

TECHNICAL REPORT MASTERING ROS THIRD EDITION

Naufal Ramadhan

Pendahuluan

Puji syukur tuhan yang maha esa yang masih memberikan kesehatan dan kebugaran terhadap saya sebagai penulis, yang masih mampu menulis technical report ini sampai selesai, pada technical report ini saya akan membahas seputar buku “Mastering Ros Third Edition”, yang mana akan saya praktikan dan saya dokumentasikan pada technical report ini, untuk memenuhi nilai UAS Robotika Universitas Telkom, Bojongsoang, Bandung

Why We Use ROS?

Kita harus menggunakan ROS (Robot Operating System) karena menawarkan sejumlah keunggulan yang membuatnya menjadi pilihan utama bagi para robotikawan dan pengembang yang bekerja pada aplikasi robotik. Salah satu keunggulan utama ROS adalah pendekatannya yang modular. Dengan ROS, pengembang dapat membuat program-program kecil yang independen, yang disebut sebagai nodes, untuk melakukan tugas-tugas spesifik. Pendekatan ini mempromosikan keberlanjutan kode dan modularitas, memudahkan penggunaan kembali kode dalam berbagai aplikasi robotik. Selain itu, keberadaan komunitas yang besar dan aktif dalam ROS memastikan adanya dukungan, kontribusi, dan kolaborasi yang kaya. ROS juga menawarkan ekosistem yang kaya dengan berbagai alat, pustaka, dan paket yang mempermudah pengembangan robotik dari navigasi hingga simulasi.

Kemampuan ROS untuk bekerja dengan berbagai platform perangkat keras membuatnya fleksibel dan tidak terikat pada satu jenis perangkat tertentu. Selain itu, infrastruktur komunikasi ROS yang kuat memungkinkan komunikasi yang lancar antara nodes, memfasilitasi pengembangan sistem yang terdistribusi. Dengan fitur-fitur seperti simulasi Gazebo dan Rviz, ROS juga memungkinkan pengembang untuk menguji dan memvalidasi algoritma dalam lingkungan virtual sebelum diterapkan pada robot nyata, mempercepat siklus pengembangan dan mengurangi risiko. Keseluruhan ini menunjukkan bahwa ROS adalah platform yang komprehensif dan skalabel untuk pengembangan aplikasi robotik modern.

2

Getting Started With Ros Programming

Setelah membahas seputar materi umum pada BAB I kita masuk pada bagian yang menyenangkan pada ROS yaitu

- Membuat ROS package
- Adding custom message dan service file
- Membuat ROS services
- Membuat launch file
- Membuat aplikasi dari topics, services dan actionlib

Creating a ROS package

Persyaratan pertama untuk bekerja dengan paket ROS adalah membuat catkin ROS Workspace. Setelah menginstal ROS, kita dapat membuat dan membangun catkin workspace yang disebut catkin_ws:

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

untuk melakukan compile, kita harus melakukan source pada ROS environment untuk mendapatkan akses ke fungsi ROS:

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

pindah ke direktori src folder yang sudah kita buat, lalu lakukan inisialisasi new catkin workspace, dalam kasus ini karna saya sudah melakukan inisialisasi jadi tidak terdapat proses apa apa.

Setelah melakukan beberapa proses diatas, sekarang kita akan membuat catkin package, dengan cara;

```
catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs  
actionlib actionlib_msgs
```

gambar seperti dibawah akan muncul ketika anda berhasil melakukan create pkg

```
Created file mastering_ros_v2_pkg/package.xml  
Created file mastering_ros_v2_pkg/CMakeLists.txt  
Created folder mastering_ros_v2_pkg/include/mastering_ros_v2_pkg  
Created folder mastering_ros_v2_pkg/src  
Successfully created files in /home/jcacace/mastering_ros_v2_pkg. Please  
adjust the values in package.xml.
```

Setelah berhasil melakukan semua step yang ada diatas, kita akan melakukan catkin_make, catkin_make disini bertujuan untuk melakukan build program, karena tanpa melakukan catkin make, program tidak akan mengenali nodes yang baru kita buat

```
root@6caed748117d:~# cd ~/catkin_ws  
root@6caed748117d:~/catkin_ws# catkin_make  
Base path: /root/catkin_ws  
Source space: /root/catkin_ws/src  
Build space: /root/catkin_ws/build  
Devel space: /root/catkin_ws/devel  
Install space: /root/catkin_ws/install  
####  
#### Running command: "make cmake_check_build_system" in "/root/catkin_ws/build"  
####  
####  
#### Running command: "make -j12 -l12" in "/root/catkin_ws/build"  
####  
[ 75%] Built target demo_topic_subscriber  
[100%] Built target demo_topic_publisher  
root@6caed748117d:~/catkin_ws# |
```

Creating ROS nodes

Pertama tama sebagai awlan, kita akan membuat demo_topic_publisher.cpp. Node ini akan menerbitkan nilai dan mengirimkannya ke server, berikut merupakan code nya

```
root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg/src# vim demo_topic_publisher.cpp|
```

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_topic_publisher");
    ros::NodeHandle node_obj;
    ros::Publisher number_publisher = node_obj.advertise<std_msgs::Int32>("/numbers", 10);
    ros::Rate loop_rate(10);
    int number_count = 0;
    while ( ros::ok() ) {
        std_msgs::Int32 msg;
        msg.data = number_count;
        ROS_INFO("%d",msg.data);
        number_publisher.publish(msg);
        loop_rate.sleep();
        ++number_count;
    }
    return 0;
}
```

Kita juga akan membuat demo_topic_client.cpp untuk menangkap nilai yang dikirimkan oleh publisher, berikut merupakan syntax dari demo_topic_client.cpp

```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

void number_callback(const std_msgs::Int32::ConstPtr& msg) {
    ROS_INFO("Received [%d]",msg->data);
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_topic_subscriber");
    ros::NodeHandle node_obj;
    ros::Subscriber number_subscriber = node_obj.subscribe("/numbers",10,number_callback);
    ros::spin();
    return 0;
}
~
~
~
~
~
~
~
```

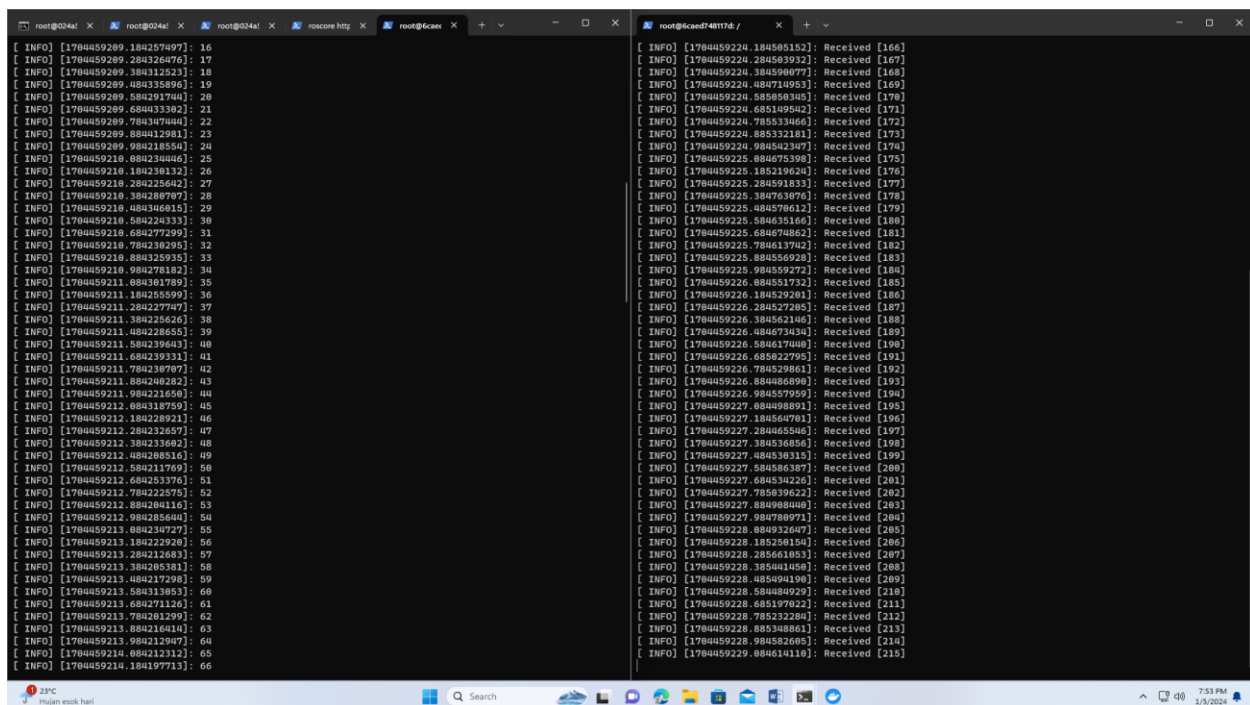
Sebelum melanjutkan, kita harus mengganti CMakeList.txt, hal ini dilakukan agar program dapat mengenali nodes yang baru kita tambahkan dengan cara add_executable

```
    ${Boost_INCLUDE_DIRS}
)
#This will create executables of the nodes
add_executable(demo_topic_publisher src/demo_topic_publisher.cpp)
add_executable(demo_topic_subscriber src/demo_topic_subscriber.cpp)
#This will link executables to the appropriate libraries
add_executable(demo_msg_publisher src/demo_msg_publisher.cpp)
add_executable(demo_msg_subscriber src/demo_msg_subscriber.cpp)

add_dependencies(demo_msg_publisher mastering_ros_demo_pkg_generate_messages_cpp)
add_dependencies(demo_msg_subscriber mastering_ros_demo_pkg_generate_messages_cpp)
target_link_libraries(demo_msg_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_msg_subscriber ${catkin_LIBRARIES})
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_topic_subscriber ${catkin_LIBRARIES})

add_executable(demo_service_server src/demo_service_server.cpp)
add_executable(demo_service_client src/demo_service_client.cpp)
add_dependencies(demo_service_server mastering_ros_demo_pkg_generate_messages_cpp)
add_dependencies(demo_service_client mastering_ros_demo_pkg_generate_messages_cpp)
```

Berikut merupakan output dari kedua nodes diatas



