

UAS ROBOTIKA

Nama : REYNANDA ADITYA

NIM : 1103202154

Untuk mengikuti bab ini hingga akhir, satu-satunya persyaratan yang diperlukan adalah sebuah komputer standar yang menggunakan sistem operasi Ubuntu 20.04 LTS atau distribusi Debian 10GNU/Linux.

Chapter 1: Introduction to ROS Programming Essentials

1. Introduction

Robot Operating System (ROS) adalah kerangka kerja middleware sumber terbuka yang dirancang untuk pengembangan perangkat lunak robotika. ROS menyediakan seperangkat alat, pustaka, dan konvensi untuk membangun dan mengelola sistem robotika. ROS Noetic adalah salah satu distribusi ROS, dan kompatibel dengan Ubuntu 20.04 LTS (Long Term Support).

Laporan teknis ini menjelaskan langkah-langkah untuk menginstal ROS Noetic di Ubuntu 20.04.

2. Langkah – Langkah instalasi

- Setup your source.list
Konfigurasi sources.list repositori Ubuntu Anda agar dapat mengakses paket ROS.
Bukaterminal dan jalankan perintah berikut:
`sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`
- Setup your Key
Dapatkan key ROS untuk memastikan keaslian paket:
`sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-keyC1CF6E31E6BADE8868B172B4F42ED6FBAB17C654`
- Update the package index
Perbarui indeks paket agar Anda memiliki akses ke paket ROS terbaru:
`sudo apt update`

```
rey@rey-VirtualBox:~$ sudo apt update
[sudo] password for rey:
Hit:1 http://packages.ros.org/ros/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://cx.archive.ubuntu.com/ubuntu focal InRelease
Get:4 http://cx.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:5 http://cx.archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 114 kB in 10s (11.2 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
```

- Install ROS Noetic
Instal paket ROS Noetic Desktop-Full, yang mencakup dependensi paling umum untuk membangun paket ROS:

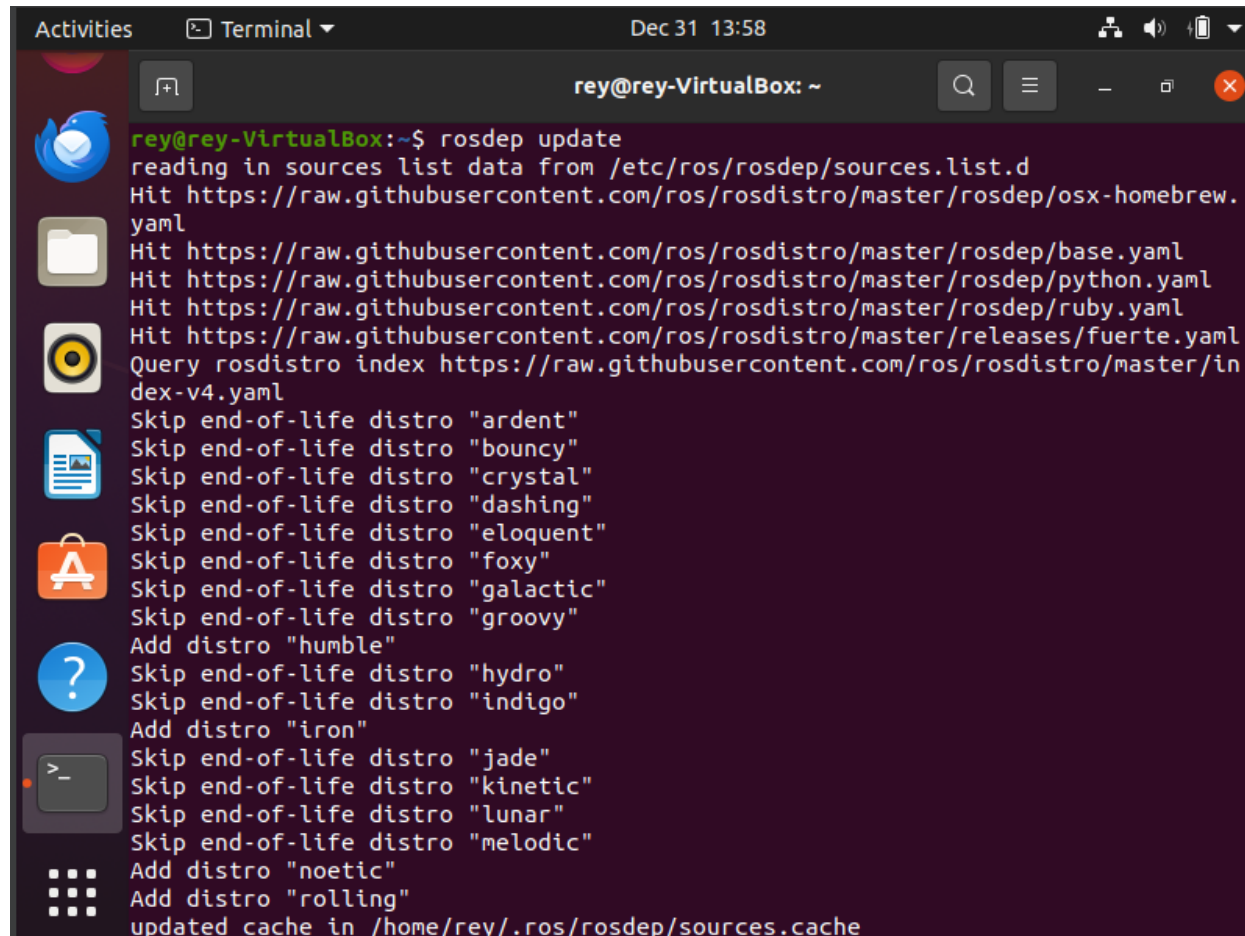
```
sudo apt install -y ros-noetic-desktop-full
```

```
rey@rey-VirtualBox:~$ sudo apt install ros-noetic-desktop-full
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-desktop-full is already the newest version (1.5.0-1focal.20231030.160753).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

- Initialize rosdep
Inisialisasi rosdep, alat untuk mengelola dependensi di

ROS:sudo rosdep init

rosdep update

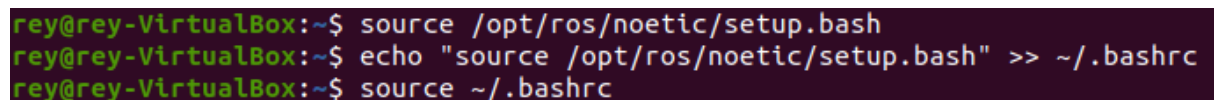


