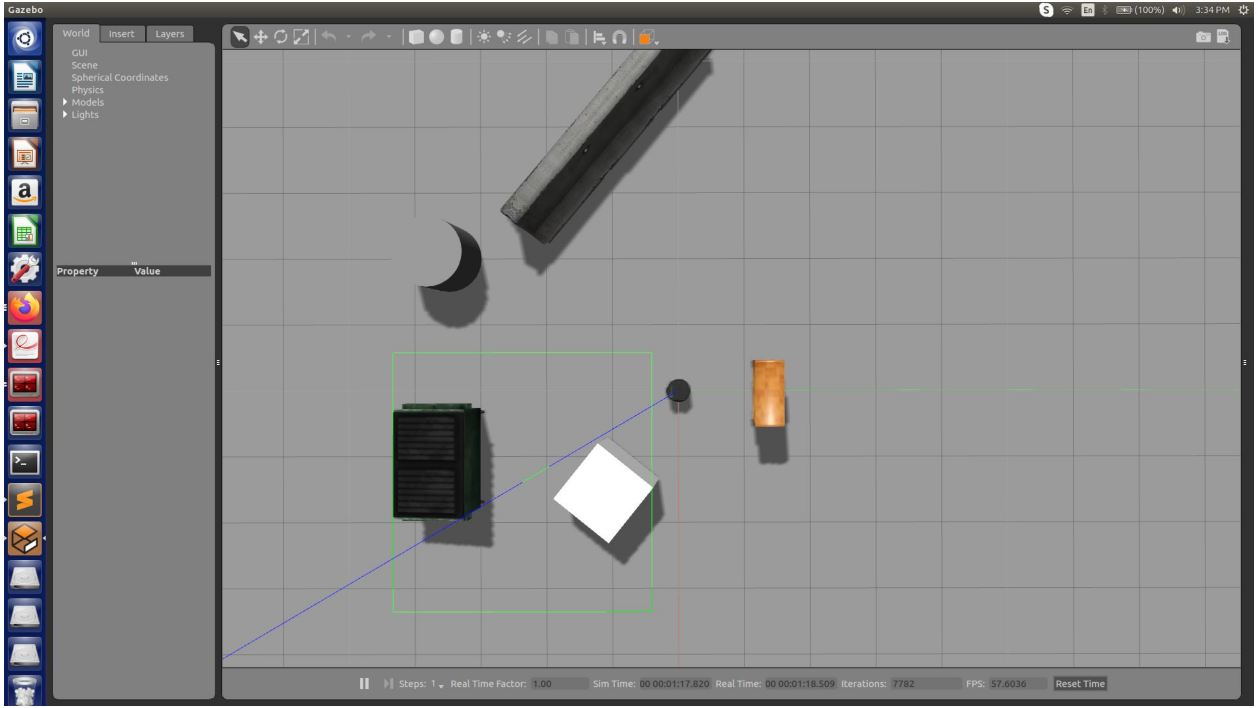
Adding a model from the database



The world file is saved with the extension ***.sdf***.

Launching Turtlebot in Gazebo

```
roslaunch turtlebot_gazebo turtlebot_world.launch
```



Try the following:

Check the active topics

Echo the topics /scan and /odom.

