

TECHNICAL REPORT MASTERING ROS2 BOOK WITH WEBOT

Penulis: Rida Putri Andini

Kelas: TK44G7

Pendahuluan:

ROS, atau Robot Operating System, adalah sebuah kerangka kerja perangkat lunak yang dirancang khusus untuk memfasilitasi pengembangan aplikasi robotika. Berbeda dengan sistem operasi konvensional, ROS menyediakan modularitas yang memungkinkan pengguna untuk mengintegrasikan berbagai komponen robotika dengan mudah, menggabungkan pustaka, alat, dan konvensi standar yang mendukung pengembangan yang efisien. Keunikan ROS terletak pada komunitas open-source yang besar, memfasilitasi pertukaran pengetahuan dan sumber daya antara pengembang dan peneliti. Dengan alat simulasi seperti Gazebo, ROS memungkinkan validasi dan pengujian algoritma dalam lingkungan virtual sebelum penerapannya pada robot fisik, mengurangi risiko dan waktu pengembangan. Selain itu, kemampuan ROS untuk diintegrasikan dengan sistem otomasi industri memperluas relevansinya, memungkinkan adaptasi solusi otomasi yang lebih efisien dan modular. Secara keseluruhan, ROS telah mendefinisikan standar baru dalam pengembangan robotika dan otomasi, memanfaatkan fleksibilitasnya untuk mempercepat inovasi dan meningkatkan fungsionalitas sistem.

Tujuan dan Ruang Lingkup:

Adapun tujuan dalam pembuatan Technical Report ini adalah;

- untuk memenuhi nilai UAS Robotic akelas TK44G7 Universitas Telkom

Ruang lingkup yang dicakup pada technical report ini adalah;

- Getting Started with ROS Programming
- Working with ROS for 3D Modeling
- Simulating Robots Using ROS and Gazebo
- Simulating Robots Using ROS , Coppeliassim, and Webots

Metodologi:

Adapun metodologi yang saya gunakan pada pengerjaan technical report ini adalah, baca, amati, riset dan praktekan. Hal pertama yang saya lakukan adalah membaca, dalam kasus ini saya pertama tama membaca buku mastering ROS, kemudian saya amati code nya dan jika terjadi error saya akan lakukan riset di internet bagaimana cara fix errornya

Hasil:

Pada bagian ini saya akan mendeskripsikan langkah langkah atau proses yang saya lakukan berdasarkan instruksi buku “Mastering ROS”. Langkah langkah akan saya jelaskan menggunakan screenshots program cmd. Saya juga akan menyertakan kode kode yang saya gunakan pada teknikal report ini.

2. Getting Started with ROS Programming

- Creating 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 envirotnment 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.

```
root@6caed748117d:/# cd ~/catkin_ws/src
root@6caed748117d:~/catkin_ws/src# catkin_init_workspace
root@6caed748117d:~/catkin_ws/src# |
```

kita dapat membangun workspace walaupun tidak terdapat package didalamnya. Kita dapat melakukan seperti command dibawah ini untuk pindah ke folder workspace, tuliskan catkin_make.

```

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# |

```

lalu tuliskan command seperti dibawah untuk membuat sample ROS package:

```

root@6caed748117d:~/catkin_ws# catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs
Created file mastering_ros_demo_pkg/package.xml
Created file mastering_ros_demo_pkg/CMakeLists.txt
Created folder mastering_ros_demo_pkg/include/mastering_ros_demo_pkg
Created folder mastering_ros_demo_pkg/src
Successfully created files in /root/catkin_ws/mastering_ros_demo_pkg. Please adjust the values in package.xml.
root@6caed748117d:~/catkin_ws# actionlib actionlib_msgs|

```

- Creating ROS nodes

Simpul pertama yang akan kita bahas adalah demo_topic_publisher.cpp. Node ini akan Menerbitkan nilai bilangan bulat pada topik yang disebut /numbers. Salin kode saat ini ke kode baru atau gunakan file yang ada ini dari repositori kode buku ini.

