

Robot Operating System

Chapter 3

Simulation, Sensing and Visualization

Pya Soan Aung

RHCSA

Rom Robitics Co.ltd

Outline

- Unified Robot Description Format
- Gazebo
 - Add robot
 - Using gazebo plugin
 - Add mobile robot
 - Add robot arm on Mobile robot
- Plotting scalar data & Viewing images
- Rviz
 - Viewing some types of data
 - Creating marker
 - Interactive marker

Universal Robot Description Format(urdf)

- Robot တွေအတွက် simulation လုပ်ဖို့ဖန်တီးထားတဲ့ model ဖြစ်ပြီး xml format ဖြစ်တယ်။
- Urdf ထဲမှာ simulation လုပ်ဖို့ visual ဖော်ပြန်အတွက် kinematic, dynamic, collision, visual model တွေပါရှိတယ်။ လက်နဲ့ချရေးဖို့မလွယ် solidwork လိုဒ်နိုင်း software သုံးပြီး ထုတ်လုပ်နိုင်ပါတယ်။ Simulation software ရဲ့ dynamic engine ဟာ အထက်ပါ data တွေကိုထည့်သွင်းတွက်ချက်မှုဖြစ်ပါတယ်။
- Robot ရဲ့ part တွေကို joint တွေနဲ့ ဆက်ထားပေးရမှာဖြစ်ပြီး joints = number of links - 1 ဖြစ်ပါတယ်။
- Urdf မှာ open chain ကိုပဲခွင့်ပြပြီးတော့ close chain ကိုခွင့်မပဲ။(parent တရာ့သာရှိနိုင်တယ်။)

```
<? xml version="1.0"?>
<robot>
    <link />
    <joint />
</robot>
```

chapter3/example.urdf

Universal Robot Description Format(urdf)

- Visual tag ကတေသ့ visual ဖော်ပြန်အတွက်ဖြစ်ပြီး 3D ပုံဆွဲပါး <include> နဲ့ ထည့်ပေးနိုင်ပါတယ်။ dynamics တွက်ချက်မှုအတွက် inertial, collision ဆိုတဲ့ tag တွေသုံးပါတယ်။
- Link တစ်ခုရဲ့ အလယ် ဗဟိုမှာ x,y,z axis ခုခုပါတဲ့ frame တစ်ခုသတ်မှတ်ပေးထားပါတယ်။ သူရဲ့ position နဲ့ orientation ကို <origin> tag မှာဖော်ပြထားပါတယ်။
- <axis xyz="0 1 0"/> ဆိုတာကတေသ့ Y-axis မှာ rotate ဖြစ်ပါမယ်။

```
<link name="link1">
  <visual>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
  </visual>
  <inertial></inertial>
  <collision></collision>
</link>
```

```
1  <?xml version="1.0"?>
2
3      <robot name="one_DOF_robot">
4          <!-- Base link-->
5          <link name="link1"/>
6          <link name="link2"/>
7
8          <joint name="joint1" type="continuous">
9              <parent link="link1"/>
10             <parent link="link2"/>
11             <origin xyz="0 0 1" rpy="0 0 0"/>
12             <axis xyz="0 1 0">
13             </joint>
14         </robot>
```

- ပုံမှာ ဆိုရင် link ဂုဏ် joint တစ်ခုနဲ့ဆက်ထားပါတယ်။ joint type တွေကိုတွေ့ fixed, continuous, revolute, prismatic စသဖြင့်ရှိပါတယ်။

နှုတ်ပုံစံ

```
<joint name="wheel_left_joint" type="continuous">
  <parent link="base_link"/>
  <child link="chefbot_wheel_left_link"/>
  <origin xyz="0 ${0.28/2} 0.026" rpy="${-M_PI/2} 0 0"/>
  <axis xyz="0 0 1"/>
</joint>
<link name="chefbot_wheel_left_link">
  <visual>
    <geometry>
      <mesh filename="package://chefbot_description/meshes/wheel.dae"/>
    </geometry>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.0206" radius="0.0352"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0"/>
  </collision>
  <inertial>
    <mass value="0.01" />
    <origin xyz="0 0 0" />
    <inertia ixx="0.001" ixy="0.0" ixz="0.0"
              iyy="0.001" iyz="0.0"
              izz="0.001" />
  </inertial>
</link>
```

```

<robot name="kobuki" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:include filename="$(find kobuki_description)/urdf/
common_properties.urdf.xacro"/>
  <xacro:include filename="$(find kobuki_description)/urdf/
kobuki_gazebo.urdf.xacro"/>

  <!-- Kobuki -->
  <xacro:macro name="kobuki">
    <link name="base_footprint"/>
  <!--
      Base link is set at the bottom of the base mould.
      This is done to be compatible with the way base link
      was configured for turtlebot 1. Refer to

```

- Urdf ကို ဖန်တီးတဲ့အခါ xml macro ပုံစံ xacro ပုံစံနဲ့လည်းရေးပါတယ်။ ဥပမာ robot မှာ ဘီးငှါးပါတယ်ဆို ရင် ဘီး သလုံးစာပဲ xml နဲ့ရေးပီးတော့ xacro နဲ့ အဲဒီ tag တွေကို ပြန်လည်အသုံးပြုတာပါ။ Urdf ကို မိမိဘယာ ရေးဖို့ ခက်ခဲတောင်းကြောင့် ပုံစံ software အသုံးပြုပြီး export ထုတ်သင့်ပါတယ်။

- Ros မှာ urdf ကို robot_description ထဲမှာ ထားလေ့ရှိကြပြီး node တွေကို launch မလုပ်ခင် urdf ဖိုင်တစ်ခုလုံးကို Parameter Server ပေါ်တင်ပေးရပါတယ်။သူကို
 - Rviz
 - Gazebo
 - robot_state_publisher node တိုက ရယူပြီး ကနဦးတွက်ချက်မှုတွေလုပ်ပါတယ်။

```

<?xml version="1.0"?>
<launch>
  <arg name="urdf_file" default="$(find xacro)/xacro.py '$(find
chefbot_description)/urdf/chefbot_base.xacro'" />
  <param name="robot_description" command="$(arg urdf_file)" />
</launch>
  
```

- အပေါက launch ဖိုင်မှာ xacro နဲ့ရေးထားတဲ့ urdf ဖိုင်ကို urdf ဖြည့်ဖို့ xacro package ကို အရင်ရှာပီးမှ python node နဲ့ urdf ဖြည့်ပြီးမှ parameter server ပေါ်တင်ပါတယ်။ xacro ကိုမသုံးပဲ urdf ဖိုင်ဘဲဆို ရင်တော့ \$(find xacro)... စတာတွေမလိုပါဘူး။

Xacro የ urdf ፈይል

\$ rosrun xacro xacro yourxacro.xacro > yoururdf.urdf

```
ls
.xacro  kobuki_gazebo.urdf.xacro  kobuki_standalone.urdf.xacro  kobuki.urdf.xacro
rosrun xacro xacro kobuki_standalone.urdf.xacro > myurdf.urdf
ls
.xacro  kobuki_gazebo.urdf.xacro  kobuki_standalone.urdf.xacro  kobuki.urdf.xacro  myurdf.urdf
█
```

Urdf ቅኑን ይሸፍ ነው፡፡

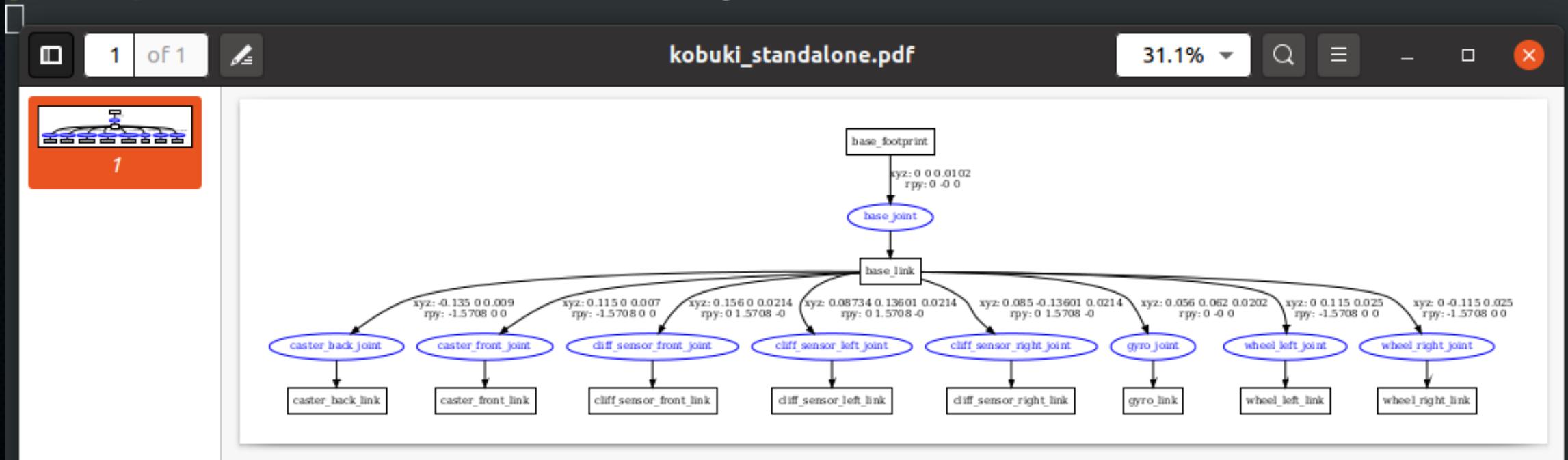
\$ check_urdf yoururdf.urdf

```
ghostman@Dell:~/urdf$ check_urdf myurdf.urdf
robot name is: kobuki_standalone
----- Successfully Parsed XML -----
root Link: base_footprint has 1 child(ren)
    child(1): base_link
        child(1): caster_back_link
        child(2): caster_front_link
        child(3): cliff_sensor_front_link
        child(4): cliff_sensor_left_link
        child(5): cliff_sensor_right_link
        child(6): gyro_link
        child(7): wheel_left_link
        child(8): wheel_right_link
ghostman@Dell:~/urdf$ █
```

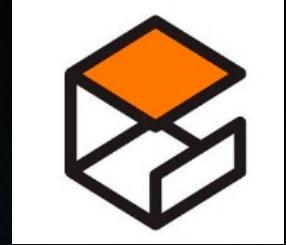
Urdf ရဲ link , joint များကို စစ်ဆေးပုံ

\$ urdf_to_graphviz myurdf.urdf

```
ghostman@Dell:~/urdf$ urdf_to_graphviz myurdf.urdf
Created file kobuki_standalone.gv
Created file kobuki_standalone.pdf
ghostman@Dell:~/urdf$ evince kobuki_standalone.pdf
```



