

Nama: Muhammad Ismail Ibadurrahman
NIM: 1103204239

UAS Robotic Technical Report – Mastering ROS for Robotics Programming

1. Chapter 1 – Introduction to ROS

Bab pertama buku ini memperkenalkan konsep dasar Robot Operating System (ROS) dan sistem manajemen pakatnya. Pembahasan melibatkan konsep-konsep seperti master ROS, node ROS, dan server parameter ROS, serta pesan dan layanan ROS. Bab ini juga mencakup instalasi ROS dan panduan awal dengan master ROS. Topik yang dibahas termasuk pentingnya mempelajari ROS, pemahaman tingkat sistem file ROS, grafik komputasi ROS, dan tingkat komunitas ROS.

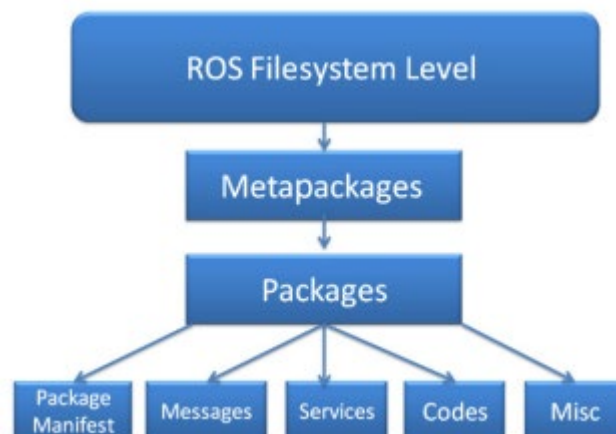
Technical Requirements :

Standard computer running Ubuntu 20.04 LTS or a Debian 10 GNU/Linux distribution

Mengapa kita harus menggunakan ROS:

Robot Operating System (ROS) adalah kerangka kerja fleksibel yang menyediakan alat dan perpustakaan untuk menulis perangkat lunak robot. Proyek ROS dimulai pada tahun 2007 dan dikembangkan di Willow Garage. ROS bertujuan menetapkan cara standar untuk memprogram robot dengan menyediakan komponen perangkat lunak siap pakai yang mudah diintegrasikan. Beberapa alasan memilih ROS termasuk kemampuan kelas atas, banyak alat untuk debugging dan simulasi, dukungan untuk sensor dan aktuator kelas atas, pengoperasian antar-platform, modularitas, dan penanganan sumber daya secara bersamaan. ROS memungkinkan pengguna untuk memanfaatkan fungsionalitas siap pakai, alat debugging yang kuat, dukungan untuk berbagai sensor dan aktuator, komunikasi antar platform, modularitas, dan penanganan sumber daya perangkat keras dengan lebih efisien.

Memahami ROS filesystem level:

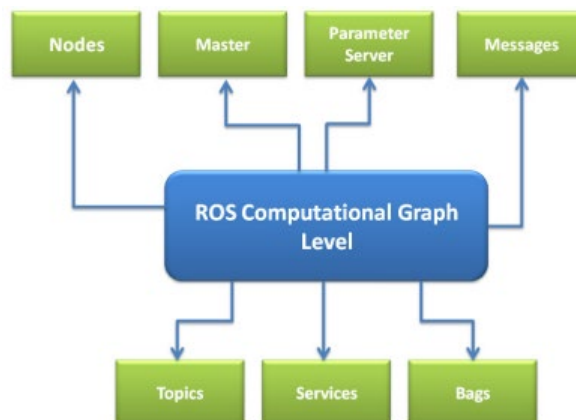


ROS bukan hanya sekadar kerangka pengembangan, tetapi dapat dianggap sebagai meta-OS karena tidak hanya menyediakan alat dan perpustakaan, tetapi juga fungsi mirip OS seperti abstraksi perangkat keras, manajemen paket, dan rantai alat pengembang. Seperti sistem operasi sungguhan, ROS mengatur file pada hard disk dengan cara tertentu, seperti yang diilustrasikan dalam diagram.

