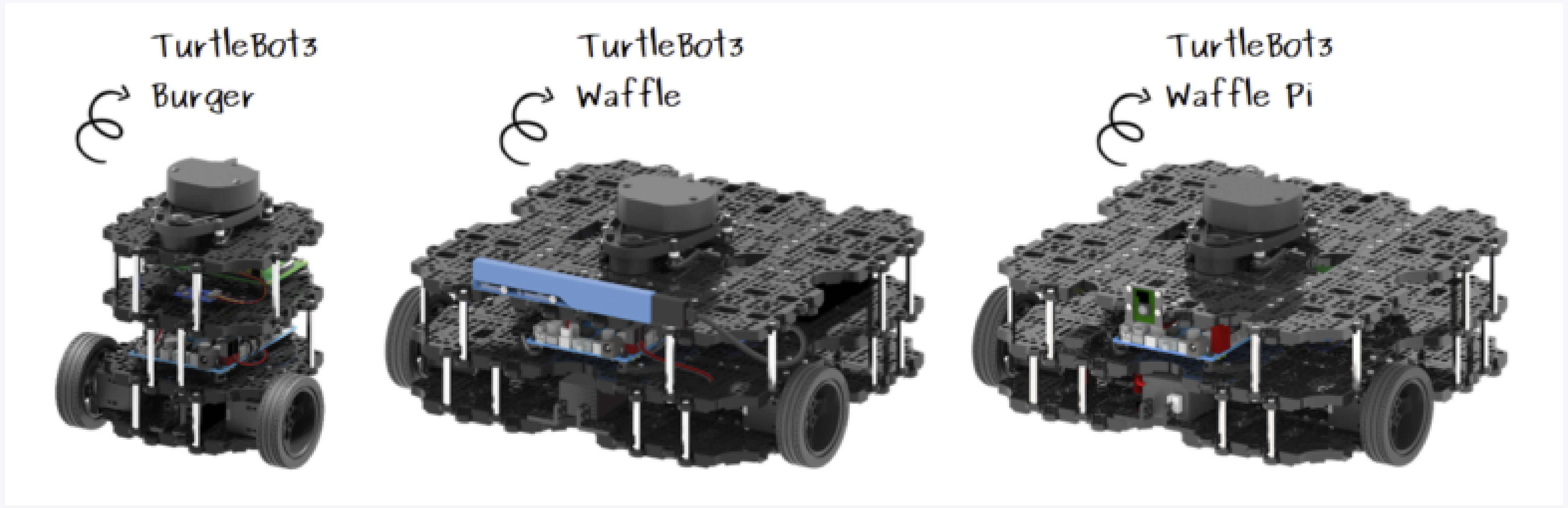
[Lab 2]

TurtleBot3, Gazebo simulation, and SLAM

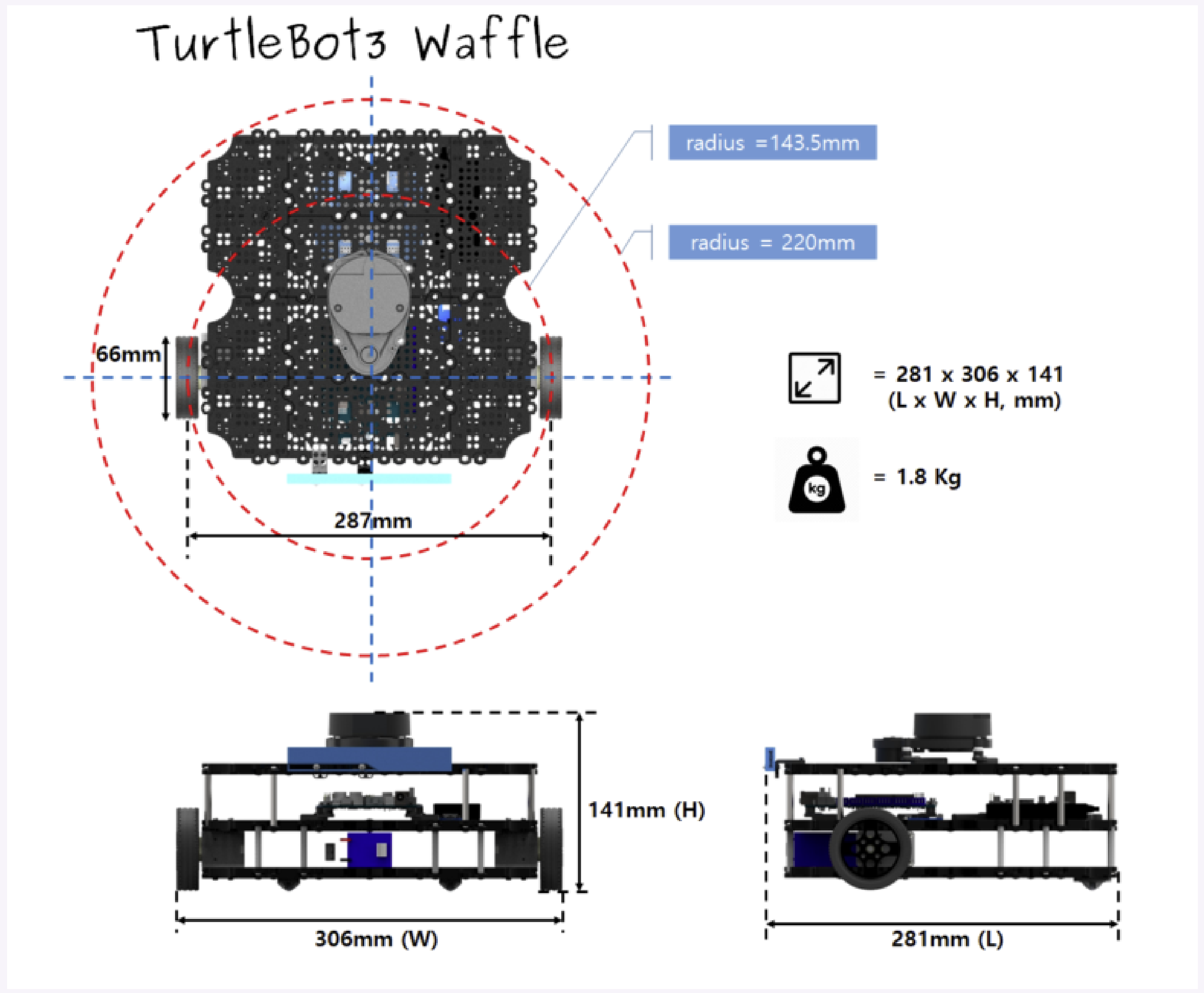
TurtleBot3

TurtleBot3 is the most affordable robot among the SLAM-able mobile robots equipped with a 360° Laser Distance Sensor LDS-01.