Berikut merupakan demo_topic_publisher.cpp yang saya buat pada folder /sr

```

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;
}
~

```

```

root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg/src# vim demo_topic_subscriber.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;
}
~
~
~
~
~
~
~
~
~
~

```

- Building the nodes

Kita harus mengedit file CMakeLists.txt dalam paket untuk mengkompilasi dan membangun sumber kode. Buka mastering_ros_demo_pkg untuk melihat CMakeLists yang ada.txt arsip. Cuplikan kode berikut dalam file ini bertanggung jawab untuk membangun dua node ini:

```
root@6caed748117d: ~/catkin_ws/mastering_ros_demo_pkg/src$ cd ..
root@6caed748117d: ~/catkin_ws/mastering_ros_demo_pkg$ ls
CMakeLists.txt  include  package.xml  src
root@6caed748117d: ~/catkin_ws/mastering_ros_demo_pkg$ vim CMakeLists.txt |
```

```
## Your package locations should be listed before other locations
include_directories(
# include
  ${catkin_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
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_topic_subscriber ${catkin_LIBRARIES})
## Declare a C++ library
# add_library(${PROJECT_NAME})
#   src/${PROJECT_NAME}/mastering_ros_demo_pkg.cpp
# )
```

Berikut merupakan tampilan ketika kita melakukan run kedua nodes yang sudah kita buat sebelumnya

```
-- Build files have been written to: /root/catkin_ws/build
####
#### Running command: "make -j12 -l12" in "/root/catkin_ws/build"
####
[ 50%] Built target demo_topic_publisher
[100%] Built target demo_topic_subscriber
root@6caed748117d:~/catkin_ws$ roscore
... logging to /root/.ros/log/deb94cc-aa43-11ee-9d9b-0242ac110002
/roslaunch-6caed748117d-1596.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://6caed748117d:33281/
ros_comm version 1.16.0

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

NODES
auto-starting new master
process[master]: started with pid [1600]
ROS_MASTER_URI=http://6caed748117d:11311/

setting /run_id to deb94cc-aa43-11ee-9d9b-0242ac110002
process[rosout-1]: started with pid [1614]
started core service [/rosout]

[ INFO] [1704291945.996651662]: 458
[ INFO] [1704291946.096636201]: 459
[ INFO] [1704291946.196615222]: 460
[ INFO] [1704291946.296620760]: 461
[ INFO] [1704291946.396646430]: 462
[ INFO] [1704291946.496643608]: 463
[ INFO] [1704291946.596656825]: 464
[ INFO] [1704291946.696536297]: 465
[ INFO] [1704291946.796601683]: 466
[ INFO] [1704291946.896535342]: 467
[ INFO] [1704291946.996520317]: 468
[ INFO] [1704291947.096523846]: 469
[ INFO] [1704291947.196558043]: 470
[ INFO] [1704291947.296683469]: 471
[ INFO] [1704291947.396668334]: 472
[ INFO] [1704291947.496647435]: 473
[ INFO] [1704291947.596547051]: 474
[ INFO] [1704291947.696580409]: 475
[ INFO] [1704291947.796538038]: 476
[ INFO] [1704291947.896743146]: 477
[ INFO] [1704291947.996570264]: 478
[ INFO] [1704291948.096522294]: 479
[ INFO] [1704291948.196515680]: 480
[ INFO] [1704291948.296541371]: 481
[ INFO] [1704291948.396705177]: 482
[ INFO] [1704291948.496581577]: 483
[ INFO] [1704291948.596537271]: 484