```
[ INFO] [1704459209.184257497]: 16
[ INFO] [1704459209.284326476]: 17
[ INFO] [1704459209.384112523]: 18
[ INFO] [1704459209.484335896]: 19
[ INFO] [1704459209.584291744]: 20
[ INFO] [1704459209.684433382]: 21
[ INFO] [1704459209.784347444]: 22
[ INFO] [1704459209.88432281]: 23
[ INFO] [1704459209.984218554]: 24
[ INFO] [1704459210.084234446]: 25
[ INFO] [1704459210.184238132]: 26
[ INFO] [1704459210.284225442]: 27
[ INFO] [1704459210.384288797]: 28
[ INFO] [1704459210.484346815]: 29
[ INFO] [1704459210.584224333]: 30
[ INFO] [1704459210.684277299]: 31
[ INFO] [1704459210.784239295]: 32
[ INFO] [1704459210.884325935]: 33
[ INFO] [1704459210.984278182]: 34
[ INFO] [1704459211.084381789]: 35
[ INFO] [1704459211.184255599]: 36
[ INFO] [1704459211.284227747]: 37
[ INFO] [1704459211.384225626]: 38
[ INFO] [1704459211.484228655]: 39
[ INFO] [1704459211.584239443]: 40
[ INFO] [1704459211.684239331]: 41
[ INFO] [1704459211.784238797]: 42
[ INFO] [1704459211.884248282]: 43
[ INFO] [1704459211.984221680]: 44
[ INFO] [1704459212.084219759]: 45
[ INFO] [1704459212.184228921]: 46
[ INFO] [1704459212.284232657]: 47
[ INFO] [1704459212.384233682]: 48
[ INFO] [1704459212.484288516]: 49
[ INFO] [1704459212.584211769]: 50
[ INFO] [1704459212.68425376]: 51
[ INFO] [1704459212.784222575]: 52
[ INFO] [1704459212.884284116]: 53
[ INFO] [1704459212.984285444]: 54
[ INFO] [1704459213.084234727]: 55
[ INFO] [1704459213.184222926]: 56
[ INFO] [1704459213.284212683]: 57
[ INFO] [1704459213.384285381]: 58
[ INFO] [1704459213.484217298]: 59
[ INFO] [1704459213.584313853]: 60
[ INFO] [1704459213.684271126]: 61
[ INFO] [1704459213.784281299]: 62
[ INFO] [1704459213.884216414]: 63
[ INFO] [1704459213.984212947]: 64
[ INFO] [1704459214.084212312]: 65
[ INFO] [1704459214.184197713]: 66
```

```
[ INFO] [1704459224.184585152]: Received [166]
[ INFO] [1704459224.284583932]: Received [167]
[ INFO] [1704459224.384589477]: Received [168]
[ INFO] [1704459224.484711853]: Received [169]
[ INFO] [1704459224.58458345]: Received [170]
[ INFO] [1704459224.685149542]: Received [171]
[ INFO] [1704459224.785533466]: Received [172]
[ INFO] [1704459224.885322121]: Received [173]
[ INFO] [1704459224.984542347]: Received [174]
[ INFO] [1704459225.084675398]: Received [175]
[ INFO] [1704459225.185219624]: Received [176]
[ INFO] [1704459225.284591833]: Received [177]
[ INFO] [1704459225.384763876]: Received [178]
[ INFO] [1704459225.484578612]: Received [179]
[ INFO] [1704459225.584635166]: Received [180]
[ INFO] [1704459225.684674862]: Received [181]
[ INFO] [1704459225.784613702]: Received [182]
[ INFO] [1704459225.884556928]: Received [183]
[ INFO] [1704459225.984559272]: Received [184]
[ INFO] [1704459226.084551732]: Received [185]
[ INFO] [1704459226.184529281]: Received [186]
[ INFO] [1704459226.284527285]: Received [187]
[ INFO] [1704459226.384562146]: Received [188]
[ INFO] [1704459226.484673434]: Received [189]
[ INFO] [1704459226.584617408]: Received [190]
[ INFO] [1704459226.684562295]: Received [191]
[ INFO] [1704459226.784529861]: Received [192]
[ INFO] [1704459226.884486890]: Received [193]
[ INFO] [1704459226.984557959]: Received [194]
[ INFO] [1704459227.084498891]: Received [195]
[ INFO] [1704459227.184564701]: Received [196]
[ INFO] [1704459227.284465546]: Received [197]
[ INFO] [1704459227.384536856]: Received [198]
[ INFO] [1704459227.484538315]: Received [199]
[ INFO] [1704459227.584586387]: Received [200]
[ INFO] [1704459227.684534226]: Received [201]
[ INFO] [1704459227.785639622]: Received [202]
[ INFO] [1704459227.884988448]: Received [203]
[ INFO] [1704459227.984788971]: Received [204]
[ INFO] [1704459228.084932647]: Received [205]
[ INFO] [1704459228.185258154]: Received [206]
[ INFO] [1704459228.28561853]: Received [207]
[ INFO] [1704459228.385641450]: Received [208]
[ INFO] [1704459228.485049198]: Received [209]
[ INFO] [1704459228.584484929]: Received [210]
[ INFO] [1704459228.685197822]: Received [211]
[ INFO] [1704459228.78523284]: Received [212]
[ INFO] [1704459228.885348661]: Received [213]
[ INFO] [1704459228.984582685]: Received [214]
[ INFO] [1704459229.084614118]: Received [215]
```

Yang terjadi diatas adalah, publisher mengirimkan data ke topic, dan subscriber menangkapnya dengan cara melakukan subscribe ke topic tersebut

Selanjutnya kita lakukan editing pada package.xml, dan melakukan uncomment pada message_generation, dan message runtime

```
<!-- <exec_depend>roscpp</exec_depend> -->
<!-- Use build_depend for packages you need at compile time: -->
  <build_depend>message_generation</build_depend>
<!-- Use build_export_depend for packages you need in order to build against this package: -->
  <build_export_depend>message_generation</build_export_depend> -->
<!-- Use buildtool_depend for build tool packages: -->
  <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
  <exec_depend>message_runtime</exec_depend>
<!-- Use test_depend for packages you need only for testing: -->
  <test_depend>gtest</test_depend> -->
<!-- Use doc_depend for packages you need only for building documentation: -->
  <doc_depend>doxygen</doc_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_export_depend>roscpp</build_export_depend>
```

Karena kita ingin mengirimkan message, kita harus melakukan deklarasi juga pada file CMakeList.txt seperti gambar dibawah ini

```
## Generate messages in the 'msg' folder
add_message_files(
  FILES
  demo_msg.msg
)
## Generate added messages and services with any dependencies
listed here
generate_messages(
)

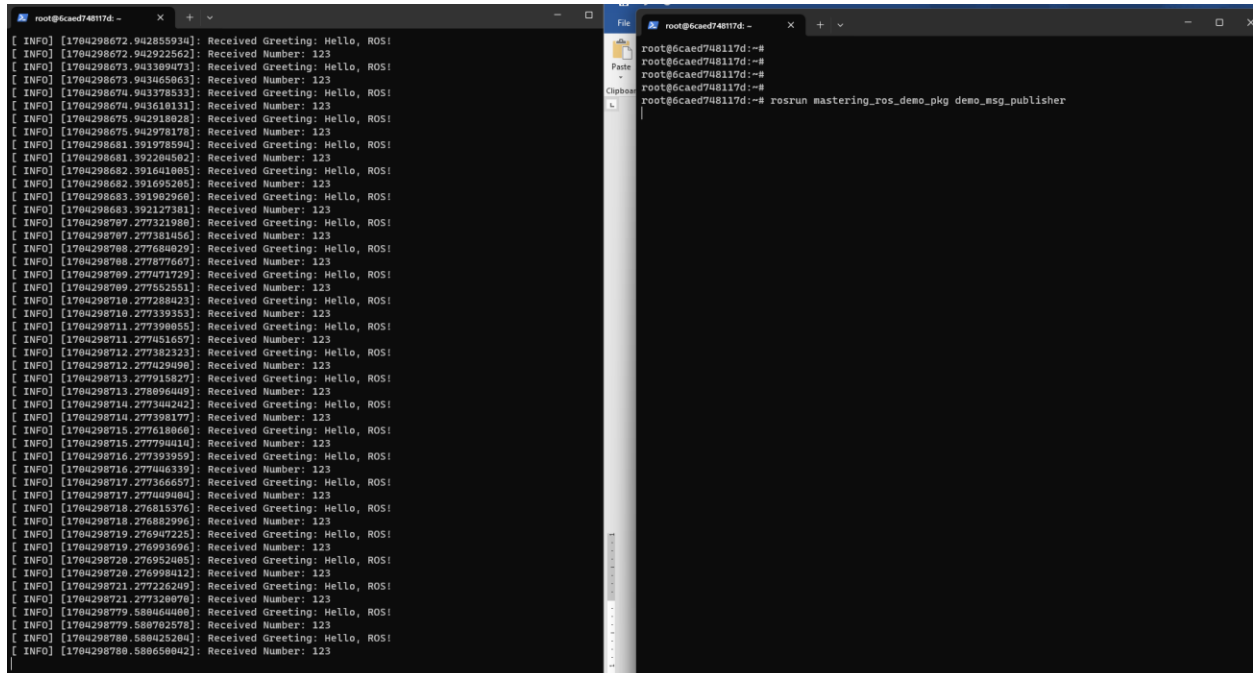
## Generate services in the 'srv' folder
```

Kemudian pastikan bahwa message file berjalan dengan baik dengan cara mengetikan rosmmsg

```
root@6caed748117d:~/catkin_ws# rosmmsg show mastering_ros_demo_pkg/demo_msg
string greeting
int32 number

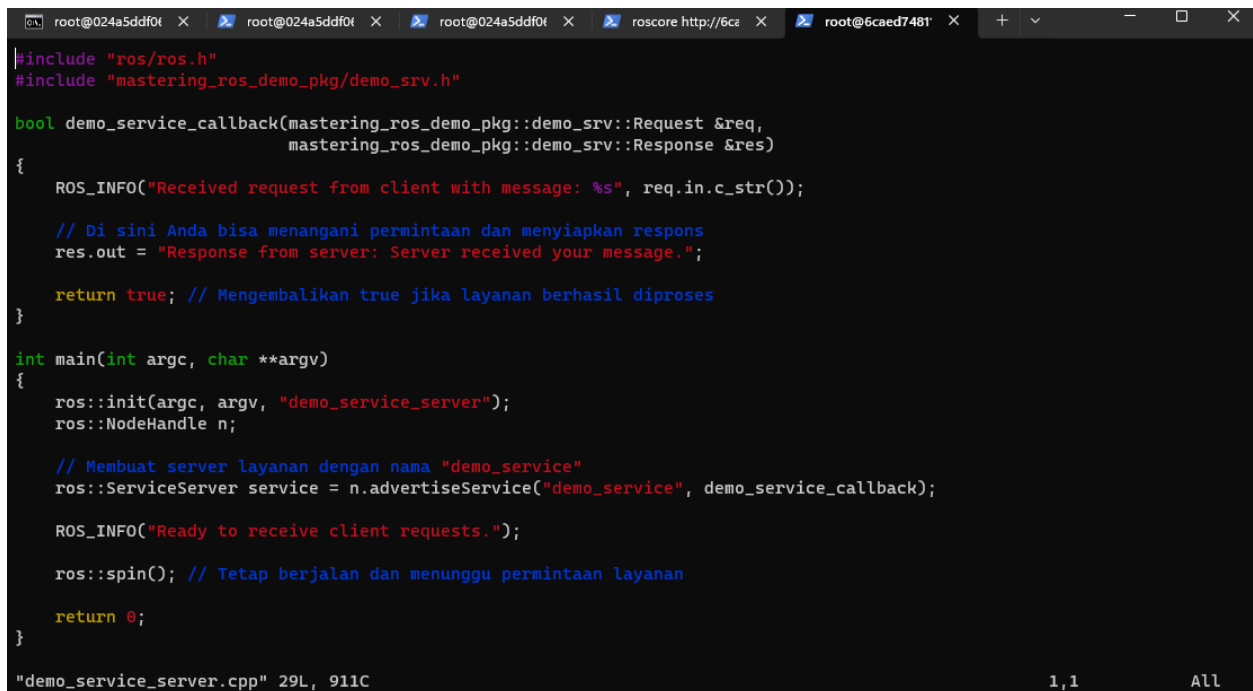
root@6caed748117d:~/catkin_ws# |
```

Berikut merupakan output dari build program