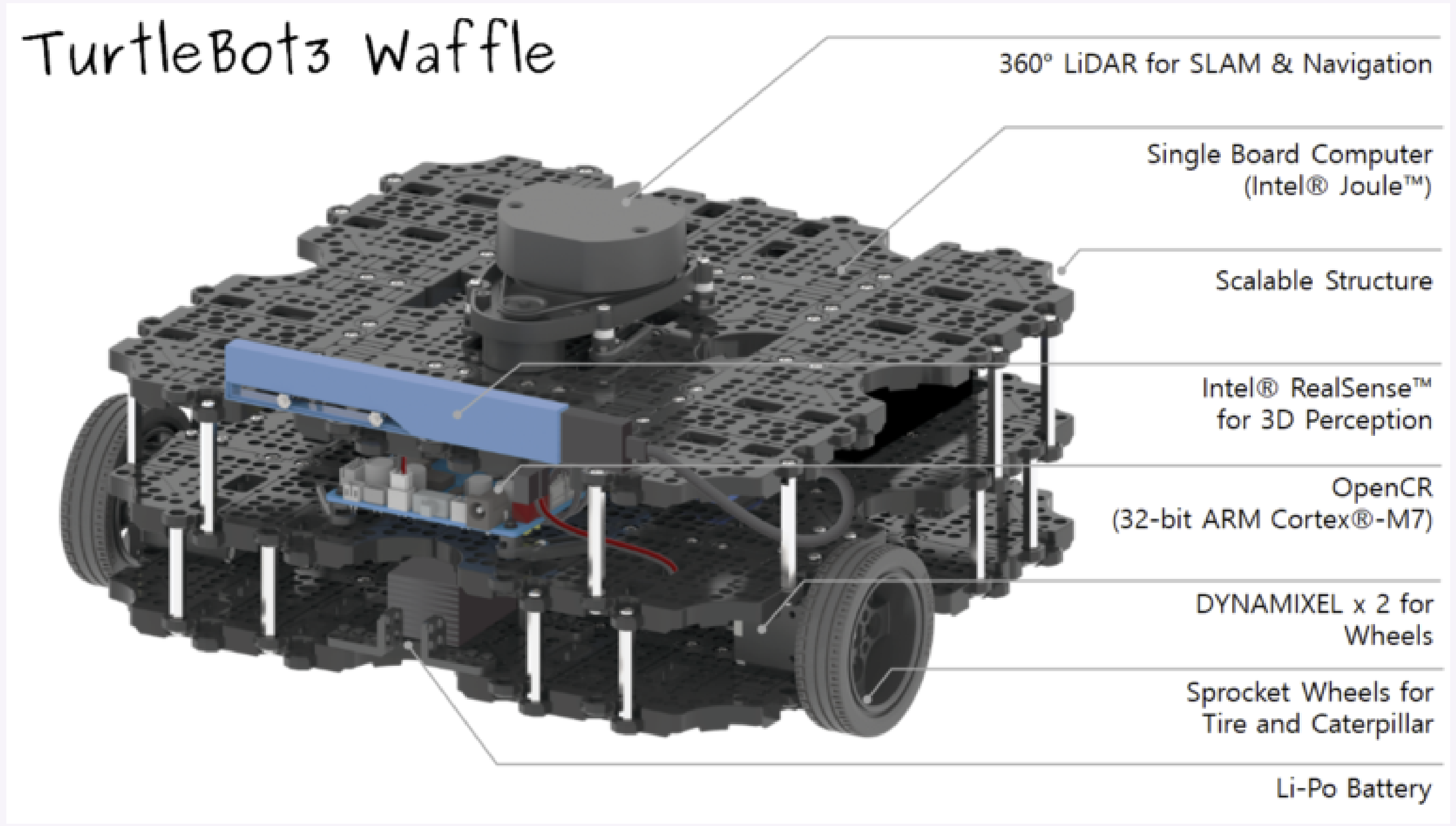
TurtleBot3

The hardware, firmware and software are open source.
All components are manufactured with injection molded plastic to achieve low cost, however, the 3D CAD data is also available for 3D printing.



TurtleBot3

- TurtleBot3 is able to get a precise spatial data by using 2 DYNAMIXEL’s in the wheel joints.
- The control board (OpenCR1.0) is open-sourced in hardware wise and in software wise for ROS communication.
- TurtleBot3 uses a 360° LiDAR and precise encoder.