Gazebo Simulator

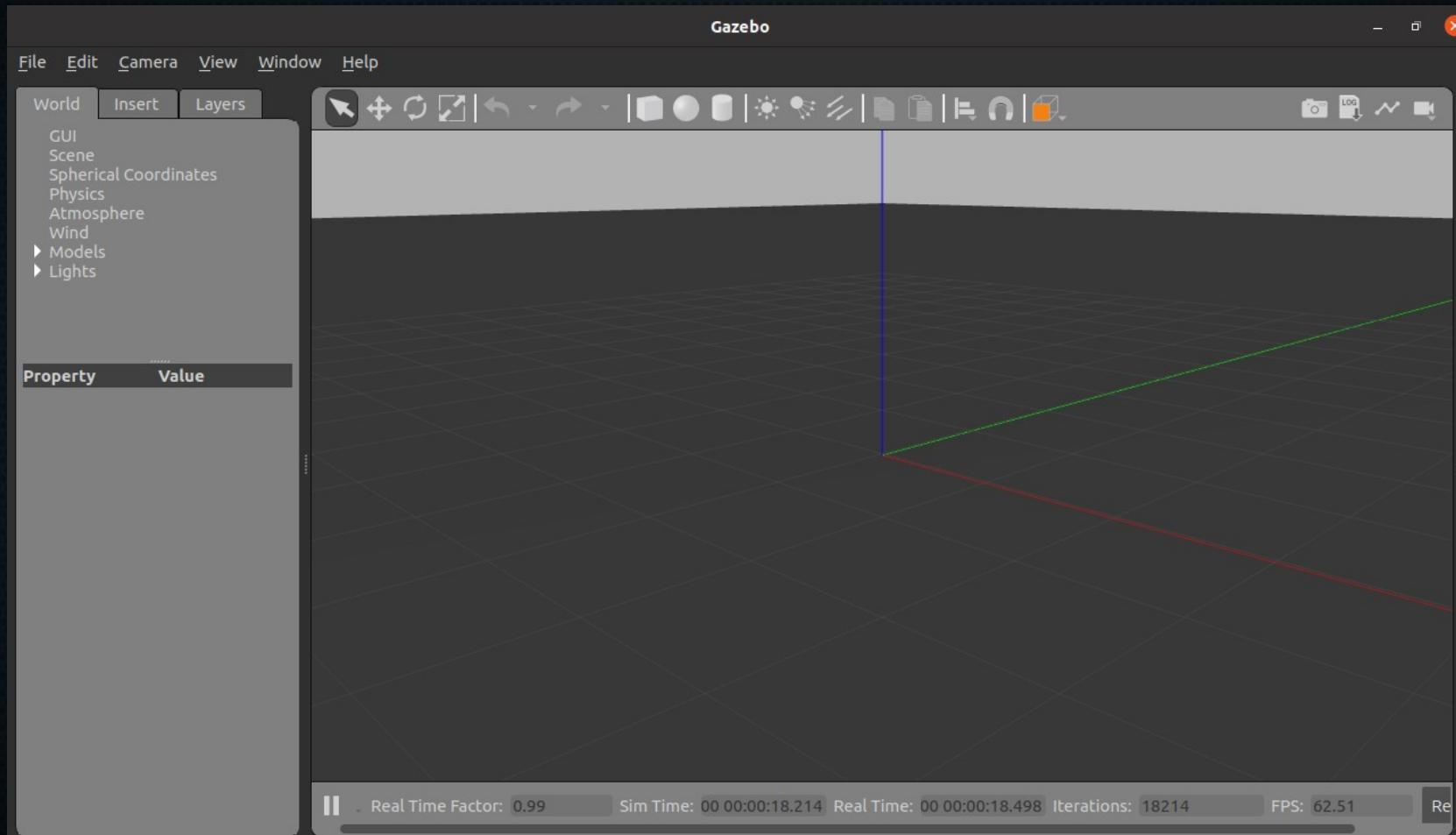


- Robot Operating System ရဲ့ အဓိကအားသာချက်တစ်ခုက open source simulator Gazebo
- Gazebo ဟာ ROS နဲ့အတူပါလာတဲ့ simulator ဖြစ်ပြီး hardware robot တည်ဆောက်ဖို့မလိုဘဲ simulation robot ကို program ရေးနိုင်ပါတယ်။
- Robot model ထည့်သွင်းလို့ရသလို Enviroment ကိုလည်းစိတ်တိုင်းကျေတည်ဆောက်နိုင်တယ်။
- Robot or Models များကိုထည့်သွင်းတဲ့အခါ CLI နဲ့ launch ဖိုင်များကတဆင့်ထည့်သွင်းနိုင်ပါတယ်။ command line ရည်တဲ့အတွက် launch ဖိုင်အသုံးပြုပြီးတော့ simulation world ထဲကို models များထည့်သွင်းသင့်ပါတယ်။
- Sensor (force,accelero,sonar,camera,lidar..) များကိုလည်း simulation လုပ်ပြီး data ရယူနိုင်တယ်။
- Gazebo simulation အတွက် gz_server နဲ့ gz_client နှစ်ခုပါဝင်ပြီးတော့ gz_client က visual display အတွက်သုံးပါတယ်။ display မပါဘဲ Headless အသုံးပြုလိုပါတယ်။
- လက်ရှိ ROS နဲ့အတူပါလာတဲ့ Gazebo version ကိုစစ်ဆေးပြီးတော့ gpu, cpu ကိုက်ညီမှုရှိမရှိစစ်ဆေးသင့်တယ်။
- Gazebo ကို စတင်ဖွင့်လိုက်ရင် internet နဲ့ချိတ်ပြီး enviroment အတွက် models များကို download ချေနေမှာ ဖြစ်လို့ hang နေနိုင်ပါတယ်။ အောက်ပါလိပ်စာကန် model များကို download လုပ်ပြီး github instruction အတိုင်းထည့်ပါ။

https://github.com/GreenGhostMan/gazebo_models

- Gazebo simulation အတွက် world တစ်ခုဖန်တီးဖိုလိုပါတယ်။ world ထဲမှာ robot တွေ sensor တွေထည့်သွင်းအသုံးပြုနိုင်ပါတယ်။ empty world တစ်ခုဖန်တီးဖို့

`~$ roslaunch gazebo_ros empty_world.launch`



အနိရောင် က x, အစိမ်းက y, အပြာက z-axis ဖြစ်ပြီး ခုန် z direction ကြောင်းပြန်အတိုင် gravity = 9.8 ရှိပါတယ်။

- Model ထွေထည့်ပြီး gazebo ကို launch ရင်တော့ left pane က insert tab မှာ ရွေးချယ်နိုင်ပါတယ်။



- ကိုယ်တိုင်ဖန်တီးထားတဲ့ model တွေကို command line ကထည့်ရန် z axis +4 meter မှာထားပဲ

~\$ rosrun gazebo_ros spawn_model yourmodel.sdf -sdf -model model_name -x 0 -y 0 -z 4

~\$ roscd example_models/rect_prism

~\$ rosrun gazebo_ros spawn_model yourmodel.sdf -sdf -model model_name -x 0 -y 0 -z 4

ထည့်ပြီးထာနဲ့ gazebo မှာသွားကြည့်တဲ့အခါ ကုပ်တံ့ဟာ gravity ကြောင့်အေက်ကို ပြေတ်ကျလာပါမယ်။

(or)

- Launch ဖိုင်ကထည့်ပဲ

```
add_rect_prism.launch x
<launch>
  <node name="spawn_sdf" pkg="gazebo_ros" type="spawn_model" args="-file $(find
example_models)/rect_prism/model-1_4.sdf -sdf -model rect_prism -x 0 -y 0 -z 4" />
</launch>
```

Simulator ဆဲ robot ဘယ်လိုထည့်မလဲ?

နမူနာ command

```
~$ roslaunch gazebo_ros empty_world.launch  
~$ roslaunch minimal_robot_description minimal_robot_description.launch
```

စတုရန်း link နဲ့ ပိုက်လုံးရည်ပံ့ခံ link ပုံစံကို joint တစ္ဆေးဆက်ထားသည့် robot arm ပုံစံကိုတွေ့ရမယ်။
အောက်က စတုရန်း link နဲ့ Gazebo world ခဲ့ floor ကိုလည်း fixed joint နဲ့ ဆက်ထား
မှသာ ပြုတ်မကျဘဲ တည်ရှုနိုင်မှာဖြစ်ပါတယ်။ urdf သိမဟုတ် xacro မှာ ပြင်ရေးလိုပါပါတယ်။

Robot arm ကို simulation လုပ်ဖို့ဘတွေလိုအပ်သလဲ?



```
minimal_robot_description.launch x  
<launch>  
  <param name="robot_description" textfile="$(find minimal_robot_description)/minimal_robot_description.urdf"  
  />  
  
  <!-- Spawn a robot into Gazebo -->  
  <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model"  
    args="-param robot_description -urdf -model one_DOF_robot" />  
</launch>
```

Using Gazebo Plugins

Real robot arm ကို control လုပ်ဖို့ Hardware controller တွေလိုသလိုပဲ Gazebo မှာလည်း software နည်းလမ်းနဲ့ controller လေးတွေလုပ်ပေးဖို့လိုပါတယ်။
Code ရေးဖို့မလို plugin တွေကို step by step ထည့်သွင်းသွားဖို့ပဲလိုပါတယ်။

Install ပြုလုပ်ရန်

```
~$ sudo apt install ros-version-controller-interface  
~$ sudo apt install ros-version-gazebo-ros-control  
~$ sudo apt install ros-version-joint-state-publisher  
~$ sudo apt install ros-version-effort-controllers
```

Robot arm အတွက် joint တွေကို simulation ပြုလုပ်နိုင်ဖို့ဆိုရင် အောက်ပါ အဆင့် ၅ ဆင့် လုပ်ရပါမယ်။

- 1) add joint limit and torque in urdf
- 2) add transmission tags for every joints in urdf
- 3) add libgazebo_ros_control.so in urdf
- 4) yaml file
- 5) launch file

1) Add joint limit and torque in urdf

```
<joint name="joint1" type="revolute">
  <parent link="link1"/>
  <child link="link2"/>
  <origin xyz="0 0 1" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit effort="10.0" lower="0.0" upper="2.0" velocity="0.5"/>
  <dynamics damping="1.0"/>
</joint>
```

Fixed joint မဟုတ်တဲ့ (linear or revolute) လူပ်ရားရမည့် joint ကိုလိုက်ရှာပြီးအပေါ်ကအတိုင်း ဖြည့်သင့်တာတွေကတော့ limit နဲ့ Dynamics ဖြစ်ပါတယ်။

Effort ကဲ torque ဖြစ်ပြီး 10Nm ထည့်ထားတာဖြစ်ပါတယ်။ velocity ကဲ 0.5 rad per second နဲ့ lower နဲ့ upper တွေကတော့ Robot arm အတွက် သွားရမယ့် limit ကန့်သတ်ထားခြင်းဖြစ်ပါတယ်။

Dynamics damping ကတော့ linear fraction ဖြစ်ပါတယ်။

Axis ကိုကြည့်ပြီး ဒီ joint1 ဟာ Y – axis မှာ revolute ဖြစ်မှာကို သိနိုင်ပါတယ်။

ကိုယ့် project နဲ့ ကိုက်ညီအောင် ဖြည့်သွင်းနိုင်ပါတယ်။

2) Add transmission tags for every joints in urdf

ဒီတစ်ခါ ထည့်ရမည့် transmission ဆိုတာက joint တွေရဲ positon , velocity, torque စတဲ့ feedback တွေကို transmit နဲ့ Receive လုပ်ပေးမှာဖြစ်ပါတယ်။

Joint name တွေ controller name တွေမှန်အောင်ပြင်ပေးဖို့ အရေးကြီးပါတယ်။ ကျွန်ုတ်ပြင်စရာမလိုပါ။

```
<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint1">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor1">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

3) Add libgazebo_ros_control.so in urdf

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/one_DOF_robot</robotNamespace>
  </plugin>
</gazebo>
<gazebo>
  <plugin name="joint_state_publisher" filename="libgazebo_ros_joint_state_publisher.so">
    <jointName>joint1</jointName>
  </plugin>
</gazebo>
```

ဒီတခုကတော့ အလုပ်လုပ်ရမည့် driver ဖြစ်တဲ့ share object ဖိုင်ကို ထည့်ပေးရမှာပါ။ များသောအားဖြင့် ဒီလို gazebo tag တွေကို urdf ရဲ့အောက်ဆုံး code line မှာရေးကြပါတယ်။ Gazebo နဲ့ဆက်သွယ်ပြီး အခိုက အလုပ်လုပ်ပေးမည့် SO တွေဖြစ်ပါတယ်။

အမည်တူ robot ၂ခု သုံးထားရင်လည်း controller များကို namespace ပေးပြီး driver မတူအောင် ခွဲခြားနိုင်တယ်။ Joint name ကိုလည်းမှန်အောင်ထည့်ဖို့လိုပါတယ်။

4) YAML File

ROS မှာ xml သာမက Yaml format (key,value) ကိုလည်း configuration အတွက်သုံးပါတယ်။ ရည်ရွယ်ချက်ကတော့ data တွေကို parameter server ပေါ်တင်ထားပြီး runtime မှာ program တွေက ယူသုံးဖို့ရန့်အတွက်ပါ။ dynamic reconfigure ဆိုတဲ့ Node နဲ့လည်း parameter server ပေါ်က value များကို ပြင်ဆင်လို့ရပါတယ်။ Yaml ကို simulation သာမက real robot တွေ အတွက်လည်း အသုံးပြုပါတယ်။

```
one_DOF_robot:  
    # Publish all joint states -----  
    joint_state_controller:  
        type: joint_state_controller/JointStateController  
        publish_rate: 50  
  
    # Position Controllers -----  
    joint1_position_controller:  
        type: effort_controllers/JointPositionController  
        joint: joint1  
        pid: {p: 10.0, i: 10.0, d: 10.0, i_clamp_min: -10.0, i_clamp_max: 10.0}
```

သတိပြုသင့်တာက robot name (ဥပမာ one_DOF_robot) , controller name များ (ဥပမာ joint_state_controller, joint1_position_controller), နဲ့ controller type များကို urdf, launch ဖိုင်ထဲက အမည်တွေနဲ့ တူအောင် ပြုလုပ်ရပါမယ်။