[ INFO] [1704291948.696558202]: 485
[ INFO] [1704291948.796520810]: 486
[ INFO] [1704291948.896577037]: 487
[ INFO] [1704291948.996626134]: 488
[ INFO] [1704291949.096582815]: 489
[ INFO] [1704291949.196658814]: 490
[ INFO] [1704291949.296590702]: 491
[ INFO] [1704291949.396612231]: 492
[ INFO] [1704291949.496606230]: 493
[ INFO] [1704291949.596616309]: 494
[ INFO] [1704291949.696539409]: 495
[ INFO] [1704291949.796610862]: 496
[ INFO] [1704291949.896629021]: 497
[ INFO] [1704291949.996652219]: 498
[ INFO] [1704291950.096646604]: 499
[ INFO] [1704291950.196730891]: 500
[ INFO] [1704291950.296647086]: 501
[ INFO] [1704291950.396640632]: 502
[ INFO] [1704291950.496624099]: 503
[ INFO] [1704291950.596601661]: 504
[ INFO] [1704291950.696603160]: 505
[ INFO] [1704291950.796549836]: 506
[ INFO] [1704291950.896548453]: 507

[ INFO] [1704291946.197689170]: Received [460]
[ INFO] [1704291946.297508568]: Received [461]
[ INFO] [1704291946.397511160]: Received [462]
[ INFO] [1704291946.497553792]: Received [463]
[ INFO] [1704291946.596811008]: Received [464]
[ INFO] [1704291946.696812253]: Received [465]
[ INFO] [1704291946.796973522]: Received [466]
[ INFO] [1704291946.896776888]: Received [467]
[ INFO] [1704291946.996886070]: Received [468]
[ INFO] [1704291947.096876258]: Received [469]
[ INFO] [1704291947.196799990]: Received [470]
[ INFO] [1704291947.297544869]: Received [471]
[ INFO] [1704291947.396863278]: Received [472]
[ INFO] [1704291947.497559523]: Received [473]
[ INFO] [1704291947.596906734]: Received [474]
[ INFO] [1704291947.696826950]: Received [475]
[ INFO] [1704291947.797097436]: Received [476]
[ INFO] [1704291947.897628182]: Received [477]
[ INFO] [1704291947.996877520]: Received [478]
[ INFO] [1704291948.096762500]: Received [479]
[ INFO] [1704291948.197590258]: Received [480]
[ INFO] [1704291948.296785832]: Received [481]
[ INFO] [1704291948.397642058]: Received [482]
[ INFO] [1704291948.497120452]: Received [483]
[ INFO] [1704291948.596886586]: Received [484]
[ INFO] [1704291948.697107649]: Received [485]
[ INFO] [1704291948.796824910]: Received [486]
[ INFO] [1704291948.897094908]: Received [487]
[ INFO] [1704291948.997096881]: Received [488]
[ INFO] [1704291949.096973764]: Received [489]
[ INFO] [1704291949.197359724]: Received [490]
[ INFO] [1704291949.297215044]: Received [491]
[ INFO] [1704291949.397685491]: Received [492]
[ INFO] [1704291949.497578883]: Received [493]
[ INFO] [1704291949.597325440]: Received [494]
[ INFO] [1704291949.696839248]: Received [495]
[ INFO] [1704291949.797675846]: Received [496]
[ INFO] [1704291949.897085178]: Received [497]
[ INFO] [1704291949.997401250]: Received [498]
[ INFO] [1704291950.097057964]: Received [499]
[ INFO] [1704291950.197698284]: Received [500]
[ INFO] [1704291950.297619517]: Received [501]
[ INFO] [1704291950.39786415]: Received [502]
[ INFO] [1704291950.497688567]: Received [503]
[ INFO] [1704291950.597607524]: Received [504]
[ INFO] [1704291950.697573304]: Received [505]
[ INFO] [1704291950.796813553]: Received [506]
[ INFO] [1704291950.896870427]: Received [507]
```