```
root@6caed748117d: ~#
[ INFO] [1704298672.942855930]: Received Greeting: Hello, ROS!
[ INFO] [1704298672.942922562]: Received Number: 123
[ INFO] [1704298673.943389473]: Received Greeting: Hello, ROS!
[ INFO] [1704298673.943465863]: Received Number: 123
[ INFO] [1704298674.943378533]: Received Greeting: Hello, ROS!
[ INFO] [1704298674.943618131]: Received Number: 123
[ INFO] [1704298675.942918028]: Received Greeting: Hello, ROS!
[ INFO] [1704298675.942978178]: Received Number: 123
[ INFO] [1704298681.391978594]: Received Greeting: Hello, ROS!
[ INFO] [1704298681.392204502]: Received Number: 123
[ INFO] [1704298682.391601065]: Received Greeting: Hello, ROS!
[ INFO] [1704298682.391695286]: Received Number: 123
[ INFO] [1704298683.391902960]: Received Greeting: Hello, ROS!
[ INFO] [1704298683.392127381]: Received Number: 123
[ INFO] [1704298707.277321980]: Received Greeting: Hello, ROS!
[ INFO] [1704298707.277381050]: Received Number: 123
[ INFO] [1704298708.277684029]: Received Greeting: Hello, ROS!
[ INFO] [1704298708.277877667]: Received Number: 123
[ INFO] [1704298709.277471729]: Received Greeting: Hello, ROS!
[ INFO] [1704298709.277552551]: Received Number: 123
[ INFO] [1704298710.277288423]: Received Greeting: Hello, ROS!
[ INFO] [1704298710.277339353]: Received Number: 123
[ INFO] [1704298711.277398055]: Received Greeting: Hello, ROS!
[ INFO] [1704298711.277451657]: Received Number: 123
[ INFO] [1704298712.277382323]: Received Greeting: Hello, ROS!
[ INFO] [1704298712.277429490]: Received Number: 123
[ INFO] [1704298713.277915929]: Received Greeting: Hello, ROS!
[ INFO] [1704298713.278096449]: Received Number: 123
[ INFO] [1704298714.277344242]: Received Greeting: Hello, ROS!
[ INFO] [1704298714.277398177]: Received Number: 123
[ INFO] [1704298715.277618860]: Received Greeting: Hello, ROS!
[ INFO] [1704298715.277790010]: Received Number: 123
[ INFO] [1704298716.277939395]: Received Greeting: Hello, ROS!
[ INFO] [1704298716.277406339]: Received Number: 123
[ INFO] [1704298717.277366657]: Received Greeting: Hello, ROS!
[ INFO] [1704298717.277449404]: Received Number: 123
[ INFO] [1704298718.276815370]: Received Greeting: Hello, ROS!
[ INFO] [1704298718.276883996]: Received Number: 123
[ INFO] [1704298719.276947225]: Received Greeting: Hello, ROS!
[ INFO] [1704298719.276993696]: Received Number: 123
[ INFO] [1704298720.276952405]: Received Greeting: Hello, ROS!
[ INFO] [1704298720.276998412]: Received Number: 123
[ INFO] [1704298721.277225209]: Received Greeting: Hello, ROS!
[ INFO] [1704298721.277328070]: Received Number: 123
[ INFO] [1704298779.580464000]: Received Greeting: Hello, ROS!
[ INFO] [1704298779.580702570]: Received Number: 123
[ INFO] [1704298780.580425204]: Received Greeting: Hello, ROS!
[ INFO] [1704298780.580650042]: Received Number: 123
root@6caed748117d: ~#
root@6caed748117d: ~#
root@6caed748117d: ~#
root@6caed748117d: ~#
root@6caed748117d: ~# roslaunch mastering_ros_demo_pkg demo_msg_publisher
```

Kemudian kita akan membuat file .src, yang bernama demo_srv.src. nodes ini digunakan bertujuan untuk server, berikut merupakan source code nya



```
root@024a5ddf0f: ~#
#include "ros/ros.h"
#include "mastering_ros_demo_pkg/demo_srv.h"

bool demo_service_callback(mastering_ros_demo_pkg::demo_srv::Request &req,
                           mastering_ros_demo_pkg::demo_srv::Response &res)
{
    ROS_INFO("Received request from client with message: %s", req.in_c_str());

    // Di sini Anda bisa menangani permintaan dan menyiapkan respons
    res.out = "Response from server: Server received your message.";

    return true; // Mengembalikan true jika layanan berhasil diproses
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "demo_service_server");
    ros::NodeHandle n;

    // Membuat server layanan dengan nama "demo_service"
    ros::ServiceServer service = n.advertiseService("demo_service", demo_service_callback);

    ROS_INFO("Ready to receive client requests.");

    ros::spin(); // Tetap berjalan dan menunggu permintaan layanan

    return 0;
}

"demo_service_server.cpp" 29L, 911C 1,1 All
```

[illegible]

Selanjutnya kita akan membuat ROS actionlib, yang terjadi pada actionlib ini adalah klien aksi akan mengirim angka sebagai tujuan. Ketika server tindakan menerima tujuan ini, itu akan dihitung dari 0 hingga nomor gol dengan ukuran langkah 1 dan dengan penundaan 1 detik. Jika selesai sebelum waktu tenggat, itu akan mengirimkan hasilnya

```
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# cd action/
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action# ls
Demo_action.action
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action# |
```

Berikut isi dari file Demo_action.action


```
root@6caed748117d: ~/catkin X + v

#goal definition
int32 count
---
#result definition
int32 final_count
---
#feedback
int32 current_number
~
~
```

Setelah melakukan konfigurasi yang tersedia pada buku berikut merupakan hasil running program dari Demo_action.action

```
root@6caed748117d: / X + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\byfish>docker exec -it 6caed748117d /bin/bash
root@6caed748117d:/# rosrun mastering_ros_demo_pkg demo_action_client 10
[ INFO] [1704363080.556879753]: Menunggu server...
[ INFO] [1704363080.796819005]: Server ditemukan, mengirim goal...
[ INFO] [1704363091.799573995]: Selesai: SUCCEEDED
root@6caed748117d:/#

root@6caed748117d: / X + v
Microsoft Windows [Version 10.0.22621.2861]
(c) Microsoft Corporation. All rights reserved.

C:\Users\byfish>docker exec -it 6caed748117d /bin/bash
root@6caed748117d:/# rosrun mastering_ros_demo_pkg demo_action_server
[ INFO] [1704363080.798063325]: demo_action: Menerima goal 10
[ INFO] [1704363091.798209621]: demo_action: Selesai, mengirimkan hasil
```

Creating launch files

Dalam konteks pengembangan sistem robotik dengan ROS, penggunaan file peluncuran memegang peran krusial dalam efisiensi dan manajemen sumber daya. Mengingat kompleksitas yang mungkin terlibat dalam implementasi robot yang melibatkan puluhan bahkan ratusan node, pendekatan manual dengan meluncurkan setiap node secara berurutan melalui terminal menjadi tidak praktis dan rentan terhadap kesalahan. Sebagai solusi yang lebih terstruktur dan efisien, ROS menyediakan mekanisme yang disebut "roslaunch," yang memungkinkan para pengembang untuk mendefinisikan dan mengelola kumpulan node yang harus diluncurkan melalui file berbasis XML yang disebut file peluncuran. Dengan pendekatan ini, pengembang dapat secara sistematis mendefinisikan hubungan antara node, parameter, dan dependensi lainnya dalam satu file, memastikan bahwa seluruh sistem dapat diluncurkan dengan koherensi dan keakuratan yang tinggi. Ini tidak hanya meningkatkan efisiensi dalam pengembangan dan pengujian, tetapi juga memfasilitasi manajemen konfigurasi yang konsisten dan penggunaan sumber daya yang optimal dalam skenario yang kompleks.

3

Working with ROS for 3D Modeling

ROS menyediakan beberapa paket untuk merancang dan membuat model robot, seperti urdf, kdl_parser, robot_state_publisher, dan collada_urdf. Paket-paket ini akan membantukami membuat deskripsi model robot 3D dengan karakteristik yang tepat dari yang sebenarnya perangkat keras.

Untuk mengikuti contoh dalam bab ini, Anda memerlukan laptop standar yang berjalan Ubuntu 20.04 dengan ROS Noetic diinstal. Kode referensi untuk bab ini dapat berupa: diunduh dari repositori Git di <https://github.com/PacktPublishing/Mastering-ROS-untuk-Robotika-Pemrograman-Ketiga-edition.git>. Itu kode terdapat di dalam Bab3/mastering_ros_robot_description_pkg/ folder. Anda dapat melihat kode bab ini beraksi di sini: <https://bit.ly/2W5jief>.

Creating the ROS package for the robot

Pertama tama kita harus melakukan gitclone pada girhub reps milik si Authors, disini authors menyediakan github reps, untuk mempermudah kita untuk mengikuti tutorial yang diberikan oleh si Authors/

```
root@024a5ddf0613:~/catkin_ws# cd src/
root@024a5ddf0613:~/catkin_ws/src# ls
CMakeLists.txt  mastering_ros_robot_description_pkg
root@024a5ddf0613:~/catkin_ws/src# cd mastering_ros_robot_description_pkg/
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg# ls -la
total 40
drwxr-xr-x 5 root root 4096 Jan  4 16:02 .
drwxr-xr-x 3 root root 4096 Jan  4 16:02 ..
-rw-r--r-- 1 root root 7239 Jan  4 16:02 CMakeLists.txt
drwxr-xr-x 2 root root 4096 Jan  4 16:02 launch
drwxr-xr-x 3 root root 4096 Jan  4 16:02 meshes
-rw-r--r-- 1 root root 3319 Jan  4 16:02 package.xml
drwxr-xr-x 3 root root 4096 Jan  4 16:14 urdf
-rw-r--r-- 1 root root 5189 Jan  4 16:02 urdf.rviz
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg#
```

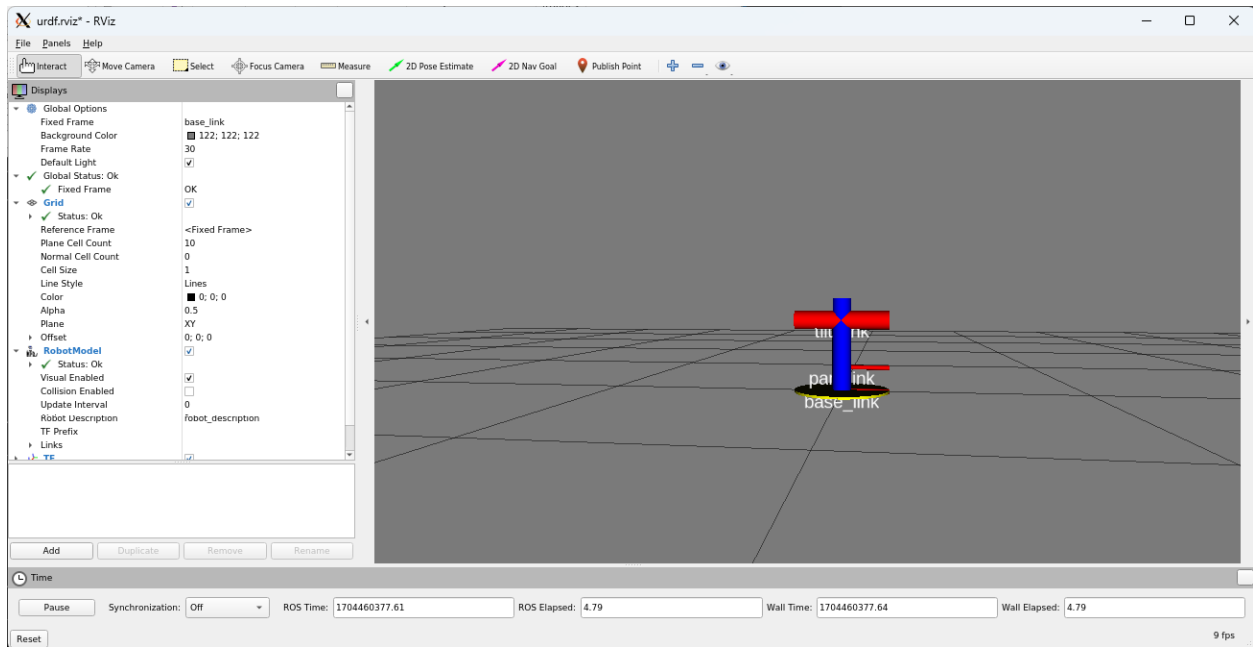
Berikut merupakan isi file catkin_ws setelah melakukan gitclon

