Moving the Robot

Estimated time to completion: **18 minutes**

5.8 Rotate the laser model by velocity around the Z-axis

Now, do the same thing, but you will add a **velocity controller** to simulate the rotation of the RPlidar. Again, it will only be cosmetic because adding the RPlidar sensor laser** will not move it. However, velocity control is key for many robot applications.

Create the needed files:

▶ Execute in Terminal 1

In [ ]:

```
cd ~/ros2_ws/src
```

In [ ]:

```
touch my_box_bot_gazebo/launch/control_position_velocity.launch.py
```

In [ ]:

```
touch my_box_bot_gazebo/launch/spawn_robot_ros2_control_velocity.launch.xml
```

In [ ]:

```
touch my_box_bot_description/config/controller_position_velocity.yaml
```

In [ ]:

```
touch my_box_bot_description/launch/urdf_visualize_control_velocity.launch.py
```

In [ ]:

```
touch my_box_bot_description/urdf/box_bot_control_complete_velocity.urdf
```

Fill in the content for the **ROS2_control** configuration file according to the following template:

controller_position_velocity.yaml

In [ ]:

```
controller_manager:
  ros__parameters:
    update_rate: 100  # Hz

    joint_trajectory_controller:
      type: joint_trajectory_controller/JointTrajectoryController

    velocity_controller:
      type: velocity_controllers/JointGroupVelocityController

    joint_state_broadcaster:
      type: joint_state_broadcaster/JointStateBroadcaster

joint_trajectory_controller:
  ros__parameters:
    joints:
      - laser_scan_link_joint
    interface_name: position
    command_interfaces:
      - position
    state_interfaces:
      - position
      - velocity

velocity_controller:
  ros__parameters:
    joints:
      - laser_scan_model_link_joint
    interface_name: velocity
    command_interfaces:
      - velocity
    state_interfaces:
      - position
      - velocity
```

You have now defined a controller to be used by the **controller_manager**, using the **velocity_controllers/JointGroupVelocityController**. Note that this must also be installed if you want to execute this on your local machine. However, in this course, everything is installed already. That is why, in the previous example, you added the dependency in the **CMakeLists.txt** of the `velocity_controllers` for this section:

In [ ]:

```
velocity_controller:
      type: velocity_controllers/JointGroupVelocityController
```

Define its values, like the **interface_name** you named **velocity**, the same name you place in the **URDF**.

In [ ]:

```
velocity_controller:
  ros__parameters:
    joints:
      - laser_scan_model_link_joint
    interface_name: velocity
    command_interfaces:
      - velocity
    state_interfaces:
      - position
      - velocity
```

Here is how you might write out the launch file that will run the controller:

control_position_velocity.launch.py

In [ ]:

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
from launch_ros.actions import Node
from launch import LaunchDescription


# this is the function launch  system will look for


def generate_launch_description():


    spawn_controller = Node(
        package="controller_manager",
        executable="spawner",
        arguments=["joint_state_broadcaster"],
        output="screen",
    )

    spawn_controller_traj = Node(
        package="controller_manager",
        executable="spawner",
        arguments=["joint_trajectory_controller"],
        output="screen",
    )

    spawn_controller_velocity = Node(
        package="controller_manager",
        executable="spawner",
        arguments=["velocity_controller"],
        output="screen",
    )



    # create and return launch description object
    return LaunchDescription(
        [
            spawn_controller,
            spawn_controller_traj,
            spawn_controller_velocity
        ]
    )
```

Load a new controller named **velocity_controller**, defined in the **YAML file**:

In [ ]:

```
spawn_controller_velocity = Node(
        package="controller_manager",
        executable="spawner",
        arguments=["velocity_controller"],
        output="screen",
    )
```

And the main launch file:

spawn_robot_ros2_control_velocity.launch.xml

In [ ]:

```
<?xml version='1.0' ?>
<launch>
    <!-- Publish URDF file in robot_description topic -->
    <include file="$(find-pkg-share my_box_bot_description)/launch/urdf_visualize_control_velocity.launch.py"/>
    <!-- Read robot_description and spawn in gazebo running sim -->
    <include file="$(find-pkg-share my_box_bot_gazebo)/launch/spawn_robot_description.launch.py"/>
    <!-- Load the controllers -->
    <include file="$(find-pkg-share my_box_bot_gazebo)/launch/control_position_velocity.launch.py"/>
</launch>
```

This launch file is to run the robot state publisher and start Rviz configured:

urdf_visualize_control_velocity.launch.py

In [ ]:

