

## LAPORAN SIMULASI MASTERING ROS FOR ROBOTICS PROGRAMMING, 3<sup>RD</sup> EDITION

### 1. Introduction to ROS

Video pertama ini menjelaskan pengertian ROS, manfaatnya, dan struktur file sistem ROS, serta pentingnya komunitas ROS bagi pengembang dan peneliti.

#### Keunggulan dari ROS

- **Kemampuan Tinggi:** ROS menyediakan fungsionalitas siap pakai untuk navigasi otonom.
- **Alat Ekosistem:** ROS dilengkapi dengan alat debugging dan simulasi.
- **Dukungan Sensor dan Aktuator:** Memungkinkan penggunaan driver perangkat untuk berbagai sensor.
- **Interoperabilitas:** Dapat diprogram dalam berbagai bahasa seperti C++, Python, dan Java.
- **Modularitas:** Pendekatan berbasis node memungkinkan sistem tetap berfungsi meskipun satu node gagal.
- **Struktur File Sistem:** Mengorganisir file dan direktori untuk kemudahan akses dan pengelolaan.
- **Komunitas ROS:** Sumber daya penting bagi pengembang dan peneliti untuk dukungan dan pengetahuan.

#### Mengapa kita menggunakan ROS?

- **Kemampuan Tinggi:** ROS menawarkan fungsionalitas yang mendukung pengembangan robotika, seperti navigasi dan perencanaan gerakan, membuatnya sangat efisien untuk pengembang.
- **Alat Ekosistem:** Dengan banyaknya alat yang tersedia, proses debugging dan visualisasi menjadi lebih mudah, meningkatkan produktivitas pengembang.
- **Dukungan Sensor dan Aktuator:** Dukungan yang luas untuk berbagai perangkat keras memungkinkan fleksibilitas dalam pengembangan robot yang kompleks.
- **Interoperabilitas:** Kemampuan untuk menggunakan berbagai bahasa pemrograman memperluas aksesibilitas dan kolaborasi antar pengembang dari berbagai latar belakang.
- **Modularitas:** Pendekatan berbasis node membuat sistem ROS lebih tahan terhadap kesalahan, meningkatkan keandalan dalam pengoperasian robot.
- **Struktur File Sistem:** Organisasi file yang jelas dalam ROS memudahkan pengelolaan dan pengembangan proyek, mengurangi waktu yang dibutuhkan untuk mencari informasi.
- **Komunitas ROS:** Komunitas yang aktif dan dukungan yang luas menjadikan ROS sebagai pilihan utama bagi pengembang dan peneliti dalam bidang robotika.

poin-poin utama seperti struktur file sistem, modularitas, dan komunitas ROS.  
Perintah ROS:

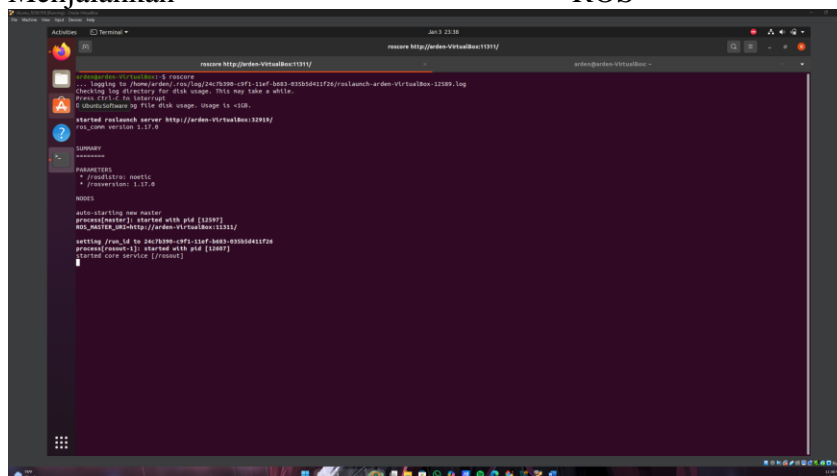
1. Membuat Paket ROS  
`catkin_create_pkg package_name [dependency1] [dependency2]`  
Contoh: `catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs actionlib actionlib_msgs`
2. Membangun Workspace  
`catkin_make` #Perintah ini digunakan untuk membangun paket-paket dalam workspace catkin.
3. Menjalankan ROS Master  
`roscore` #Perintah ini memulai ROS master, yang diperlukan untuk menjalankan node ROS.
4. Mendaftar Topik Aktif  
`rostopic list` #Perintah ini mencantumkan semua topik aktif dalam sistem ROS.
5. Menerbitkan ke Topik  
`rostopic pub /topic_name message_type args`  
Contoh: `rostopic pub /numbers std_msgs/Int32 10`
6. Menyubscribe ke Topik  
`rostopic echo /topic_name` #Perintah ini menampilkan pesan yang diterbitkan ke topik yang ditentukan.
7. Memeriksa Pesan  
`rosmmsg show message_type`  
Contoh: `rosmmsg show std_msgs/Int32`
8. Bekerja dengan Layanan Memanggil Layanan:  
`rosservice call /service_name args`  
Mendaftar Layanan: `rosservice list`  
Mendapatkan Info Layanan: `rosservice info /service_name`
9. Menggunakan Parameter Server  
Mengatur Parameter: `rosparam set parameter_name value`  
Mendapatkan Parameter: `rosparam get parameter_name`  
Mendaftar Parameter: `rosparam list`

- Contoh Comand

Menjalankan

ROS

Master



```
roscore http://arden-VirtualBox:11311/
roscore http://arden-VirtualBox:11311/
roscore http://arden-VirtualBox:11311/
... logging to /home/arden/.ros/log/24c7b300-c9f1-11ef-b883-02554ac1f226/roslaunch-arden-VirtualBox-12389.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
0 done (2000000) on 4112 disk usage. Stage 12 of 120
started roslaunch server: http://arden-VirtualBox:11311/
ros_core version 1.17.0

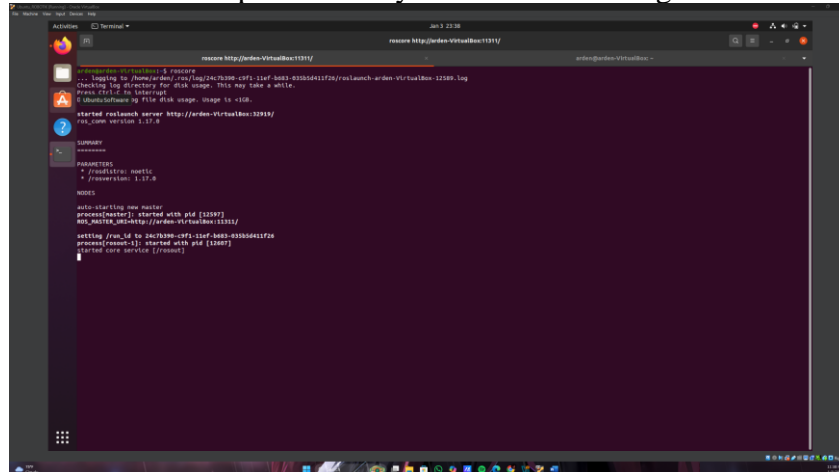
SUMMARY
-----
PARAMETERS
 * /roslaunch: roscpp
 * /roslaunch: 1.17.0

NODES
auto-starting new master
process[roscpp]: started with pid [12389]
MSG_MQTT_MQTT: http://arden-VirtualBox:11311/
setting /run_id to 24c7b300-c9f1-11ef-b883-02554ac1f226
process[roscpp]: started with pid [12389]
started core service [/roscpp]
```