```
rey@rey-VirtualBox: ~  
rey@rey-VirtualBox:~$ rosdep update  
reading in sources list data from /etc/ros/rosdep/sources.list.d  
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.  
yaml  
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml  
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/python.yaml  
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/ruby.yaml  
Hit https://raw.githubusercontent.com/ros/rosdistro/master/releases/fuerte.yaml  
Query rosdistro index https://raw.githubusercontent.com/ros/rosdistro/master/in  
dex-v4.yaml  
Skip end-of-life distro "ardent"  
Skip end-of-life distro "bouncy"  
Skip end-of-life distro "crystal"  
Skip end-of-life distro "dashing"  
Skip end-of-life distro "eloquent"  
Skip end-of-life distro "foxy"  
Skip end-of-life distro "galactic"  
Skip end-of-life distro "groovy"  
Add distro "humble"  
Skip end-of-life distro "hydro"  
Skip end-of-life distro "indigo"  
Add distro "iron"  
Skip end-of-life distro "jade"  
Skip end-of-life distro "kinetic"  
Skip end-of-life distro "lunar"  
Skip end-of-life distro "melodic"  
Add distro "noetic"  
Add distro "rolling"  
updated cache in /home/rey/.ros/rosdep/sources.cache
```

- Setup Environment variables
Tambahkan variabel lingkungan ROS ke sesi bash Anda dengan
menambahkan barisberikut ke file ~/.bashrc:

echo "source /opt/ros/noetic/setup.bash" >>

~/.bashrcsource ~/.bashrc



```
rey@rey-VirtualBox:~$ source /opt/ros/noetic/setup.bash  
rey@rey-VirtualBox:~$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc  
rey@rey-VirtualBox:~$ source ~/.bashrc
```

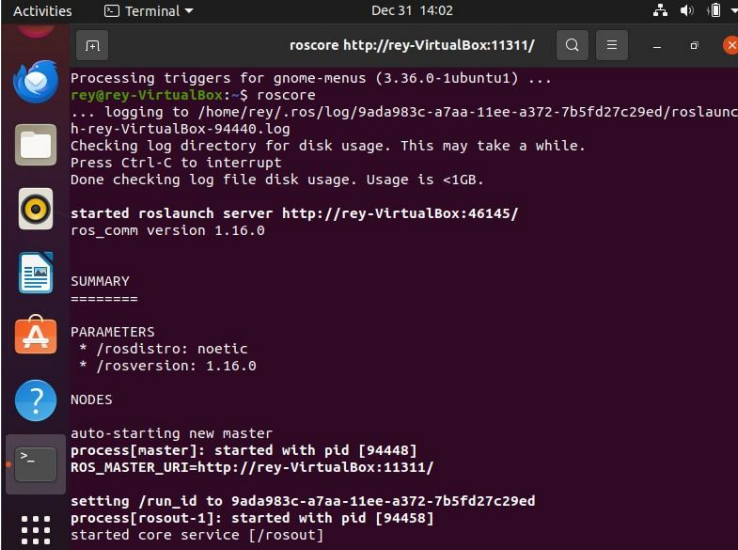
- Install additional dependencies
Instal dependensi sistem tambahan yang mungkin diperlukan oleh paket ROS:

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

```
rey@rey-VirtualBox:~$ sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
build-essential set to manually installed.
python3-rosdep is already the newest version (0.22.2-1).
The following additional packages will be installed:
  brz bzip2 git git-man liberror-perl libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libserf-1-1 libsvn1 libutf8proc2 mercurial
  mercurial-common python2 python2-minimal python2.7 python2.7-minimal
  python3-breezy python3-configobj python3-deprecated python3-dulwich
  python3-fastimport python3-github python3-gitlab python3-gpg
  python3-rosdistro python3-vcstools python3-wrapt subversion
Suggested packages:
  brz-doc python3-breezy.tests git-daemon-run | git-daemon-sysvinit git-doc
  git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn kdiff3
  | kdiff3-qt | kompare | meld | tkcvs | mgdiff qct python-mysqldb
  python-openssl python-pygments python2-doc python-tk python2.7-doc
  python3-breezy-dbg python3-kerberos python-configobj-doc python-gitlab-doc
  db5.3-util libapache2-mod-svn subversion-tools
The following NEW packages will be installed:
  brz git git-man liberror-perl libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libserf-1-1 libsvn1 libutf8proc2 mercurial
  mercurial-common python2 python2-minimal python2.7 python2.7-minimal
  python3-breezy python3-configobj python3-deprecated python3-dulwich
  python3-fastimport python3-github python3-gitlab python3-gpg
  python3-rosdistro python3-rosinstall python3-rosinstall-generator
```

- Testing the Installation
Untuk memverifikasi bahwa ROS Noetic terinstal dengan benar, buka terminal baru dan jalankan perintah berikut:

Roscore



```
Activities Terminal Dec 31 14:02
roscore http://rey-VirtualBox:11311/
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
rey@rey-VirtualBox:~$ roscore
... logging to /home/rey/.ros/log/9ada983c-a7aa-11ee-a372-7b5fd27c29ed/roslaunch
h-rey-VirtualBox-94440.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://rey-VirtualBox:46145/
ros_comm version 1.16.0

SUMMARY
=====
PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES
auto-starting new master
process[master]: started with pid [94448]
ROS_MASTER_URI=http://rey-VirtualBox:11311/

setting /run_id to 9ada983c-a7aa-11ee-a372-7b5fd27c29ed
process[rosout-1]: started with pid [94458]
started core service [/rosout]
```

Jika ROS terinstal dengan benar, Anda seharusnya melihat ROS Master berjalan tanpakesalahan

Chapter 2: Getting Started with ROS Programming

Persyaratan Teknis:

- Memerlukan laptop standar dengan sistem operasi Ubuntu 20.04 dan ROS Noetic terinstal.
- Kode referensi untuk bab ini dapat diunduh dari repositori GitHub: <https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git>.