```python
import os

from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.substitutions import Command
from launch_ros.actions import Node

# this is the function launch  system will look for
def generate_launch_description():

    ####### DATA INPUT ##########
    urdf_file = 'box_bot_control_complete_velocity.urdf'
    #xacro_file = "box_bot.xacro"
    package_description = "my_box_bot_description"

    ####### DATA INPUT END ##########
    print("Fetching URDF ==>")
    robot_desc_path = os.path.join(get_package_share_directory(package_description), "urdf", urdf_file)

    # Robot State Publisher

    robot_state_publisher_node = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        name='my_robot_state_publisher_node',
        emulate_tty=True,
        parameters=[{'use_sim_time': True, 'robot_description': Command(['xacro ', robot_desc_path])}],
        output="screen"
    )

    # RVIZ Configuration
    rviz_config_dir = os.path.join(get_package_share_directory(package_description), 'rviz', 'urdf_vis.rviz')


    rviz_node = Node(
            package='rviz2',
            executable='rviz2',
            output='screen',
            name='rviz_node',
            parameters=[{'use_sim_time': True}],
            arguments=['-d', rviz_config_dir])

    # create and return launch description object
    return LaunchDescription(
        [
            robot_state_publisher_node,
            rviz_node
        ]
    )
```

The robot description file:

box_bot_control_complete_velocity.urdf

In [ ]:

```
<?xml version="1.0"?>
<robot name="box_bot">

  <material name="red">
      <color rgba="1.0 0.0 0.0 1"/>
  </material>

  <material name="green_light">
      <color rgba="0.0 1.0 0.0 1"/>
  </material>

  <material name="green_dark">
    <color rgba="0.0 0.5 0.0 1"/>
  </material>

  <material name="blue">
      <color rgba="0.0 0.0 1.0 1"/>
  </material>

  <link name="base_link">
  </link>


  <!-- Body -->
  <link name="chassis">
    <visual>
      <geometry>
        <mesh filename="package://my_box_bot_description/meshes/cute_cube.dae" scale="0.1 0.1 0.1"/>
      </geometry>
    </visual>

    <collision>
      <geometry>
        <box size="0.1 0.1 0.1"/>
      </geometry>
    </collision>

    <inertial>
      <mass value="0.5"/>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <inertia ixx="0.0008333333333333335" ixy="0" ixz="0" iyy="0.0008333333333333335" iyz="0" izz="0.0008333333333333335"/>
    </inertial>

  </link>

  <joint name="base_link_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0" />
    <parent link="base_link" />
    <child link="chassis" />
  </joint>

  <!-- Wheel Left -->
  <link name="left_wheel">
      <visual>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.035"/>
        </geometry>
        <material name="red"/>
      </visual>

      <collision>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.035"/>
        </geometry>
      </collision>

      <inertial>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <mass value="0.05"/>
        <inertia ixx="1.531666666666667e-05" ixy="0" ixz="0" iyy="1.531666666666667e-05" iyz="0" izz="3.0625000000000006e-05"/>
      </inertial>

  </link>

  <gazebo reference="left_wheel">
    <kp>1000000000000000000000000000.0</kp>
    <kd>1000000000000000000000000000.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Green</material>
  </gazebo>


  <joint name="joint_left_wheel" type="continuous">
    <origin rpy="0 0 0" xyz="0 0.05 -0.025"/>
    <child link="left_wheel"/>
    <parent link="chassis"/>
    <axis rpy="0 0 0" xyz="0 1 0"/>
    <limit effort="10000" velocity="1000"/>
    <joint_properties damping="1.0" friction="1.0"/>
  </joint>

  <!-- Wheel Right -->
  <link name="right_wheel">
      <visual>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.035"/>
        </geometry>
        <material name="green"/>
      </visual>

      <collision>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.035"/>
        </geometry>
      </collision>