TurtleBot3 Waffle
360° LiDAR for SLAM & Navigation
Single Board Computer (Intel® Joule™)
Scalable Structure
Intel® RealSense™ for 3D Perception
OpenCR (32-bit ARM Cortex®-M7)
DYNAMIXEL x 2 for Wheels
Sprocket Wheels for Tire and Caterpillar
Li-Po Battery

Resources

[1] GitHub of the lab: https://github.com/emarche/Mobile-robotics-4S009023-UniVR

Requirements

- Setup of L1

Simulation package

This package requires *turtlebot3* and *turtlebot3_msgs* packages installed during L1

```
$ cd ~/catkin_ws/src/
$ git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

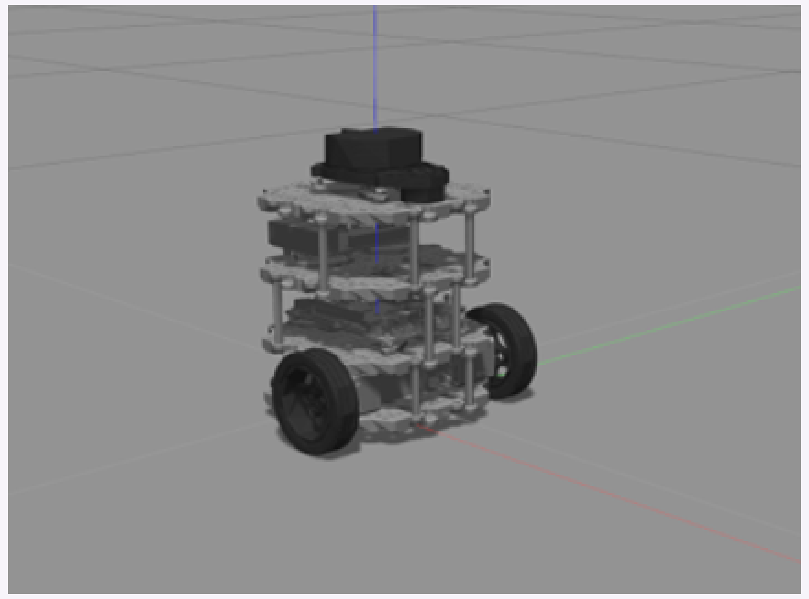
Simulation world

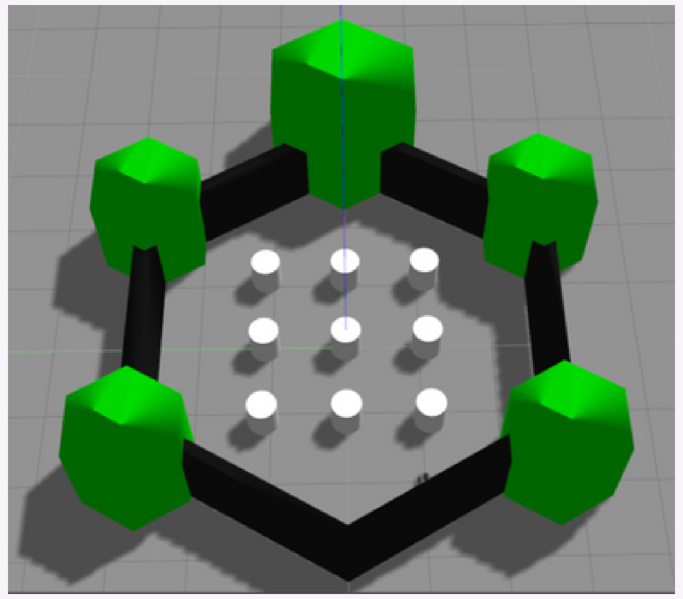
Gazebo is the standard simulation environment provided with the Turtlebot3. There are three environments ready to use:

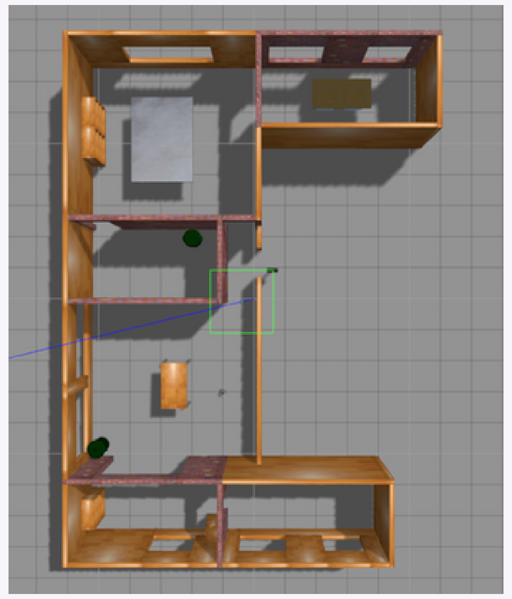
```
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

```
$ roslaunch turtlebot3_gazebo turtlebot3_house.launch
```







Teleoperation

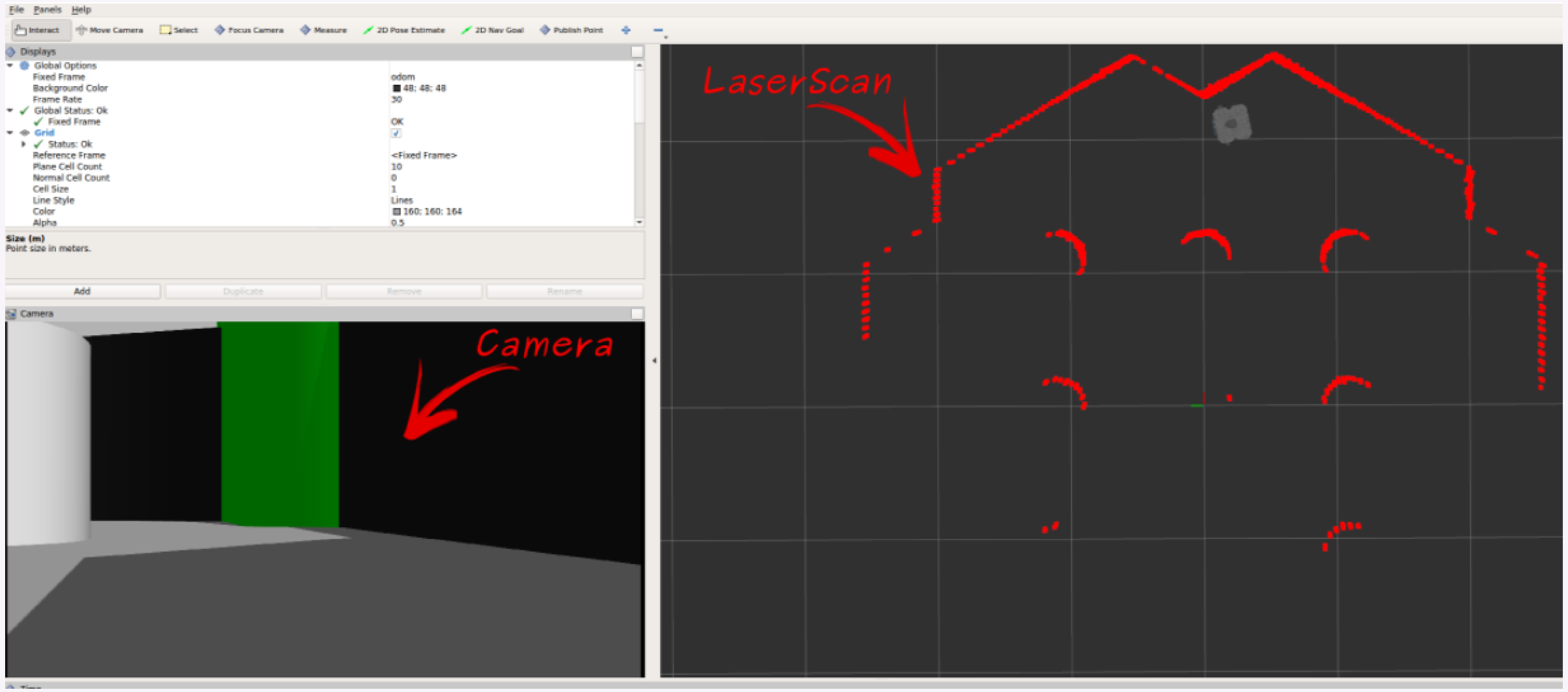
Launch the teleoperation node to move around the Turtlebot3 with the keyboard.