5) Launch File

```
<launch>
  <!-- Load joint controller configurations from YAML file to parameter server -->
  <rosparam file="$(find minimal_robot_description)/control_config/one_dof_ctl_params.yaml" command="load"/>

  <param name="robot_description" textfile="$(find minimal_robot_description)/minimal_robot_w_jnt_pub.urdf"/>

  <!-- Spawn a robot into Gazebo -->
  <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model" args="-param robot_description -urdf -model one_DOF_robot" />

  <!--start up the controller plug-ins via the controller manager -->
  <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
    output="screen" ns="/one_DOF_robot" args="joint_state_controller joint1_position_controller"/>

</launch>
```

နောက်ဆုံးအဆင့် launch ဖိုင်ပါ။ <rosparam> က config ဖိုင်တစ္ဆုလုံးကို param server ပေါ်တင်တာဖြစ်ပြီး <param> ကတော့ Single key value အတွဲခုကို တင်တာပါ။

ပထမ Node မှာ robot ကို Gazebo ထဲထည့်ပေးမည့် spawn_node ဖြစ်ပြီး param server ပေါ်က robot_description ဆိုတဲ့ Key ကိုပဲညွှန်းထားပါတယ်။

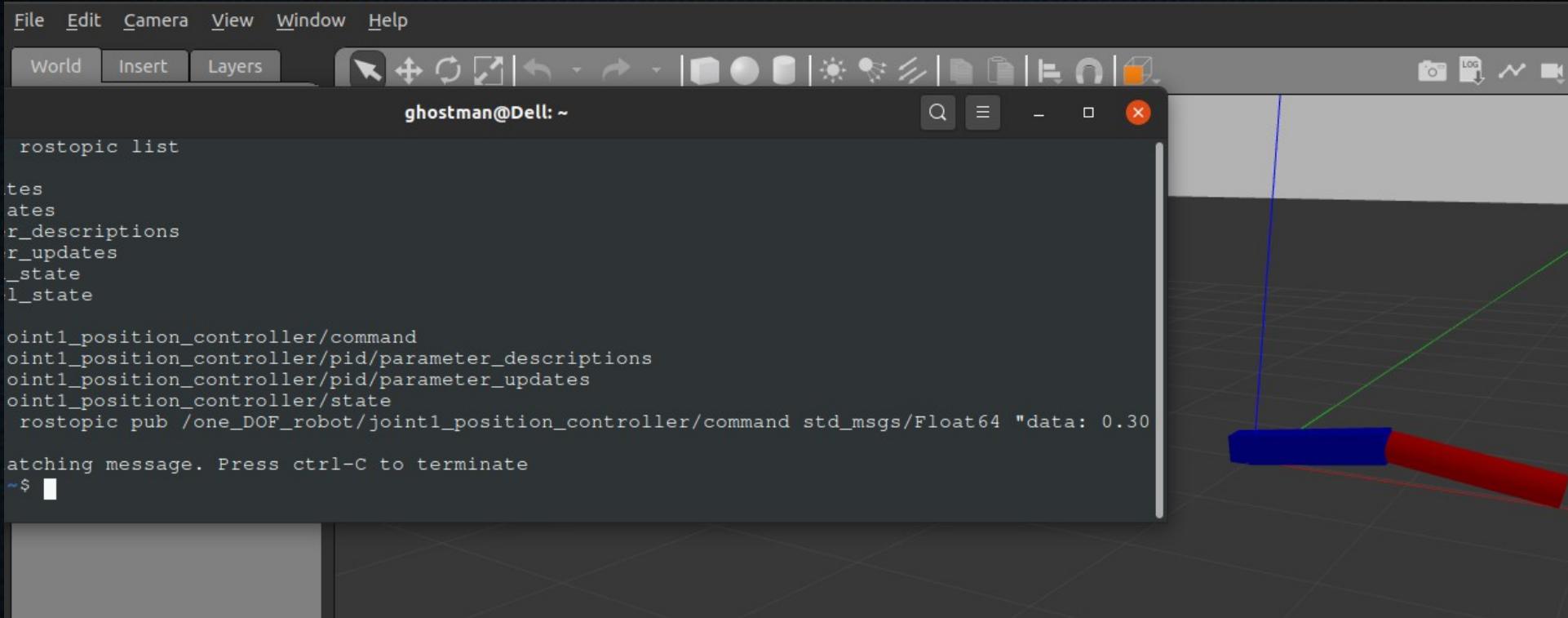
နောက် Node တစ်ခုဖြစ်တဲ့ controller manager ကတော့ args ထဲမှာပါတဲ့ controller တွေကို load လုပ်ပေးမှာဖြစ်ပြီး yaml ဖိုင် ထဲက controller အမည်များပဲဖြစ်ပါတယ်။

Namespace ဖြစ်တဲ့ ns မှာတော့ robot name ကို မှန်အောင်ဖြည့်ပေးရမှာပါ။

အားလုံးပြင်ဆင်ပြီးသွားတဲ့အောက်ပါ launch ကို run ရင်

~\$ rosrun gazebo_ros empty_world.launch

~\$ rosrun minimal_robot_description minimal_robot_w_jnt_ctl.launch



Gazebo မှာတော့ visual အရ ထူးစားမှုမရှိပါဘူး။ rostopic ခေါ်ကြည့်တဲ့အခါမှသာ real hardware robot တွေလို့ driver Topic တွေ joint command တွေပေးလို့ပြဖော်လို့ program ရေးလို့ပြဖော်ပါတယ်။ joint1 position ကို 0.3 radian ပေးကြည့်တဲ့ အခါ robot arm ပြတ်ကျသွားတာဟာ link1 နဲ့ world မဲ့ floor ကို fixed joint တာနဲ့ ဆက်မထားလို့ဖြစ်ပါတယ်။ joint1 အတွက် Pid control လုပ်ချင်ရင်လည်း dynamic reconfigure နဲ့ rqt_plot ကို အသုံးပြန်ပြီဖြစ်ပါတယ်။

Gazebo Simulator ထဲကို mobile robot သယ်လိုထည့်မလဲ?

~\$ rosrun gazebo_ros empty_world.launch

~\$ rosrun mobot_urdf mobot.launch

အရင်ဆုံး launch မလုပ်ခင် mobot ၏ xacro ဖိုင်ကိုမှာ လိုတာလေးတွေထည့်ပါမယ်။ robot arm တုန်းက urdf နဲ့ ရေးထားလို့ urdf ထဲမှာပြင်ရသလို ခု မီobile robot အတွက် xacro နဲ့ ရေးထားလို့ xacro ကိုပြင်ရမှာပါ။
မည်သို့ပင် xacro နဲ့ ရေးစေကောမူ parameter server ပေါ်တင်တဲ့အခါ urdf ပြောင်းပီး တင်ပေးပါတယ်။

- နမူနာ mobot.xacro

```
<?xml version="1.0"?>
<robot
    xmlns:xacro="http://www.ros.org/wiki/xacro" name="mobot">
    <xacro:include filename="$(find mobot_urdf)/urdf/mobot_xacro_defs.xacro" />
    <xacro:include filename="$(find mobot_urdf)/urdf/mobot_static_links.xacro" />
    <xacro:include filename="$(find mobot_urdf)/urdf/mobot_wheels.xacro" />
    <xacro:include filename="$(find mobot_urdf)/urdf/casters.xacro" />
    <xacro:include filename="$(find mobot_urdf)/urdf/gazebo_tags.xacro" />
</robot>
```

Mobile robot မှာတော့ robot arm လောက် အဆင့်မများတော့ပါ။

include လုပ်ပြီးရေးထားတဲ့ ဖိုင်များကို လော့လာကြည့်သင့်ပါတယ်။

အရင် slide မှာ Gazebo အတွက်ပြင်ဆင်မှုများကို urdf မှာသွားရေးသော်လည်း xacro မှာတော့ Gazebo အတွက် သီးခြား Configuration လုပ်ပေးမည့် gazebo_tags.xacro လို့ ခဲ့ခြားရေးသားနိုင်ပါတယ်။

mobot_xacro_defs.xacro

```
<xacro:property name="M_PI" value="3.1415926535897931" />
<xacro:property name="boschwidth" value="0.0381" />
<xacro:property name="casterdrop" value="0.125" />
<xacro:property name="bracketwidth" value="0.1175" />
<xacro:property name="bracketheight" value="0.16" />
<xacro:property name="bracketthick" value="0.0508" />
<xacro:property name="bracketangle" value="0.7854" />
<xacro:property name="casterwidth" value="0.0826" />
<xacro:property name="casterdiam" value="0.2286" />

<!--here is a default inertia matrix with small, but legal values; use this
when don't need accuracy for I -->
<!--model will assign inertia matrix dominated by main body box -->
<xacro:macro name="default_inertial" params="mass">
    <inertial>
        <mass value="${mass}" />
        <inertia ixx="0.01" ixy="0.0" ixz="0.0"
                 iyy="0.01" iyz="0.0"
                 izz="0.01" />
    </inertial>
</xacro:macro>
</robot>
```

<xacro:property .../> ဆိုတာ မကြေခဏအသုံးပြုရတဲ့ variable တွေကိုသိမ်းထားဖို့သုံးပါတယ်။

<xacro:macro> ဆိုတာကတော့ မကြေခဏအသုံးပြုရတဲ့ xml parent and child tree ကို သိမ်းထားတာပါ။

gazebo_tags.xacro

```
<?xml version="1.0"?>
<robot
    xmlns:xacro="http://www.ros.org/wiki/xacro" name="gazebo_tags">

<gazebo>
    <plugin name="differential_drive_controller" filename=
        "libgazebo_ros_diff_drive.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>100</updateRate>
        <leftJoint>right_wheel_joint</leftJoint>
        <rightJoint>left_wheel_joint</rightJoint>
        <wheelSeparation>${track}</wheelSeparation>
        <wheelDiameter>${tirediam}</wheelDiameter>
        <torque>200</torque>
        <commandTopic>cmd_vel</commandTopic>
        <odometryTopic>odom</odometryTopic>
        <odometryFrame>odom</odometryFrame>
        <robotBaseFrame>base_link</robotBaseFrame>
        <publishWheelTF>true</publishWheelTF>
        <publishWheelJointState>true</publishWheelJointState>
    </plugin>
</gazebo>

</robot>
```

ဒါကတော့ robot arm တုန်းကလို mobile robot အတွက် သူနဲ့သက်ဆိုင်ရာ driver ကိုထည့်တာဖြစ်ပါတယ်။ differential drive Mobile robot တွေအတွက် driver ဖြစ်တဲ့ libgazebo_ros_diff_drive.so ကို ထည့်ပေးရပါတယ်။ Left နဲ့ right ဘီး joint တွေကိုမှန်အောင်ထည့်ပေးရသလို သက်ဆိုင်ရာ topic name များကိုလည်းပြောင်းပေးနိုင်ပါတယ်။

- Simulator ထဲမထည့်ခင် xacro ဖိုင်ကို urdf ပြောင်းပြီး joint တွေ link တွေကို ပြင်ဆောင်ကြည့်နိုင်ပါတယ်။ Robot model တစ်ခုလုံးပါတဲ့ xacro ဖိုင်ဖြစ်ဖို့လိုပါတယ်။
- Xacro ဖိုင်ကို urdf ပြောင်းပို့ xacro package ထဲက xacro.py သို့မဟုတ် xacro ဆိုတဲ့ node ကို run ပြီး Output ကို myrobot.urdf အဖြစ်ဖန်တီးပါတယ်။
- Urdf မှာ child, parent များ မှန်ကန်မှုရှိမရှိ check_urdf ဖြင့် စစ်ဆေးသည်။
- Urdf ဖိုင်ကို urdf_to_graphviz ဖြင့် generate လုပ်ပါက mobot.gv နဲ့ robot.pdf ဖိုင်းချုပ်လည်။ Evince ဖြင့် pdf ဖိုင်ကို ကြည့်ပြီး စစ်ဆေးနိုင်ပါသည်။