      <inertial>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <mass value="0.05"/>
        <inertia ixx="1.5316666666666667e-05" ixy="0" ixz="0" iyy="1.5316666666666667e-05" iyz="0" izz="3.0625000000000006e-05"/>
      </inertial>
  </link>

  <gazebo reference="right_wheel">
```

```
    <kp>1000000000000000000000000000.0</kp>
    <kd>1000000000000000000000000000.0</kd>
    <mu1>10.0</mu1>
    <mu2>10.0</mu2>
    <material>Gazebo/Orange</material>
  </gazebo>

  <joint name="joint_right_wheel" type="continuous">
    <origin rpy="0 0 0" xyz="0 -0.05 -0.025"/>
    <child link="right_wheel"/>
    <parent link="chassis"/>
    <axis rpy="0 0 0" xyz="0 1 0"/>
    <limit effort="10000" velocity="1000"/>
    <joint_properties damping="1.0" friction="1.0"/>
  </joint>


  <!-- Caster Wheel Front -->
  <link name="front_yaw_link">
      <visual>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
        <material name="blue"/>
      </visual>

      <collision>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
      </collision>

      <inertial>
          <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
          <mass value="0.001"/>
          <inertia ixx="5.145833333333334e-09" ixy="0" ixz="0" iyy="5.145833333333334e-09" iyz="0" izz="1.0125000000000003e-08"/>
      </inertial>
```

```
</link>

<joint name="front_yaw_joint" type="continuous">
  <origin rpy="0 0 0" xyz="0.04 0 -0.05" />
  <parent link="chassis" />
  <child link="front_yaw_link" />
  <axis xyz="0 0 1" />
  <limit effort="1000.0" velocity="100.0" />
  <dynamics damping="0.0" friction="0.1"/>
</joint>

  <gazebo reference="front_yaw_link">
      <material>Gazebo/Blue</material>
  </gazebo>



<link name="front_roll_link">
    <visual>
      <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.001" radius="0.0045000000000000005"/>
      </geometry>
      <material name="red"/>
    </visual>

    <collision>
      <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.001" radius="0.0045000000000000005"/>
      </geometry>
    </collision>

    <inertial>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <mass value="0.001"/>
        <inertia ixx="5.145833333333334e-09" ixy="0" ixz="0" iyy="5.145833333333334e-09" iyz="0" izz="1.0125000000000003e-08"/>
    </inertial>
</link>
```

```
<joint name="front_roll_joint" type="continuous">
  <origin rpy="0 0 0" xyz="0 0 0" />
  <parent link="front_yaw_link" />
  <child link="front_roll_link" />
  <axis xyz="1 0 0" />
  <limit effort="1000.0" velocity="100.0" />
  <dynamics damping="0.0" friction="0.1"/>
</joint>

  <gazebo reference="front_roll_link">
      <material>Gazebo/Red</material>
  </gazebo>


<link name="front_pitch_link">
  <visual>
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <geometry>
      <sphere radius="0.010"/>
    </geometry>
    <material name="green_dark"/>
  </visual>

  <collision>
    <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
    <geometry>
      <sphere radius="0.010"/>
    </geometry>
  </collision>

  <inertial>
      <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
      <mass value="0.001"/>
      <inertia ixx="4e-08" ixy="0" ixz="0" iyy="4e-08" iyz="0" izz="4e-08"/>
  </inertial>
</link>

<gazebo reference="front_pitch_link">
  <kp>1000000000000000000000000000.0</kp>
  <kd>1000000000000000000000000000.0</kd>
```

```
    <mu1>0.5</mu1>
    <mu2>0.5</mu2>
    <material>Gazebo/Purple</material>
  </gazebo>

  <joint name="front_pitch_joint" type="continuous">
    <origin rpy="0 0 0" xyz="0 0 0" />
    <parent link="front_roll_link" />
    <child link="front_pitch_link" />
    <axis xyz="0 1 0" />
    <limit effort="1000.0" velocity="100.0" />
    <dynamics damping="0.0" friction="0.1"/>
  </joint>

<!-- Caster Wheel Back -->
  <link name="back_yaw_link">
    <visual>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
        <material name="blue"/>
      </visual>

      <collision>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
      </collision>