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Visualize topics

RViz is used to visualize the published topics during the simulation.

```
$ roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch
```



Global Options	
Fixed Frame	odom
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
✓ Status: Ok	
Reference Frame	<Fixed Frame>
Plane Cell Count	10
Normal Cell Count	0
Cell Size	1
Line Style	Lines
Color	■ 160; 160; 164
Alpha	0.5

Size (m)
Point size in meters.

Exercise

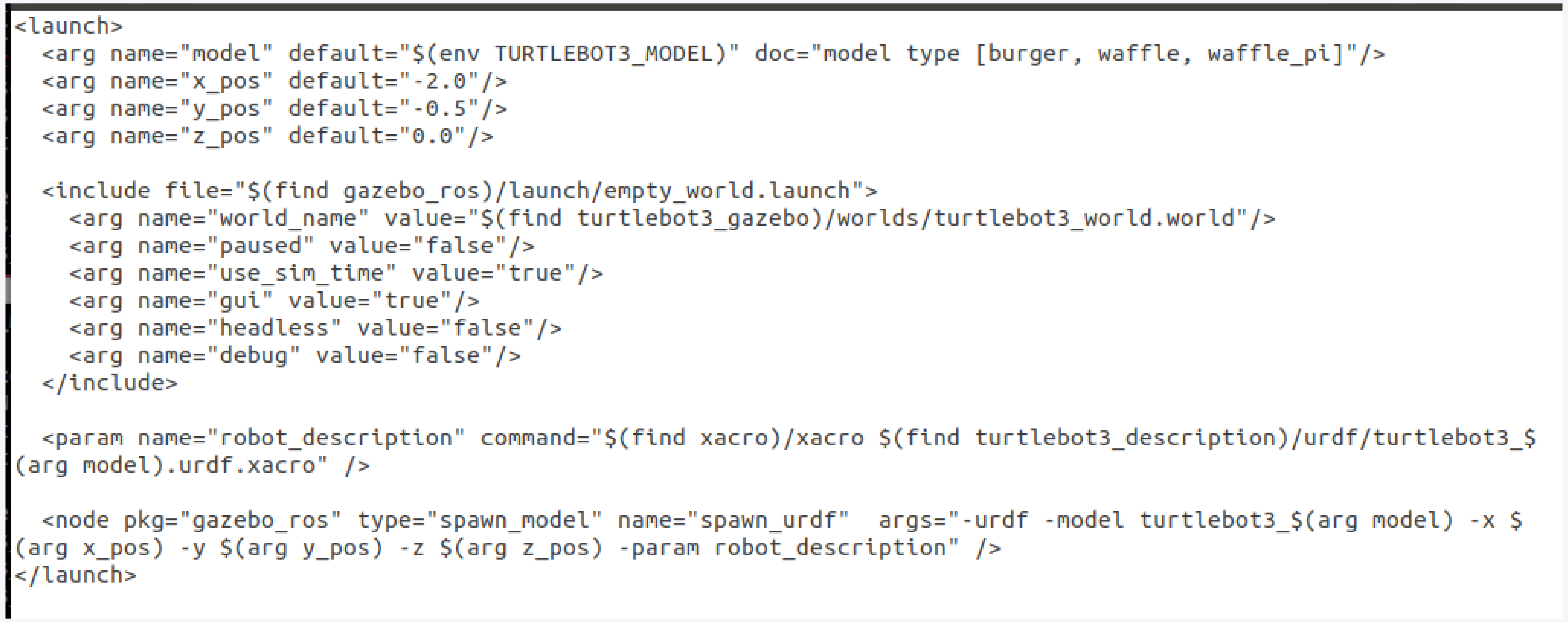
- Try to navigate in the different environments using the teleoperation node.
- Visualize different topics' information using RViz.

- Test the collision avoidance node (remember to terminate the teleoperation node):

```
$ roslaunch turtlebot3_gazebo turtlebot3_simulation.launch
```

Create a world

We can analyze the *turtlebot3_gazebo* package to understand the design of a world model, using the launch files.