```
rey@rey-VirtualBox:~$ sudo apt install ros-noetic-desktop-full
[sudo] password for rey:
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-desktop-full is already the newest version (1.5.0-1focal.20231030.160753).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
```

Membuat Paket ROS:

- ROS packages adalah unit dasar dari program ROS.

- Dapat membuat, membangun, dan merilis paket ROS.
- Menggunakan sistem build catkin pada distribusi ROS Noetic.

Membuat Workspace Catkin:

- Buat workspace catkin dengan perintah **mkdir -p ~/catkin_ws/src**.
- Sumber lingkungan ROS perlu diaktifkan dengan perintah **source /opt/ros/noetic/setup.bash**.
- Inisialisasi workspace catkin dengan perintah **catkin_init_workspace**.

Membangun Workspace:

- Pindah ke folder workspace src dengan perintah **cd ~/catkin_ws/src**.
- Jalankan **catkin_make** untuk membangun workspace.
- Sumberkan file setup.bash setiap kali sesi bash baru dimulai.

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

- Setelah membuat paket ini, buat paket tanpa menambahkan node apa pun dengan menggunakan perintah catkin_make. Perintah ini harus dijalankan dari ruang kerja catkin jalur. Perintah berikut menunjukkan cara membuat paket ROS kosong kami:

```
cd ~/catkin_ws && catkin_make
```

Membuat Node ROS:

node pertama yang akan kita bahas adalah `demo_topic_publisher.cpp`. Node ini akan mempublikasikan nilai integer pada topik yang disebut `/numbers`. Salin kode saat ini ke yang baru paket atau gunakan file yang ada dari repositori kode buku ini.

Membuat Paket ROS:

- Gunakan perintah **`catkin_create_pkg`** untuk membuat paket ROS.
- Contoh: **`catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs actionlib actionlib_msgs`**.
- Tambahkan dependensi sesuai kebutuhan.

Penggunaan `roscore.xml`:

- File `roscore.xml` mengonfigurasi roscore dan menyimpan parameter serta node dalam grup dengan namespace `/`.

Memahami Output roscore:

- Periksa topik, parameter, dan layanan ROS setelah menjalankan roscore dengan perintah **rostopic list**, **rosparam list**, dan **rosservice list**.

Pengerjaan ROS Nodes:

- Gunakan perintah **catkin_make** untuk membangun paket setelah membuatnya.
- Tambahkan ROS nodes ke folder src dalam paket.

Mengerjakan ROS Topics:

- Topik digunakan sebagai metode komunikasi antara node ROS.
- Gunakan **demo_topic_publisher.cpp** untuk mempublikasikan topik dan **demo_topic_subscriber.cpp** untuk berlangganan.

Node demo_topic_publisher.cpp:

- Menginisialisasi node dan Nodehandle.
- Membuat publisher untuk topik `"/numbers"` dengan tipe pesan `std_msgs::Int32`.
- Mengatur frekuensi utama dan loop untuk mempublikasikan nilai integer ke topik `"/numbers"`.
- Menggunakan Ctrl + C untuk menghentikan loop.

Node Publisher (demo_topic_publisher.cpp):

- Memanfaatkan **ROS_INFO** untuk mencetak data pesan.
- Menggunakan **number_publisher.publish(msg)** untuk mempublikasikan pesan ke jaringan ROS.
- Menggunakan **loop_rate.sleep()** untuk memberikan penundaan dan mencapai frekuensi 10 Hz.
- Membahas publisher node yang mempublikasikan nilai integer ke topik `"/numbers"`.

Node Subscriber (demo_topic_subscriber.cpp):

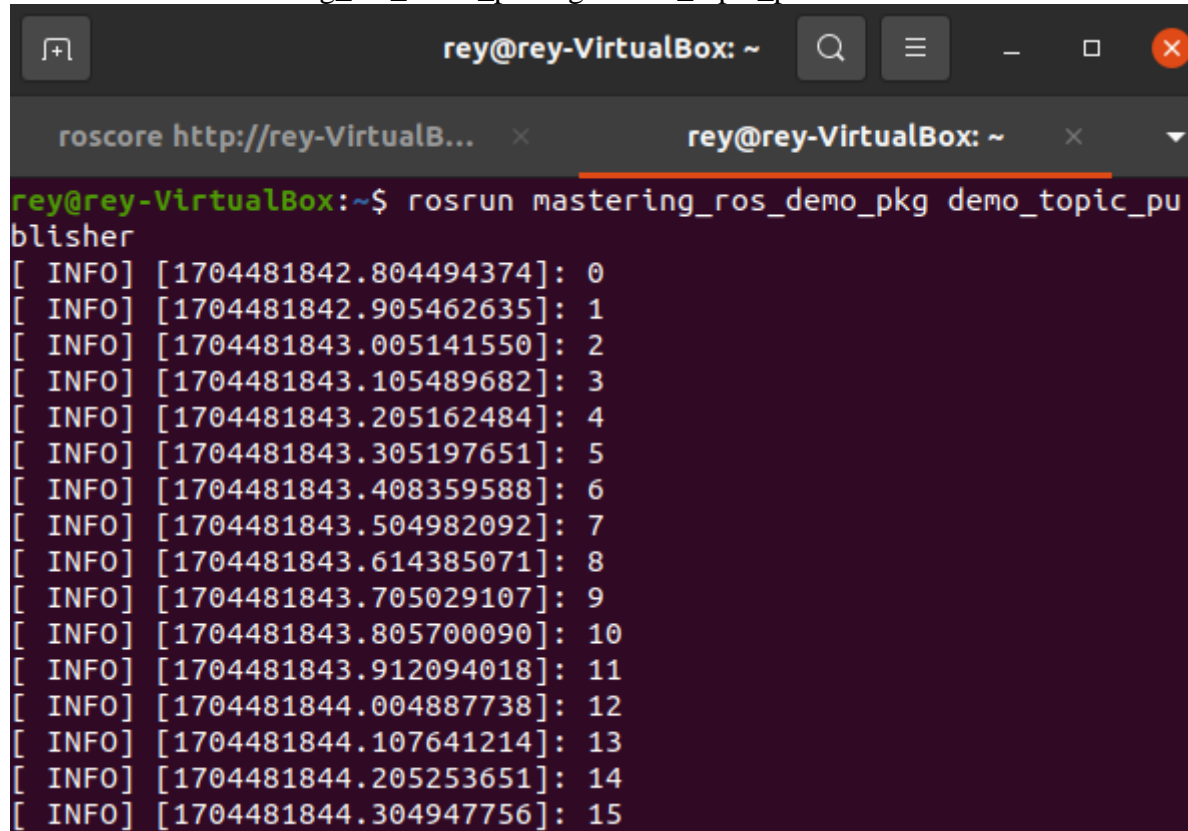
- Menggunakan **ros::Subscriber** untuk berlangganan ke topik `"/numbers"`.
- Mendefinisikan fungsi **number_callback** yang dijalankan saat pesan datang.
- Menampilkan nilai data dari pesan yang diterima.
- Menggunakan **ros::spin()** untuk menjaga agar node tetap berjalan.

Membangun Nodes:

- Mengedit file **CMakeLists.txt** di dalam paket untuk membangun dan mengompilasi kode sumber.

- Menggunakan perintah **catkin_make** di dalam workspace.
- Sekarang, jalankan kedua perintah dalam dua shell. Di penerbit yang sedang berjalan, jalankan yang berikut ini memerintah:

roslaunch mastering_ros_demo_package demo_topic_publisher

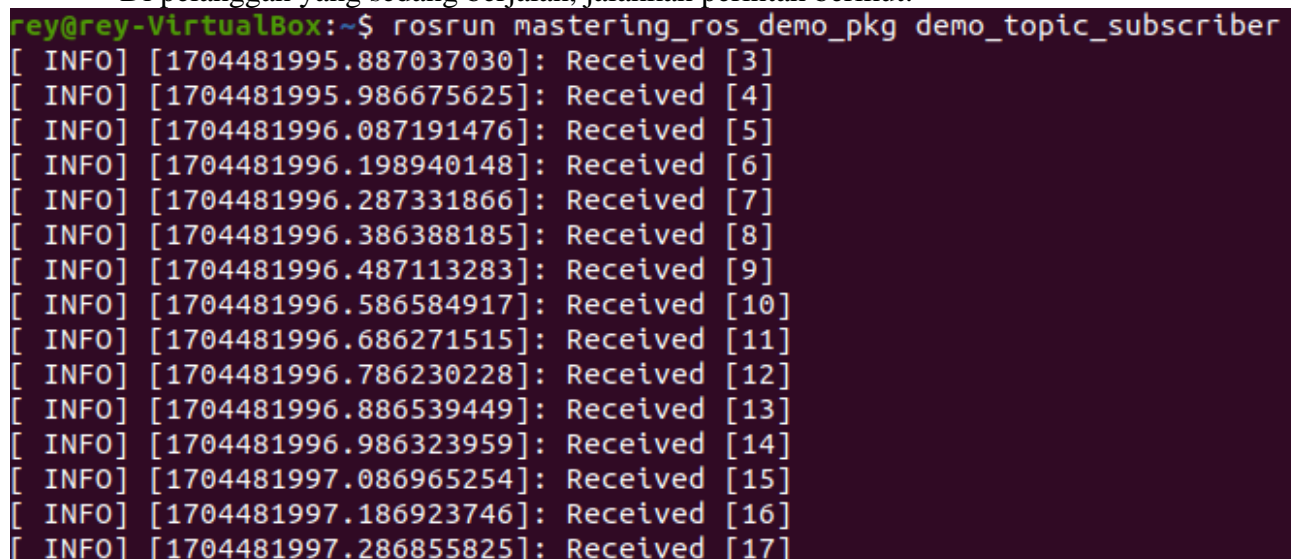