## 2. Getting Started with ROS

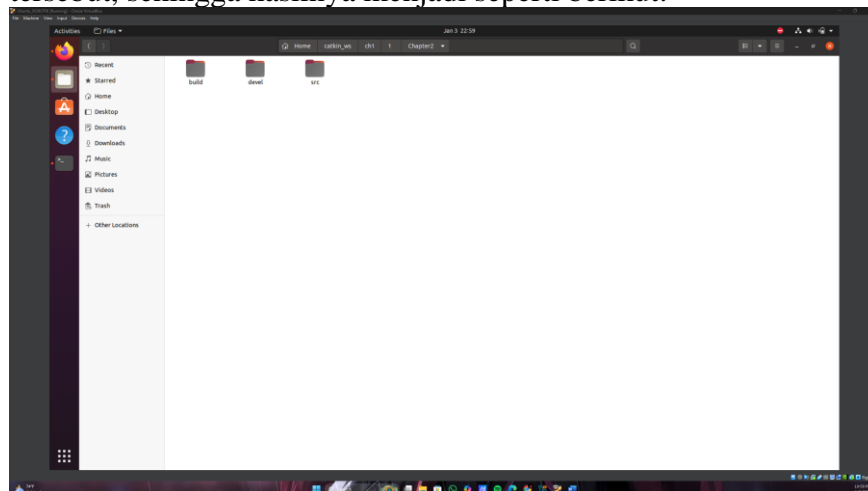
Disini saya akan menjelaskan bagaimana untuk setup ROS sampai berhasil run Publisher dan Subscribarnya.

1. Pertama kita perlu nyalakan ros dengan comen roscore

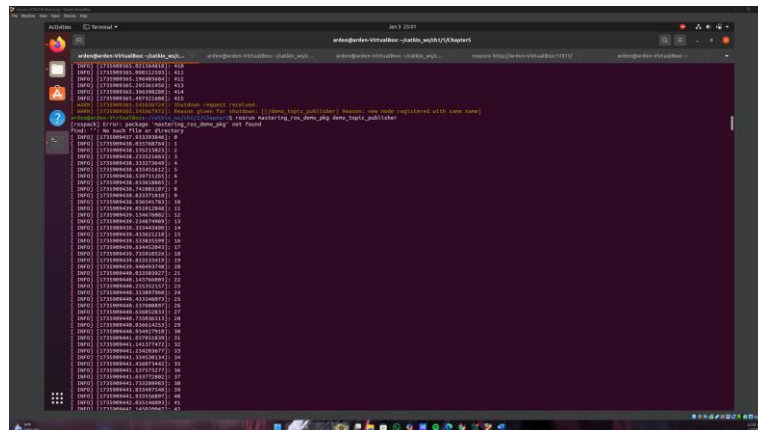


```
roscore http://arden-VirtualBox:11311/
: loading the theme arden.../usr/lib/74c79200-c0ff:11af:0002:033050412f2a/roslaunch-arden-VirtualBox-12389.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
! UbuntuSoftware File disk usage: usage is <10%.
started roslaunch server http://arden-VirtualBox:11311/
^C
SUMMARY
roscore
PARAMETERS
  * /roscore: roscore
  * /roscore: 1.17.8
roscore
auto-starting new master
process[master]: started with pid [12387]
MSG_MQTTM: started http://arden-VirtualBox:11311/
setting /roscore to 24c79200-c0ff:11af:0002:033050412f2a
process[roscore-1]: started with pid [12087]
started core service [/roscore]
```

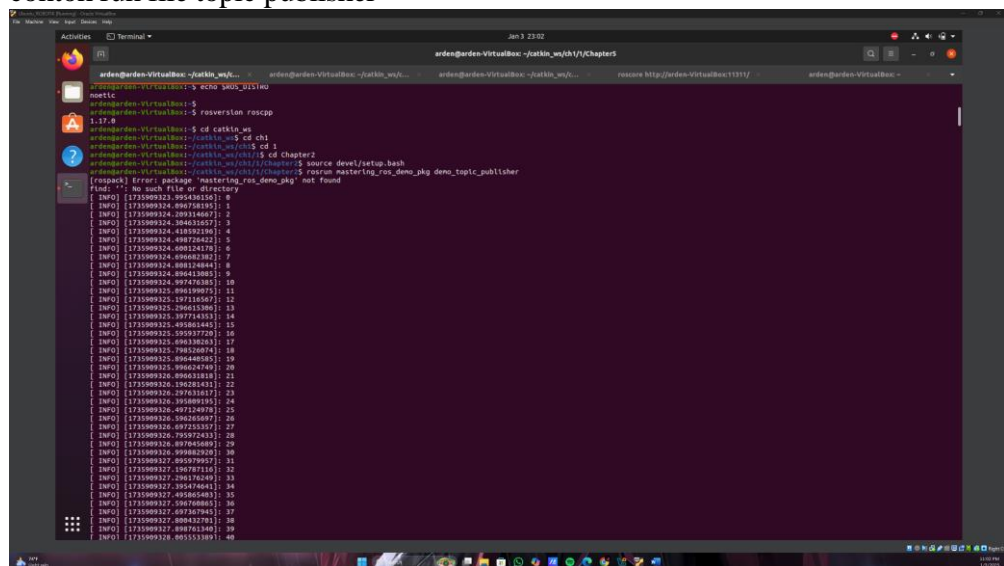
2. clone File Github dari repositori berikut dengan Comen:  
git clone <https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition>
3. Selanjutnya, Maukan Folder yang berisi file yang akan kita run nantinya dan kemudian kita masuk ke folder Chapter 2 dengan comen cd Chapter2, lalu ketik catkin\_make di terminal untuk build file yang ada di folder tersebut, sehingga hasilnya menjadi seperti berikut.