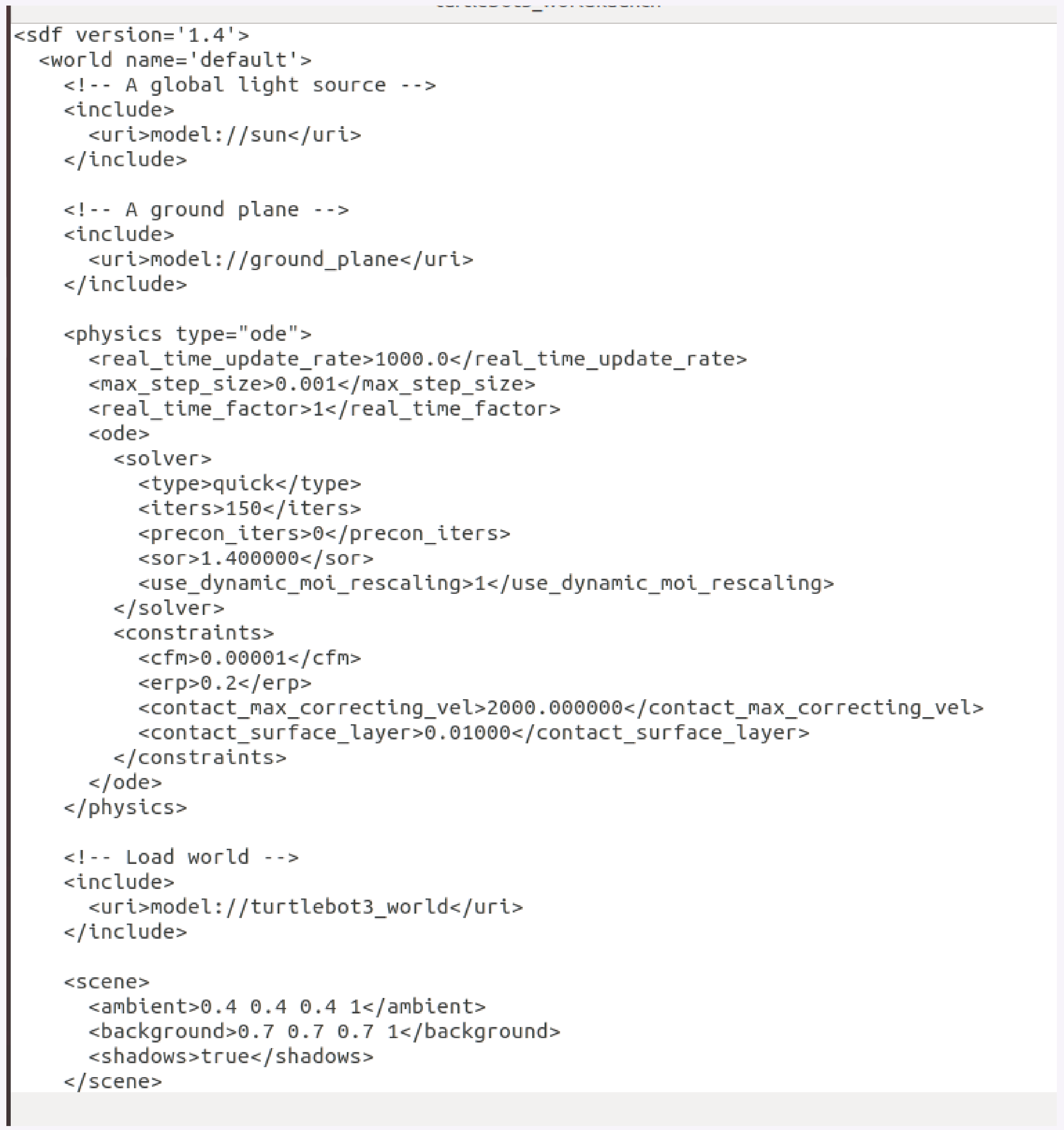
```
<launch>
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <arg name="x_pos" default="-2.0"/>
  <arg name="y_pos" default="-0.5"/>
  <arg name="z_pos" default="0.0"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find turtlebot3_gazebo)/worlds/turtlebot3_world.world"/>
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="true"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>

  <param name="robot_description" command="$(find xacro)/xacro $(find turtlebot3_description)/urdf/turtlebot3_$
(arg model).urdf.xacro" />

  <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf"  args="-urdf -model turtlebot3_$(arg model) -x $
(arg x_pos) -y $(arg y_pos) -z $(arg z_pos) -param robot_description" />
</launch>
```

.world

The actual models are stored in the *worlds* folder.

The .world files specify different components of the environment:

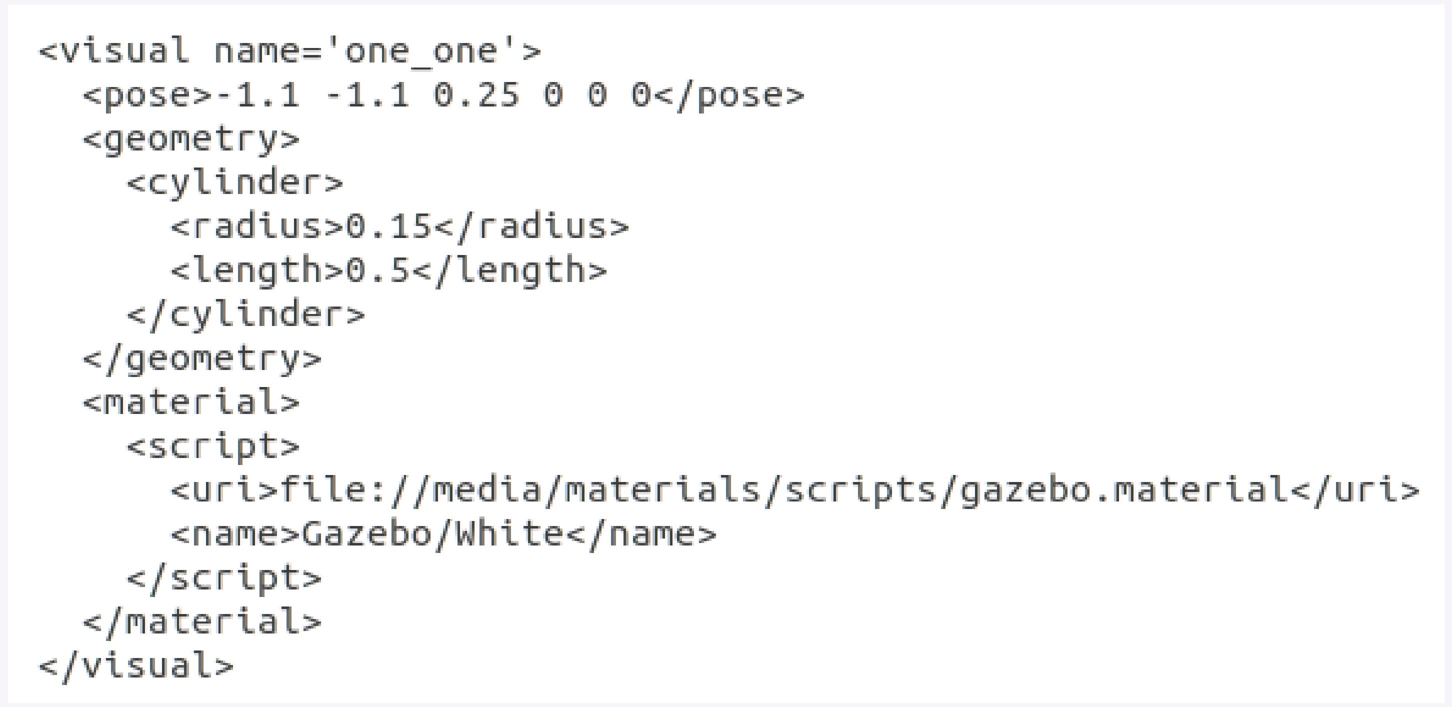
- Physics
- Lights
- Camera
- World models



```
<sdf version='1.4'>
  <world name='default'>
    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>

    <physics type="ode">
      <real_time_update_rate>1000.0</real_time_update_rate>
      <max_step_size>0.001</max_step_size>
      <real_time_factor>1</real_time_factor>
      <ode>
        <solver>
          <type>quick</type>
          <iters>150</iters>
          <precon_iters>0</precon_iters>
          <sor>1.400000</sor>
          <use_dynamic_moi_rescaling>1</use_dynamic_moi_rescaling>
        </solver>
        <constraints>
          <cfm>0.00001</cfm>
          <erp>0.2</erp>
          <contact_max_correcting_vel>2000.000000</contact_max_correcting_vel>
          <contact_surface_layer>0.01000</contact_surface_layer>
        </constraints>
      </ode>
    </physics>

    <!-- Load world -->
    <include>
      <uri>model://turtlebot3_world</uri>
    </include>

    <scene>
      <ambient>0.4 0.4 0.4 1</ambient>
      <background>0.7 0.7 0.7 1</background>
      <shadows>true</shadows>
    </scene>
```

World models

The models are specified in the *.stl* format inside the *models* folder.



```
<visual name='one_one'>
  <pose>-1.1 -1.1 0.25 0 0 0</pose>
  <geometry>
    <cylinder>
      <radius>0.15</radius>
      <length>0.5</length>
    </cylinder>
  </geometry>
  <material>
    <script>
      <uri>file://media/materials/scripts/gazebo.material</uri>
      <name>Gazebo/White</name>
    </script>
  </material>
</visual>
```

We can avoid to manually specify every object, by using Gazebo to design the environment.

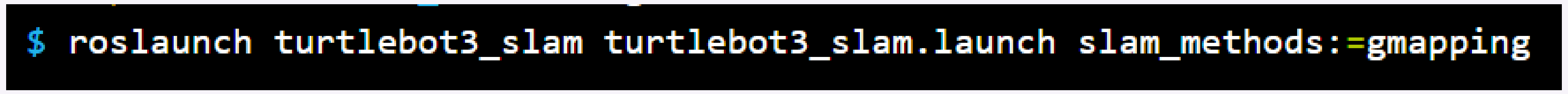
Exercise

- Create a new environment in Gazebo and export it.
- Launch a Turtlebot3 simulation in such environment

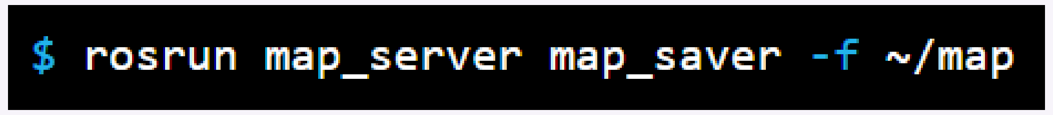
SLAM

The SLAM (Simultaneous Localization and Mapping) is a technique to draw a map by estimating current location in an arbitrary space.

- Launch a Gazebo environment
- Run SLAM node
- Run teleoperation node
- Save the map



```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
```



```
$ rosrun map_server map_saver -f ~/map
```

Try to create the map of one of the provided Gazebo environments.