```
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg/urdf# ls
diff_wheeled_robot.urdf      pan_tilt.pdf      sensors          seven_dof_arm_with_rgbd.xacro
diff_wheeled_robot.xacro    pan_tilt.urdf    seven_dof_arm.urdf  wheel.urdf.xacro
diff_wheeled_robot_with_laser.xacro pan_tilt.xacro  seven_dof_arm.xacro
pan_tilt.gv                 pan_tilt_generated.urdf seven_dof_arm_moveit.urdf
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg/urdf#
```

Kita juga dapat melakukan export file urdf ke bentuk pdf untuk melihat bentukan persendian dari robot tersebut

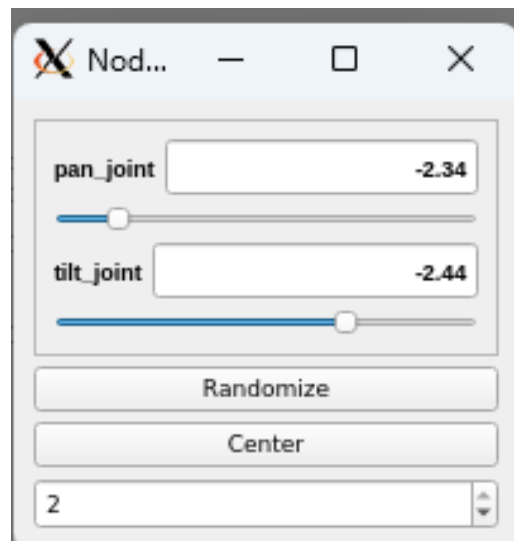
```
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg/urdf# urdf_to_graphviz pan_tilt.urdf
Created file pan_tilt.gv
Created file pan_tilt.pdf
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg/urdf#
```

Setelah melakukan beberapa konfigurasi mengikuti tutorial yang diberikan di buku, selanjutnya kita lakukan run program



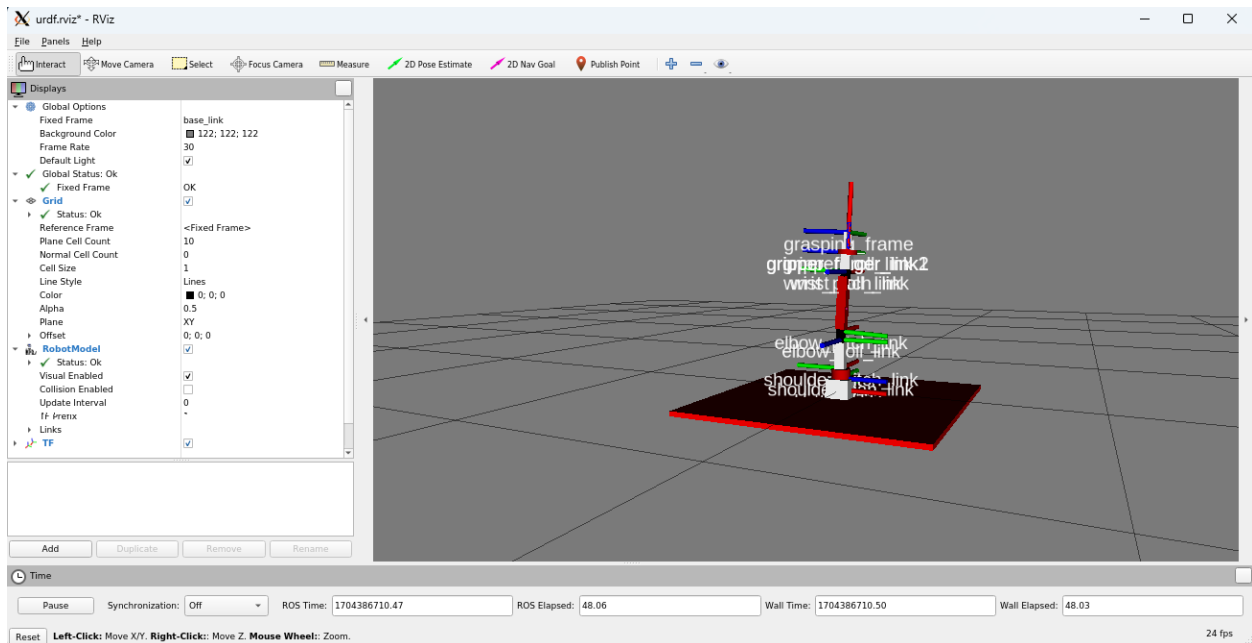
Dalam versi ROS sebelumnya, GUI joint_state_publisher diaktifkan berkat parameter ROS yang disebut `use_gui`. Untuk memulai GUI di file peluncuran, ini parameter harus diatur ke `true` sebelum memulai node `joint_state_publisher`. Dalam versi ROS saat ini, file peluncuran harus diperbarui untuk meluncurkan `joint_state_publisher_gui` alih-alih menggunakan `joint_state_publisher` dengan `use_gui` parameter.

Kita juga dapat mengerjakan persendian diatas menggunakan GUI yang disediakan oleh Rviz



Creating the robot description for a seven-DOF robot manipulator

Kita sudah bisa membuat beberapa robot kompleks menggunakan URDF dan xacro. Robot selanjutnya yang akan kita buat adalah seven doff arm robot. Berikut merupakan penampakan dari seven doff arm robot setelah melakukan beberapa konfigurasi dan running

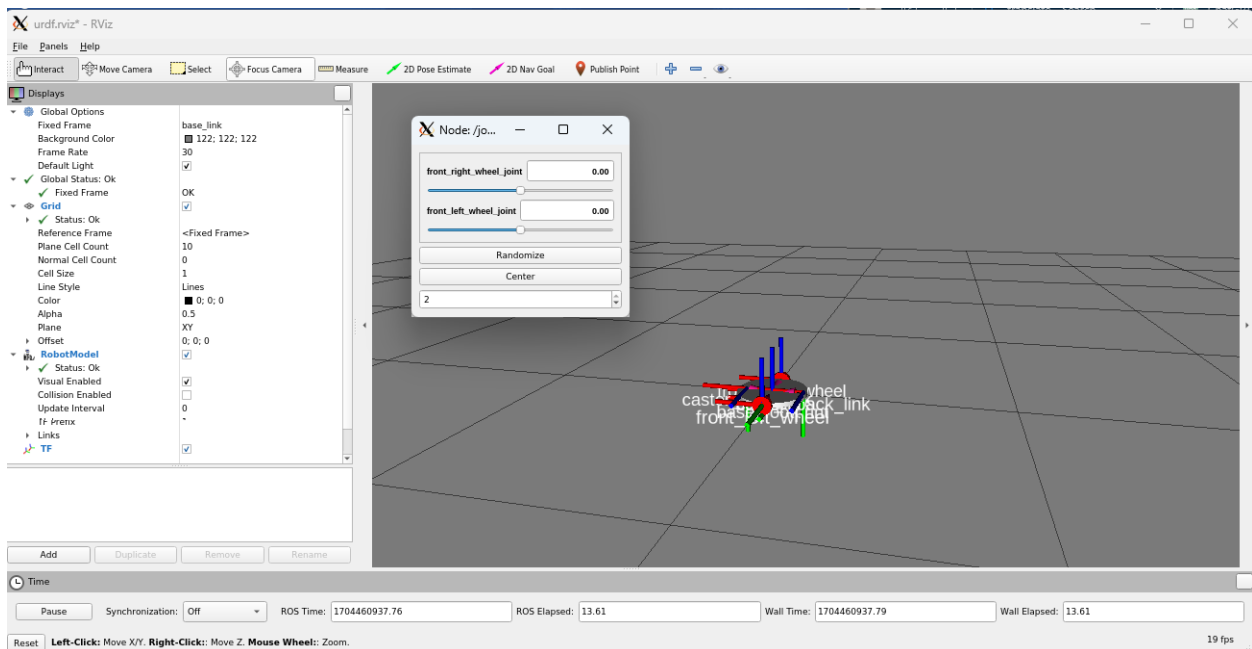


Paket `joint_state_publisher` dalam kerangka kerja ROS (Robot Operating System) telah menjadi salah satu komponen esensial yang banyak dimanfaatkan dalam pengembangan aplikasi robotik untuk mengelola interaksi dengan sendi-sendi robot. Paket ini, yang mencakup node penerbit `joint_state`, dirancang khusus untuk menavigasi struktur yang fleksibel dari model URDF (Unified Robot Description Format) dan mengirimkan informasi status yang terpadu dari setiap sendi melalui pesan dengan format `sensor_msgs/JointState`. Dalam konteks implementasi, paket ini seringkali dikombinasikan dengan `robot_state_publisher`, memungkinkan publikasi posisi yang akurat dari seluruh sendi robot. Adapun konfigurasi nilai sendi, berbagai sumber dapat digunakan untuk mengatur parameter ini. Sebagai contoh, dalam skenario pengujian, pendekatan yang efektif melibatkan penggunaan antarmuka GUI berbasis slider untuk modifikasi nilai. Namun, untuk aplikasi yang lebih formal atau spesifik, node dapat diatur untuk berlangganan topik `JointState`, memberikan fleksibilitas dalam penyesuaian dan integrasi nilai sendi dalam sistem robotik yang lebih kompleks.

Creating a robot model for the differential drive mobile robot

Sebuah robot beroda diferensial dirancang dengan dua roda yang dipasang pada sisi berlawanan dari kerangka robot, yang biasanya didukung oleh satu atau dua roda kastor untuk stabilitas. Sistem kontrol robot ini didasarkan pada prinsip diferensial, di mana kecepatan dan arah pergerakan robot ditentukan oleh perbedaan kecepatan antara kedua roda. Dalam operasi standar, apabila kedua motor roda diatur untuk beroperasi pada kecepatan yang seragam, robot akan bergerak secara linear, baik maju maupun mundur, tergantung pada orientasi roda. Namun, untuk mengimplementasikan gerakan belok atau manuver, kecepatan roda pada satu sisi diatur berbeda dari sisi lainnya. Sebagai contoh, untuk mendorong robot berbelok ke arah kiri, akan dilakukan pengurangan kecepatan pada roda kiri, sementara roda kanan tetap beroperasi pada kecepatan yang lebih tinggi atau stabil. Prinsip ini memungkinkan robot beroda diferensial untuk menavigasi dengan fleksibilitas dan presisi sesuai dengan instruksi atau kebutuhan aplikasi yang ditentukan.

Setelah melakukan perintah dibuku terkait konfigurasi robot berikut merupakan tampilan dari robot jika running menggunakan Rviz



4

Simulating Robots Using Ros And Gazebo

Gazebo adalah simulator multi-robot untuk simulasi robot indoor dan outdoor yang kompleks. Kita dapat mensimulasikan robot kompleks, sensor robot, dan berbagai objek 3D. Gazebo sudah memiliki model simulasi robot populer, sensor, dan berbagai objek 3D di ITS repositori (https://bitbucket.org/osrf/gazebo_models/). Kita bisa langsung Gunakan model ini tanpa harus membuat yang baru.

Gazebo terintegrasi sempurna dengan ROS berkat antarmuka ROS yang tepat, yang mengekspos kontrol penuh Gazebo di ROS. Kita bisa memasang Gazebo tanpa ROS, tapi kita harus menginstal antarmuka ROS-Gazebo untuk berkomunikasi dari ROS ke Gazebo.

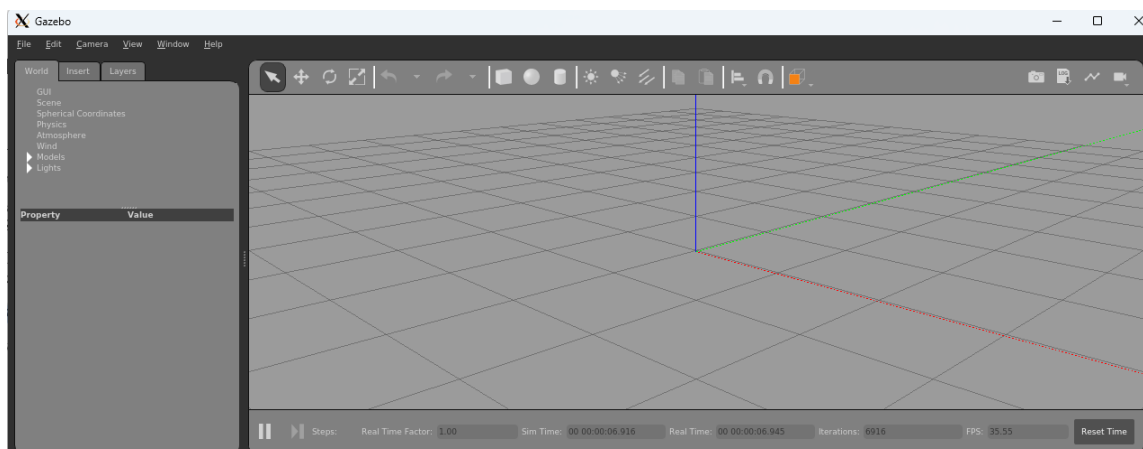
Simulating the robotic arm using Gazebo and ROS

Dalam bab sebelumnya, telah dirancang sebuah manipulator dengan tujuh derajat kebebasan (DOF). Pada bagian ini, fokusnya adalah pada simulasi robot menggunakan lingkungan simulasi Gazebo yang terintegrasi dengan kerangka kerja ROS (Robot Operating System). Sebelum memasuki tahapan simulasi dengan Gazebo dan integrasinya dengan ROS, langkah awal yang kritis adalah memastikan instalasi paket-paket yang esensial untuk mendukung operasionalitas kedua platform ini. Oleh karena itu, untuk memastikan integrasi dan fungsi yang optimal, diperlukan instalasi paket-paket khusus yang disarankan dan relevan untuk kerja sama antara Gazebo dan ROS.

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