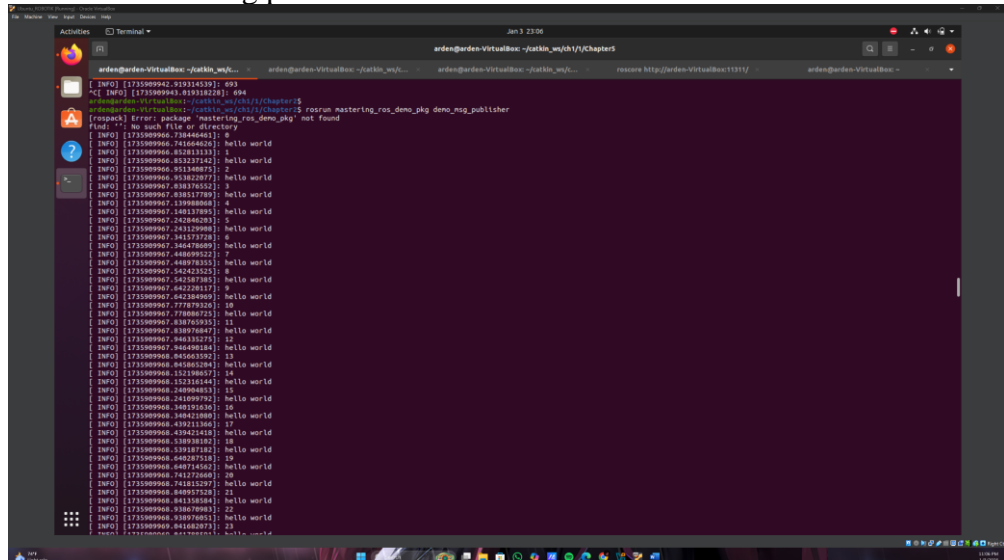
4. Setelah direktori berhasil di build, kita bisa menjalankan program di dalamnya. Pertama run roscore terlebih dahulu dengan cara ketik roscore jika belum di run di terminal Lalu kita bisa jalankan file nya dengan cara rosrn <nama folder> <nama file>, contohnya:



5. Hasil di run  
contoh run file topic publisher



### contoh run file msg publisher



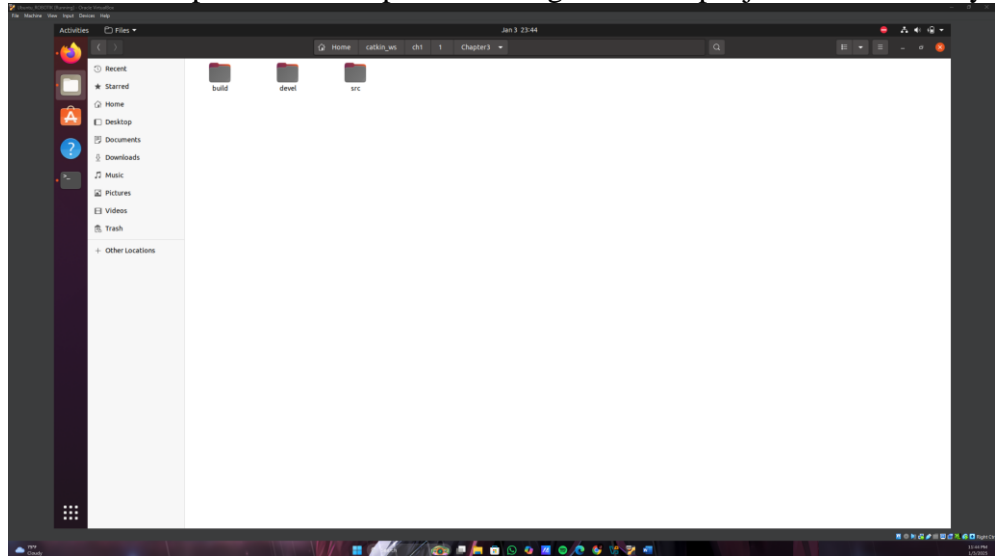


### 3. 3D Modeling with ROS

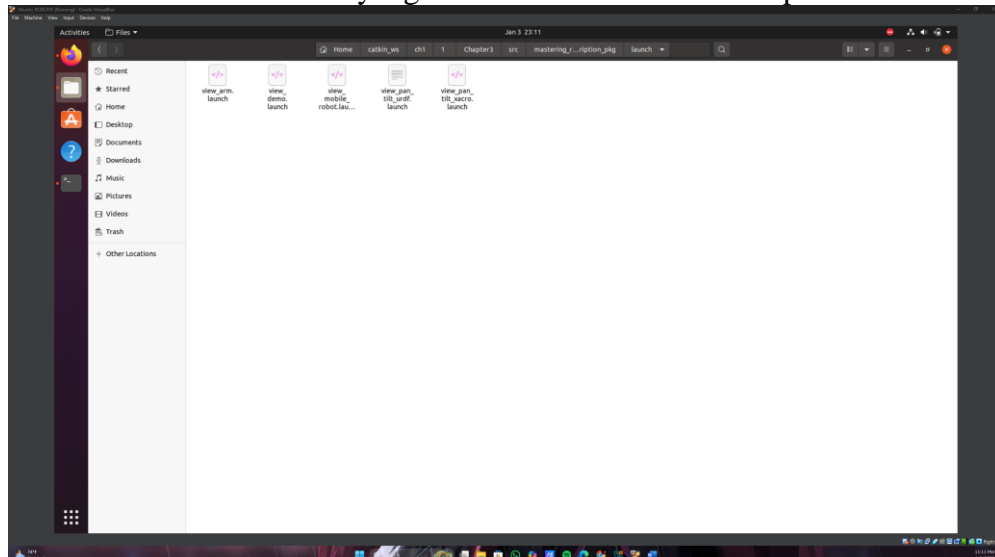
Bagian ini membahas pemodelan robot dalam format 3D menggunakan ROS, dengan fokus pada penggunaan **URDF (Unified Robot Description Format)** dan alat visualisasi seperti **RViz**. Pemodelan 3D memungkinkan pengembang untuk mendesain, memvisualisasikan, dan memeriksa model robot secara mendetail sebelum melakukan simulasi atau pengujian fisik.