Check the message type of these topics

```
header:
  seq: 37365
  stamp:
    secs: 373
    nsecs: 790000000
  frame_id: "odom"
child_frame_id: "base_footprint"
pose:
  pose:
    position:
      x: 4.06977937151e-07
      y: 5.86244132415e-07
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: -0.0703769729762
      w: 0.997520466795
  covariance: [0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000
.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1000000.0, 0.0, 0
.0, 0.0, 0.0, 0.0, 0.0, 0.05]
twist:
  twist:
    linear:
      x: -6.80888721205e-06
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: 0.000139781101681
  covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
 0.0, 0.0]
---
```

```
84414673, 1.3767646551132202, 1.3774455785751343, 1.378131628036499, 1.3788
30419158936, 1.3795197010040283, 1.3802217245101929, 1.380928874015808, 1.3
16413879394531, 1.3823591470718384, nan, 1.3830821514129639, 1.383810758590
982, 1.3845443725585938, 1.3852834701538086, 1.3860279321670532, 1.38677775
5983276, 1.3875333070755005, 1.3882942199707031, 1.3890607357025146, 1.3898
2854270935, 1.3906105756759644, 1.3913936614990234, 1.3921825885772705, 1.3
29768800735474, 1.3937770128250122, 1.394582748413086, 1.395394206047058, 1
3962113857269287, 1.3970341682434082, 1.3978627920150757, 1.398697376251220
, 1.3995379209518433, nan, 1.4003840684890747, 1.4012360572814941, 1.402093
873291016, 1.4029574394226074, 1.4038267135620117, 1.4047021865844727, 1.40
583381652832, 1.4064706563949585, 1.407363772392273, 1.4082629680633545, 1.
091678857803345, 1.4100786447525024, 1.4109957218170166, 1.4119187593460083
 1.4128477573394775, 1.413783311843872, nan, 1.4147250652313232, 1.41567277
083252, 1.4166269302368164, 1.417587161064148, 1.4185537099838257, 1.419526
769958496, 1.4205061197280884, 1.4214918613433838, 1.422484040260315, 1.423
824180603027, 1.4244871139526367, 1.4254982471466064, 1.426515817642212, 1.
27539587020874, nan, 1.428570032119751, 1.4296067953109741, 1.4306502342224
2, 1.4316999912261963, 1.4327560663223267, 1.4338186979293823, 1.4348878860
73633, 1.4359638690948486, 1.4370464086532593, 1.4381358623504639, 1.439231
917678833, 1.4403350353240967, 1.441444754600525, nan, 1.442561149597168, 1
4436845779418945, 1.444814682006836, 1.4459518194198608, 1.4470958709716797
 1.4482468366622925, 1.4494047164916992, 1.4505692720413208, 1.451740741729
363, 1.452919363975525, 1.4541047811508179, 1.4552973508834839, nan, 1.4564
68347549438, 1.457703948020935, 1.4589180946350098, 1.4601396322250366, 1.4
13680839538574, 1.4626036882400513, 1.4638463258743286, 1.465096116065979,
.4663532972335815, 1.4676176309585571, 1.4688893556594849, nan, nan, nan, n
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan
 nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan
 nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan
 nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan
 nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, n
n, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, na
, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan
 nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
an, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan]
intensities: []
---
```

Turtlebot TeleOp

The package **turtlebot_teleop** allows us to control a turtlebot using keyboard keys.

```
roslaunch turtlebot_teleop keyboard_teleop.launch
```

The node **turtlebot_teleop_keyboard** publishes velocity commands through the **Twist** message on the topic **cmd_vel.**

If **turtlebot_gazebo** has a world running, gazebo subscribes to this topic and makes the turtlebot simulation move accordingly.

Exercise - Simulate a Wanderbot

Turtlebot-gazebo subscribes to **cmd_vel** for getting velocity commands.
Similarly, the LIDAR on the simulated turtlebot publishes the obstacle data to the topic **scan.**

Write a node that subscribes to this laser scan and publishes velocity commands.

The node should allow the turtlebot to do the following:

1. Keep going forward for a random duration of time (greater than 2 seconds)
   *Use the **random** library and the **time** library of Python.*
2. After this duration: stop, turn by a random amount, and carry out step 1 again.
3. If an obstacle is detected within 0.5m in the front, carry out step 2.

Launch the wanderer in turtlebot_world.

References:

Lidar 101: https://www.youtube.com/watch?v=NZKvf1cXe8s

Morgan Quigley Chapter 6 (part 3) and Chapter 7.

ROSWiki Tutorials