Teks tersebut menjelaskan beberapa elemen kunci dalam sistem file ROS:

1. **Paket:** Paket ROS merupakan unit sentral dari perangkat lunak ROS, berisi program (node), perpustakaan, file konfigurasi, dll., yang disatukan sebagai satu kesatuan. Paket adalah versi atom dan item rilis dalam perangkat lunak ROS.
2. **Manifes Paket:** File manifes paket berada di dalam paket dan berisi informasi tentang paket, seperti penulis, lisensi, dependensi, flag kompilasi, dll. File `package.xml` di dalam paket adalah file manifes paket.
3. **Metapackage:** Metapackage mengacu pada satu atau lebih paket terkait yang dapat dikelompokkan secara longgar. Metapackages adalah paket virtual yang tidak berisi kode sumber atau file khas.
4. **Manifes Metapackage:** Manifes metapackage mirip dengan manifes paket, mungkin menyertakan paket di dalamnya sebagai dependensi waktu proses dan mendeklarasikan tag ekspor.
5. **Pesan (.msg):** Pesan khusus dapat ditentukan dalam folder `msg` di dalam paket. File pesan memiliki perpanjangan `.msg`.
6. **Layanan (.srv):** Tipe data balasan dan permintaan untuk layanan ditentukan dalam folder `srv` di dalam paket. File layanan memiliki perpanjangan `.srv`.
7. **Repositori:** Sebagian besar paket ROS dikelola menggunakan Sistem Kontrol Versi (VCS) seperti Git, Subversion (SVN), atau Mercurial (hg). Repositori mengandung satu set file yang merepresentasikan paket-paket ROS.

Memahami *ROS computation graph level*:



Teks tersebut menjelaskan beberapa elemen kunci dalam ROS:

Node: Node adalah proses yang memiliki komputasi dalam ROS. Setiap node ditulis menggunakan perpustakaan klien ROS dan dapat menerapkan berbagai fungsionalitas, seperti komunikasi antar node. Tujuan node adalah membangun proses yang sederhana, mudah untuk di-debug, dan memiliki fungsi yang diinginkan.

Master: Master ROS menyediakan proses registrasi dan pencarian nama node lainnya. Tanpa master, node tidak dapat menemukan satu sama lain, bertukar pesan, atau menjalankan layanan. Dalam sistem terdistribusi, master dijalankan di satu komputer, dan node jarak jauh dapat berkomunikasi dengan master tersebut.

Server Parameter: Server parameter memungkinkan penyimpanan data di lokasi pusat. Semua node dapat mengakses dan mengubah nilai-nilai ini, dan server parameter merupakan bagian dari master ROS.

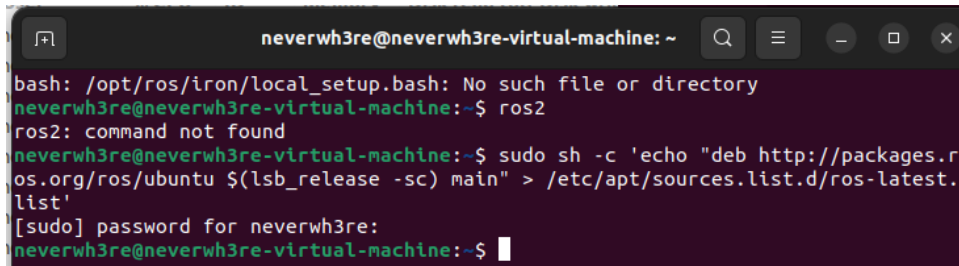
Topik: Pesan di ROS diangkut menggunakan bus bernama topik. Node yang mengirim pesan melalui topik disebut sebagai penerbit, sedangkan node yang

menerima pesan dari topik disebut sebagai pelanggan. Produksi dan konsumsi informasi melalui topik adalah terpisah, dan setiap topik memiliki nama unik.

Logging: ROS menyediakan sistem logging untuk menyimpan data, seperti data sensor, yang mungkin sulit dikumpulkan tetapi diperlukan untuk pengembangan dan pengujian algoritma robot. Fitur ini, dikenal sebagai bagfile, sangat berguna dalam bekerja dengan mekanisme robot yang kompleks.