      <inertial>
          <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
          <mass value="0.001"/>
          <inertia ixx="5.145833333333334e-09" ixy="0" ixz="0" iyy="5.145833333333334e-09" iyz="0" izz="1.0125000000000003e-08"/>
      </inertial>
  </link>

  <joint name="back_yaw_joint" type="continuous">
    <origin rpy="0 0 0" xyz="-0.04 0 -0.05" />
    <parent link="chassis" />
    <child link="back_yaw_link" />
    <axis xyz="0 0 1" />
    <limit effort="1000.0" velocity="100.0" />
    <dynamics damping="0.0" friction="0.1"/>
  </joint>

    <gazebo reference="back_yaw_link">
        <material>Gazebo/Blue</material>
    </gazebo>



  <link name="back_roll_link">
      <visual>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
        <material name="red"/>
      </visual>

      <collision>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <geometry>
          <cylinder length="0.001" radius="0.0045000000000000005"/>
        </geometry>
      </collision>

      <inertial>
          <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
          <mass value="0.001"/>
          <inertia ixx="5.145833333333334e-09" ixy="0" ixz="0" iyy="5.145833333333334e-09" iyz="0" izz="1.0125000000000003e-08"/>
      </inertial>
  </link>

  <joint name="back_roll_joint" type="continuous">
    <origin rpy="0 0 0" xyz="0 0 0" />
    <parent link="back_yaw_link" />
    <child link="back_roll_link" />
    <axis xyz="1 0 0" />
    <limit effort="1000.0" velocity="100.0" />
    <dynamics damping="0.0" friction="0.1"/>
  </joint>

    <gazebo reference="back_roll_link">
        <material>Gazebo/Red</material>
    </gazebo>



  <link name="back_pitch_link">
    <visual>
      <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
      <geometry>
        <sphere radius="0.010"/>
      </geometry>
      <material name="green_light"/>
    </visual>

    <collision>
      <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
      <geometry>
        <sphere radius="0.010"/>
      </geometry>
    </collision>

    <inertial>
        <origin rpy="0 1.5707 1.5707" xyz="0 0 0"/>
        <mass value="0.001"/>
        <inertia ixx="4e-08" ixy="0" ixz="0" iyy="4e-08" iyz="0" izz="4e-08"/>
    </inertial>
  </link>

  <gazebo reference="back_pitch_link">
    <kp>1000000000000000000000000000.0</kp>
    <kd>1000000000000000000000000000.0</kd>
    <mu1>0.5</mu1>
    <mu2>0.5</mu2>
    <material>Gazebo/Yellow</material>
```

```
</gazebo>

<joint name="back_pitch_joint" type="continuous">
  <origin rpy="0 0 0" xyz="0 0 0" />
  <parent link="back_roll_link" />
  <child link="back_pitch_link" />
  <axis xyz="0 1 0" />
  <limit effort="1000.0" velocity="100.0" />
  <dynamics damping="0.0" friction="0.1"/>
</joint>

<!-- PLUGINS -->

<!-- JOINT PUBLISHER -->
<gazebo>
  <plugin name="box_bot_joint_state" filename="libgazebo_ros_joint_state_publisher.so">
    <ros>
        <remapping>~/out:=joint_states</remapping>
    </ros>
    <update_rate>30</update_rate>

    <joint_name>joint_left_wheel</joint_name>
    <joint_name>joint_right_wheel</joint_name>
    <joint_name>front_yaw_joint</joint_name>
    <joint_name>back_yaw_joint</joint_name>
    <joint_name>front_roll_joint</joint_name>
    <joint_name>back_roll_joint</joint_name>
    <joint_name>front_pitch_joint</joint_name>
    <joint_name>back_pitch_joint</joint_name>

  </plugin>
</gazebo>

<!-- Differential drive -->
<gazebo>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">

    <!-- wheels -->
    <left_joint>joint_left_wheel</left_joint>
    <right_joint>joint_right_wheel</right_joint>
    <!-- kinematics -->
    <wheel_separation>0.1</wheel_separation>
    <wheel_diameter>0.07</wheel_diameter>

    <!-- limits -->
    <max_wheel_torque>1.0</max_wheel_torque>
    <max_wheel_acceleration>2.0</max_wheel_acceleration>

    <!-- output -->
    <publish_odom>true</publish_odom>
    <publish_odom_tf>true</publish_odom_tf>