```

rey@rey-VirtualBox: ~$ roslaunch mastering_ros_demo_package demo_topic_publisher
[ INFO] [1704481842.804494374]: 0
[ INFO] [1704481842.905462635]: 1
[ INFO] [1704481843.005141550]: 2
[ INFO] [1704481843.105489682]: 3
[ INFO] [1704481843.205162484]: 4
[ INFO] [1704481843.305197651]: 5
[ INFO] [1704481843.408359588]: 6
[ INFO] [1704481843.504982092]: 7
[ INFO] [1704481843.614385071]: 8
[ INFO] [1704481843.705029107]: 9
[ INFO] [1704481843.805700090]: 10
[ INFO] [1704481843.912094018]: 11
[ INFO] [1704481844.004887738]: 12
[ INFO] [1704481844.107641214]: 13
[ INFO] [1704481844.205253651]: 14
[ INFO] [1704481844.304947756]: 15

```

Di pelanggan yang sedang berjalan, jalankan perintah berikut:



```

rey@rey-VirtualBox: ~$ roslaunch mastering_ros_demo_package demo_topic_subscriber
[ INFO] [1704481995.887037030]: Received [3]
[ INFO] [1704481995.986675625]: Received [4]
[ INFO] [1704481996.087191476]: Received [5]
[ INFO] [1704481996.198940148]: Received [6]
[ INFO] [1704481996.287331866]: Received [7]
[ INFO] [1704481996.386388185]: Received [8]
[ INFO] [1704481996.487113283]: Received [9]
[ INFO] [1704481996.586584917]: Received [10]
[ INFO] [1704481996.686271515]: Received [11]
[ INFO] [1704481996.786230228]: Received [12]
[ INFO] [1704481996.886539449]: Received [13]
[ INFO] [1704481996.986323959]: Received [14]
[ INFO] [1704481997.086965254]: Received [15]
[ INFO] [1704481997.186923746]: Received [16]
[ INFO] [1704481997.286855825]: Received [17]

```

Menambahkan file .msg dan .srv khusus

Edit file CMakeLists.txt saat ini dan tambahkan baris message_generasi, sebagai berikut:

```
find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  message_generation
)
```

Batalkan komentar pada baris berikut dan tambahkan file pesan khusus:

```
--
53 add_message_files(
54   FILES
55   demo_msg.msg
56 )

71 ## Generate added messages and services with any dependencies listed here
72 generate_messages(
```

Untuk memeriksa apakah pesan telah dibuat dengan benar, kita dapat menggunakan perintah rosmmsg:

Mari buat paket menggunakan catkin_make dan uji node dengan mengikuti langkah-langkah berikut:

- Run roscore:
roscore
- Mulai simpul penerbit pesan khusus:

```
rey@rey-VirtualBox:~$ rosrund mastering_ros_demo_pkg demo_msg_publisher
[ INFO] [1704482414.784470639]: 0
[ INFO] [1704482414.785439627]: hello world
[ INFO] [1704482414.891753012]: 1
[ INFO] [1704482414.891879177]: hello world
[ INFO] [1704482415.024866792]: 2
[ INFO] [1704482415.024980470]: hello world
[ INFO] [1704482415.084463493]: 3
[ INFO] [1704482415.084575793]: hello world
[ INFO] [1704482415.186107887]: 4
[ INFO] [1704482415.186251607]: hello world
[ INFO] [1704482415.295895959]: 5
[ INFO] [1704482415.296031093]: hello world
[ INFO] [1704482415.386055763]: 6
[ INFO] [1704482415.386103650]: hello world
[ INFO] [1704482415.485145395]: 7
[ INFO] [1704482415.485444412]: hello world
[ INFO] [1704482415.584526632]: 8
[ INFO] [1704482415.584662157]: hello world
[ INFO] [1704482415.685831965]: 9
[ INFO] [1704482415.685953654]: hello world
[ INFO] [1704482415.784713731]: 10
[ INFO] [1704482415.784847877]: hello world
```

- Mulai node pelanggan pesan khusus:

```

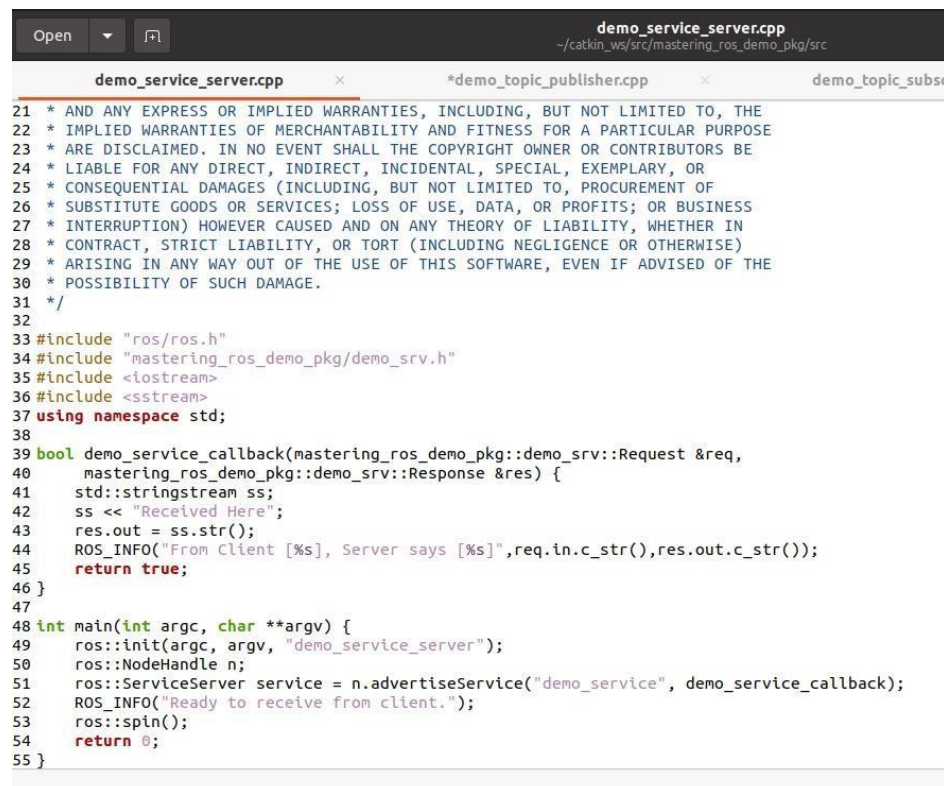
rey@rey-VirtualBox:~$ roslaunch mastering_ros_demo_pkg demo_msg_subscriber
[ INFO] [1704482502.267782970]: Received greeting [hello world ]
[ INFO] [1704482502.269670805]: Received [3]
[ INFO] [1704482502.366859134]: Received greeting [hello world ]
[ INFO] [1704482502.367008953]: Received [4]
[ INFO] [1704482502.468276755]: Received greeting [hello world ]
[ INFO] [1704482502.468986341]: Received [5]
[ INFO] [1704482502.566847087]: Received greeting [hello world ]
[ INFO] [1704482502.566936779]: Received [6]
[ INFO] [1704482502.666637437]: Received greeting [hello world ]
[ INFO] [1704482502.666720842]: Received [7]
[ INFO] [1704482502.766820507]: Received greeting [hello world ]
[ INFO] [1704482502.766907194]: Received [8]
[ INFO] [1704482502.866566523]: Received greeting [hello world ]
[ INFO] [1704482502.866595862]: Received [9]
[ INFO] [1704482502.967700777]: Received greeting [hello world ]
[ INFO] [1704482502.967832039]: Received [10]
[ INFO] [1704482503.068893524]: Received greeting [hello world ]

```

Bekerja dengan Layanan ROS

Arahkan ke `mastering_ros_demo_pkg/src` dan temukan `demo_service_node` `server.cpp` dan `demo_service_client.cpp`.

`demo_service_server.cpp` adalah server, dan definisinya adalah sebagai berikut:



```

demo_service_server.cpp
~/catkin_ws/src/mastering_ros_demo_pkg/src

demo_service_server.cpp  *demo_topic_publisher.cpp  demo_topic_subsc

21 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
22 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
23 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
24 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
25 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
26 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
27 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
28 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
29 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
30 * POSSIBILITY OF SUCH DAMAGE.
31 */
32
33 #include "ros/ros.h"
34 #include "mastering_ros_demo_pkg/demo_srv.h"
35 #include <iostream>
36 #include <sstream>
37 using namespace std;
38
39 bool demo_service_callback(mastering_ros_demo_pkg::demo_srv::Request &req,
40   mastering_ros_demo_pkg::demo_srv::Response &res) {
41   std::stringstream ss;
42   ss << "Received Here";
43   res.out = ss.str();
44   ROS_INFO("From Client [%s], Server says [%s]", req.in.c_str(), res.out.c_str());
45   return true;
46 }
47
48 int main(int argc, char **argv) {
49   ros::init(argc, argv, "demo_service_server");
50   ros::NodeHandle n;
51   ros::ServiceServer service = n.advertiseService("demo_service", demo_service_callback);
52   ROS_INFO("Ready to receive from client.");
53   ros::spin();
54   return 0;
55 }

```

Mari kita jelaskan kode ini. Pertama, kami menyertakan file header untuk mendefinisikan layanan yang kami ingin digunakan dalam kode:

```
demo_service_client.cpp x demo_service_server.cpp x
34 #include "mastering_ros_demo_pkg/demo_srv.h"
35 #include <iostream>
36 #include <sstream>
37 using namespace std;
38
39 int main(int argc, char **argv)
40 {
41     ros::init(argc, argv, "demo_service_client");
42     ros::NodeHandle n;
43     ros::Rate loop_rate(10);
44     ros::ServiceClient client = n.serviceClient<mastering_ros_demo_pkg::demo_srv>("demo_service");
45     while (ros::ok())
46     {
47         mastering_ros_demo_pkg::demo_srv srv;
48         std::stringstream ss;
49         ss << "Sending from Here";
50         srv.request.in = ss.str();
51         if (client.call(srv))
52         {
53             ROS_INFO("From Client [%s], Server says [%s]", srv.request.in.c_str(), srv.response.out.c_str());
54         }
55         else
56         {
57             ROS_ERROR("Failed to call service");
58             return 1;
59         }
60     }
61     ros::spinOnce();
62     loop_rate.sleep();
63 }
64 return 0;
65 }
```

Chapter 3: Working with ROS for 3D Modeling

Persyaratan Teknis:

Untuk mengikuti contoh-contoh pada bab ini, diperlukan laptop standar yang menjalankan Ubuntu 20.04 dengan ROS Noetic terinstal. Kode referensi untuk bab ini dapat diunduh dari repositori Git di <https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git>.

Kode tersebut terdapat di dalam folder Chapter3/mastering_ros_robot_description_pkg/.

Paket-paket ROS untuk Pemodelan Robot:

Paket-paket ROS yang penting untuk membangun dan memodelkan robot termasuk urdf, joint_state_publisher, joint_state_publisher_gui, kdl_parser, robot_state_publisher, dan xacro. Paket-paket ini sangat penting untuk membuat, memvisualisasikan, dan berinteraksi dengan model robot.

Memahami Pemodelan Robot menggunakan URDF:

tag tautan: Mewakili satu tautan robot, termasuk properti seperti ukuran, bentuk, warna, dan properti dinamis. Terdiri dari bagian inersia, visual, dan tabrakan.

tag sambungan: Merepresentasikan sendi robot yang menghubungkan dua tautan. Mendukung berbagai jenis sambungan (berputar, kontinu, prismatic, tetap, mengambang, planar). Mendefinisikan kinematika, dinamika, dan batas gerakan.

tag robot: Mengenkapsulasi seluruh model robot, yang berisi link dan sendi.

tag gazebo: Digunakan untuk menyertakan parameter simulasi Gazebo di dalam URDF, termasuk plugin gazebo dan properti material.

Visualisasi elemen URDF termasuk tautan, sambungan, dan model robot.

Tag URDF dan detail lebih lanjut dapat ditemukan di <http://wiki.ros.org/urdf/XML>.

Langkah selanjutnya:

Bagian selanjutnya akan melibatkan pembuatan paket ROS baru yang berisi deskripsi robot yang berbeda.

Membuat paket ROS untuk robot keterangan:

Sebelum membuat file URDF untuk robot

Paket ini terutama bergantung pada paket urdf dan xacro. Jika paket ini punya belum diinstal pada sistem Anda,

Membuat model URDF pertama kami

Mari kita lihat kode URDF dari mekanisme ini. Arahkan ke `mastering_direktori/robo_robot_description_pkg/urdf` dan buka `pan_tilt.urdf`.

Menjelaskan file URDF

Simpan kode URDF sebelumnya sebagai pan_tilt.urdf dan periksa apakah file urdf mengandung kesalahan

Berinteraksi dengan sambungan pan-and-tilt

Kita dapat memasukkan node ini ke dalam file peluncuran menggunakan pernyataan berikut. Batasan dari pan-and-tilt harus disebutkan di dalam tag gabungan:


```

28 <joint name="pan_joint" type="revolute">
29   <parent link="base_link"/>
30   <child link="pan_link"/>
31   <origin xyz="0 0 0.1"/>
32   <axis xyz="0 0 1"/>
33   <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/>
34   <dynamics damping="50" friction="1"/>
35 </joint>
36
37 <link name="pan_link">
38   <visual>
39     <geometry>
40       <cylinder length="0.4" radius="0.04"/>
41     </geometry>
42     <origin rpy="0 0 0" xyz="0 0 0.09"/>
43     <material name="red">
44       <color rgba="0 0 1 1"/>
45     </material>
46   </visual>
47   <collision>
48     <geometry>
49       <cylinder length="0.4" radius="0.06"/>
50     </geometry>
51     <origin rpy="0 0 0" xyz="0 0 0.09"/>
52   </collision>
53   <inertial>
54     <mass value="1"/>
55     <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
56   </inertial>
57 </link>

```

```

59 <joint name="tilt_joint" type="revolute">
60   <parent link="pan_link"/>
61   <child link="tilt_link"/>
62   <origin xyz="0 0 0.2"/>
63   <axis xyz="0 1 0"/>
64   <limit effort="300" velocity="0.1" lower="-4.64" upper="-1.5"/>
65   <dynamics damping="50" friction="1"/>
66 </joint>
67
68 <link name="tilt_link">
69   <visual>
70     <geometry>
71       <cylinder length="0.4" radius="0.04"/>
72     </geometry>
73     <origin rpy="0 1.5 0" xyz="0 0 0"/>
74     <material name="green">
75       <color rgba="1 0 0 1"/>
76     </material>
77   </visual>
78   <collision>
79     <geometry>
80       <cylinder length="0.4" radius="0.06"/>
81     </geometry>
82     <origin rpy="0 1.5 0" xyz="0 0 0"/>
83   </collision>
84   <inertial>
85     <mass value="1"/>
86     <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
87   </inertial>

```

Chapter 4: Simulating Robots Using ROS and Gazebo

Berikut ini adalah perincian yang disederhanakan dari teks yang diberikan:

Persyaratan Teknis:

- Laptop standar dengan Ubuntu 20.04 dan ROS Noetic.
- Kode tersedia di Git: [tautan](#).
- Model simulasi dalam folder Bab4/seven_dof_arm_gazebo.
- Lihat kode yang sedang bekerja: [tautan](#).

Simulasi Lengan Robot di Gazebo dan ROS:

- Merancang lengan tujuh DOF pada bab sebelumnya.
- Simulasi di Gazebo menggunakan ROS.
- Menginstal paket-paket yang diperlukan untuk Gazebo dan ROS.
- Setelah instalasi, periksa apakah Gazebo sudah terpasang dengan benar menggunakan

Membuat Model Simulasi Lengan Robot untuk Gazebo:

- Membuat paket untuk mensimulasikan lengan robot.
- Model simulasi dalam file seven_dof_arm.xacro.

Menambahkan Warna dan Tekstur pada Model Robot Gazebo:

- Tentukan warna dan tekstur dalam file .xacro.

Menambahkan Tag Transmisi untuk Menggerakkan Model:

- Tentukan elemen transmisi untuk menghubungkan aktuator ke sendi.

Menambahkan Plugin Gazebo_ros_control:

- Tambahkan plugin gazebo_ros_control untuk mengurai tag transmisi.

Menambahkan Sensor Visi 3D ke Gazebo:

- Mengintegrasikan sensor penglihatan 3D (Asus Xtion Pro) di Gazebo.

Mensimulasikan Lengan Robot dengan Xtion Pro:

- Luncurkan simulasi lengkap dengan sensor Xtion Pro.

Memvisualisasikan Data Sensor 3D:

- Melihat gambar RGB, IR, dan kedalaman.
- Memvisualisasikan data point cloud di RViz.

Menggerakkan Sendi Robot menggunakan Pengontrol ROS di Gazebo:

- Mengonfigurasi pengontrol ROS untuk status dan posisi sendi.
- Gambaran umum tentang pengontrol ROS dan antarmuka perangkat keras.
- Interaksi pengontrol ROS dengan Gazebo.

Menghubungkan Pengontrol Status Gabungan dan Pengontrol Posisi Gabungan:

- File konfigurasi untuk pengontrol status dan posisi bersama.
- Definisi pengontrol untuk setiap sambungan dengan penguatan PID.