```
Setting up libserd-0-0:amd64 (0.30.2-1) ...  
Setting up libdart6 (6.9.2-2build4) ...  
Setting up libkmldev1:amd64 (1.3.0-8build1) ...  
Setting up libpoppler-dev:amd64 (0.86.1-0ubuntu1.4) ...  
Setting up libspatialite7:amd64 (4.3.0a-6build1) ...  
Setting up libignition-common3-events:amd64 (3.14.2-1~focal) ...  
Setting up mesa-vdpau-drivers:amd64 (21.2.6-0ubuntu0.1~20.04.2) ...  
Setting up libzvb10:amd64 (0.2.35-17) ...  
Setting up libkadm5clnt-mit11:amd64 (1.17-6ubuntu4.4) ...  
Setting up libqwt-qt5-dev (6.1.4-1.1build1) ...  
Setting up libcharls-dev:amd64 (2.0.0+dfsg-1build1) ...  
Setting up libzip-dev:amd64 (1.5.1-0ubuntu1) ...  
Setting up libcfitsio-dev:amd64 (3.470-3) ...  
Setting up libopencv-core4.2:amd64 (4.2.0+dfsg-5) ...  
Setting up libzmq5:amd64 (4.3.2-2ubuntu1) ...  
Setting up libkmlengine1:amd64 (1.3.0-8build1) ...  
Setting up libdc1394-22-dev:amd64 (2.2.5-2.1) ...  
Setting up qt5-qmake:amd64 (5.12.8+dfsg-0ubuntu2.1) ...  
Setting up libpcre2-dev:amd64 (10.34-7ubuntu0.1) ...  
Setting up libignition-fuel-tools4:amd64 (4.6.0-1~focal) ...  
Setting up libkmlconvenience1:amd64 (1.3.0-8build1) ...  
Setting up libignition-common3-profiler:amd64 (3.14.2-1~focal) ...  
Setting up ros-noetic-controller-interface (0.20.0-1focal.20231030.153718) ...  
Setting up libcdio-cdda2:amd64 (10.2+2.0.0-1) ...  
Setting up libselinux1-dev:amd64 (3.0-1build2) ...  
Setting up libfreexl-dev:amd64 (1.0.5-3) ...  
Setting up libopencv-imgproc4.2:amd64 (4.2.0+dfsg-5) ...  
Setting up libcdio-paranoia2:amd64 (10.2+2.0.0-1) ...  
Setting up libfyba-dev:amd64 (4.1.1-6build1) ...
```

Setelah terinstall kita lakukan pengujian apakah gazebo terinstall dengan sempurna



Creating the robotic arm simulation model for Gazebo

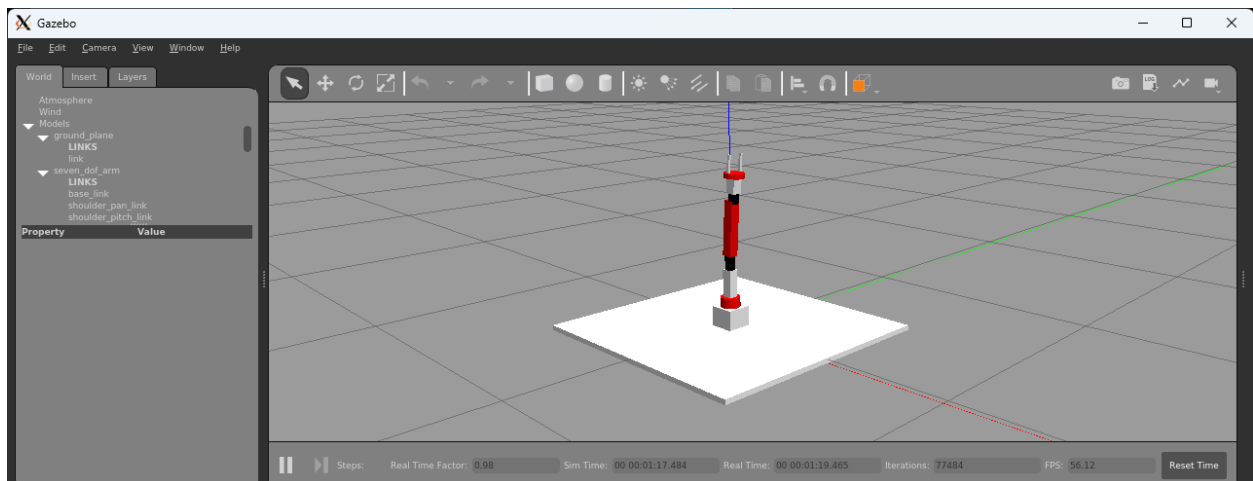
Untuk menghasilkan model simulasi dari lengan robot, pendekatan yang dianjurkan adalah dengan memodifikasi deskripsi robot yang ada dengan menambahkan parameter-parameter khusus yang mendukung simulasi. Dengan memanfaatkan kerangka kerja ROS (Robot Operating System), proses ini dapat dilaksanakan melalui pembuatan paket khusus yang dirancang untuk tujuan simulasi lengan robot. Untuk inisiasi proses ini, langkah pertama yang harus diambil adalah melaksanakan perintah-perintah tertentu yang akan memfasilitasi pembuatan dan konfigurasi paket tersebut sesuai kebutuhan simulasi yang diinginkan.

```
catkin_create_pkg seven_dof_arm_gazebo gazebo_msgs gazebo_plugins  
gazebo_ros gazebo_ros_control mastering_ros_robot_description_pkg
```

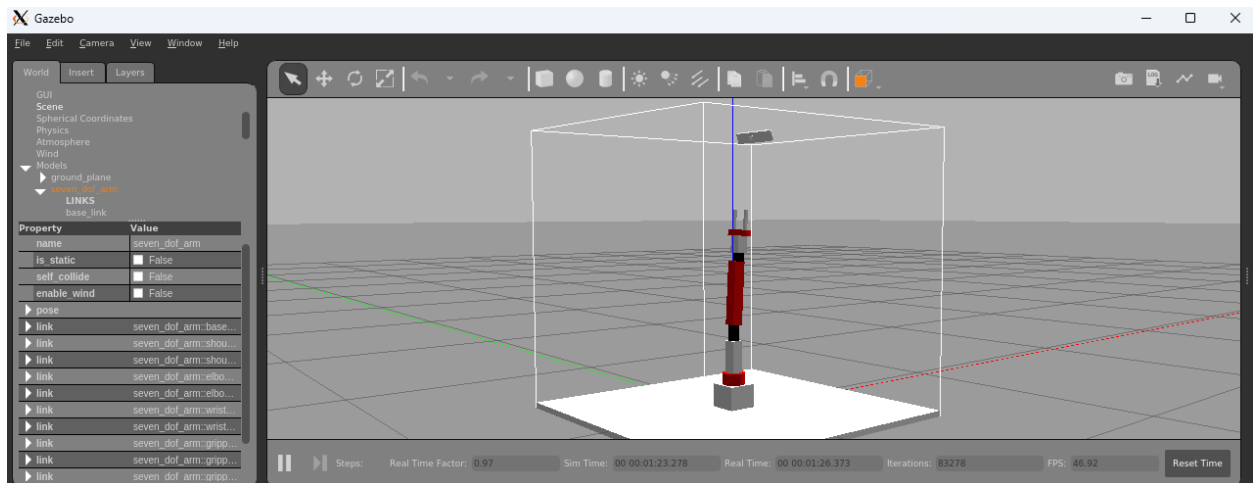
agar mempermudah proses praktek, saya melakukan clone dari github repos, berikut merupakan tampilan dari direktori saya