~\$ roscd mobot_urdf/urdf

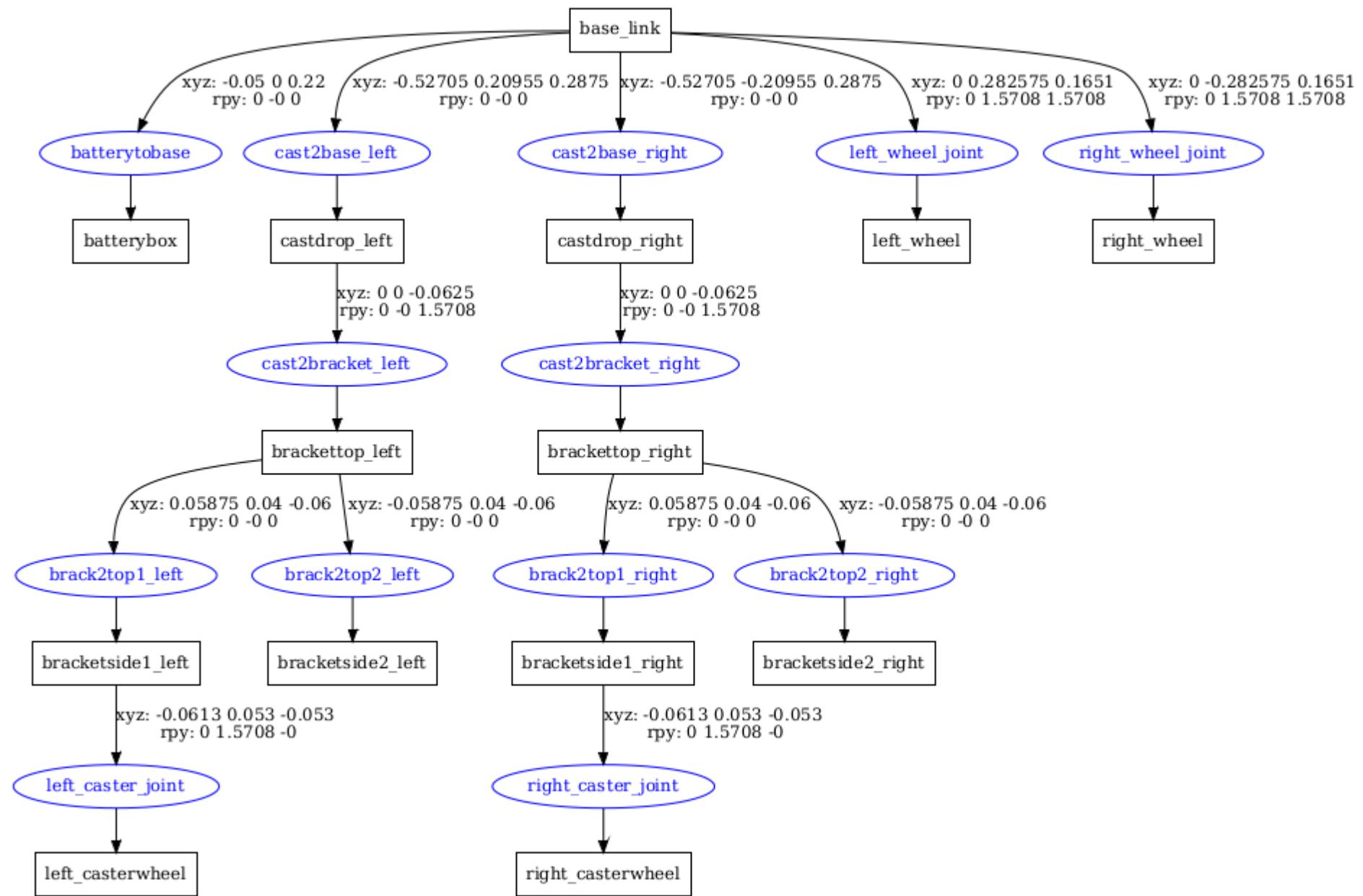
~\$ rosrun xacro xacro.py mobot.xacro > myrobot.urdf

~\$ check_urdf myrobot.urdf

~\$ urdf_to_graphviz myrobot.urdf

~\$ evince myrobot.pdf

Robot link
joint များ
တည်ဆောက်ပုံ



```
~$ roslaunch gazebo_ros empty_world.launch  
~$ roslaunch mobot_urdf mobot.launch
```

Insert tab
gas
Station
သည်ထားပုံ



cmd_vel
Topic ကို
Publish
လုပ်ပြီး Robot
ကို Drive
လုပ်နိုင်သည်။

Sensors

Sensors supported by ROS

- 1D range finders (IR, Sonar..)
- 2D range finders (Lidar..)
- 3D range finders (Lidar, 3D camera..)
- Audion Speech Recognition
- Camera
- Force, Torque, Touch..
- Motion capture device..
- Pose Estimation device (GPS, IMU..)
- Power Supply
- RFID
- Radar..

How to user these popular sensors?

- Lidar
- 3D Camera

Both Real Hardware and Simulation

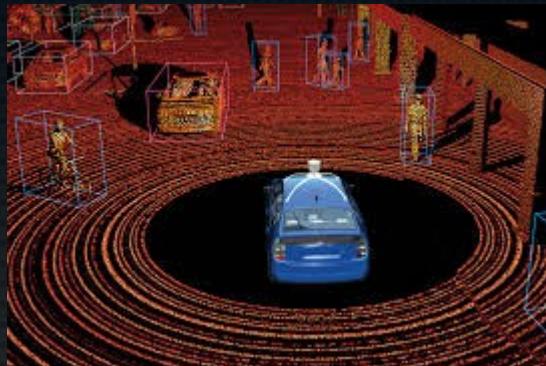
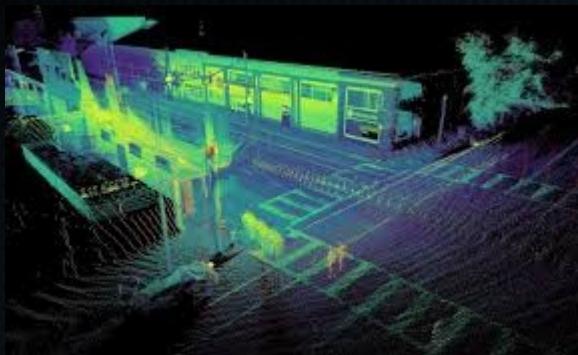
Light Detection And Ranging(LIDAR)

A1



Laser ray တစ်ခုကို ထုတ်ပြီးတော့ reflect ဖြစ်လာတဲ့ ကြာခိုန်ကို တွက်ချက်ပြီးတော့ Distance တိုင်းတာပါတယ်။
ပတ်လည် ၃၆၀ ဖတ်နှိုင်သလို 3D ဖတ်နှိုင်တဲ့ Lidar များလည်းရှိပါတယ်။ မြေပြင်၊ ဆောင်ရွက်အောက် တိုင်းတာခြင်းများမှာအသုံးပြုပါတယ်။
ရုံးလာတဲ့ Lidar data များကို gyroscope + imu တို့နဲ့ fusion လုပ်ပြီးအသုံးပြုကြပါတယ်။

Light Detection And Ranging(LIDAR)

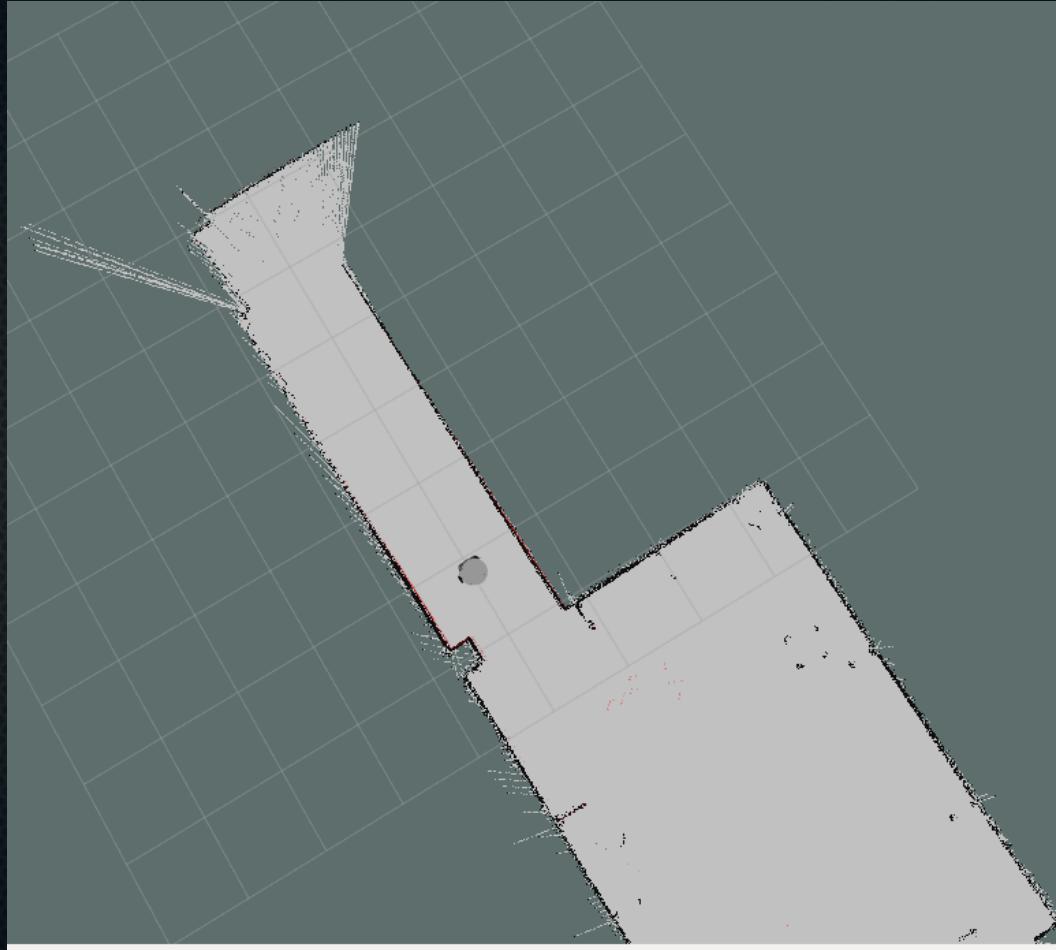


ROS နဲ့ တွဲဖက်ပြီး lidar အသုံးပြုမယ်ဆိုရင် ROS ကို Support လုပ်တဲ့ driver package တွေပါပါတယ်။ Driver ဆဲ က node များကို run ခြင်းဖြင့် စတင်အသုံးပြုနိုင်မှာဖြစ်ပါတယ်။

Lidar sensor အတွက် အသုံးပြုတဲ့ data type သည် sensor_msgs/LaserScan ဖြစ်ပြီး /scan topic ဖြင့် publish လုပ်ပေးပါတယ်။ ဥပမာ ရှေ့ slide က rplidar A1 အတွက် rplidar_ros package ကို အသုံးပြပြီး driver ကို launch လုပ်နိုင်ပြုဖြစ်ပါတယ်။

Real Hardware ဖြင့် အသုံးပြခြင်းမှာ ရိုးရှင်းလွယ်ကူပြီး ထိရောက်မှုရှိတဲ့အတွက် မောင်းသူမဲ့ ကား project တွေမှာစမ်းသပ်အသုံးပြုနိုင်ပါတယ်။

Light Detection And Ranging(LIDAR)