Berikut Langkah-langkah untuk mengakses Rviz untuk 3D modeling:

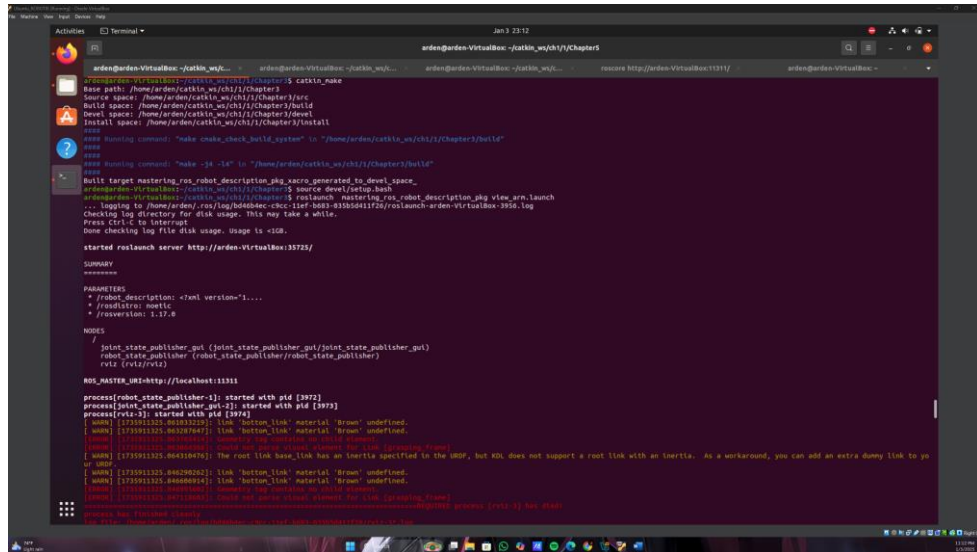
1. Pertama kita perlu build seperti di Langkah ke-2 penjelasan sebelumnya.



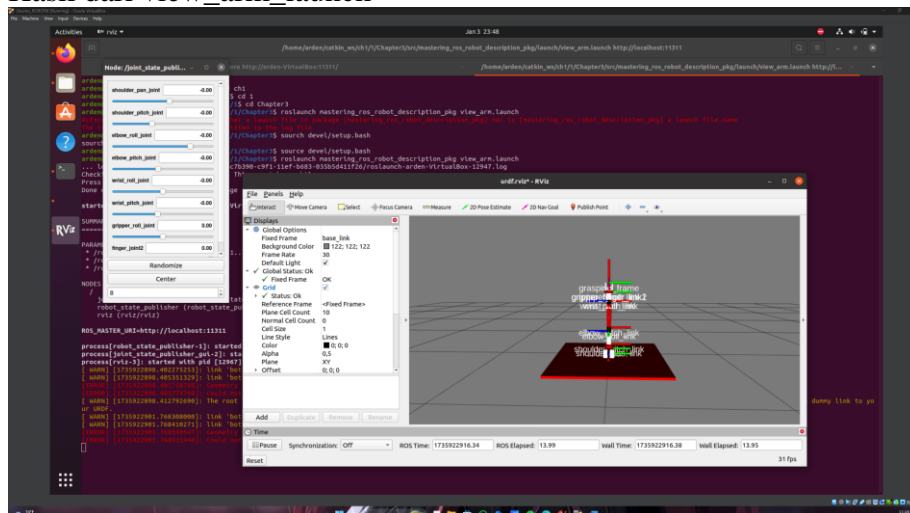
2. Lalu kita bisa run file yang ada di folder launch seperti berikut ini



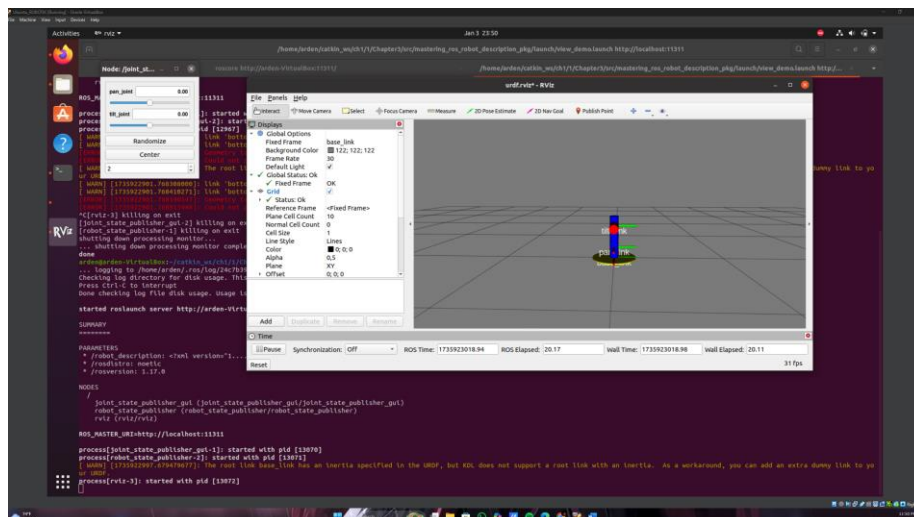
3. Misalkan kita ingin membuka file `view_arm` maka kita perlu ketik command ini di terminal



4. Lalu akan terbuka window Rviz seperti berikut ini  
Hasil dari view\_arm\_launch



Hasil dari file view\_demo\_\_launch





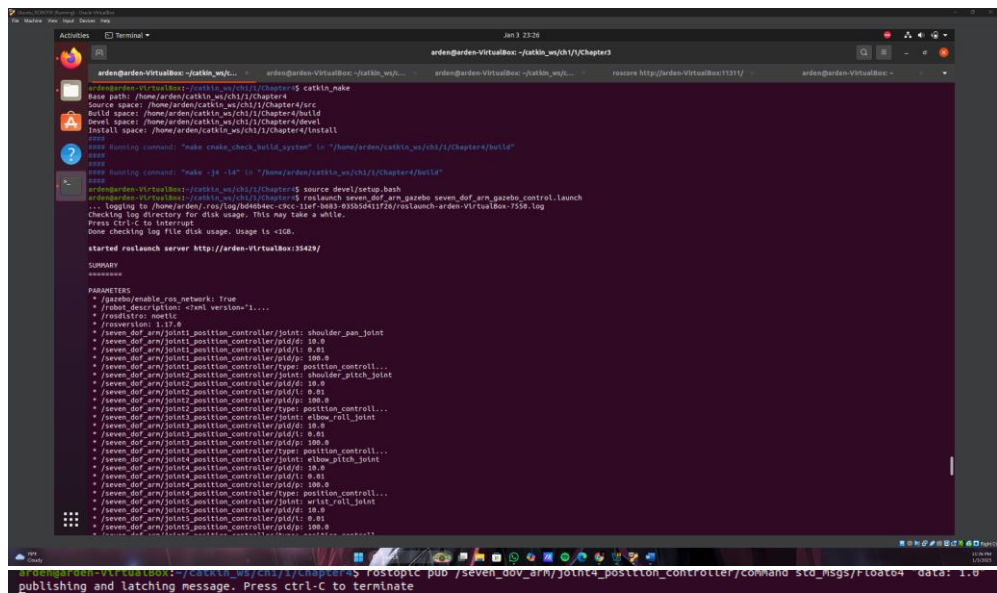




#### 4. Simulation Robot with ROS and Gazebo

Pada Chapter sebelumnya, kita telah mempelajari cara memodelkan robot menggunakan format URDF dan memvisualisasikannya dalam RViz. Pada tahap ini, kita akan melangkah lebih jauh dengan memanfaatkan **Gazebo**, sebuah simulator fisik yang dirancang untuk mensimulasikan robot di lingkungan virtual. Gazebo menyediakan integrasi mendalam dengan ROS, memungkinkan pengembang untuk menguji fungsionalitas robot secara realistis sebelum diimplementasikan pada perangkat keras sesungguhnya.