```
root@024a5ddf0613:~/catkin_ws/src/seven_dof_arm_gazebo/launch# ls  
seven_dof_arm_bringup_moveit.launch    seven_dof_arm_gazebo_control.launch    seven_dof_arm_with_rgbd_world.launch  
seven_dof_arm_bringup_obstacle_moveit.launch  seven_dof_arm_obstacle_world.launch    seven_dof_arm_world.launch  
root@024a5ddf0613:~/catkin_ws/src/seven_dof_arm_gazebo/launch#
```

Setelah melakukan beberapa konfigurasi yang dibutuhkan selanjutnya kita lakukan running pada program robot



Sekarang kita telah belajar tentang definisi plugin kamera di Gazebo, kita dapat meluncurkan Simulasi lengkap kami menggunakan perintah berikut seperti gambar dibawah ini



Kita juga dapat melihat simulasi pada robot bahwa robot memiliki warna yang berbeda, warna ini bisa kita ubah sesuai dengan keinginan kita, berikut merupakan syntax yang mengutip permasalahan warna

```
<gazebo reference="bottom_link">
<material>Gazebo/White</material>
</gazebo>

<gazebo reference="base_link">
<material>Gazebo/White</material>
</gazebo>

<gazebo reference="shoulder_pan_link">
<material>Gazebo/Red</material>
</gazebo>
```

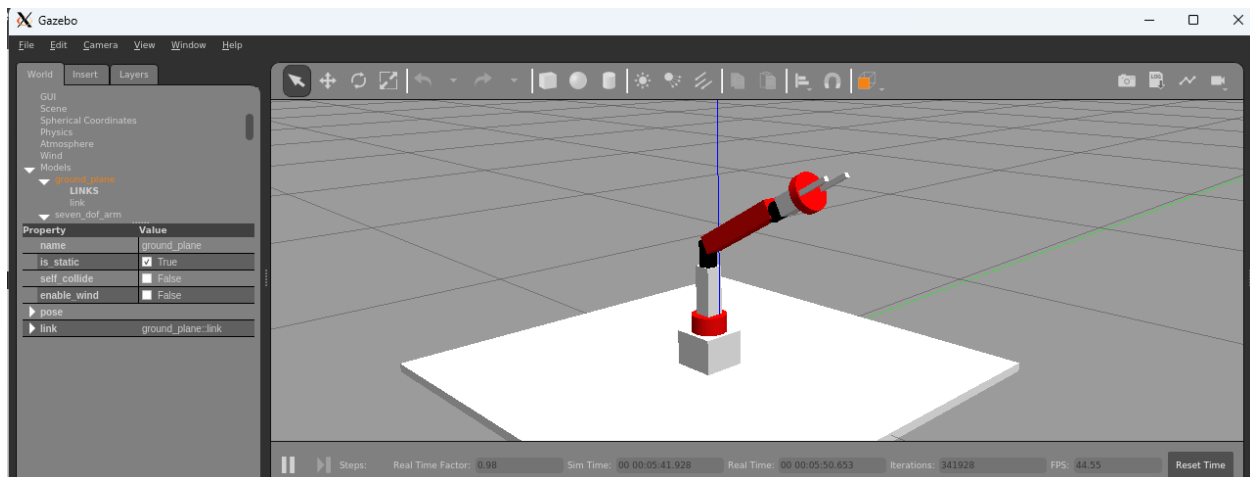
File peluncuran yang telah disusun bertujuan untuk menginisiasi simulasi Gazebo dari lengan robot. Dalam proses ini, file tersebut mengintegrasikan berbagai konfigurasi yang esensial untuk operasionalitas yang optimal. Hal ini mencakup, namun tidak terbatas pada, konfigurasi pengontrol, pemuatan pengontrol status bersama, serta pengontrol posisi bersama. Sebagai tahap penutup, file peluncuran akan menginisialisasi robot penerbit negara, yang bertugas untuk mempublikasikan status bersama dan transformasi (TF). Setelah seluruh proses simulasi dijalankan melalui file peluncuran ini, langkah berikutnya yang kritis adalah melakukan pemeriksaan mendetail terhadap topik-topik pengontrol yang dihasilkan untuk memastikan konsistensi, akurasi, dan integritas informasi yang diterbitkan dalam lingkungan simulasi.