2019 သင်တန်းသားများ
hotel sincere smile yangon
မှာ scan ဖတ်ထားတာဖြစ်ပါတယ်။

LIDAR Simulation

Hardware lidar sensor မရှိဘဲ Gazebo မှာ simulation ပြလုပ်ပြီးအသုံးပြန်ပါတယ်။

Xacro သို့မဟုတ် urdf မှာ lidar link တစ်ခု၊ တွဲဆက်မည့် joint တစ်ခု နဲ့ driver ထည့်ပေးရမှာဖြစ်ပါတယ်။

```
<link name="lidar_link">
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>

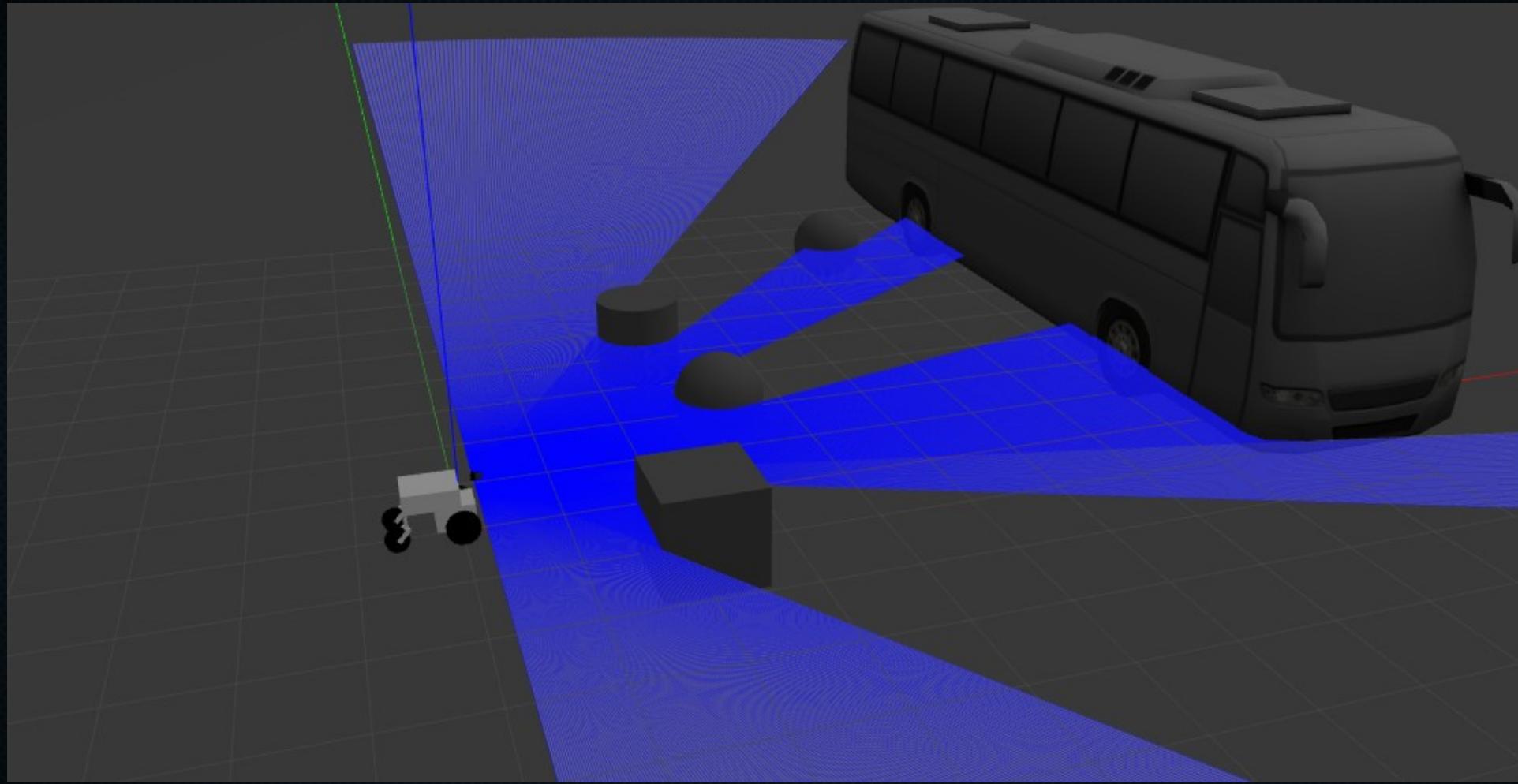
<joint name="lidar_joint" type="fixed">
  <axis xyz="0 1 0" />
  <origin xyz="0.15 0 0.56" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="lidar_link"/>
</joint>

<!--plugin name="gazebo_ros_lidar_controller"
filename="libgazebo_ros_gpu_laser.so"-->
<plugin name="gazebo_ros_head_hokuyo_controller" filename="
libgazebo_ros_laser.so">
  <topicName>/scan</topicName>
  <frameName>lidar_link</frameName>
</plugin>
```

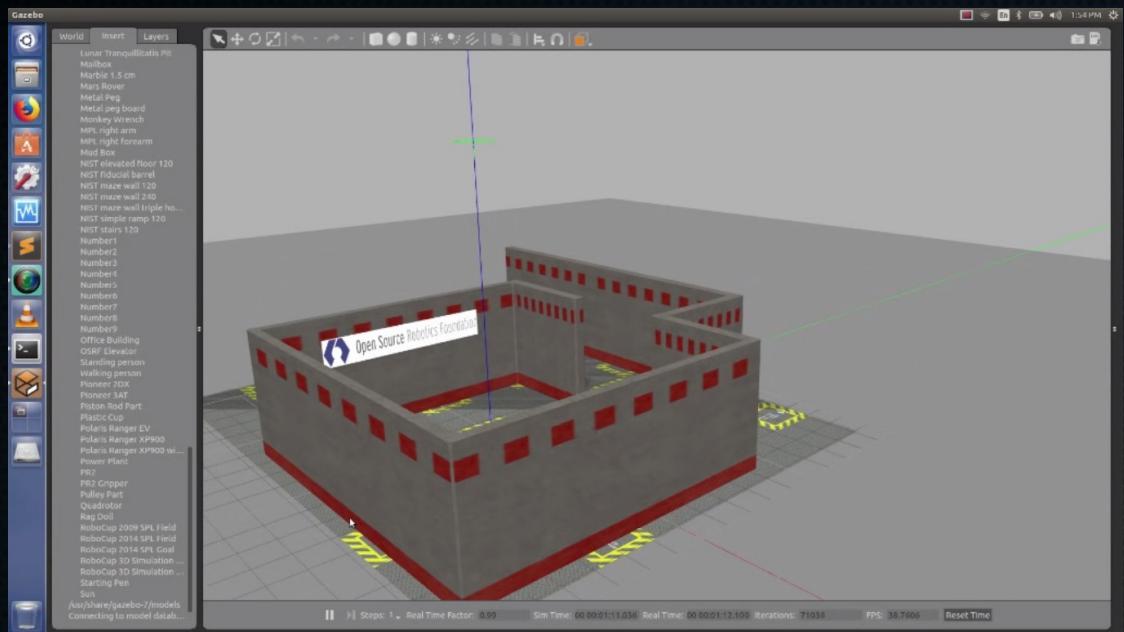
Real Hardware နဲ့သုံးမယ်ဆိုရင်တော့ ဒီအဆင့်တွေမလိုပါဘူး။

Lidar တပ်ပြီးသော်လည်း simulation လုပ်ရမှာ /scan topic မှာ data မရခဲ့ပါက libgazebo_ros_laser.so အစား libgazebo_ros_laser.so ကြောင်းပြီးစမ်းသပ်ပါ။

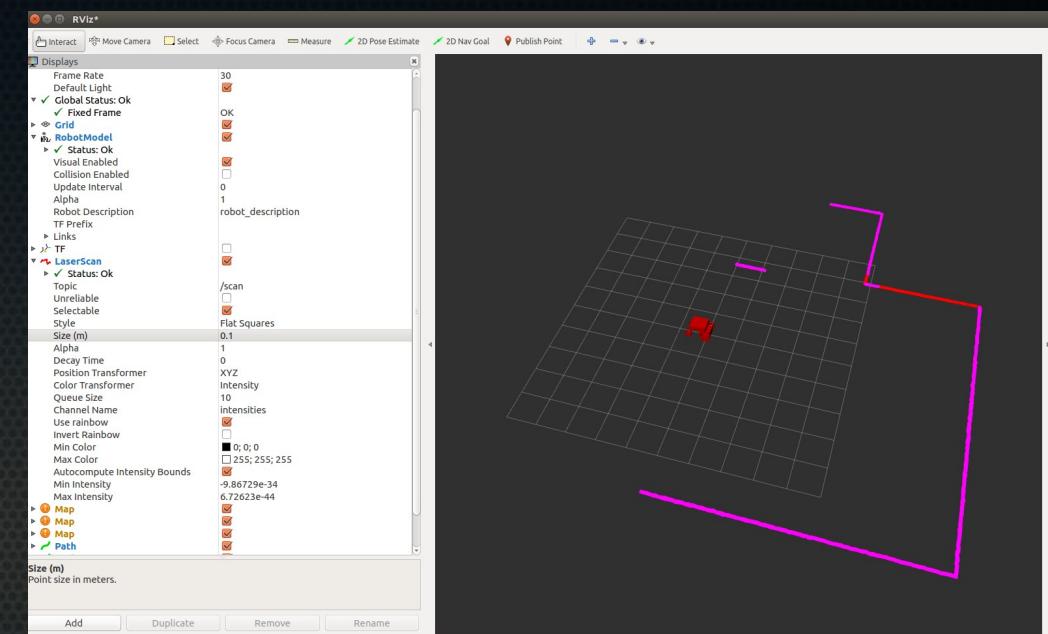
```
~$ roslaunch gazebo_ros empty_world.launch  
~$ roslaunch mobot_urdf mobot_w_lidar.launch
```



Gazebo



Rviz



Lidar sensor simulation ပြလုပ်ခြင်းကို
Pc တိုင်းတွင် အသုံးပြနိုင်မည်မဟုတ်ပါ။



3D Camera or Depth Camera



- ROS support လုပ်တဲ့ camera, stereo camera, 3D camera များစွာရှိပြီး 3D camera များသည်အသုံးပြုရလွယ်ပါတယ်။
- Image, point cloud , depth စသည့် **topic** များစွာပါဝင်ပါတယ်။
- Object Detection များတွင် အသုံးပြုကြပြီး calibration လုပ်ပေးဖို့လိုပါတယ်။
- ကင်မရာဂလုံး ir sensor နှင့် mic အပါအဝင် အခြား sensor များလဲပါဝင်နိုင်ပါတယ်။
- 3D camera မှ depth image ကို အသုံးပြုပြီး fake laser ပြုလုပ်နိုင်ပါသည်။
- Hardware အထွက် Driver package အား compile လုပ်ပြီး launch လုပ်နိုင်သည်။



Astra Camera

Install driver for Orbbec astra 3D camera

```
~$ sudo apt install ros-version-astra-camera ros-version-astra-launch  
~$ roslaunch astra_launch astra.launch
```

Body tracking with 3D camera



Camera/3D Camera Simulation

Camera

Gazebo robot တွင် Camera တပ်ဆင်လိုပါက camera link တစ်ခု joint တစ်ခုတပ်ဆင်ပေးဖို့လိုသည်။ ပြီးနောက် driver တပ်ဆင်ရန် Pulgin အဖြစ် libgazebo_ros_camera.so ကိုထည့်ပါ။

```
<!--here is the plug-in that does the work of camera emulation-->
<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>10.0</updateRate> <!--can set the publication rate-->
  <cameraName>example_camera</cameraName> <!--topics will be
example_camera/... -->
  <!--listen to the following topic name to get streaming images-->
  <imageTopicName>image_raw</imageTopicName>
  <!--the following topic carries info about the camera, e.g.
640x480, etc-->
  <cameraInfoTopicName>camera_info</cameraInfoTopicName>
  <!--frameName must match gazebo reference name...seems redundant-->
  <!-- this name will be the frame_id name in header of published
frames-->
  <frameName>camera_link</frameName>
  <distortionK1>0.0</distortionK1>
  <distortionK2>0.0</distortionK2>
  <distortionK3>0.0</distortionK3>
  <distortionT1>0.0</distortionT1>
  <distortionT2>0.0</distortionT2>
</plugin>
</sensor>
</gazebo>
```

Full code → chapter03/mobot_urdf/urdf/example_camera.xacro

3D Camera

Gazebo robot တွင် 3D Camera တပ်ဆင်လိုပါက camera link တစ်ခု joint တစ်ခုတပ်ဆင်ပေးဖို့လိုသည်။ ပြီးနောက် driver တပ်ဆင်ရန် Pulgin အဖြစ် libgazebo_ros_openni_kinect.so ကိုထည့်ပါ။

```
<!--here is the plug-in that does the work of kinect emulation-->
<plugin name="camera_controller" filename=
libgazebo_ros_openni_kinect.so>
    <alwaysOn>true</alwaysOn>
    <updateRate>10.0</updateRate>
    <cameraName>kinect</cameraName>
    <frameName>kinect_depth_frame</frameName>

    <imageTopicName>rgb/image_raw</imageTopicName>
    <depthImageTopicName>depth/image_raw</depthImageTopicName>
    <pointCloudTopicName>depth/points</pointCloudTopicName>
    <cameraInfoTopicName>rgb/camera_info</cameraInfoTopicName>

    <depthImageCameraInfoTopicName>depth/camera_info</
depthImageCameraInfoTopicName>
    <pointCloudCutoff>0.4</pointCloudCutoff>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
    <CxPrime>0.0</CxPrime>
    <Cx>0.0</Cx>
    <Cy>0.0</Cy>
    <focalLength>0.0</focalLength>
</plugin>
</sensor>
</gazebo>
```