- Adding custom .msg and .srv files

Langkah pertama adalah mengedit file .xml paket dari paket saat ini dan hapus komentar

Garis `<build_depend>message_generation</build_depend>`
`<exec_depend>message_runtime</exec_depend>`.

dan

```
<!-- <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_depend>roscpp</build_depend>
```

Selanjutnya edit CMakeLists.txt file, seperti gambar di bawah 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
```

Selanjutnya kita lakukan build package

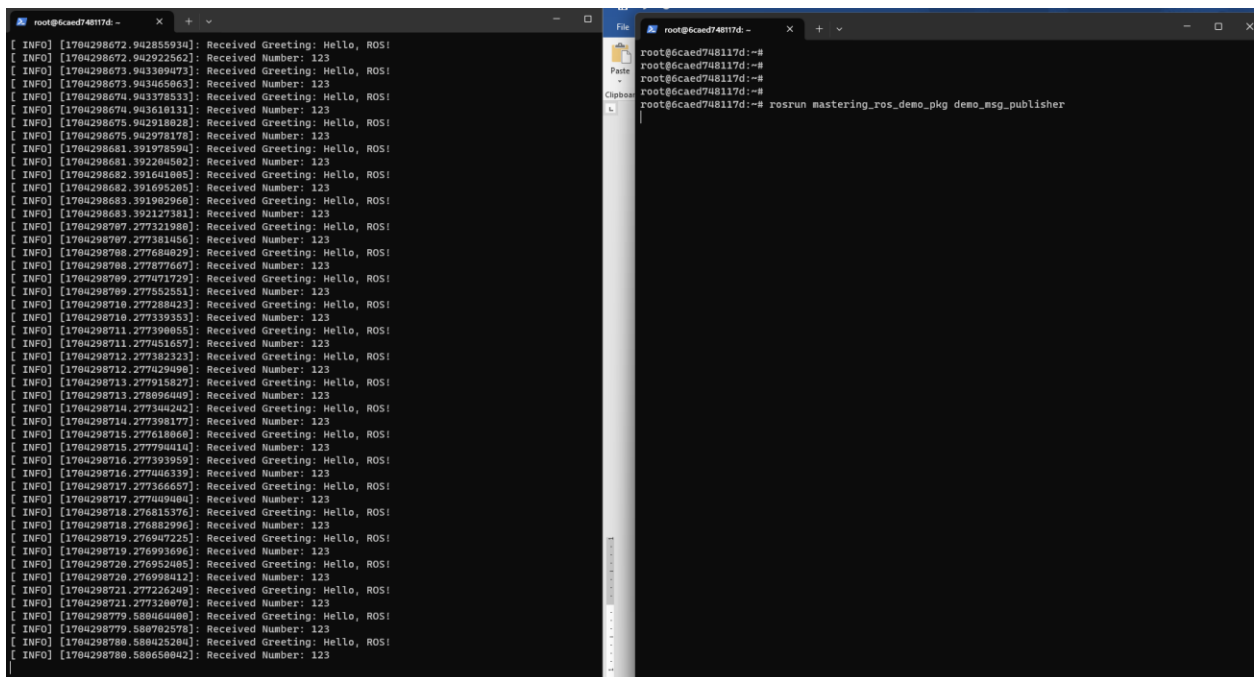
```
-- Using Debian Python package layout
-- Using empy: /usr/lib/python3/dist-packages/em.py
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /root/catkin_ws/build/test_results
-- Forcing gtest/gmock from source, though one was otherwise available.
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Found gmock sources under '/usr/src/gtest': gmock will be built
-- Found PythonInterp: /usr/bin/python3 (found version "3.8.10")
-- Using Python nosetests: /usr/bin/nosetests3
-- catkin 0.8.10
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- traversing 1 packages in topological order:
--   - mastering_ros_demo_pkg
-- +++ processing catkin package: 'mastering_ros_demo_pkg'
-- ==> add_subdirectory(mastering_ros_demo_pkg)
-- Configuring done
-- Generating done
-- Build files have been written to: /root/catkin_ws/build
####
#### Running command: "make -j12 -l12" in "/root/catkin_ws/build"
####
[ 50%] Built target demo_topic_subscriber
[100%] Built target demo_topic_publisher
root@6caed748117d:~/catkin_ws#
```

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

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

root@6caed748117d:~# |
```