```
Open  seven_dof_arm_gazebo_control.yaml
~/Downloads/Mastering-ROS-for-Rob...pter4/seven_dof_arm_gazebo/config

1 seven_dof_arm:
2   # Publish all joint states -----
3   joint_state_controller:
4     type: joint_state_controller/JointStateController
5     publish_rate: 50
6
7   # Position Controllers -----
8   joint1_position_controller:
9     type: position_controllers/JointPositionController
10    joint: shoulder_pan_joint
11    pid: {p: 100.0, i: 0.01, d: 10.0}
12  joint2_position_controller:
13    type: position_controllers/JointPositionController
14    joint: shoulder_pitch_joint
15    pid: {p: 100.0, i: 0.01, d: 10.0}
16  joint3_position_controller:
17    type: position_controllers/JointPositionController
18    joint: elbow_roll_joint
19    pid: {p: 100.0, i: 0.01, d: 10.0}
20  joint4_position_controller:
21    type: position_controllers/JointPositionController
22    joint: elbow_pitch_joint
23    pid: {p: 100.0, i: 0.01, d: 10.0}
24  joint5_position_controller:
25    type: position_controllers/JointPositionController
26    joint: wrist_roll_joint
27    pid: {p: 100.0, i: 0.01, d: 10.0}
28  joint6_position_controller:
29    type: position_controllers/JointPositionController
30    joint: wrist_pitch_joint
31    pid: {p: 100.0, i: 0.01, d: 10.0}
32  joint7_position_controller:
33    type: position_controllers/JointPositionController
34    joint: gripper_roll_joint
35    pid: {p: 100.0, i: 0.01, d: 10.0}
36
```

Meluncurkan pengendali ROS dengan Gazebo:

- Buat berkas peluncuran di direktori seven_dof_arm_gazebo/launch.
- Sertakan peluncuran Gazebo dan muat konfigurasi pengontrol bersama dari file YAML.
- Muat pengontrol menggunakan paket controller_manager.
- Jalankan penerbit status robot untuk status gabungan dan transformasi.

```
Open  seven_dof_arm_gazebo_control.launch
~/Downloads/Mastering-ROS-for-Ro...ter4/seven_dof_arm_gazebo/launch

1 <?xml version="1.0" ?>
2
3 <launch>
4   <!-- Launch Gazebo -->
5   <include file="$(find seven_dof_arm_gazebo)/launch/seven_dof_arm_world.launch" />
6
7
8   <!-- Load joint controller configurations from YAML file to parameter server -->
9   <rosparam file="$(find seven_dof_arm_gazebo)/config/seven_dof_arm_gazebo_control.yaml" command="load"/>
10
11
12   <!-- load the controllers -->
13   <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
14     output="screen" ns="/seven_dof_arm" args="joint_state_controller
15     joint1_position_controller
16     joint2_position_controller
17     joint3_position_controller
18     joint4_position_controller
19     joint5_position_controller
20     joint6_position_controller
21     joint7_position_controller"/>
22
23
24   <!-- convert joint states to TF transforms for rviz, etc -->
25   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
26     respawn="false" output="screen"
27     <remap from="/joint_states" to="/seven_dof_arm/joint_states" />
28   </node>
29
30 </launch>
```

Memeriksa Topik Pengontrol:

- Gunakan `roslaunch seven_dof_arm_gazebo seven_dof_arm_gazebo_control.launch` untuk memeriksa topik pengontrol.
- Konfirmasikan peluncuran yang berhasil dengan pesan spesifik di terminal.
- Topik yang dihasilkan termasuk perintah pengontrol posisi untuk setiap sendi.

Menggerakkan Sendi Robot:

- Perintahkan setiap sendi dengan menerbitkan nilai yang diinginkan ke topik perintah pengontrol posisi sendi.
- Contoh: `rostopic pub /seven_dof_arm/joint4_position_controller/command std_msgs/Float64 1.0`.
- Lihat status gabungan dengan `rostopic echo /seven_dof_arm/joint_states`.

Mensimulasikan Robot Beroda Diferensial di Gazebo:

- Siapkan simulasi untuk robot beroda diferensial.
- Buat file peluncuran di `diff_wheeled_robot_gazebo/launch`.
- Luncurkan menggunakan `roslaunch diff_wheeled_robot_gazebo diff_wheeled_gazebo.launch`.
- Memvisualisasikan robot di Gazebo.

Menambahkan Pemindai Laser ke Gazebo:

- Ubah `diff_wheeled_robot.xacro` untuk menyertakan pemindai laser.
- Konfigurasi informasi khusus Gazebo untuk plugin pemindai laser.
- Memvisualisasikan data pemindai laser dengan objek yang ditambahkan di Gazebo.

Memindahkan Robot Bergerak di Gazebo:

- Tambahkan plugin `libgazebo_ros_diff_drive.so` untuk perilaku penggerak diferensial.
- Tentukan parameter seperti sambungan roda, pemisahan, diameter, dll.
- Sertakan penerbit status gabungan dalam file peluncuran.

Node Teleop ROS:

- Gunakan node `diff_wheeled_robot_key` untuk teleoperasi.
- Sesuaikan skala linier dan sudut.

- Luncurkan teleop dengan `roslaunch diff_wheeled_robot_control keyboard_teleop.launch`.

Visualisasi di RViz:

- Gunakan RViz untuk memvisualisasikan status robot dan data laser.
- Atur Bingkai Tetap ke /odom dan tambahkan Pemindaian Laser dengan topik /scan.
- Tambahkan model Robot untuk dilihat.

Memindahkan Robot:

- Gunakan tombol di terminal teleop (U, I, O, J, K, L, M, koma, titik) untuk penyesuaian arah.
- Gunakan tombol lain (Q, Z, W, X, E, C, K, spasi) untuk penyesuaian kecepatan.

Menjelajahi Area:

- Robot hanya bergerak jika tombol yang sesuai ditekan di terminal simpul teleop.
- Jelajahi area menggunakan robot yang dioperasikan secara teleop dan visualisasikan data laser di RViz.

Chapter 5: Simulating Robots Using ROS, Coppeliasim, and Webots

Persyaratan Teknis:

- Laptop standar dengan Ubuntu 20.04 dan ROS Noetic.
- Unduh kode dari: Menguasai-ROS-untuk-Pemrograman-Robotika-Edisi-ketiga. Gunakan kode dari folder Bab5/csim_demo_pkg dan Bab5/webost_demo_pkg.
- Melihat kode yang sedang bekerja: Kode Bab 5.

Menyiapkan Coppeliasim dengan ROS:

- Unduh dan ekstrak Coppeliasim 4.2.0 dari halaman unduhan Coppelia Robotics, pilih versi edu untuk Linux.
- Pindah ke folder unduhan dan jalankan: `tar vxf Coppeliasim_Edu_V4_2_0_Ubuntu20_04.tar.xz`
- Ganti nama folder untuk kenyamanan: `mv Coppeliasim_Edu_V4_2_0_Ubuntu20_04 Coppeliasim`.
- Atur variabel lingkungan COPPELIASIM_ROOT: `echo "export COPPELIASIM_ROOT=/path/to/Coppeliasim/folder" >> ~/.bashrc`.

Mode Coppeliasim untuk Robot Simulasi:

- API jarak jauh: Fungsi yang dapat dipanggil dari aplikasi eksternal (C/C++, Python, Lua, MATLAB). Membutuhkan sisi klien (aplikasi eksternal) dan server (skrip Coppeliasim).
- RosInterface: Antarmuka saat ini untuk komunikasi ROS, menggantikan plugin ROS yang sudah usang. Mereplikasi fungsi API jarak jauh.

Memulai Coppeliasim dengan ROS:

- Jalankan roscore sebelum membuka Coppeliasim.
- Memulai Coppeliasim: `cd $COPPELIASIM_ROOT && ./coppeliasim.sh`.

Memeriksa Pengaturan:

- Verifikasi node ROS yang aktif setelah meluncurkan Coppeliasim.

Berinteraksi dengan Coppeliasim menggunakan Topik ROS:

- Gunakan topik ROS untuk mengirim/menerima data ke/dari objek simulasi.
- Objek CoppeliaSim dapat dikaitkan dengan skrip Lua untuk dieksekusi selama simulasi.
- Skrip Lua menggunakan simROS untuk berinteraksi dengan ROS.

Memahami Plugin RosInterface:

- Bagian dari kerangka kerja API CoppeliaSim.
- Plugin ROS harus dimuat selama startup CoppeliaSim.
- Jelajahi fungsi plugin RosInterface menggunakan scene plugin_publisher_subscriber.ttt.

Berinteraksi dengan CoppeliaSim menggunakan Topik ROS (lanjutan):

- Gunakan skrip Lua untuk mempublikasikan dan berlangganan topik ROS.
- Contoh: skrip dummy_publisher dan dummy_subscriber menukar data bilangan bulat pada topik /number.

Bekerja dengan Pesan ROS:

- Membungkus pesan ROS dalam skrip Lua untuk menerbitkan dan mengekstrak informasi.
- Contoh: Menerbitkan gambar dari sensor kamera dalam adegan simulasi.

Mensimulasikan Lengan Robotik menggunakan CoppeliaSim dan ROS:

- Mengimpor model URDF lengan tujuh-DOF ke dalam CoppeliaSim.
- Aktifkan motor untuk gerakan sendi. Menyetel penguatan PID untuk kinerja loop kontrol.
- Menguji gerakan sendi dengan mengatur posisi target.

Menambahkan Antarmuka ROS ke Pengontrol Bersama CoppeliaSim:

- Antarmuka lengan tujuh-DOF dengan plugin RosInterface untuk kontrol bersama.
- Gunakan skrip Lua untuk mempublikasikan status gabungan dan berlangganan perintah gabungan melalui topik ROS.
- Contoh: sysCall_init menginisialisasi penanganan sambungan dan mengatur penerbit/pelanggan.
- Mengontrol sambungan menggunakan perintah ROS, misalnya, rostopic pub /csim_demo/seven_dof_arm/elbow_pitch/cmd std_msgs/Float32 "data: 1.0".

- Memantau status sambungan, misalnya, rostopic echo /csim_demo/seven_dof_arm/elbow_pitch/state.

Membuat Webots dengan ROS

1. **Instalasi Webots:** Unduh Webots dari situs web resmi (<http://www.cyberbotics.com/#download>) atau gunakan Debian/Ubuntu APT package manager dengan langkah-langkah berikut:


```
wget -qO- https://cyberbotics.com/Cyberbotics.asc | sudo apt-key add - sudo apt-add-repository 'deb https://cyberbotics.com/debian/ binary-amd64/' sudo apt-get update sudo apt-get install webots
```
2. **Mulai Webots:** Ketikkan perintah berikut untuk membuka antarmuka pengguna Webots:


```
$ webots
```
3. **Mengenal Simulasi Webots:**
 - Konfigurasi Dunia: Gunakan file konfigurasi dunia (.wbt) untuk mendefinisikan lingkungan simulasi.
 - Kontroler: Setiap simulasi diatur oleh satu atau lebih program kontroler yang dapat diimplementasikan dalam bahasa seperti C, C++, Python, atau Java.
 - Plugin Fisik: Modifikasi perilaku fisik simulasi menggunakan plugin yang ditulis dalam bahasa yang sama dengan kontroler.

Simulasi Robot Bergerak dengan Webots

1. **Membuat Adegan Simulasi:** Gunakan wizard untuk membuat adegan simulasi baru dengan memilih Wizards | New Project Directory dan mengonfigurasi direktori dan nama proyek.
2. **Menambahkan Objek ke Adegan:**
 - Pilih RectangleArea dari panel hirarki.
 - Klik tombol + untuk menambahkan objek dan pilih PROTO nodes | objects | factory | containers | WoodenBox (Solid).

- Pilih RectangleArea dan tambahkan robot e-puck dengan memilih (Webots Projects) / robots / gctronic / e-puck / E-puck PROTO.

3. Konfigurasi Objek:

- Klik dua kali pada RectangleArea untuk mengubah ukuran lantai.
- Konfigurasi objek, seperti WoodenBox, melalui properti.

Menulis Kontroler Pertama Anda

1. Buat Kontroler Baru:

- Gunakan Wizards | New Robot Controller dan pilih C++.
- Compile kontroler menggunakan tombol Build.

2. Implementasikan Kontroler:

- Gunakan kontroler berikut untuk menggerakkan robot e-puck secara maju-mundur:
- // Kode kontroler C++ di sini

Simulasi Lengan Robot dengan Webots dan ROS

1. Instalasi Webots-ROS Package:

```
sudo apt-get install ros-noetic-webots-ros
```

2. Mengganti Kontroler Webots:

- Ganti kontroler robot Webots dengan kontroler ROS.

Menulis Node Teleop Menggunakan webots_ros

1. Buat Node ROS:

- Implementasikan node ROS untuk mengontrol kecepatan roda e-puck berdasarkan pesan geometry_msgs::Twist.
- Gunakan layanan Webots untuk mengatur kecepatan dan posisi roda.

Memulai Webots dengan Berkas Peluncuran

1. Berkas Peluncuran Webots:

- Gunakan berkas peluncuran yang disertakan di webots_ros untuk memulai Webots dan konfigurasi adegan simulasi.

Pastikan untuk menyesuaikan setiap langkah dengan versi perangkat lunak yang digunakan dan rincian spesifik sistem Anda.