Full code → chapter03/mobot_urdf/urdf/example_kinect.xacro

3939

```
~$ roslaunch gazebo_ros empty_world.launch  
~$ roslaunch mobot_urdf mobot_w_lidar_and_camera.launch  
Or  
~$ roslaunch mobot_urdf mobot_w_lidar_and_kinect.launch  
ကို launch လုပ်ပြီး စတင်အသုံးပြန်ပြီဖြစ်ပါသည်။
```

```
~$ roslaunch mobot_urdf mobot_w_lidar_and_camera.launch
```



ပုံမှာဆိုရင် Gazebo world ထဲမှာ coke can နဲ့ table တစ်ခုထည့်ထားပါတယ်။

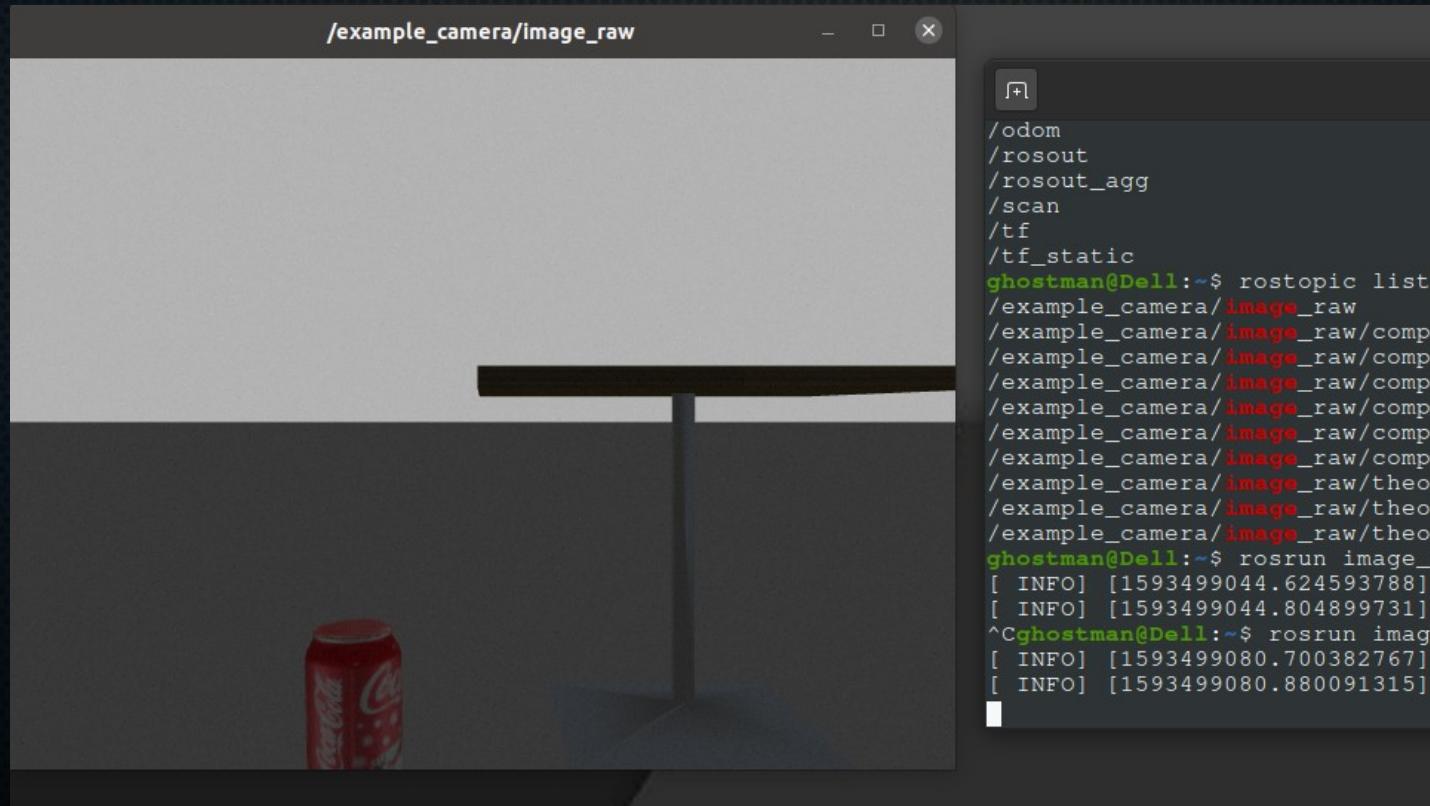
Visualizing

1) Image

ROS မှ 1 dimension, 2 dimensions, 3 dimensions data များကို ကြည့်ဖို့ အတွက် tool မျိုးစုံပါရိုပါတယ်။ image, video တွေကို ကြည့်တဲ့ အခါ image_view package and node ကို အသုံးပြန်သလို Rviz ကနေလည်းကြည့်နိုင်ပါတယ်။ mobot_w_lidar_and_camera.launch ကို launch လုပ်ပြီး

```
~$ rostopic list | grep image
```

```
~$ rosrun image_view image_view image:=/your/image_topic
```

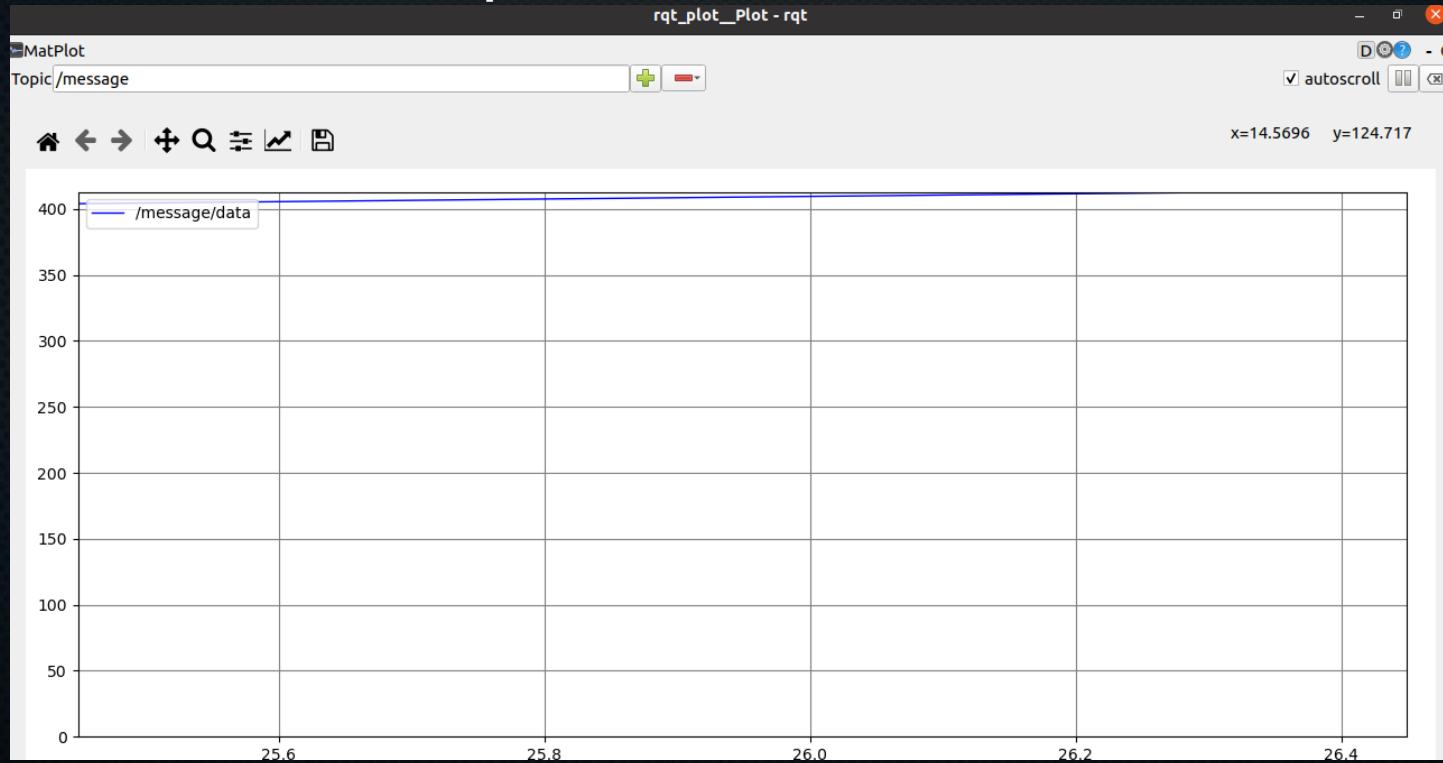


2) Plotting

ROS မှ scalar data များကို ကြည့်ဖို့အတွက် rqt_plot ဆိုတဲ့ package and node ကို အသုံးပြန်သလို Rviz ကနေလည်းကောင်းမြန်ပါတယ်။ chapter1_tutorial ထဲက node_tuto package ထဲက pub_node ကို runပြီး

```
~$ rostopic list
```

```
~$ rosrun node_tuto pub_node
```

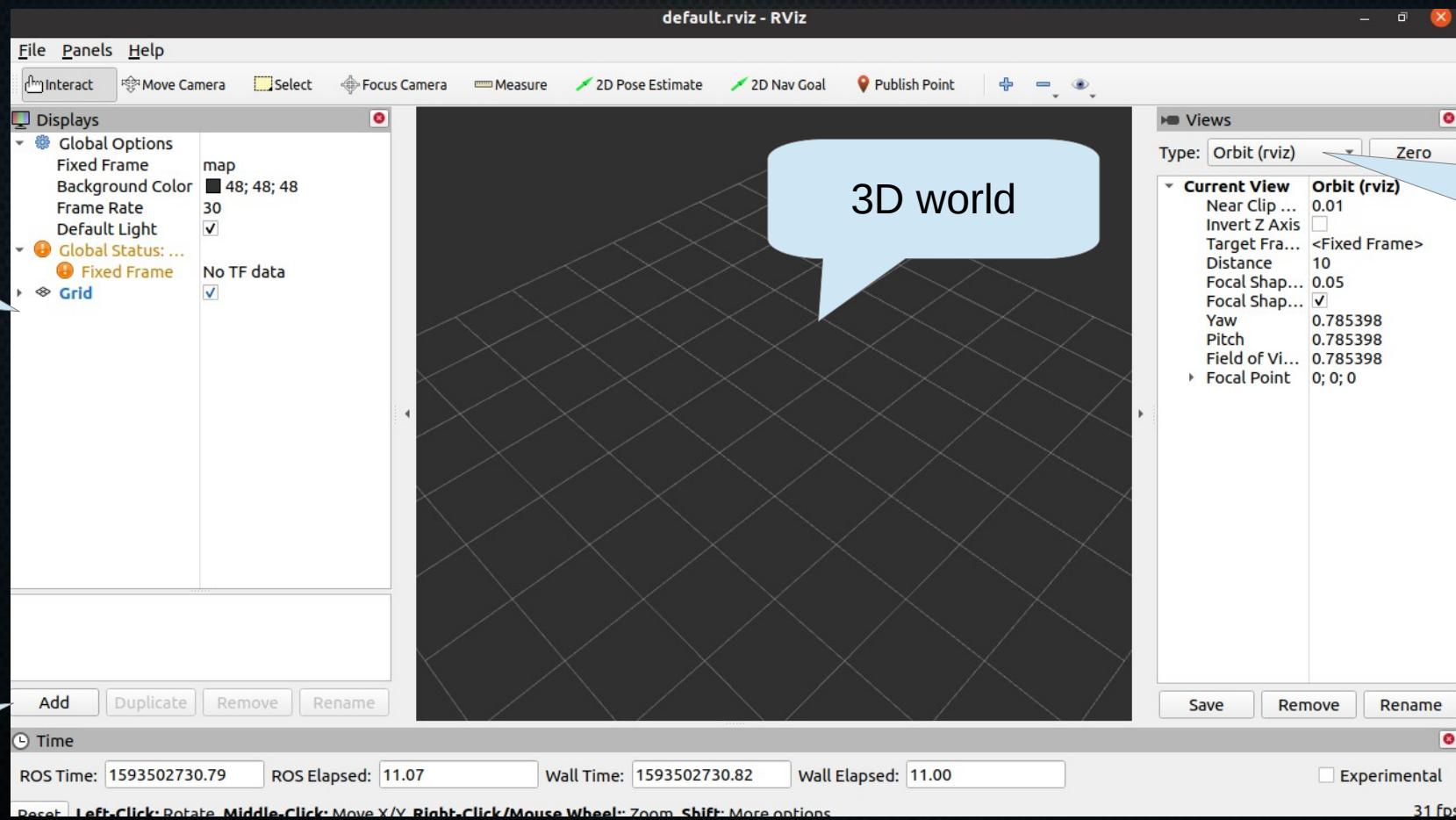


ကြည့်ချင်တဲ့ topic များကို + ကိုနိုပ်ပြီးကြည့်နိုင်သလို - ကို နိုပ်ပြီး remove လုပ်နိုင်ပါတယ်။