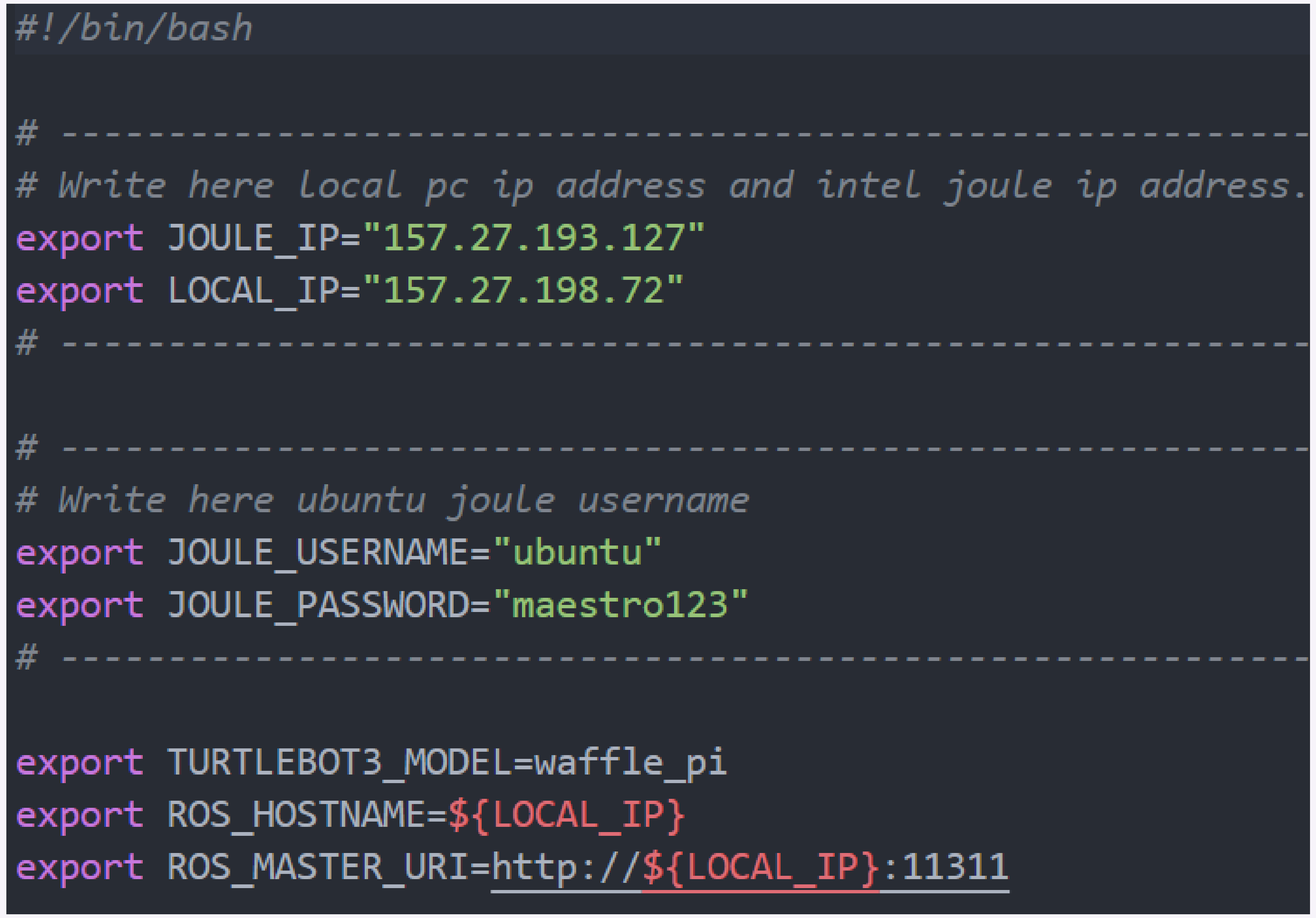
Bringup

We provide a set of scripts for the bringup of the real robot in the course's repository [1]:

- config.sh
- 00sendconfiguration.sh
- 01hostpc_bringup.sh
- 02turtlebot_bringup.sh

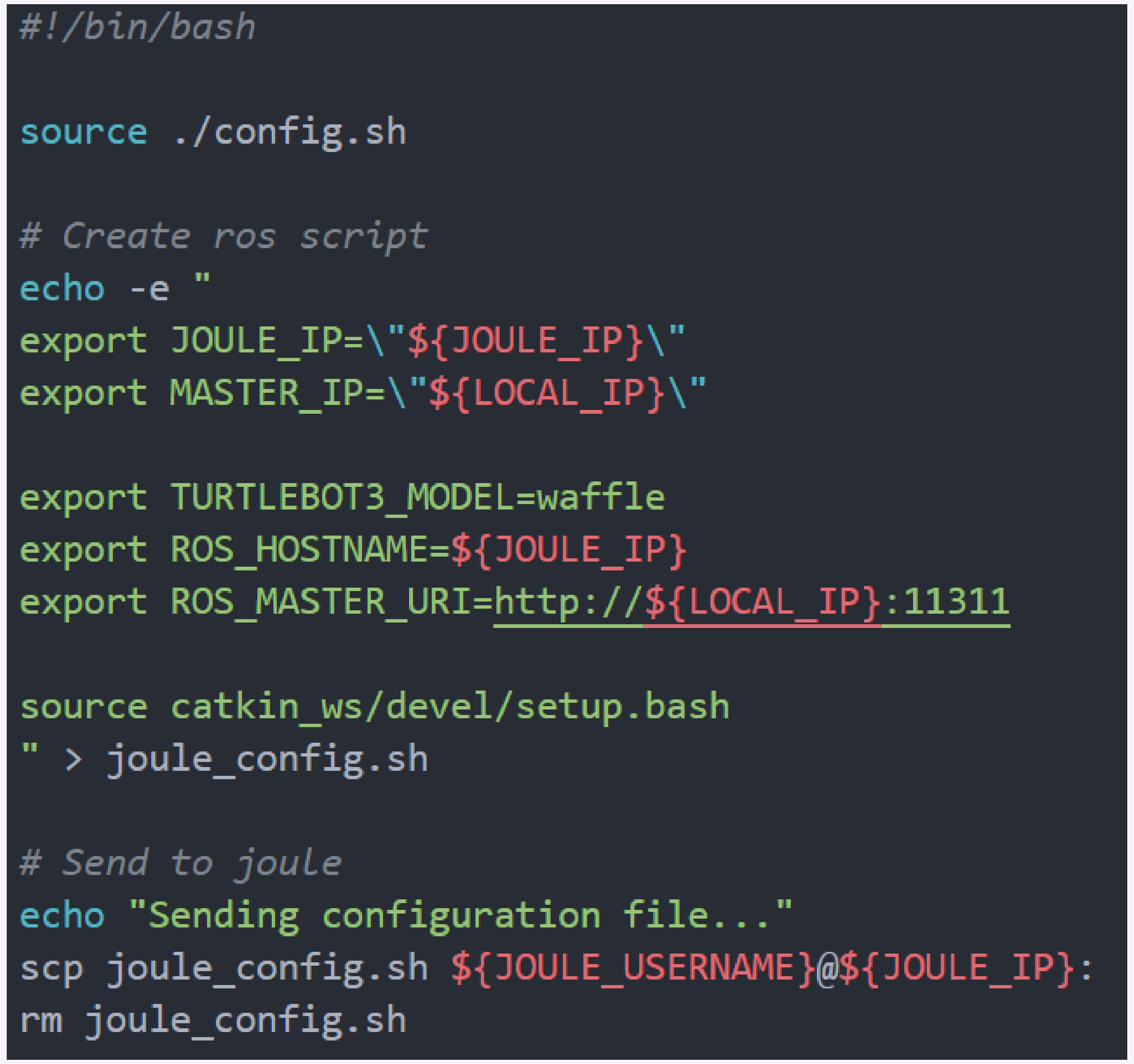
These are used to configure the system and pass the configuration files to the robot.