1. Pada folder Chapter 4 di repositori, kita perlu build dulu file nya seperti di chapter sebelumnya.
2. Bila sudah di build, maka kita bisa langsung launch file yang sudah ada seperti berikut:



```

ardengarden-VirtualBox-7catkin_ws$ catkin_make
Base path: /home/arden/catkin_ws/ch1/Chapter4
Source space: /home/arden/catkin_ws/ch1/Chapter4/src
Build space: /home/arden/catkin_ws/ch1/Chapter4/build
Devel space: /home/arden/catkin_ws/ch1/Chapter4/devel
Install space: /home/arden/catkin_ws/ch1/Chapter4/install

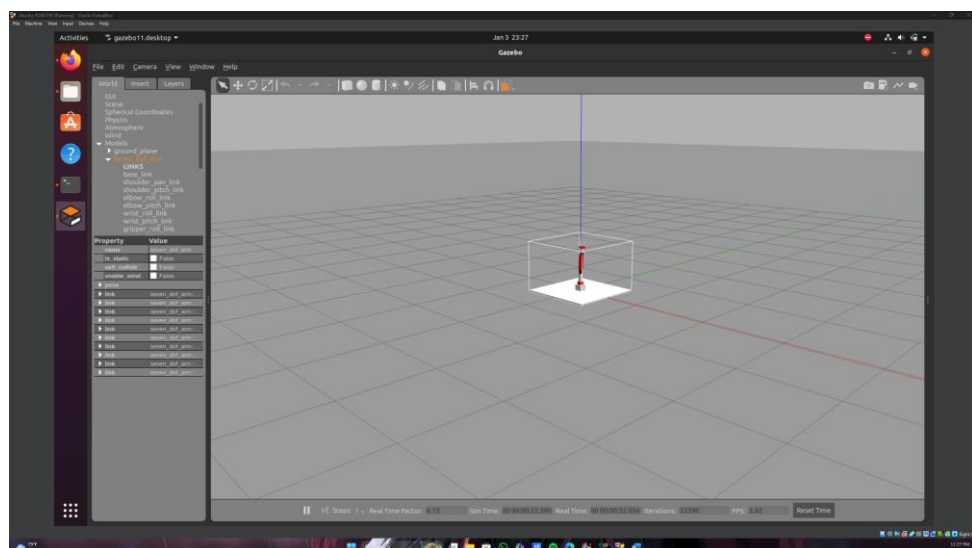
#### Running command: "make make_check_build_system" in "/home/arden/catkin_ws/ch1/Chapter4/build"
####
#### Running command: "make -j4 -l4" in "/home/arden/catkin_ws/ch1/Chapter4/build"
####
ardengarden-VirtualBox-7catkin_ws$ source devel/setup.bash
ardengarden-VirtualBox-7catkin_ws$ rosrun roslaunch seven_dof_arm_gazebo seven_dof_arm_gazebo_control.launch
... Logging to /home/arden/.ros/log/2024-01-03/14:00:00.000000/roslaunch-arden-VirtualBox-75d6.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://arden-VirtualBox:35429/

SUMMARY
=====
PARAMETERS
 * gazebo/enable_ros_network: True
 * robot_description: <xml version="1.0" ...
 * rosdistro: noetic
 * rosversion: 1.17.0
 * seven_dof_arm/joint1_position_controller/joint: shoulder_pan_joint
 * seven_dof_arm/joint1_position_controller/pid/d: 10.0
 * seven_dof_arm/joint1_position_controller/pid/i: 0.01
 * seven_dof_arm/joint1_position_controller/pid/p: 100.0
 * seven_dof_arm/joint1_position_controller/type: position_controll...
 * seven_dof_arm/joint2_position_controller/joint: shoulder_pitch_joint
 * seven_dof_arm/joint2_position_controller/pid/d: 10.0
 * seven_dof_arm/joint2_position_controller/pid/i: 0.01
 * seven_dof_arm/joint2_position_controller/pid/p: 100.0
 * seven_dof_arm/joint2_position_controller/type: position_controll...
 * seven_dof_arm/joint3_position_controller/joint: elbow_roll_joint
 * seven_dof_arm/joint3_position_controller/pid/d: 10.0
 * seven_dof_arm/joint3_position_controller/pid/i: 0.01
 * seven_dof_arm/joint3_position_controller/pid/p: 100.0
 * seven_dof_arm/joint3_position_controller/type: position_controll...
 * seven_dof_arm/joint4_position_controller/joint: elbow_pitch_joint
 * seven_dof_arm/joint4_position_controller/pid/d: 10.0
 * seven_dof_arm/joint4_position_controller/pid/i: 0.01
 * seven_dof_arm/joint4_position_controller/pid/p: 100.0
 * seven_dof_arm/joint4_position_controller/type: position_controll...
 * seven_dof_arm/joint5_position_controller/joint: wrist_roll_joint
 * seven_dof_arm/joint5_position_controller/pid/d: 10.0
 * seven_dof_arm/joint5_position_controller/pid/i: 0.01
 * seven_dof_arm/joint5_position_controller/pid/p: 100.0
 * seven_dof_arm/joint5_position_controller/type: position_controll...

publishing and latching message. Press ctrl-C to terminate
  
```

3. Ketika sudah, maka akan terbuka window Gazebo nya seperti ini:



4. Bila sudah terbuka Gazebo nya, maka kita sudah bisa simulasikan robot yang sudah kita buat sebelumnya sesuai skenario yang kita inginkan.

**Analisis:** Fokus utama adalah pada simulasi robot:

1. **Integrasi dengan Gazebo:** Gazebo digunakan untuk simulasi fisik, memungkinkan robot berinteraksi dengan lingkungan virtual.
2. **Penggunaan Sensor:** Sensor seperti LIDAR dan kamera 3D ditambahkan ke model untuk pengujian fungsi otonom.
3. **Kontrol Robot:** Simulasi melibatkan pengendalian robot melalui ROS Controller.