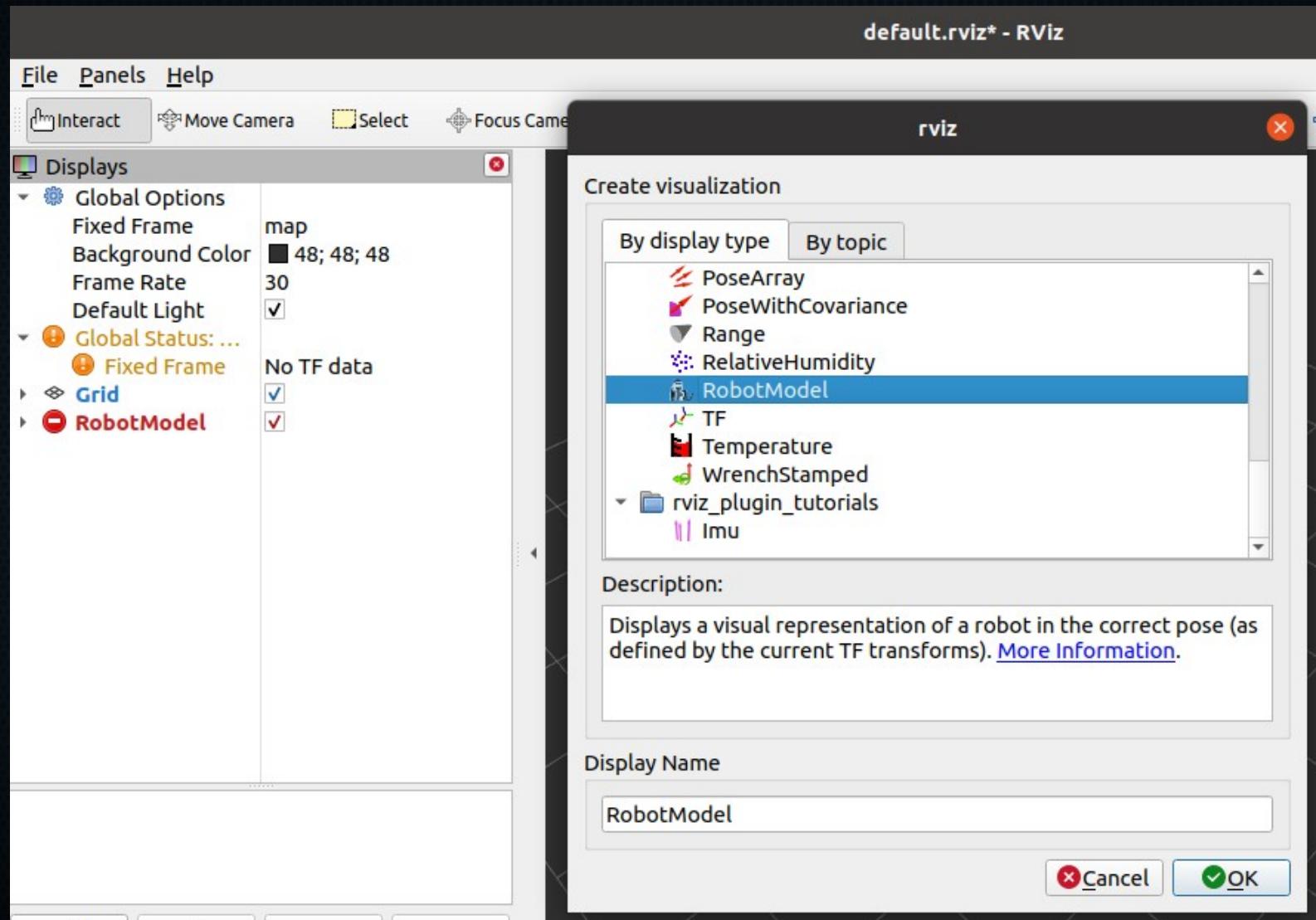
Rviz

- ROS မှ sensor data တွေနဲ့ robot state တွေကို visualize လုပ်ဖို့ Rviz ပါဝင်ပြီး data များစွာကို ကြည့်ရှုနိုင်ပါတယ်။
- ROBOT ARM တွေကို Planning, Moving ပြုလုပ်ခြင်းနဲ့ Mobile Robot တွေကိုလဲ အသာတူလုပ်ပေးနိုင်ပါတယ်။
- Rviz ဟာ Real Hardware နဲ့ Simulation နှစ်ခုလုံးက data တွေကို ပြသပေးနိုင်ပါတယ်။

~\$ rosrun rviz rviz



- Add ကို နှိပ်ပါက Left pane ထဲထည့်သွင်းနိုင်သည့် Sensors data type များကို တွေ့ရမည့်ဖြစ်ပြီး နူးနာ robot model ထည့်ပြထားပါတယ်။

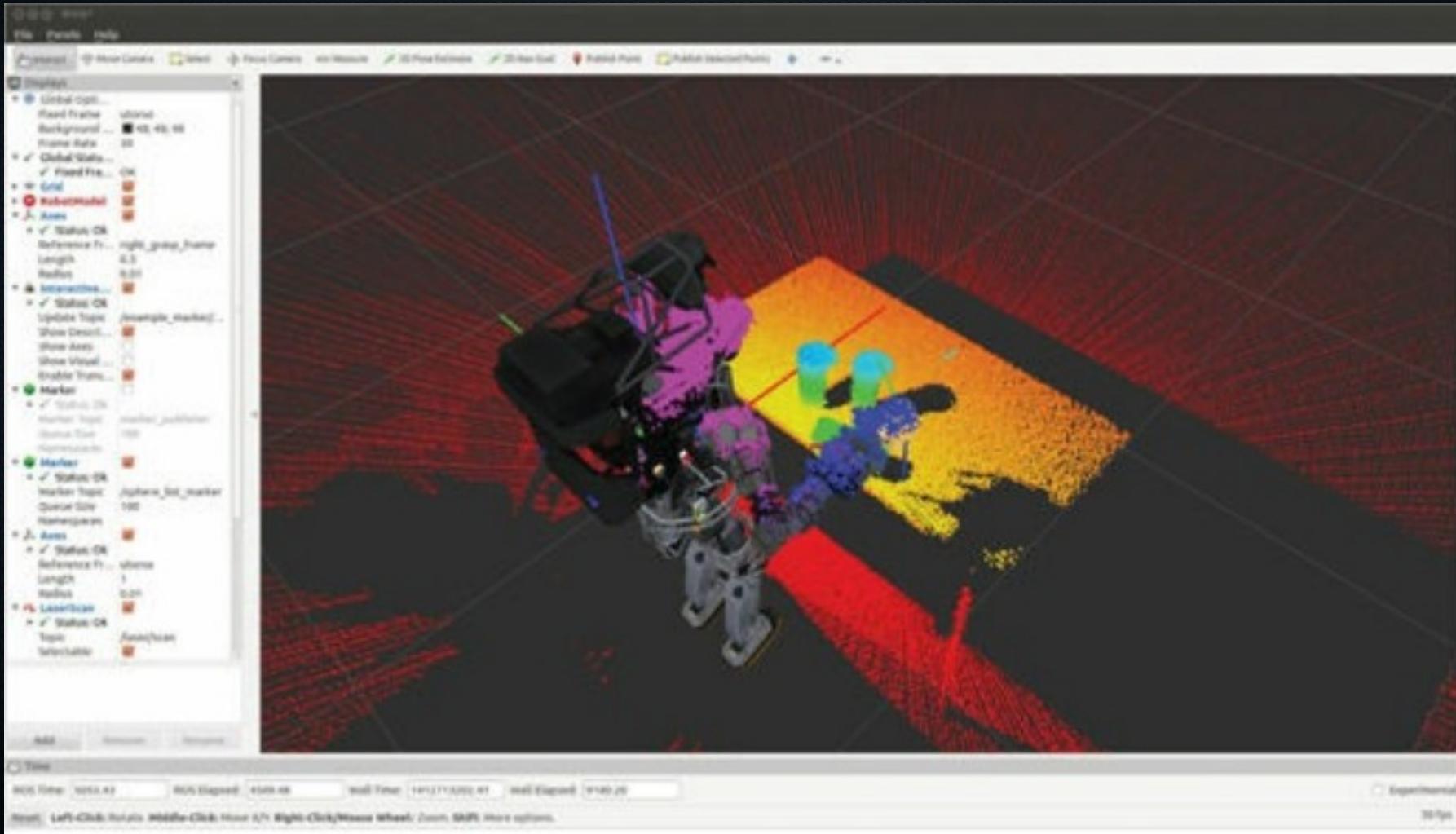


Atalas Robot ကဲ Sensor များမှတဆင့် visualize လုပ်ပုံ

Add → RobotModel → Choose from parameter server (robot_description)

Add → LaserScan (select topic name)

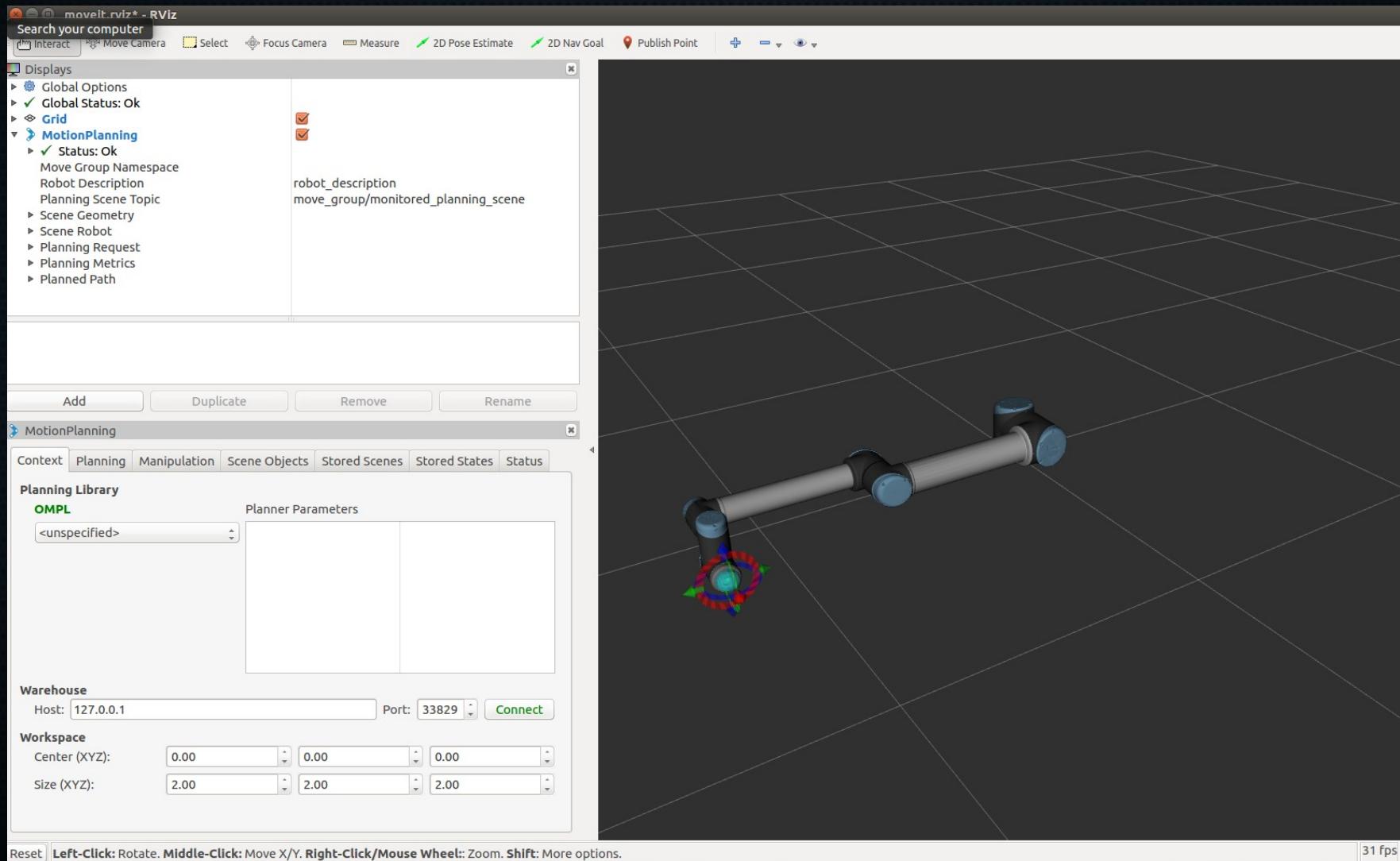
Add → Point cloud (select topic name)