```

[INFO] [1704389771.071951, 0.395900000]: Loaded gazebo_ros_control.
[INFO] [1704389771.078084, 0.396000]: wait_for_service(/seven_dof_arm/controller_manager/load_controller)
able to contact [rospc://024a5ddf0613:43219]
[INFO] [1704389771.079262, 0.397000]: Controller Spawner: Waiting for service controller_manager/switch_
[INFO] [1704389771.081880, 0.400000]: Controller Spawner: Waiting for service controller_manager/unload_
[INFO] [1704389771.084078, 0.402000]: Loading controller: joint_state_controller
[INFO] [1704389771.089823, 0.408000]: Loading controller: joint1_position_controller
[INFO] [1704389771.097109, 0.415000]: Loading controller: joint2_position_controller
[INFO] [1704389771.102195, 0.420000]: Loading controller: joint3_position_controller
[INFO] [1704389771.106977, 0.425000]: Loading controller: joint4_position_controller
[INFO] [1704389771.111964, 0.430000]: Loading controller: joint5_position_controller
[INFO] [1704389771.117180, 0.435000]: Loading controller: joint6_position_controller
[INFO] [1704389771.122155, 0.440000]: Loading controller: joint7_position_controller
[INFO] [1704389771.127185, 0.445000]: Controller Spawner: Loaded controllers: joint_state_controller, jo
ntroller, joint2_position_controller, joint3_position_controller, joint4_position_controller, joint5_pos
, joint6_position_controller, joint7_position_controller
[INFO] [1704389771.130198, 0.448000]: Started controllers: joint_state_controller, joint1_position contr
ition controller, joint3_position controller, joint4_position controller, joint5_position controller, j

```

Setelah melakukan apa yang harus dilakukan pada instruksi buku, terdapat perintah untuk menggerakkan lengan robot dengan cara melakukan publish topic dengan data float, yang menjadi parameter berapa derajatnya perpindahan lengan robot tersebut, berikut merupakan tampilan robot setelah kita mengirimkan topicnya

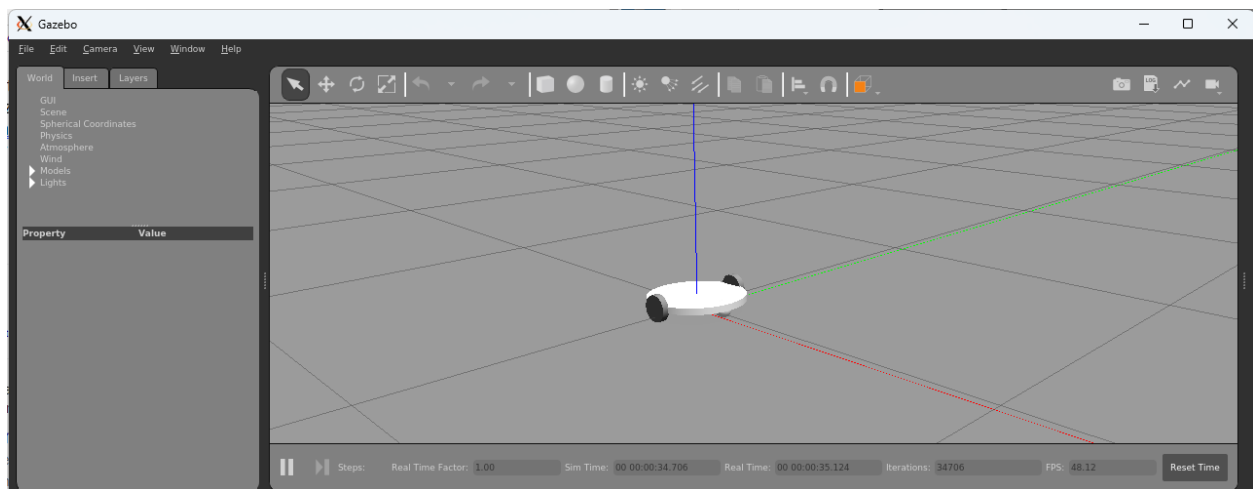


Simulating a differential wheeled robot in Gazebo

Dalam eksplorasi sebelumnya, telah disajikan simulasi dari lengan robot dengan detail tertentu. Pada bagian ini, fokus berpindah untuk mengatur simulasi robot beroda diferensial yang sebelumnya telah dirancang dalam bab sebelumnya. Untuk referensi dan detail lebih lanjut, deskripsi lengkap dari robot beroda diferensial dapat ditemukan dalam file `diff_wheeled_robot.xacro`, yang terletak di direktori `mastering_ros_robot_description_pkg/URDF`.

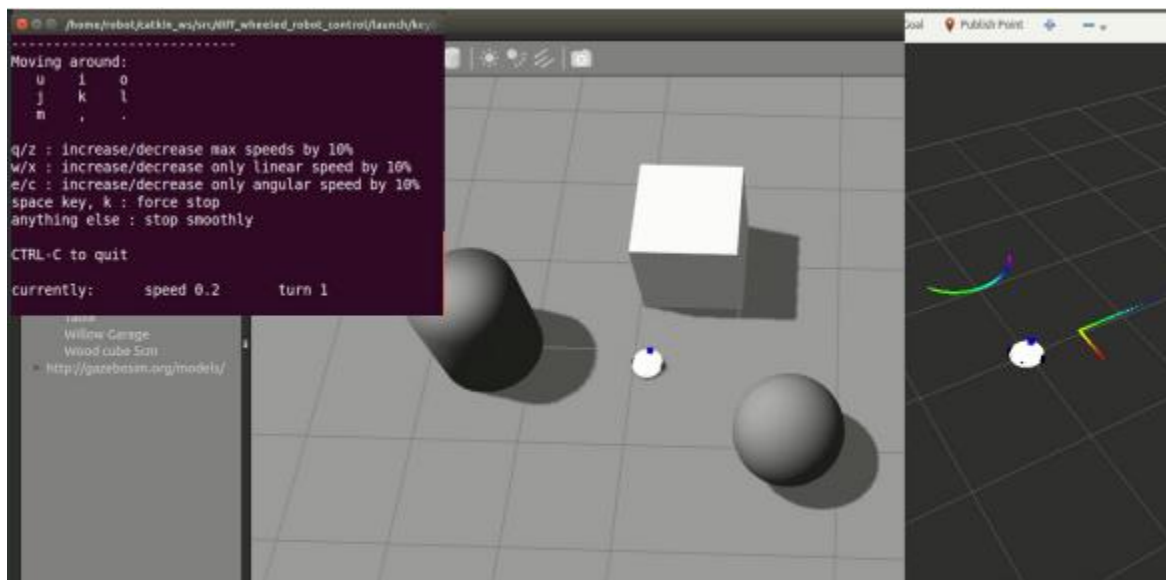
Langkah selanjutnya adalah penyusunan file peluncuran yang akan memfasilitasi inisiasi model simulasi dalam lingkungan Gazebo. Sebagaimana pendekatan yang diterapkan pada simulasi lengan robot, kita akan memanfaatkan kerangka kerja ROS dengan membuat paket khusus yang berintegrasi dengan dependensi paket `seven_dof_arm_gazebo`. Jika Anda telah mengakses atau mengkloning repositori kode, paket tersebut seharusnya sudah ada dalam repositori yang Anda miliki. Namun, jika belum, opsi alternatif adalah untuk mengkloning seluruh kode dari repositori Git yang relevan atau memperoleh paket tersebut melalui sumber yang disediakan dalam materi buku.

Setelah semua konfigurasi diatur dan disiapkan, langkah berikutnya yang krusial adalah melaksanakan percobaan simulasi. Hasil atau output dari eksekusi program ini akan memberikan gambaran menyeluruh mengenai performa dan integritas model simulasi



Node `teleop` dalam kerangka kerja ROS berfungsi sebagai penerbit perintah berdasarkan masukan dari keyboard, menghasilkan pesan ROS `Twist` yang mencakup informasi tentang kecepatan linier dan sudut dari robot. Sebagai praktek yang umum dan efisien, implementasi node `teleop` yang standar sudah tersedia dalam berbagai paket ROS. Dalam konteks ini, untuk kontrol robot beroda diferensial, kita dapat memanfaatkan dan memodifikasi simpul yang telah ada untuk memenuhi kebutuhan spesifik.

Implementasi node teleop untuk robot beroda diferensial dikemas dalam paket yang disebut `diff_wheeled_robot_control`. Di dalam struktur direktori paket, Anda akan menemukan node khusus bernama `diff_wheeled_robot_key`, yang berperan sebagai node teleop untuk mengontrol robot. Sebagai tindak lanjut dari proses ini, Anda dapat memperoleh paket ini dengan mengunduhnya dari repositori Git yang relevan. Dengan akses ke repositori tersebut, Anda dapat mengkloning paket tersebut ke lingkungan kerja ROS Anda dengan menggunakan perintah yang disediakan.



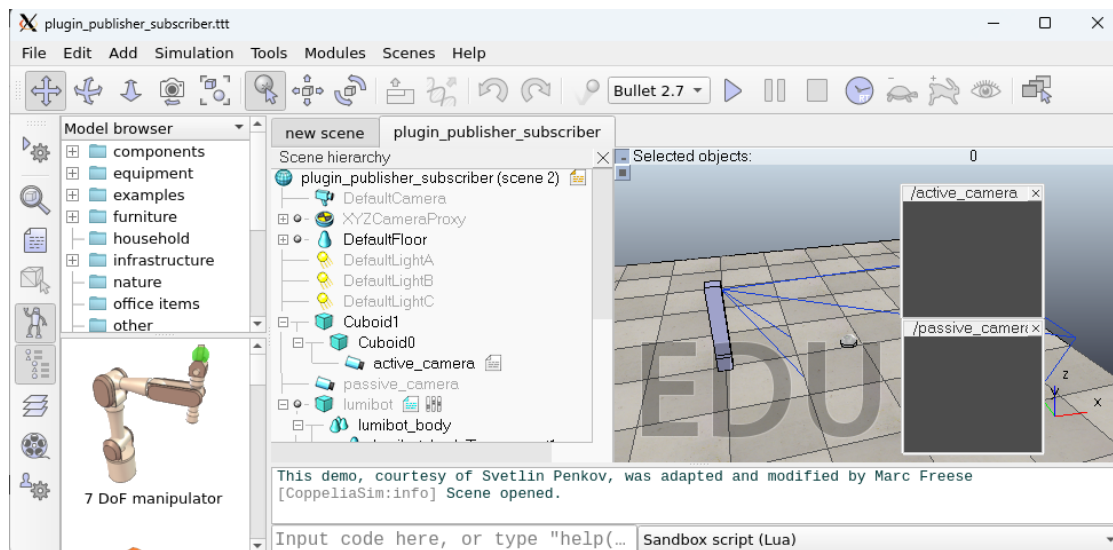
Simulating Robots Using Ros Coppeliasim, and Webots

Ini adalah simulator robot multiplatform. CoppeliaSim dikembangkan oleh CoppeliaSim Robotika. Ini menawarkan banyak model simulasi robot industri dan mobile populer yang siap untuk digunakan, dan berbagai fungsi yang dapat dengan mudah diintegrasikan dan digabungkan melalui antarmuka pemrograman aplikasi (API) khusus. Selain itu, bisa beroperasi dengan Robot Operating System (ROS) menggunakan antarmuka komunikasi yang tepat. Itu memungkinkan kita untuk mengontrol adegan simulasi dan robot melalui topik dan servis.

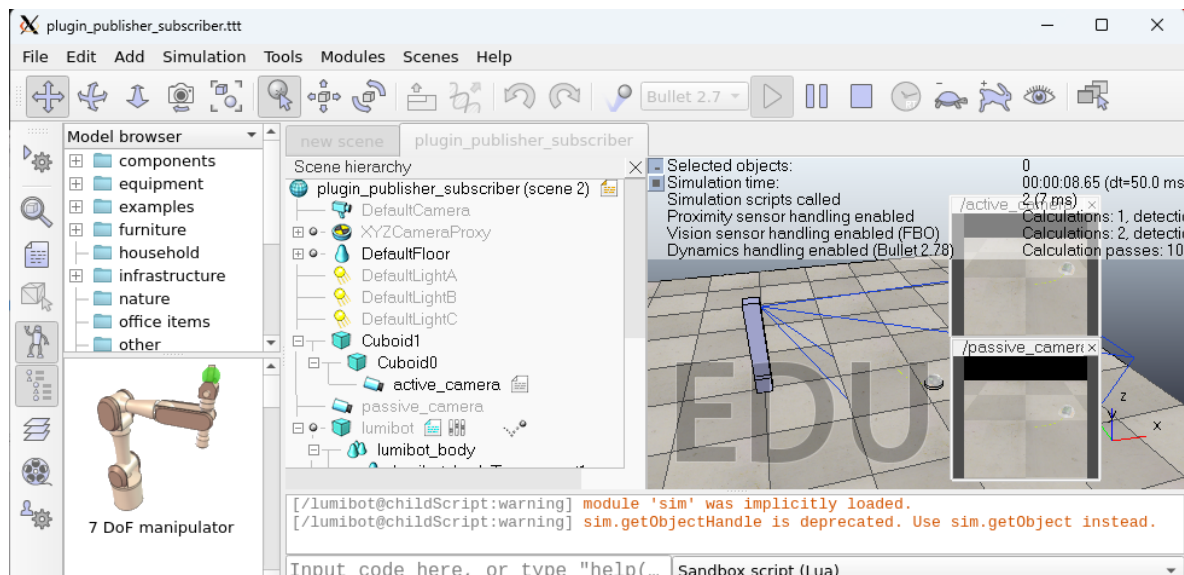
Simulating Robots Using ROS, CoppeliaSim, and Webots

Sebelum memulai integrasi dengan CoppeliaSim, langkah awal yang diperlukan adalah menginstal perangkat lunak tersebut di sistem yang akan digunakan. Selain proses instalasi, penting untuk mengkonfigurasi lingkungan kerja agar dapat mendukung integrasi yang efektif antara ROS (Robot Operating System) dan lingkungan simulasi CoppeliaSim.

CoppeliaSim dirancang sebagai perangkat lunak lintas platform, mendukung berbagai sistem operasi termasuk Windows, macOS, dan distribusi Linux. Perangkat lunak ini dikembangkan oleh Coppelia Robotics GmbH dan tersedia dalam dua model lisensi: pendidikan dan komersial. Menariknya, CoppeliaSim menawarkan opsi lisensi gratis untuk keperluan pendidikan dan komersial, memfasilitasi aksesibilitas dan penerapan yang lebih luas di kalangan pengembang dan peneliti robotika.

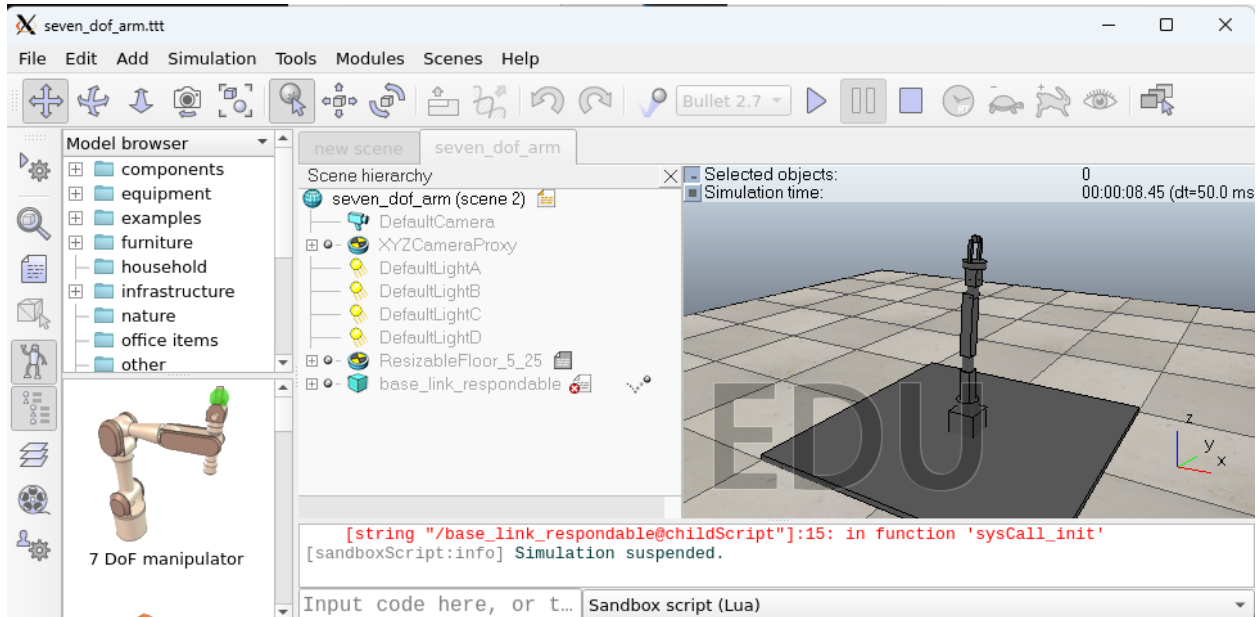


Kemudian kita lakukan running agar kamera menyala dan kita bisa melihat apa yang disaksikan oleh robot tersebut



Simulating a robotic arm using Coppeliassim and ROS

Selanjutnya kita akan mencoba melakukan simulasi seven doff arm pada copeliasim ini, sebelumnya kita sudah melakukan simulasi ini pada Rviz, berikut merupakan tampilan seven doff arm pada copeliasim



Setelah berhasil mengimpor model URDF, langkah berikutnya adalah melakukan serangkaian konfigurasi yang diperlukan agar lengan robot dapat beroperasi sesuai keinginan. Konfigurasi ini meliputi penyesuaian properti objek dalam scene, termasuk parameter dinamis dari setiap sendi yang ada.

Dalam konteks ini, metode `setJointTargetPosition` digunakan sebagai instrumen utama untuk menginisiasi perubahan posisi pada sendi tertentu. Parameter yang diperlukan untuk metode ini meliputi penanganan objek bersama (joint) serta nilai yang spesifik yang ingin ditetapkan untuk sendi tersebut.

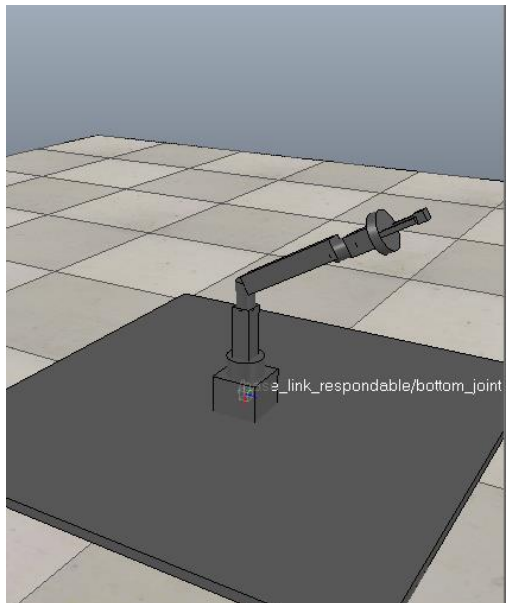
Ketika simulasi telah dimulai, operasi lanjutan dapat dilakukan dengan memanipulasi sendi yang ditargetkan. Sebagai contoh, untuk sendi dengan label "elbow_pitch joint", pengguna dapat menerbitkan nilai yang sesuai melalui antarmuka baris perintah. Implementasi praktis dari operasi ini dapat ditemukan dalam contoh kode yang disajikan berikut ini.