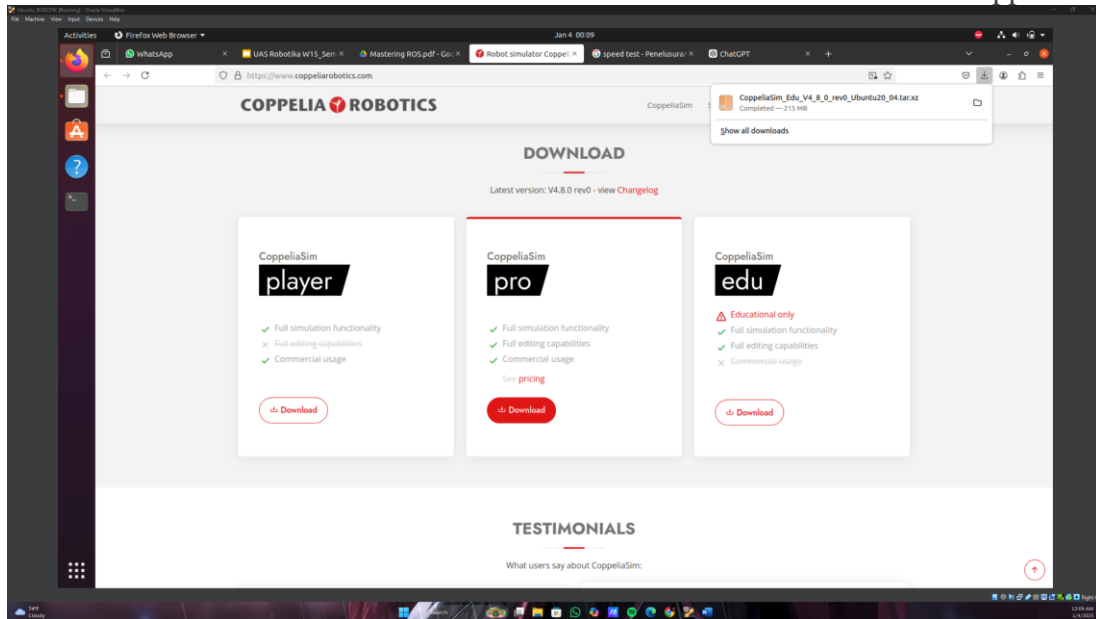
**Kesimpulan:** Gazebo adalah platform yang kuat untuk simulasi robotika, memungkinkan pengujian skenario kompleks tanpa perangkat keras.

## 5. Simulating Robots Using ROS, CoppeliaSim, and Webots

CoppeliaSim dan Webots: Integrasi dengan platform simulasi lain seperti CoppeliaSim dan Webots memberikan fleksibilitas yang lebih besar dalam memilih alat yang sesuai dengan kebutuhan simulasi. CoppeliaSim menawarkan antarmuka yang sangat kuat untuk simulasi robot, sementara Webots adalah alat lain yang sangat populer di dunia robotika. Langkah-langkah :

### 1. Download

CoppeliaSim



- ekstrak coppeliasim: tar vxf CoppeliaSim\_Edu\_V4\_8\_0\_rev0\_Ubuntu20\_04.tar
- Ganti Nama Folder (Opsional): Agar lebih mudah, Anda dapat mengganti nama folder hasil ekstraksi. Misalnya, menjadi CoppeliaSim. Comen:

```
mv CoppeliaSim_Edu_V4_8_0_rev0_Ubuntu20_04 CoppeliaSim
```

- Setel Variabel Lingkungan: Untuk memudahkan akses ke folder CoppeliaSim, atur variabel lingkungan COPPELIASIM\_ROOT dengan menambahkan path folder CoppeliaSim ke file .bashrc Anda. Edit file .bashrc dengan perintah berikut:  
`echo "export COPPELIASIM_ROOT=$HOME/Downloads/CoppeliaSim" >> ~/.bashrc`  
`source ~/.bashrc`  
 Pastikan untuk mengganti ~/Downloads/CoppeliaSim dengan path yang sesuai jika Anda memindahkan folder ke lokasi lain.
- Periksa Kebutuhan Dependensi: CoppeliaSim memerlukan beberapa dependensi seperti Lua dan Qt5. Pastikan Anda telah menginstal dependensi tersebut. Anda dapat menginstalnya menggunakan perintah berikut:  
`sudo apt-get update`  
`sudo apt-get install liblua5.3-dev qt5-qmake qtbase5-dev`
- Jalankan ROS Core: Sebelum menjalankan CoppeliaSim, Anda perlu menjalankan roscore di terminal yang terpisah.
- Jalankan CoppeliaSim: Sekarang Anda dapat menjalankan CoppeliaSim dengan perintah berikut:

```
cd $COPPELIASIM_ROOT
./coppeliaSim.sh
```

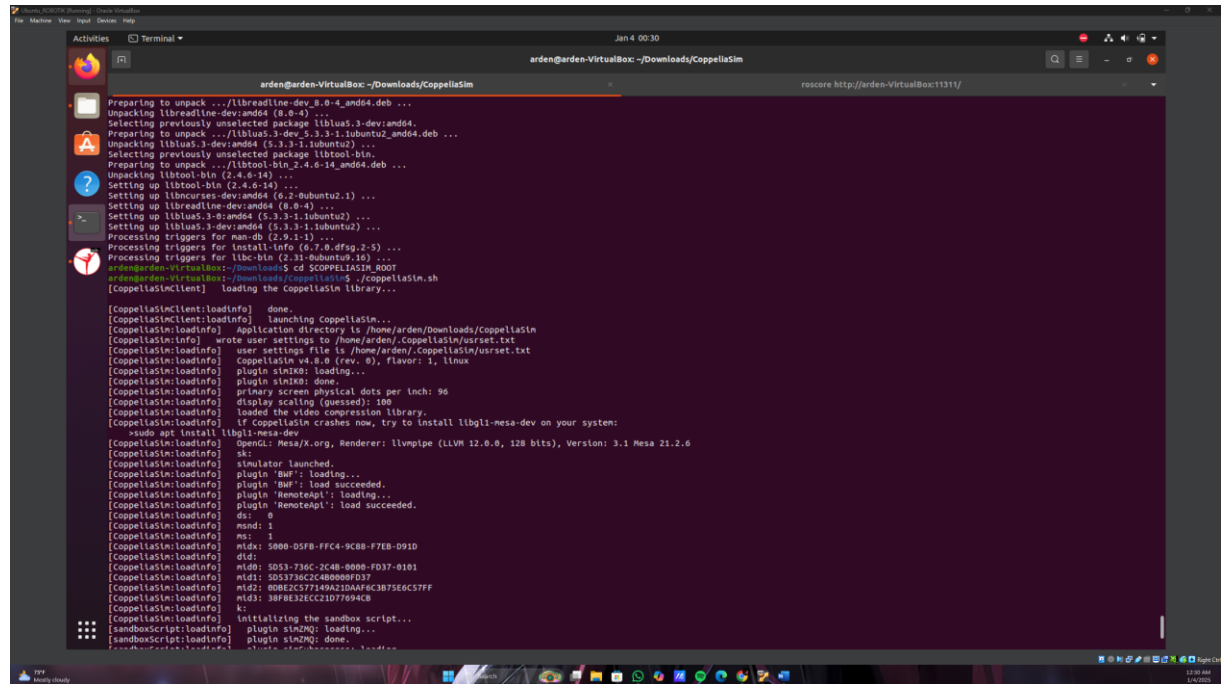
Jika semuanya berjalan dengan baik, CoppeliaSim akan terbuka dan siap digunakan. Anda akan melihat jendela CoppeliaSim dengan scene simulasi.