Industrial Robot Arm (UR 10,5 Series) őrvislualize őjönbő

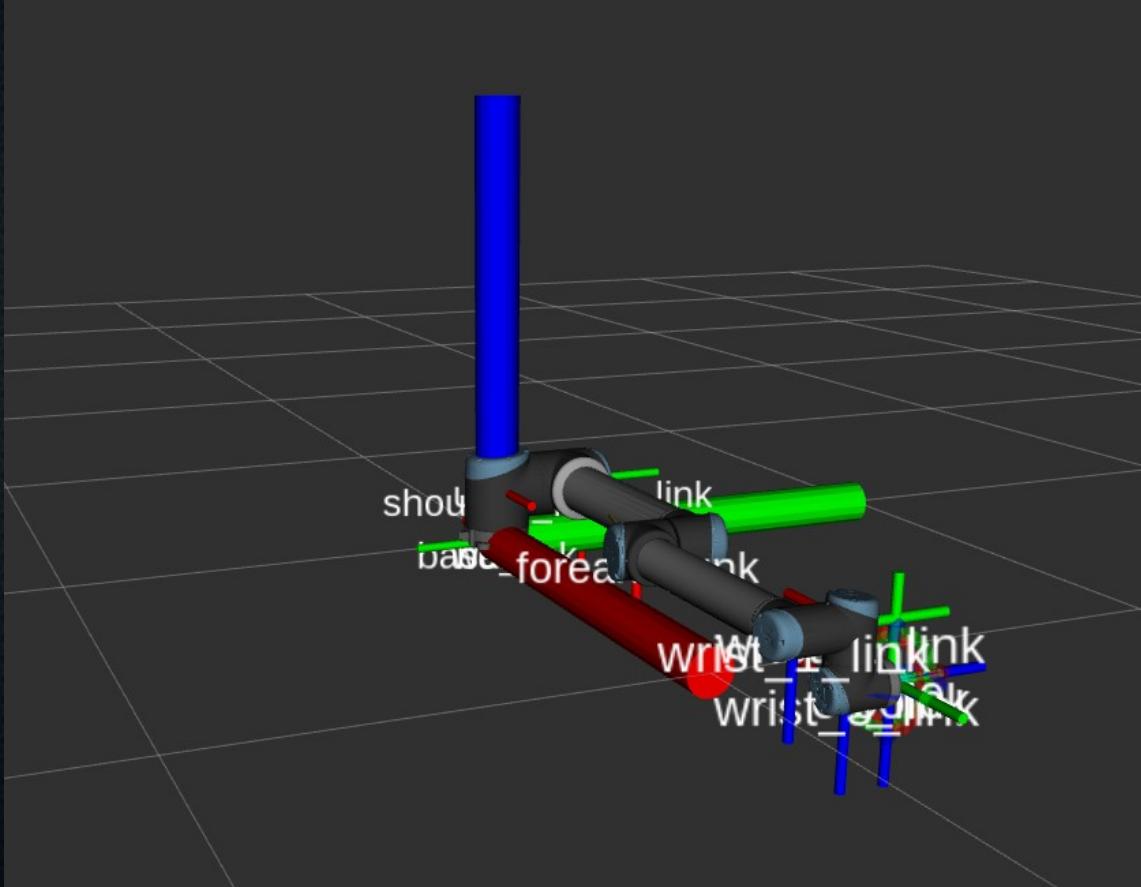
Add → RobotModel → Choose from parameter server (robot_description)

Add → Moveit motion planning plugins

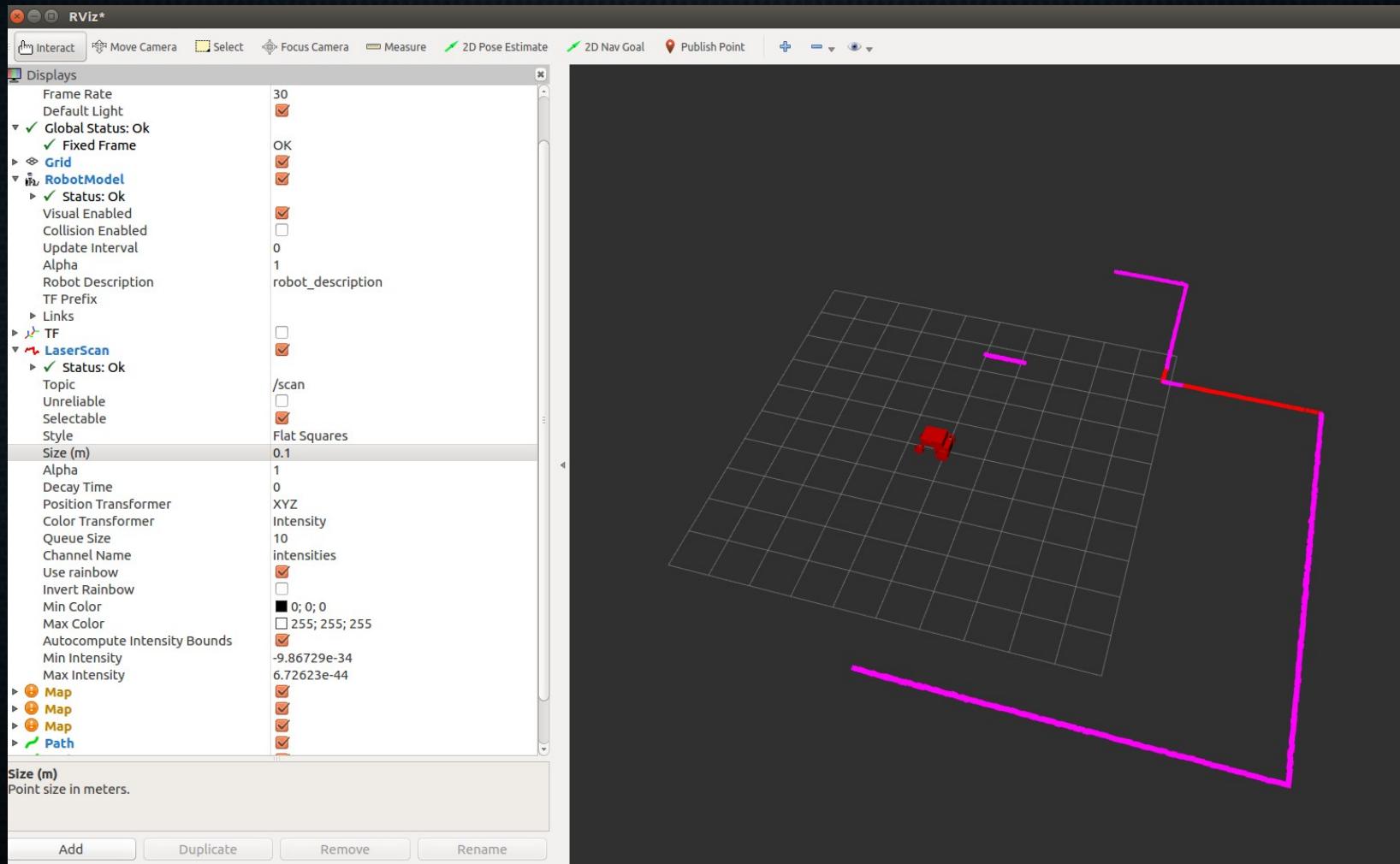


4646

Industrial Robot Arm (UR 10,5 Series) ရဲ့ tf frame များကိစစ်ဆေးပုံ
Add → tf →

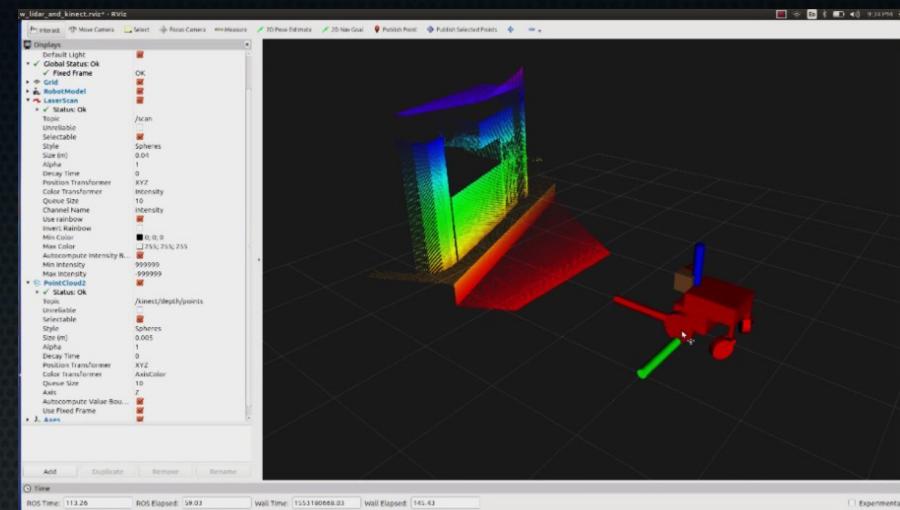
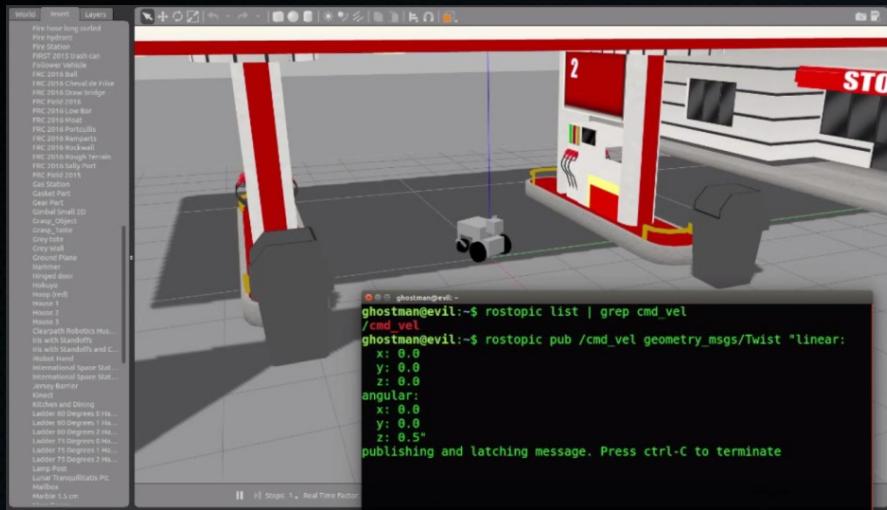


Gazebo မှ simulation robot နှင့် Laser data များကို visualize လုပ်ပုံ
Add → RobotModel → robot_description
Add → LasetScan (topic name: /scan)



Gazebo , Rviz Point Cloud view from 3D Camera

Image View



Gazebo တို့ Rviz တို့ မှာ object ထွေ marker တွေကို Programming နဲဖန်တီးနိုင်ပါတယ်။

ဒီသင်ခန်းစာများမှာ တွေ့ Gazebo Programming နည်းလမ်းများကို မဖော်ပြတော့ဘဲ အသုံးပြုပုံများကိုသာ ဖော်ပြထားတာဖြစ်ပါတယ်။

Rviz အတွက်လဲ Marker , Interactive Marker များကို programm ရေးဖန်တီးပုံကို ချပ်လှန်ခဲ့ပါတယ်။



ROM Robotics
Robotics Company
5.1 miles · No(215/2) Zewasoe street, (14/2) ward,
Okkalapa (South) Township, Yangon · Always Open

✓ Liked ▾



ROBOT OPERATING SYSTEM MYANMAR
127 subscribers

SUBSCRIBE

- Facebook** - <https://www.facebook.com/ROMROBOTS/>
- Website** - <http://romrobots.com/>
- YouTube** - <https://www.youtube.com/channel/UCq6OHEpbQ9RQIHV2U6W8qAQ>
- GitHub** - <https://github.com/GreenGhostMan>
- Phone** - 09259288229 , 09259288230