Selanjutnya kita jalankan program publisher dan subscriber



```
root@6caed748117d:~# roscat --node /demo_subscriber
[ INFO ] [1704298672.942855934]: Received Greeting: Hello, ROS!
[ INFO ] [1704298672.942922562]: Received Number: 123
[ INFO ] [1704298673.943389473]: Received Greeting: Hello, ROS!
[ INFO ] [1704298673.943465063]: Received Number: 123
[ INFO ] [1704298674.943378533]: Received Greeting: Hello, ROS!
[ INFO ] [1704298674.943610131]: Received Number: 123
[ INFO ] [1704298675.942918028]: Received Greeting: Hello, ROS!
[ INFO ] [1704298675.942978178]: Received Number: 123
[ INFO ] [1704298681.391978598]: Received Greeting: Hello, ROS!
[ INFO ] [1704298681.392204502]: Received Number: 123
[ INFO ] [1704298682.391641085]: Received Greeting: Hello, ROS!
[ INFO ] [1704298682.391695205]: Received Number: 123
[ INFO ] [1704298683.391982968]: Received Greeting: Hello, ROS!
[ INFO ] [1704298683.392177381]: Received Number: 123
[ INFO ] [1704298787.277321988]: Received Greeting: Hello, ROS!
[ INFO ] [1704298787.277381456]: Received Number: 123
[ INFO ] [1704298788.277684029]: Received Greeting: Hello, ROS!
[ INFO ] [1704298788.277877667]: Received Number: 123
[ INFO ] [1704298789.277471729]: Received Greeting: Hello, ROS!
[ INFO ] [1704298789.277532533]: Received Number: 123
[ INFO ] [1704298710.277288423]: Received Greeting: Hello, ROS!
[ INFO ] [1704298710.277339353]: Received Number: 123
[ INFO ] [1704298711.277398055]: Received Greeting: Hello, ROS!
[ INFO ] [1704298712.277451657]: Received Number: 123
[ INFO ] [1704298712.277382323]: Received Greeting: Hello, ROS!
[ INFO ] [1704298712.277429498]: Received Number: 123
[ INFO ] [1704298713.277915827]: Received Greeting: Hello, ROS!
[ INFO ] [1704298713.278096449]: Received Number: 123
[ INFO ] [1704298714.277344242]: Received Greeting: Hello, ROS!
[ INFO ] [1704298714.277398177]: Received Number: 123
[ INFO ] [1704298715.277618868]: Received Greeting: Hello, ROS!
[ INFO ] [1704298715.277794410]: Received Number: 123
[ INFO ] [1704298716.277393959]: Received Greeting: Hello, ROS!
[ INFO ] [1704298716.277446339]: Received Number: 123
[ INFO ] [1704298717.277366657]: Received Greeting: Hello, ROS!
[ INFO ] [1704298717.277409408]: Received Number: 123
[ INFO ] [1704298718.276815376]: Received Greeting: Hello, ROS!
[ INFO ] [1704298718.276882996]: Received Number: 123
[ INFO ] [1704298719.276947225]: Received Greeting: Hello, ROS!
[ INFO ] [1704298719.276993696]: Received Number: 123
[ INFO ] [1704298720.276995408]: Received Greeting: Hello, ROS!
[ INFO ] [1704298720.276998412]: Received Number: 123
[ INFO ] [1704298721.277226249]: Received Greeting: Hello, ROS!
[ INFO ] [1704298721.277320078]: Received Number: 123
[ INFO ] [1704298779.588064408]: Received Greeting: Hello, ROS!
[ INFO ] [1704298779.588070257]: Received Number: 123
[ INFO ] [1704298780.588042508]: Received Greeting: Hello, ROS!
[ INFO ] [1704298780.588058042]: Received Number: 123

root@6caed748117d:~# roslaunch mastering_ros_demo_pkg demo_msg_publisher
root@6caed748117d:~#
```

Selanjutnya, kita dapat menambahkan file `.srv` ke paket. Buat folder baru bernama `srv` di saat ini folder paket dan tambahkan file `.srv` bernama `demo_srv.srv`. Definisi file ini adalah sebagai Berikut:

```
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# rossrv show mastering_ros_demo_pkg/demo_srv
string in
---
string out

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

- Working with ROS services

Pertama tama kita tambahkan file demo_service_server

```
#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
```

Selanjutnya kita tambahkan juga file demo_service_client

```
#include "ros/ros.h"
#include <iostream>
#include "mastering_ros_demo_pkg/demo_srv.h"
#include <sstream>

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_service_client");
    ros::NodeHandle n;
    ros::Rate loop_rate(10);
    ros::ServiceClient client = n.serviceClient<mastering_ros_demo_pkg::demo_srv>("demo_service"); // Perhatikan peng

    std::stringstream ss; // Deklarasi stringstream
    while (ros::ok()) {
        mastering_ros_demo_pkg::demo_srv srv;
        ss << "Sending from Here";
        srv.request.in = ss.str();

        if (client.call(srv)) {
            ROS_INFO("From Client [%s], Server says [%s]", srv.request.in.c_str(), srv.response.out.c_str());
        } else {
            ROS_ERROR("Failed to call service");
            return 1;
        }
    }

    ros::spinOnce();
    loop_rate.sleep();
    ss.str(""); // Bersihkan stringstream setelah penggunaan

    "demo_service_client.cpp" 32L, 980C
```


Setelah semuanya sudah konfigurasi CMakelist.txt dan lakukan catkin_make untuk melakukan build program