Instalasi ROS Noetic:

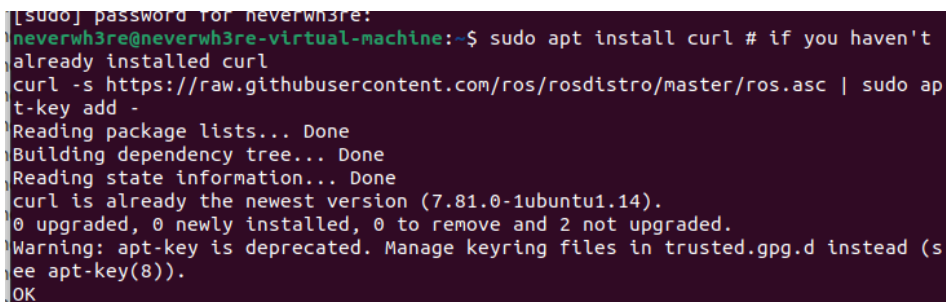
1.1 Setup sources.list

A terminal window titled 'neverwh3re@neverwh3re-virtual-machine: ~' showing the following commands and output:

```
bash: /opt/ros/iron/local_setup.bash: No such file or directory
neverwh3re@neverwh3re-virtual-machine:~$ ros2
ros2: command not found
neverwh3re@neverwh3re-virtual-machine:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
[sudo] password for neverwh3re:
neverwh3re@neverwh3re-virtual-machine:~$
```

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

1.2 Setup keys

A terminal window showing the following commands and output:

```
[sudo] password for neverwh3re:
neverwh3re@neverwh3re-virtual-machine:~$ sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.14).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

```
sudo apt install curl # if you haven't already installed curl
curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -
```

1.3 Instalasi

```
sudo apt update
sudo apt install ros-noetic-desktop-full
```

1.4 Environment Setup

```
source /opt/ros/noetic/setup.bash
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

1.5 Dependencies

```
sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
```

1.6 Initialize rosdep

```
sudo apt install python3-rosdep  
sudo rosdep init  
rosdep update
```

2. Chapter 2 – Getting Started with ROS Programming

Bab ini membahas langkah-langkah untuk membuat dan membangun paket ROS, dengan penekanan pada implementasi sistem komunikasi ROS melalui node, topik, pesan, layanan, dan actionlib. Topik-topik yang dibahas meliputi:

1. Membuat paket ROS: Langkah-langkah untuk membuat paket ROS, yang merupakan unit sentral perangkat lunak ROS.
2. Menambahkan pesan khusus dan file layanan: Penjelasan tentang cara menambahkan pesan khusus dalam folder msg dan file layanan dalam folder srv di dalam paket ROS.
3. Bekerja dengan layanan ROS: Pemahaman tentang cara bekerja dengan layanan ROS untuk pertukaran informasi antara node.
4. Membuat file peluncuran: Pembahasan tentang pembuatan file peluncuran untuk mengorganisir dan menjalankan node-node ROS.
5. Penerapan topik, layanan, dan actionlib: Implementasi konsep topik, layanan, dan actionlib dalam pengembangan paket ROS.

Bab ini memberikan wawasan tentang langkah-langkah praktis dalam pengembangan perangkat lunak menggunakan ROS, dengan fokus pada komunikasi antar-node dan komponen-komponen utama ROS.

Technical Requirements:

Ubuntu 20.04 dengan ROS Noetic terinstal. Kode referensi untuk bab ini dapat diunduh dari repositori GitHub berikut: <https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git> kode yang diperlukan terdapat di folder Chapter2/mastering_ros_demo_pkg

Membuat ROS Package:

Package ROS merupakan unit dasar program dalam sistem ROS. Proses pembuatan, pembangunan, dan rilis paket ROS dapat dilakukan dengan menggunakan sistem build catkin. Distribusi ROS yang digunakan saat ini adalah Noetic Ninjamys. Sistem build catkin bertanggung jawab untuk menghasilkan target, baik itu dapat dieksekusi atau perpustakaan, dari sumber kode tekstual yang dapat digunakan oleh pengguna akhir. Rosbuild adalah sistem pembangunan pada distribusi lama seperti Electric dan Fuerte, namun karena kekurangannya, catkin muncul sebagai penggantinya. Catkin juga memungkinkan pendekatan yang lebih dekat dengan sistem kompilasi ROS Cross Platform Make (CMake). Menggunakan sistem build catkin dalam paket ROS

memiliki keuntungan, termasuk kemampuan untuk melakukan porting paket ke sistem operasi (OS) lain, seperti Windows. Jika OS tersebut mendukung CMake dan Python, paket yang dibangun dengan catkin dapat dengan mudah dipindahkan ke sana. Dengan demikian, catkin memberikan fleksibilitas dalam memperluas kompatibilitas paket ROS ke berbagai sistem operasi.