Verifikasi Integrasi dengan ROS:

Untuk memastikan bahwa CoppeliaSim terhubung dengan ROS, Anda dapat memeriksa daftar node yang aktif setelah menjalankan CoppeliaSim:

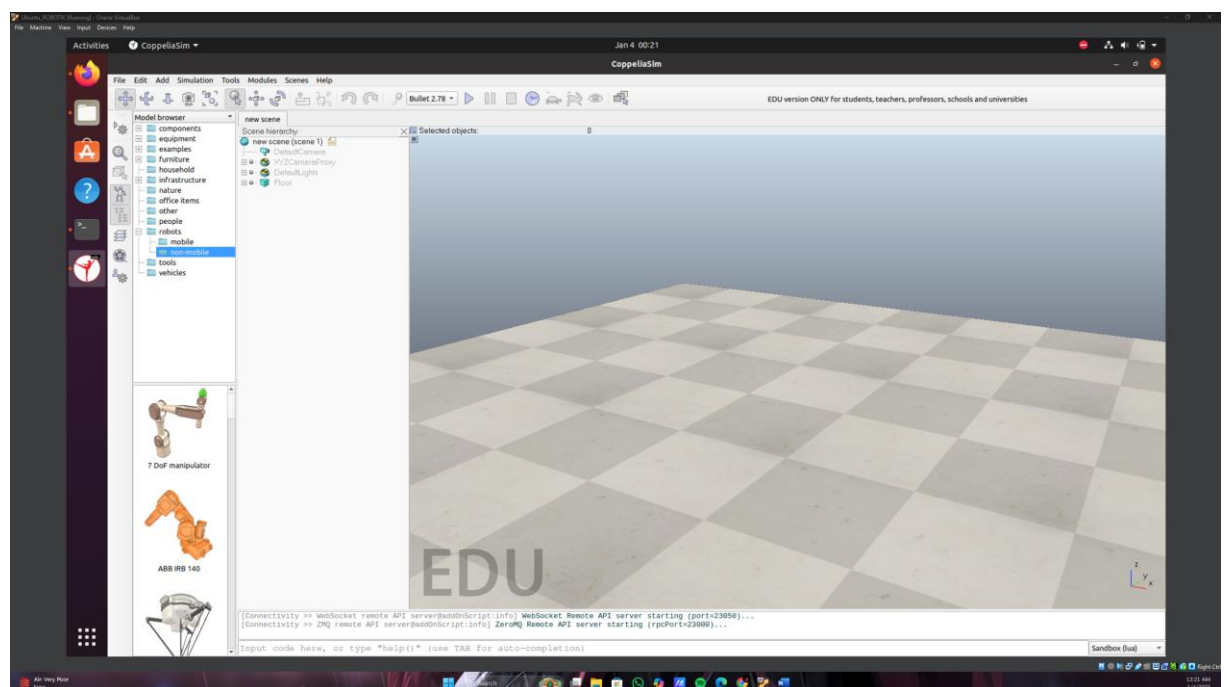
`rostopic list`

Jika `sim_ros_interface` muncul dalam daftar node, itu berarti integrasi dengan ROS berhasil.