    <odometry_frame>odom</odometry_frame>
    <robot_base_frame>base_link</robot_base_frame>

  </plugin>
</gazebo>


<!-- Laser Position Control-->

<link name="laser_scan_link">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <box size="0.02 0.02 0.02"/>
    </geometry>
  </visual>

  <collision>
     <origin rpy="0 0 0" xyz="0 0 0.0204"/>
    <geometry>
      <box size="0.02 0.02 0.02"/>
    </geometry>
  </collision>

  <inertial>

    <mass value="0.01"/>
```

```
      <origin rpy="0 0 0" xyz="0 0 0.0204"/>
      <inertia ixx="6.066578520833334e-06" ixy="0" ixz="0" iyy="6.072950163333333e-06" iyz="0" izz="9.365128684166666e-06"/>
    </inertial>
  </link>

  <joint name="laser_scan_link_joint" type="prismatic">
    <origin rpy="0 0 0" xyz="0.0 0.0 0.05"/>
    <parent link="chassis"/>
    <child link="laser_scan_link"/>
    <axis xyz="0 0 1"/>
    <limit lower="-0.1" upper="0.0" effort="20.0" velocity="2.0"/>
    <dynamics damping="0.1" friction="1.0"/>
  </joint>

  <link name="laser_scan_frame">
  </link>

  <joint name="laser_scan_frame_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0.03"/>
    <parent link="laser_scan_link"/>
    <child link="laser_scan_frame"/>
    <axis xyz="0 0 0"/>
  </joint>

  <!-- Visual Laser Model to be rotated -->
  <link name="laser_scan_model_link">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://my_box_bot_description/meshes/sensors/rplidar.dae" scale="1.0 1.0 1.0"/>
      </geometry>
    </visual>

    <collision>
       <origin rpy="0 0 0" xyz="0 0 0.0204"/>
      <geometry>
        <cylinder length="0.0408" radius="0.037493"/>
      </geometry>
    </collision>
    <inertial>

      <mass value="0.01"/>
      <origin rpy="0 0 0" xyz="0 0 0.0204"/>
      <inertia ixx="6.066578520833334e-06" ixy="0" ixz="0" iyy="6.072950163333333e-06" iyz="0" izz="9.365128684166666e-06"/>
    </inertial>
  </link>

  <joint name="laser_scan_model_link_joint" type="continuous">
    <origin rpy="0 0 0" xyz="0.0 0.0 0.0"/>
    <parent link="laser_scan_link"/>
    <child link="laser_scan_model_link"/>
    <axis xyz="0 0 1"/>
    <limit effort="10.0" velocity="2.0"/>
    <dynamics friction="0.01"/>
  </joint>

  <!-- Position Config -->
    <ros2_control name="GazeboSystem" type="system">
      <hardware>
        <plugin>gazebo_ros2_control/GazeboSystem</plugin>
      </hardware>

      <joint name="laser_scan_link_joint">
        <command_interface name="position">
          <param name="min">-0.05</param>
          <param name="max">0.0</param>
        </command_interface>
        <state_interface name="position"/>
        <state_interface name="velocity"/>
        <state_interface name="effort"/>
      </joint>

      <joint name="laser_scan_model_link_joint">
        <command_interface name="velocity">
          <param name="min">0.0</param>
          <param name="max">2.0</param>
        </command_interface>
        <state_interface name="position"/>
        <state_interface name="velocity"/>
        <state_interface name="effort"/>
      </joint>

  </ros2_control>


  <gazebo>
    <plugin filename="libgazebo_ros2_control.so" name="gazebo_ros2_control">
      <parameters>$(find my_box_bot_description)/config/controller_position_velocity.yaml</parameters>
      <robot_param_node>/my_robot_state_publisher_node</robot_param_node>
    </plugin>
  </gazebo>


</robot>
```

You have changed several things to allow the movement of the **prismatic joint** and the **rotation of the 3D model by velocity**. Still, the sensors attached do not rotate with the 3D model because you do not need that in simulation. So you can have a 360º degree laser without having to do that.

In [ ]:

```
<link name="laser_scan_link">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 0"/>
      <geometry>
        <box size="0.02 0.02 0.02"/>
      </geometry>
    </visual>