Persyaratan pertama untuk bekerja dengan paket ROS adalah membuat ruang kerja ROS catkin. Setelah menginstal ROS, kita dapat membuat dan *workspace* catkin bernama `catkin_ws`

```
mkdir -p ~/catkin_ws/src
```

Untuk mencompile workspace ini, kita perlu menetapkan source the ROS environment untuk mendapatkan akses, berikut commandnya.

```
source /opt/ros/noetic/setup.bash
```

Pindah direktori

```
cd ~/catkin_ws/src
```

Inisialisasi ruang kerja catkin baru:

```
catkin_init_workspace
```

Kita bisa membuat *workspace* meskipun tidak ada *packages*. Perintah berikut untuk pindah ke direktori *workspace*

```
cd ~/catkin_ws
```

Perintah berikut akan mem-*build workspace* tersebut

```
catkin_make
```

Perintah berikut akan membuat direktori "devel" dan "build" di ruang kerja catkin. File setup terletak di dalam folder "devel". Untuk menambahkan ruang kerja ROS ke lingkungan ROS, kita dapat mengambil salah satu file setup tersebut. Selain itu, kita dapat menjalankan perintah berikut untuk mengambil file setup ruang kerja ini setiap kali sesi bash baru dimulai.

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc source ~/.bashrc
```

Setelah menyiapkan ruang kerja catkin, langkah selanjutnya adalah membuat paket khusus dengan contoh node untuk mengilustrasikan penggunaan topik, pesan, layanan, dan actionlib di ROS. Penting untuk memastikan bahwa ruang kerja telah diatur dengan benar sebelum menggunakan perintah ROS. Dalam konteks ini, perintah `'catkin_create_pkg'` adalah pilihan yang paling nyaman untuk membuat paket ROS yang akan digunakan sebagai demo konsep-konsep dasar di dalam ROS.

Beralih ke folder src ruang kerja catkin dan buat paket dengan menggunakan perintah berikut:

```
catkin_create_pkg package_name [dependency1] [dependency2]
```

Kode sumber lengkap untuk proyek ini dapat diklon dari GitHub buku ini gudang. Perintah berikut akan mengkloning repositori proyek:

```
git clone https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git  
  
cd Mastering-ROS-for-Robotics-Programming-Third-edition/ Chapter2/
```

Dalam bab ini, disajikan berbagai contoh node ROS yang menunjukkan implementasi fitur ROS melalui topik, layanan, dan tindakan ROS. Alat-alat ini umumnya digunakan di setiap paket ROS, baik yang sudah ada dalam repositori ROS maupun yang dibuat pengguna. Pembahasan juga mencakup cara membuat dan mengkompilasi paket ROS menggunakan pesan khusus, baik yang disediakan secara kustom maupun standar. Karena paket yang berbeda menggunakan pesan khusus untuk mengelola data dari node mereka, pemahaman yang baik terhadap manajemen pesan khusus ini menjadi penting dalam pengembangan perangkat lunak ROS.

3. Chapter 3 – Working with ROS for 3D Modelling

Fase awal dalam pembuatan robot melibatkan perancangan dan pemodelan menggunakan alat CAD seperti Autodesk Fusion 360, SolidWorks, dan Blender. Pemodelan ini, yang dilakukan untuk dua jenis robot - manipulator tujuh Derajat Kebebasan (DOF) dan robot penggerak diferensial, memiliki tujuan utama dalam simulasi. Alat simulasi memungkinkan pemeriksaan kelemahan kritis dalam desain robot sebelum masuk tahap pembuatan, memastikan bahwa robot akan berfungsi dengan baik. Bab ini akan membahas proses desain kedua robot tersebut, dan bab-bab berikutnya akan menjelajahi simulasi, pembuatan perangkat keras nyata, dan antarmuka dengan ROS. Pemahaman dan pembuatan model robot dalam ROS menjadi krusial, karena model tersebut dapat digunakan untuk simulasi, kontrol robot, visualisasi, dan mendapatkan informasi mengenai struktur dan kinematika robot.

Dalam bab ini, kita mempelajari topik-topik berikut:

- Paket ROS untuk pemodelan robot
- Memahami pemodelan robot menggunakan Unified Robot Description Format (URDF)
- Membuat paket ROS untuk deskripsi robot
- Membuat model URDF pertama kami
- Menjelaskan file URDF
- Memvisualisasikan model robot 3D di RViz
- Menambahkan properti fisik dan tumbukan ke model URDF
- Memahami pemodelan robot menggunakan XML Macro (Xacro)
- Mengonversi xacro ke URDF
- Membuat deskripsi robot untuk manipulator robot tujuh DOF
- Menjelaskan model xacro dari lengan tujuh DOF
- Membuat model robot untuk mobile robot penggerak diferensial

Technical**Requirements:**

Ubuntu 20.04 dengan ROS Noetic diinstal. Kode referensi untuk bab ini bisa diunduh dari repositori Git pada chapter sebelumnya

Dalam bab ini, fokus utamanya adalah pada pentingnya pemodelan robot dan bagaimana kita dapat mencapainya melalui pemodelan di ROS. Pembahasan mencakup paket-paket dalam ROS yang digunakan untuk memodelkan struktur robot, seperti urdf, xacro, dan joint_state_publisher bersama dengan antarmuka grafisnya. URDF, xacro, dan tag utama URDF menjadi pusat perhatian, dengan pembahasan mendalam mengenai penggunaannya. Model contoh dibuat dalam URDF dan xacro, sambil mempertimbangkan perbedaan di antara keduanya. Selanjutnya, sebuah manipulator robot dengan tujuh Derajat Kebebasan (DOF) dikonstruksi, dan penerapan joint_state_publisher dan robot_state_publisher dieksplorasi. Bab ini juga menyajikan panduan desain untuk robot penggerak diferensial menggunakan xacro, yang akan diteruskan dalam simulasi menggunakan Gazebo dalam bab selanjutnya.

4. Chapter 4 –Simulating Robots Using ROS and Gazebo

Setelah merancang model 3D robot, langkah berikutnya adalah melakukan simulasi menggunakan simulator Gazebo. Simulasi ini memberikan gambaran tentang operasi robot dalam lingkungan virtual. Gazebo, simulator multi-robot untuk simulasi robot kompleks dalam dan luar ruangan, digunakan untuk mensimulasikan robot dengan tujuh Derajat Kebebasan (DOF) dan robot bergerak. Gazebo mendukung simulasi robot, sensor, dan objek 3D dengan menyediakan model-model yang sudah populer dalam repositori resminya. Keunggulan Gazebo adalah integrasinya yang sempurna dengan ROS melalui antarmuka ROS yang tepat, memungkinkan kontrol penuh Gazebo di ROS. Meskipun Gazebo dapat diinstal tanpa ROS, instalasi antarmuka ROS-Gazebo diperlukan untuk berkomunikasi dari ROS ke Gazebo.

Pada bab ini, akan dibahas simulasi robot tujuh lengan DOF dan robot beroda diferensial, termasuk penggunaan pengontrol ROS untuk mengendalikan koneksi robot di Gazebo. Topik-topik yang akan dijelaskan meliputi pemahaman tentang simulasi robot dan Gazebo, simulasi model lengan robot, simulasi lengan robot dengan sensor kedalaman, pengontrol ROS untuk menggerakkan sambungan robot di Gazebo, serta simulasi robot beroda diferensial dan pengoperasiannya di Gazebo.

Technical**Requirements:**

Ubuntu 20.04 dengan ROS Noetic diinstal. Kode referensi untuk bab ini bisa diunduh dari repositori Git pada chapter sebelumnya

Mensimulasikan Tangan Robotik Menggunakan Gazebo dan Ros

Sebelum memulai dengan Gazebo dan ROS, kita perlu menginstall package berikut agar Gazebo dan ROS bisa bekerja:

```
sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-msgs ros-noetic-gazebo-plugins ros-noetic-gazebo-ros-control
```

lalu dilanjut dengan command berikut

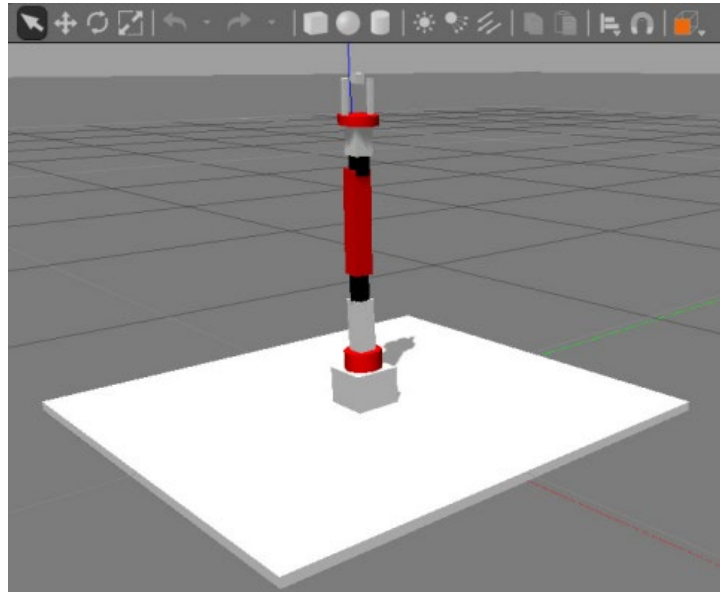
```
roscore & rosrun gazebo_ros gazebo
```

Command tersebut akan membuka GUI Gazebo, lalu bisa masuk ke tahap selanjutnya.

Membuat Tangan Robotik Model Simulasi untuk Gazebo

Kita bisa membuat simulasi model tangan dengan memperbarui deskripsi robot dengan menambahkan parameter simulasi. Kita bisa membuat package yang dibutuhkan untuk mensimulasikan tangan robotik menggunakan *command* berikut

```
sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-msgs ros-noetic-gazebo-plugins ros-noetic-gazebo-ros-control
```



Apabila tidak ada kendala, Gazebo akan memunculkan tangan robotik ini setelah kita me launch command berikut

```
roslaunch seven_dof_arm_gazebo seven_dof_arm_world.launch
```

Yang dimana isi dari file 'seven_dof_arm_world.launch' berisi sebagai berikut

```
<launch>
  <!-- these are the arguments you can pass this launch file,
  for example paused:=true -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>
  <!-- We resume the logic in empty_world.launch -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
  <arg name="debug" value="$(arg debug)" />
```



```

<arg name="gui" value="$ (arg gui)" />
<arg name="paused" value="$ (arg paused)"/>
<arg name="use_sim_time" value="$ (arg use_sim_time)"/>
<arg name="headless" value="$ (arg headless)"/>
</include>

<!-- Load the URDF into the ROS Parameter Server -->
<param name="robot_description" command="$ (find xacro)/xacro
'$ (find mastering_ros_robot_description_pkg)/urdf/seven_dof_
arm.xacro'" />

<!-- Run a python script to the send a service call to
gazebo_ros to spawn a URDF robot -->

<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model"
respawn="false" output="screen"
args="-urdf -model seven_dof_arm -param robot_description"/>
</launch>

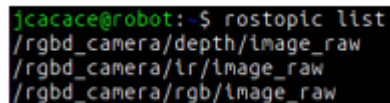
```

Visualizing the 3D sensor data

Setelah sukses menjalankan simulasi, kita bisa mengecek topik yang di generate oleh plugin sensor dengan command :

```
rostopic list
```

yang nantinya kurang lebih tampilannya seperti gambar berikut



```

jcacace@robot:~$ rostopic list
/rgb_camera/depth/image_raw
/rgb_camera/ir/image_raw
/rgb_camera/rgb/image_raw
/rgb_camera/rgb/image_raw

```

lalu untuk melihat data gambar 3D vision sensor, bisa menggunakan tools yang disebut image_view.do seperti sebagai berikut :

➔ Melihat RGB raw image

```
roslaunch image_view image:=/rgb_camera/rgb/image_raw
```

➔ Melihat IR raw image

```
roslaunch image_view image:=/rgb_camera/ir/image_raw
```

➔ Melihat depth image