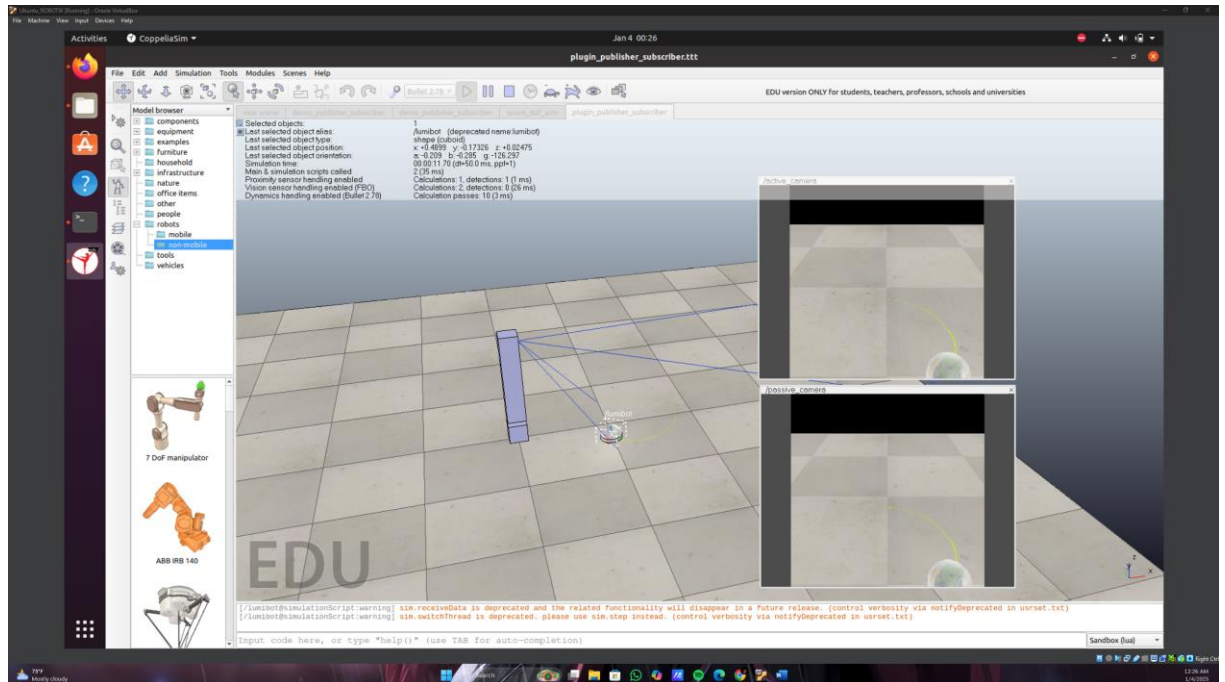


```
ardn@arden-VirtualBox: ~/Downloads/CoppeliaSim
Preparing to unpack .../libreadline-dev_8.0-4_amd64.deb ...
Unpacking libreadline-dev:amd64 (8.0-4) ...
Selecting previously unselected package liblua5.3-dev:amd64.
Preparing to unpack .../liblua5.3-dev_5.3.3-1ubuntu2_amd64.deb ...
Unpacking liblua5.3-dev:amd64 (5.3.3-1ubuntu2) ...
Selecting previously unselected package libtool-bin.
Preparing to unpack .../libtool-bin_2.4.6-14_amd64.deb ...
Unpacking libtool-bin (2.4.6-14) ...
Setting up libtool-bin (2.4.6-14) ...
Setting up libncurses-dev:amd64 (6.2-0ubuntu2.1) ...
Setting up liblua5.3-dev:amd64 (5.3.3-1ubuntu2) ...
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
Processing triggers for install-info (6.7-0ubuntu2) ...
Processing triggers for libe-bin (2.31-0ubuntu9.16) ...
ardn@arden-VirtualBox: ~/Downloads$ cd $COPPELIASIM_ROOT
ardn@arden-VirtualBox: ~/Downloads/CoppeliaSim$ ./CoppeliaSim.sh
[CoppeliaSimClient] loading the CoppeliaSim library...
[CoppeliaSimClient:loadInfo] done.
[CoppeliaSimClient:loadInfo] launching CoppeliaSim...
[CoppeliaSim:loadInfo] Application directory is /home/arden/Downloads/CoppeliaSim
[CoppeliaSim:info] wrote user settings to /home/arden/.CoppeliaSim/user.txt
[CoppeliaSim:loadInfo] user settings file is /home/arden/.CoppeliaSim/user.txt
[CoppeliaSim:loadInfo] CoppeliaSim v4.8.0 (rev. 0), Flavor: 1, Linux
[CoppeliaSim:loadInfo] plugin simIO: loading...
[CoppeliaSim:loadInfo] plugin simIO: done.
[CoppeliaSim:loadInfo] primary screen physical dots per inch: 96
[CoppeliaSim:loadInfo] display scaling (guessed): 100
[CoppeliaSim:loadInfo] loaded the video compression library.
[CoppeliaSim:loadInfo] if CoppeliaSim crashes now, try to install libgl1-mesa-dev on your system:
$ sudo apt install libgl1-mesa-dev
[CoppeliaSim:loadInfo] OpenGL: Mesa/X.org, Renderer: llvmpipe (LLVM 12.0.0, 128 bits), Version: 3.1 Mesa 21.2.0
sk:
sk:
[CoppeliaSim:loadInfo] simulator launched.
[CoppeliaSim:loadInfo] plugin 'simIO': loading...
[CoppeliaSim:loadInfo] plugin 'simIO': load succeeded.
[CoppeliaSim:loadInfo] plugin 'RemoteAPI': loading...
[CoppeliaSim:loadInfo] plugin 'RemoteAPI': load succeeded.
[CoppeliaSim:loadInfo] ds: 0
[CoppeliaSim:loadInfo] msd: 1
[CoppeliaSim:loadInfo] ms: 1
[CoppeliaSim:loadInfo] midx: 5000-05F8-FFC4-9C8B-F7E8-091D
[CoppeliaSim:loadInfo] dtd:
[CoppeliaSim:loadInfo] midb: 5053-736C-2C48-0000-FD37-0101
[CoppeliaSim:loadInfo] midt: 5053736C2C480000FD37
[CoppeliaSim:loadInfo] mid2: 8082577149A21D0AFC3075E6C57FF
[CoppeliaSim:loadInfo] mid3: 38F8E32EC21D77694CB
[CoppeliaSim:loadInfo] k:
[CoppeliaSim:loadInfo] Initializing the sandbox script...
[CoppeliaSim:loadInfo] plugin sim2MQ: loading...
[CoppeliaSim:loadInfo] plugin sim2MQ: done.
```

Tampilan coppeliasim



Selanjutnya pilih file di pojok kiri atas dan pilih opsi open scene dan pilih file yg akan di jalankan jika sudah contohnya akan seperti gambar di bawah



#### Perbandingan CoppeliaSim dan Webots:

Aspek	CoppeliaSim	Webots
Fokus	Simulasi manipulasi dan kolaborasi robot.	Simulasi robot bergerak dan multi-robot.
Plugin ROS	RosInterface plugin.	Dukungan langsung melalui webots_ros.
Kemudahan Integrasi	Memerlukan konfigurasi tambahan.	Mudah dengan dukungan bawaan.
Bahasa Pemrograman	Python, Lua, C++.	Python, C++, Java.

Dengan CoppeliaSim dan Webots, pengembang memiliki fleksibilitas untuk memilih simulator sesuai kebutuhan proyek mereka.

- **CoppeliaSim** unggul dalam simulasi manipulasi robotik dan kontrol tingkat rendah.
- **Webots** lebih cocok untuk robot bergerak dan simulasi multi-robot.

#### KESIMPULAN:

ROS (Robot Operating System) adalah kerangka kerja yang fleksibel dan kuat untuk pengembangan robotika, menawarkan berbagai alat dan pustaka yang mendukung seluruh siklus hidup proyek robotik, mulai dari desain hingga implementasi. Pada tahap awal, ROS menyediakan dasar yang kuat melalui konsep modularitas berbasis node, komunikasi antar-node menggunakan topik dan layanan, serta pengelolaan parameter server. Kemampuan ini memungkinkan sistem tetap berfungsi meskipun terdapat kegagalan pada salah satu komponennya. Pemodelan 3D menggunakan URDF dan RViz memberikan pengembang kemampuan untuk merancang dan memvalidasi spesifikasi robot sebelum simulasi lebih lanjut, sementara simulasi di Gazebo memungkinkan pengujian fungsi robot secara realistis di lingkungan virtual yang kompleks tanpa memerlukan perangkat keras. Selain itu, integrasi dengan simulator lain seperti CoppeliaSim dan Webots memperluas fleksibilitas pengembangan, memungkinkan simulasi manipulasi robotik hingga skenario multi-robot. Dengan dukungan komunitas yang luas, ekosistem alat debugging yang lengkap, dan kompatibilitas lintas platform, ROS menjadi solusi utama dalam penelitian dan pengembangan robotika modern, memberikan efisiensi, keandalan, dan fleksibilitas yang luar biasa untuk berbagai aplikasi robotik.