```
rostopic pub /csim_demo/seven_dof_arm/elbow_pitch/cmd std_msgs/ Float32  
"data: 1.0"
```

At the same time, we can get the position of the joint listening on the state topic, as follows:

```
rostopic echo /csim_demo/seven_dof_arm/elbow_pitch/state
```


berikut merupakan output dari kode diatas

[illegible]

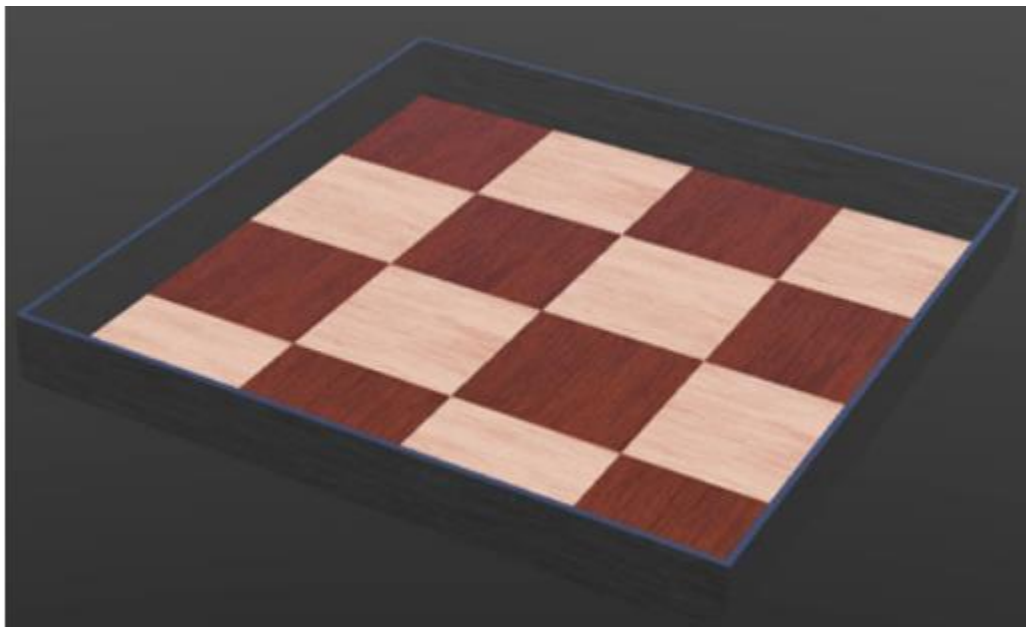
Simulating a mobile robot with Webots

Misi dari bagian ini adalah untuk mengembangkan sebuah adegan simulasi dari awal yang melibatkan objek-objek tertentu serta robot beroda yang bergerak dinamis. Untuk memulai, langkah pertama adalah menciptakan sebuah lingkungan simulasi kosong yang baru. Sebagai alat bantu, kita dapat memanfaatkan opsi wizard yang disediakan untuk inisiasi adegan simulasi.

Adegan simulasi yang dimaksud sudah tersedia dalam kode sumber buku, ditempatkan dalam paket dengan nama ``webots_demo_pkg``. Untuk menginisiasi adegan simulasi baru, navigasi ke menu bilah atas dan pilih opsi "Wizards" diikuti oleh "Direktori Proyek Baru". Interface applet yang disediakan akan memandu Anda melalui proses konfigurasi.

Selanjutnya, pilih "Berikutnya" untuk menentukan direktori proyek yang akan digunakan. Anda akan diminta untuk menentukan lokasi folder; pada tahap ini, masukkan "mobile_robot" sebagai nama folder proyek. Selanjutnya, spesifikasikan nama untuk adegan simulasi yang akan dibuat, sebagai contoh "robot_motion_controller.wbt". Sebagai tambahan, pastikan untuk menyesuaikan parameter lain sesuai kebutuhan, seperti menambahkan opsi area persegi panjang.

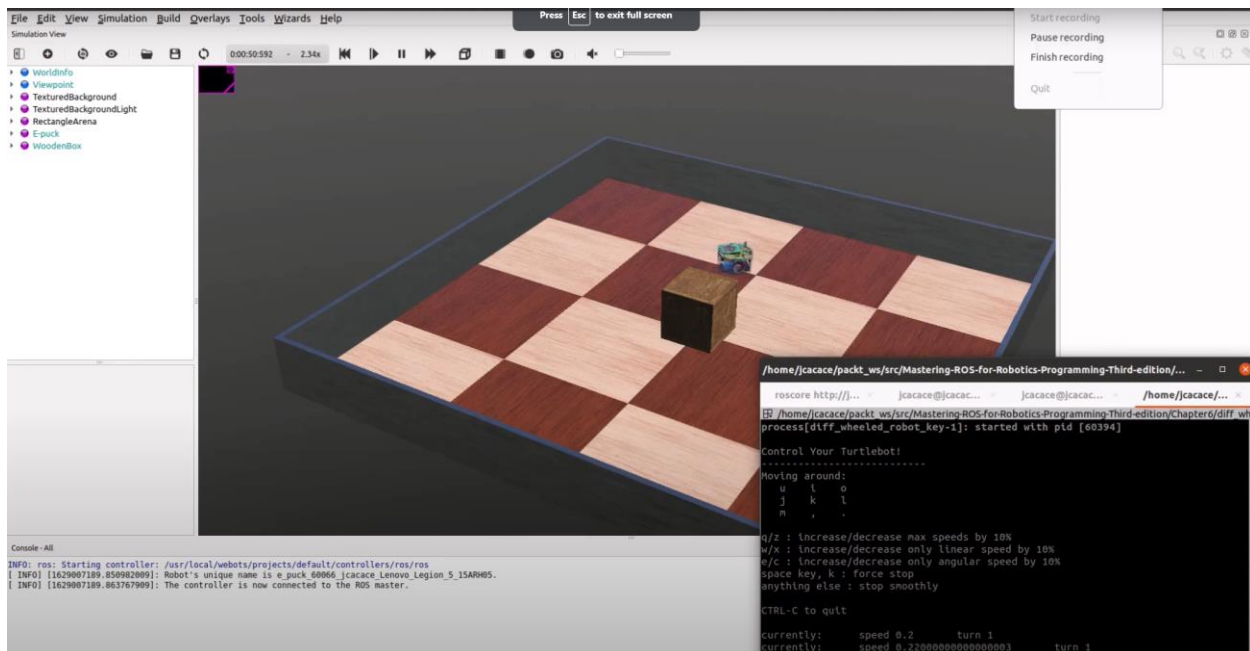
Setelah semua pengaturan selesai, konfirmasi pilihan Anda dengan mengklik "Selesai". Setelah proses ini selesai, adegan simulasi yang telah dikonfigurasi seharusnya akan muncul di layar sesuai dengan tampilan yang diharapkan, sebagaimana ditampilkan dalam ilustrasi layar yang tersedia.



```
jcacace@jcacace-Lenovo-Legion-5-15ARH05: ~/dev/coppeliaSim$ rosservice call /e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/robot/time_step
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/robot/wait_for_user_input_event
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/robot/wwi_receive_text
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/robot/wwi_send_text
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/set_logger_level
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/get_engine
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/get_language
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/get_model
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/get_name
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/get_node_type
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/is_sound_playing
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/is_speaking
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/play_sound
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/set_engine
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/set_language
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/speak
/e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/speaker/stop
/rosout/get_loggers
/rosout/set_logger_level
jcacace@jcacace-Lenovo-Legion-5-15ARH05:~/dev/coppeliaSim$ rosservice call /e_puck_59313_jcacace_Lenovo_Legion_5_15ARH05/
```

e-puck adalah sebuah robot beroda kecil yang populer di dalam komunitas penelitian robotika. Dalam konteks penggunaan Webots atau lingkungan simulasi robot lainnya, e-puck manager atau manajer untuk e-puck biasanya merujuk pada antarmuka atau alat yang memungkinkan pengguna untuk mengelola, mengkonfigurasi, atau mengendalikan simulasi atau perangkat nyata dari robot e-puck.

Selanjutnya kita akan mendemonstrasikan teleop dalam webots, teleop adalah semacam controller yang dapat kita gerakan menggunakan keyboard kita



Simulating the robotic arm using Webots and ROS

Integrasi Webots-ROS membutuhkan dua sisi: sisi ROS dan sisi Webots. The ROS sisi diimplementasikan melalui paket ROS `webots_ros`, sementara Webots mendukung ROS Secara native berkat pengontrol standar yang dapat ditambahkan ke model robot apa pun. Untuk menggunakan Webots dengan ROS, Anda perlu menginstal paket `webots_ros`. Ini dapat dilakukan dengan menggunakan APT, sebagai berikut:

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

Kesimpulan

Dalam bab ini, fokus utama kami adalah untuk mereplikasi dan memperluas eksplorasi yang telah kami lakukan pada bab sebelumnya dengan platform simulasi Gazebo. Kali ini, kami memperkenalkan dua platform simulasi tambahan yang populer, yaitu CoppeliaSim dan Webots. Kedua platform ini menonjol sebagai perangkat lunak simulasi multiplatform yang canggih, yang menggabungkan teknologi lanjutan dengan fleksibilitas yang luar biasa.

Salah satu keunggulan utama dari CoppeliaSim dan Webots adalah antarmuka pengguna (UI) yang intuitif, yang memfasilitasi adaptasi cepat bagi pengguna baru dan mahir. Dalam eksplorasi simulasi ini, kami berfokus pada dua model robot spesifik. Pertama, kami memanfaatkan file URDF untuk merepresentasikan model lengan dengan tujuh derajat kebebasan (DOF), yang telah dirancang dan dianalisis dalam bab sebelumnya. Sementara itu, model robot kedua adalah representasi dari robot beroda diferensial, sebuah entitas populer yang tersedia dalam framework simulasi Webots.

Pentingnya interaksi antara ROS dan model simulasi menjadi sorotan utama, di mana kami memandu pembaca melalui proses pengontrolan sendi pada model lengan dan pengendalian pergerakan robot beroda diferensial melalui topik yang relevan di ROS. Sebagai kelanjutan dari eksplorasi ini, bab selanjutnya akan memfokuskan pada integrasi antara lengan robot dengan paket ROS MoveIt, sementara juga mengeksplorasi kemampuan navigasi untuk robot beroda diferensial melalui paket navigasi yang disediakan.