```
roslaunch image_view image:=/rgb_camera/depth/image_raw
```

Pada bab ini, dilakukan simulasi dua jenis robot: lengan robot dengan tujuh Derajat Kebebasan (DOF) dan robot bergerak beroda diferensial. Pembahasan dimulai dengan robot lengan, mencakup tag Gazebo tambahan yang diperlukan untuk

meluncurkannya, penambahan sensor penglihatan 3D, pembuatan file peluncuran, dan penambahan pengontrol ke setiap sambungan. Proses serupa dilakukan untuk robot bergerak beroda diferensial, dengan pembuatan URDF dan penambahan plugin Gazebo-ROS untuk pemindai laser dan mekanisme penggerak diferensial. Setelah menyelesaikan model simulasi, dilakukan peluncuran menggunakan berkas peluncuran kustom. Terakhir, dijelaskan cara menggerakkan robot menggunakan node teleop.

5. Chapter 5 – Simulating Robots Using ROS, CoppeliaSim, and Webots

Pada bab ini, setelah mempelajari simulasi robot dengan Gazebo, akan dibahas penggunaan dua perangkat lunak simulasi robot canggih lainnya, yaitu CoppeliaSim dan Webots. CoppeliaSim, dikembangkan oleh Coppelia Robotics, menyediakan berbagai model simulasi robot industri dan seluler yang siap pakai, serta fungsi yang dapat diintegrasikan melalui antarmuka pemrograman aplikasi (API) khusus. CoppeliaSim juga dapat beroperasi dengan ROS menggunakan antarmuka komunikasi yang tepat. Webots, sebagai perangkat lunak sumber terbuka dan gratis, dikembangkan oleh Cyberbotics Ltd. dan dapat digunakan untuk mensimulasikan robot 3D. Seperti CoppeliaSim, Webots dapat terhubung dengan ROS dengan mudah.

Technical

Requirements:

Ubuntu 20.04 dengan ROS Noetic diinstal. Kode referensi untuk bab ini bisa diunduh dari repositori Git pada chapter sebelumnya

Setting Up CoppeliaSim dengan ROS

Sebelum memulai, kita perlu install terlebih dahulu pada sistem kita dengan mendownload versi terbaru dari CoppeliaSim di link berikut <http://www.coppeliarobotics.com/downloads.html>, lalu menjalankan command berikut setelah di ekstrak

```
tar vxf CoppeliaSim_Edu_V4_2_0_Ubuntu20_04.tar.xz
```

Ganti nama folder menjadi lebih intuitif

```
mv CoppeliaSim_Edu_V4_2_0_Ubuntu20_04 CoppeliaSim .
```

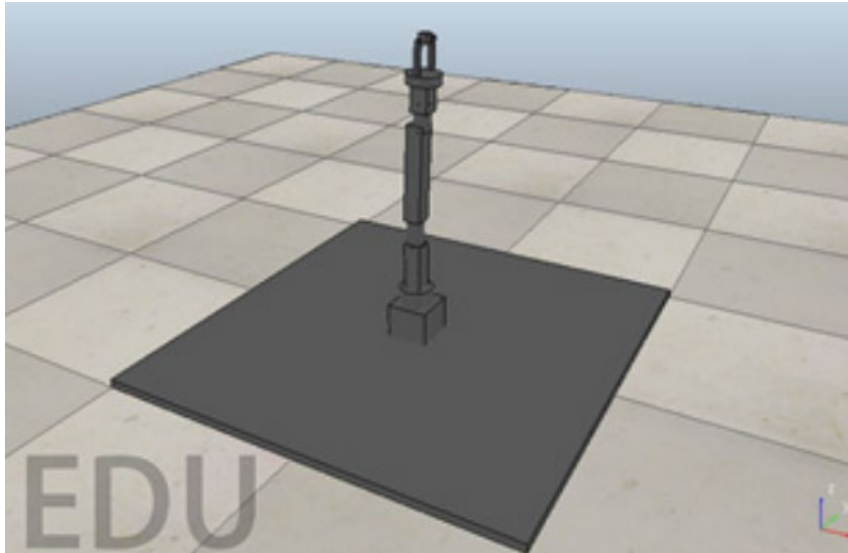
Agar mengakses CoppeliaSim bisa dengan mudah, berikan perintah ini

```
echo "export COPPELIASIM_ROOT=/path/to/CoppeliaSim/folder >  
> ~/.bashrc"
```

Simulating a robotic Arm using CoppeliaSim and ROS

Kita perlu convert Xacro model arm ke URDF file dengan command

Kurang lebih apabila berhasil tampilannya akan seperti ini



```
roslaunch xacro seven_dof_arm.xacro > /path/to/csim_demo_pkg/  
urdf/seven_dof_arm.urdf
```