config



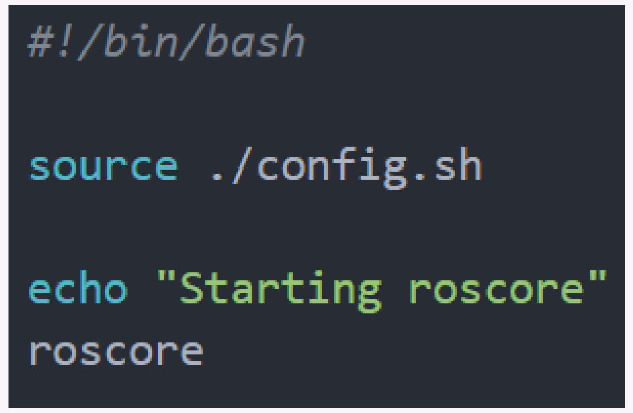
```
#!/bin/bash

# ------------------------------------------------------------
# Write here local pc ip address and intel joule ip address.
export JOULE_IP="157.27.193.127"
export LOCAL_IP="157.27.198.72"
# ------------------------------------------------------------


# ------------------------------------------------------------
# Write here ubuntu joule username
export JOULE_USERNAME="ubuntu"
export JOULE_PASSWORD="maestro123"
# ------------------------------------------------------------

export TURTLEBOT3_MODEL=waffle_pi
export ROS_HOSTNAME=${LOCAL_IP}
export ROS_MASTER_URI=http://${LOCAL_IP}:11311
```

00sendconfiguration



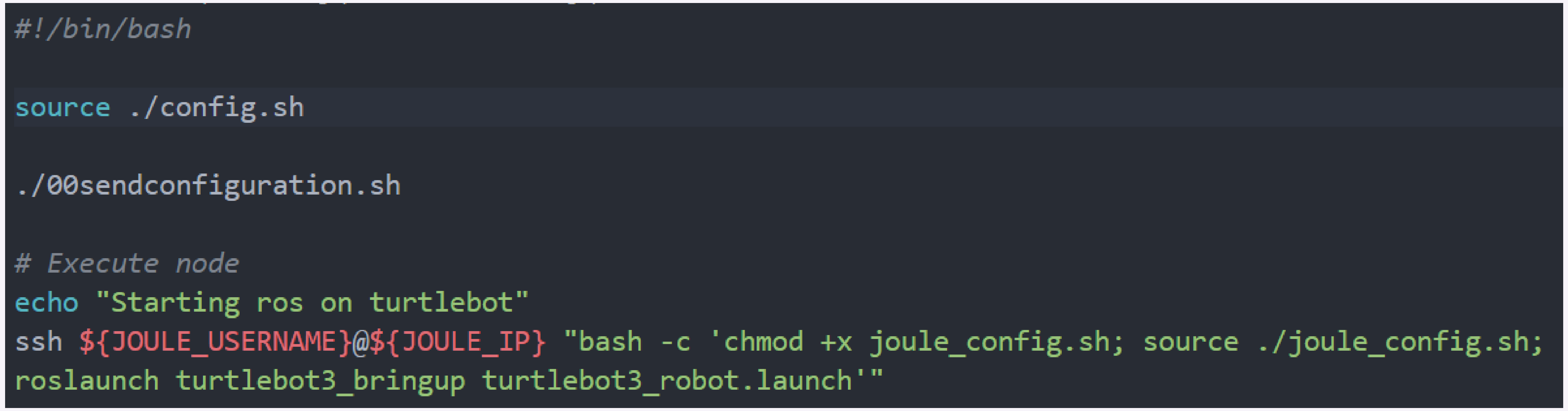
```bash
#!/bin/bash

source ./config.sh

# Create ros script
echo -e "
export JOULE_IP=\"${JOULE_IP}\"
export MASTER_IP=\"${LOCAL_IP}\"

export TURTLEBOT3_MODEL=waffle
export ROS_HOSTNAME=${JOULE_IP}
export ROS_MASTER_URI=http://${LOCAL_IP}:11311

source catkin_ws/devel/setup.bash
" > joule_config.sh

# Send to joule
echo "Sending configuration file..."
scp joule_config.sh ${JOULE_USERNAME}@${JOULE_IP}:
rm joule_config.sh
```

01hostpc_bringup



```bash
#!/bin/bash

source ./config.sh

echo "Starting roscore"
roscore
```

02turtlebot_bringup



```bash
#!/bin/bash

source ./config.sh

./00sendconfiguration.sh

# Execute node
echo "Starting ros on turtlebot"
ssh ${JOULE_USERNAME}@${JOULE_IP} "bash -c 'chmod +x joule_config.sh; source ./joule_config.sh;
roslaunch turtlebot3_bringup turtlebot3_robot.launch'"
```

Exercise

- Try to navigate the real turtlebot using the 03hostpc_keyboard_teleop.sh