    <collision>
       <origin rpy="0 0 0" xyz="0 0 0.0204"/>
      <geometry>
        <box size="0.02 0.02 0.02"/>
      </geometry>
    </collision>

    <inertial>

      <mass value="0.01"/>
      <origin rpy="0 0 0" xyz="0 0 0.0204"/>
      <inertia ixx="6.066578520833334e-06" ixy="0" ixz="0" iyy="6.072950163333333e-06" iyz="0" izz="9.365128684166666e-06"/>
    </inertial>
  </link>
```

Now, the **laser_scan_link** is a tiny box linked to the chassis. You will not see it, but you give it a visual, collision, and inertia to behave better with physics in the simulator.

In [ ]:

```
<!-- Visual Laser Model to be rotated -->
<link name="laser_scan_model_link">
  <visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://my_box_bot_description/meshes/sensors/rplidar.dae" scale="1.0 1.0 1.0"/>
    </geometry>
  </visual>

  <collision>
     <origin rpy="0 0 0" xyz="0 0 0.0204"/>
    <geometry>
      <cylinder length="0.0408" radius="0.037493"/>
    </geometry>
  </collision>

  <inertial>

    <mass value="0.01"/>
    <origin rpy="0 0 0" xyz="0 0 0.0204"/>
    <inertia ixx="6.066578520833334e-06" ixy="0" ixz="0" iyy="6.072950163333333e-06" iyz="0" izz="9.365128684166666e-06"/>
  </inertial>
</link>

<joint name="laser_scan_model_link_joint" type="continuous">
  <origin rpy="0 0 0" xyz="0.0 0.0 0.0"/>
  <parent link="laser_scan_link"/>
  <child link="laser_scan_model_link"/>
  <axis xyz="0 0 1"/>
  <limit effort="10.0" velocity="2.0"/>
  <dynamics friction="0.01"/>
</joint>
```

It is the **laser_scan_model_link** that you rotate with **velocity control**. It is attached to the **laser_scan_link** to go up and down with the prismatic joint.

In [ ]:

```
<ros2_control name="GazeboSystem" type="system">
      <hardware>
        <plugin>gazebo_ros2_control/GazeboSystem</plugin>
      </hardware>

      <joint name="laser_scan_link_joint">
        <command_interface name="position">
          <param name="min">-0.05</param>
          <param name="max">0.0</param>
        </command_interface>
        <state_interface name="position"/>
        <state_interface name="velocity"/>
        <state_interface name="effort"/>
      </joint>

      <joint name="laser_scan_model_link_joint">
        <command_interface name="velocity">
          <param name="min">0.0</param>
          <param name="max">2.0</param>
        </command_interface>
        <state_interface name="position"/>
        <state_interface name="velocity"/>
        <state_interface name="effort"/>
      </joint>

  </ros2_control>
```

Here, add the new joint **laser_scan_model_link_joint**. You are setting:

- min and max velocities
- In this case, the interface's name is **velocity**, but the name could be anything here. Be sure to name it the same way in the YAML file.

In [ ]:

```
  <gazebo>
    <plugin filename="libgazebo_ros2_control.so" name="gazebo_ros2_control">
      <parameters>$(find my_box_bot_description)/config/controller_position_velocity.yaml</parameters>
      <robot_param_node>/my_robot_state_publisher_node</robot_param_node>
    </plugin>
  </gazebo>
```

- Load the new **controller_position_velocity.yaml** to use both **position and velocity controllers**.
- Do not forget to set the `robot_param_node` name to the one of the **Robot State Publisher**.

Launch and see the result:

▶ Execute in Terminal 1

In [ ]:
```
cd ~/ros2_ws
```

In [ ]:
```
colcon build
```

In [ ]:
```
source install/setup.bash
```

In [ ]:
```
ros2 launch my_box_bot_gazebo start_world.launch.py
```

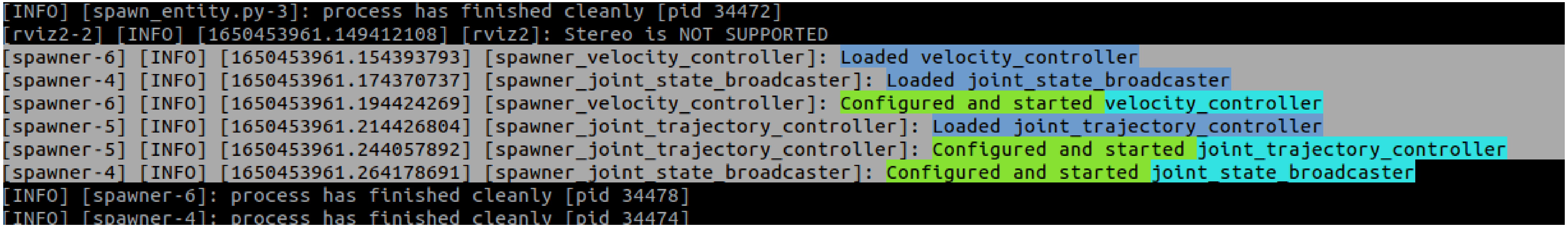
▶ Execute in Terminal 2

In [ ]:
```
cd ~/ros2_ws
```

In [ ]:
```
source install/setup.bash
```

In [ ]:
```
ros2 launch my_box_bot_gazebo spawn_robot_ros2_control_velocity.launch.xml
```

These messages are a good indication that the **plugins loaded correctly**:



```
[INFO] [spawn_entity.py-3]: process has finished cleanly [pid 34472]
[rviz2-2] [INFO] [1650453961.149412108] [rviz2]: Stereo is NOT SUPPORTED
[spawner-6] [INFO] [1650453961.154393793] [spawner_velocity_controller]: Loaded velocity_controller
[spawner-4] [INFO] [1650453961.174370737] [spawner_joint_state_broadcaster]: Loaded joint_state_broadcaster
[spawner-6] [INFO] [1650453961.194424269] [spawner_velocity_controller]: Configured and started velocity_controller
[spawner-5] [INFO] [1650453961.214426804] [spawner_joint_trajectory_controller]: Loaded joint_trajectory_controller
[spawner-5] [INFO] [1650453961.244057892] [spawner_joint_trajectory_controller]: Configured and started joint_trajectory_controller
[spawner-4] [INFO] [1650453961.264178691] [spawner_joint_state_broadcaster]: Configured and started joint_state_broadcaster
[INFO] [spawner-6]: process has finished cleanly [pid 34478]
[INFO] [spawner-4]: process has finished cleanly [pid 34474]
```

▶ Execute in Terminal 3

In [ ]:

```
cd ~/ros2_ws
```

In [ ]:

```
source install/setup.bash
```

In [ ]:

```
ros2 topic info /velocity_controller/commands
```

You get the proto message to send the speed to the velocity control topic:

In [ ]:

```
ros2 interface proto std_msgs/msg/Float64MultiArray
```

Send the speed value to start moving the lidar:

In [ ]:

```
ros2 topic pub /velocity_controller/commands std_msgs/msg/Float64MultiArray "layout:
  dim: []
  data_offset: 0
data: [20.0]
"
```

Move in the other direction:

In [ ]:

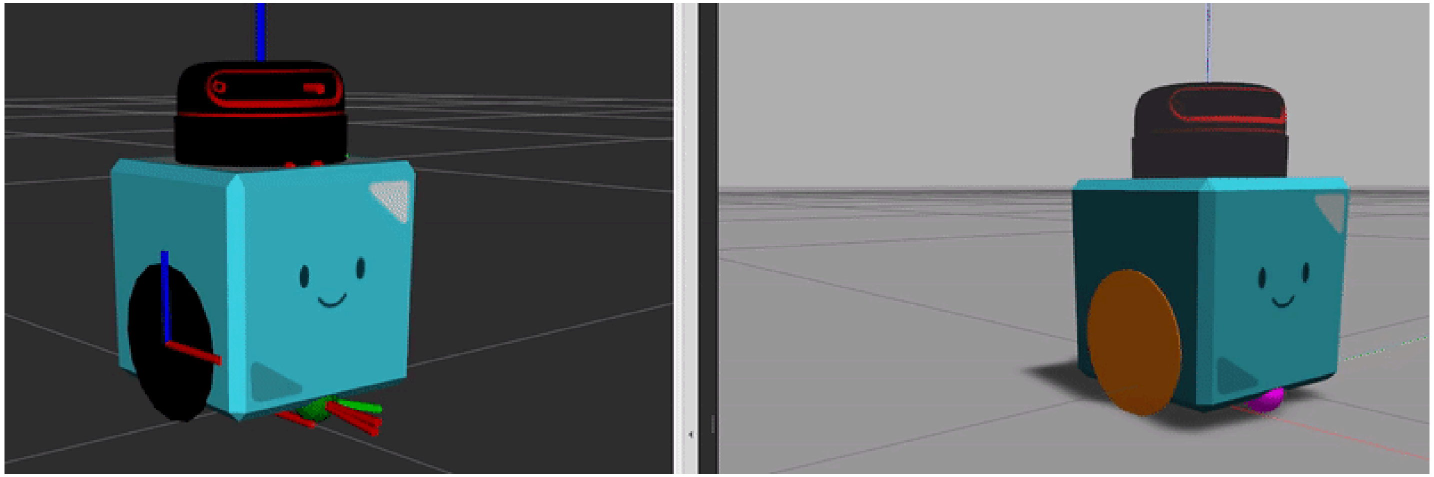
```
ros2 topic pub /velocity_controller/commands std_msgs/msg/Float64MultiArray "layout:
  dim: []
  data_offset: 0
data: [-20.0]
"
```

To stop:

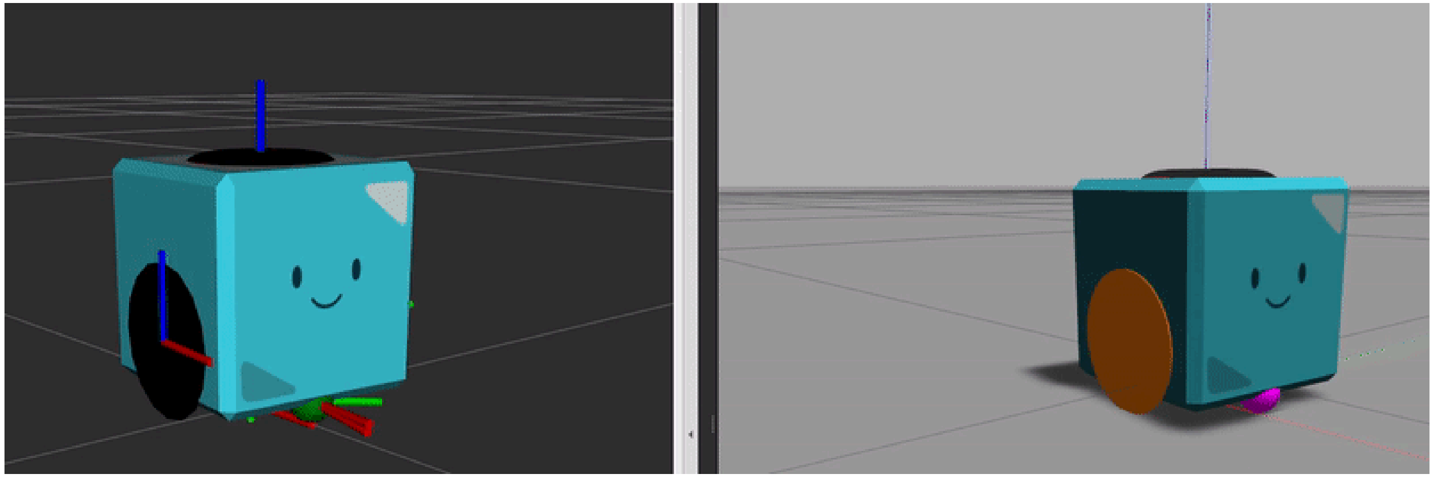
In [ ]:

```
ros2 topic pub /velocity_controller/commands std_msgs/msg/Float64MultiArray "layout:
  dim: []
  data_offset: 0
data: [0.0]
"
```

You should see something similar to this:



You can also test that the position script works simultaneously:



Check the **ROS2_Control Framework** course to learn more about **ros2_control**, including the step-by-step process of writing a new hardware interface required to use it on a real robot.

As for the current robot modeling course, it is time to add sensors to your robot's model.

ENGLISH PROOFREAD

16/11/2023