```
#####
##### Running command: "make -j12 -l12" in "/root/catkin_ws/build"
#####
[ 16%] Built target demo_topic_publisher
[ 16%] Built target demo_topic_subscriber
[ 16%] Built target _mastering_ros_demo_pkg_generate_messages_check_deps_demo_srv
[ 16%] Built target _mastering_ros_demo_pkg_generate_messages_check_deps_demo_msg
[ 36%] Built target mastering_ros_demo_pkg_generate_messages_py
[ 40%] Built target mastering_ros_demo_pkg_generate_messages_cpp
[ 48%] Built target mastering_ros_demo_pkg_generate_messages_nodejs
[ 60%] Built target mastering_ros_demo_pkg_generate_messages_eus
[ 68%] Built target mastering_ros_demo_pkg_generate_messages_lisp
Scanning dependencies of target demo_service_server
Scanning dependencies of target demo_service_client
[ 68%] Built target mastering_ros_demo_pkg_generate_messages
[ 76%] Built target demo_msg_subscriber
[ 84%] Built target demo_msg_publisher
[ 88%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_service_server.dir/src/demo_service_server.cpp.o
[ 92%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_service_client.dir/src/demo_service_client.cpp.o
[ 96%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_service_client
[ 96%] Built target demo_service_client
[100%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_service_server
[100%] Built target demo_service_server
root@6caed748117d:~/catkin_ws# roscore
... logging to /root/.ros/log/af0fbbb4-aa59-11ee-bc68-0242ac110002/roslaunch-6caed748117d-3671.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.
```

Kemudian jalankan file demo_service_client.cpp dan demo_service_server.cpp, untuk melakukan pertukaran data

```
root@6caed748117d:~/catkin_ws# rosrun mastering_ros_demo_pkg demo_service_client
[ INFO] [1704301290.249108229]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.249353539]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.449352444]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.549601030]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.649218045]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.749603711]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.849138978]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301290.949328979]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.049287515]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.149297301]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.249396645]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.349634710]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.449161370]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.549249059]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.649331858]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.749025741]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.849317998]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301291.948232110]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.049334355]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.148925035]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.247465242]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.348573298]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.446816180]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
[ INFO] [1704301292.546934071]: From Client [Sending from Here], Server says [Response from server:
Server received your message.]
root@6caed748117d:~/catkin_ws# rosrun mastering_ros_demo_pkg demo_service_server
[ INFO] [1704301277.311403951]: Ready to receive client requests.
[ INFO] [1704301288.146334728]: Received request from client with message: Sending from Here
[ INFO] [1704301288.248784020]: Received request from client with message: Sending from Here
[ INFO] [1704301288.348601431]: Received request from client with message: Sending from Here
[ INFO] [1704301288.448599724]: Received request from client with message: Sending from Here
[ INFO] [1704301288.547432588]: Received request from client with message: Sending from Here
[ INFO] [1704301288.648657322]: Received request from client with message: Sending from Here
[ INFO] [1704301288.748628464]: Received request from client with message: Sending from Here
[ INFO] [1704301288.847309882]: Received request from client with message: Sending from Here
[ INFO] [1704301288.948565687]: Received request from client with message: Sending from Here
[ INFO] [1704301289.048523395]: Received request from client with message: Sending from Here
[ INFO] [1704301289.148821502]: Received request from client with message: Sending from Here
[ INFO] [1704301289.248185639]: Received request from client with message: Sending from Here
[ INFO] [1704301289.348433833]: Received request from client with message: Sending from Here
[ INFO] [1704301289.448714999]: Received request from client with message: Sending from Here
[ INFO] [1704301289.548444036]: Received request from client with message: Sending from Here
[ INFO] [1704301289.648618205]: Received request from client with message: Sending from Here
[ INFO] [1704301289.748657331]: Received request from client with message: Sending from Here
[ INFO] [1704301289.848962574]: Received request from client with message: Sending from Here
[ INFO] [1704301289.949013952]: Received request from client with message: Sending from Here
[ INFO] [1704301290.048726951]: Received request from client with message: Sending from Here
[ INFO] [1704301290.148795640]: Received request from client with message: Sending from Here
[ INFO] [1704301290.248460172]: Received request from client with message: Sending from Here
[ INFO] [1704301290.348750472]: Received request from client with message: Sending from Here
[ INFO] [1704301290.448833813]: Received request from client with message: Sending from Here
[ INFO] [1704301290.548965993]: Received request from client with message: Sending from Here
[ INFO] [1704301290.648782671]: Received request from client with message: Sending from Here
[ INFO] [1704301290.748985083]: Received request from client with message: Sending from Here
[ INFO] [1704301290.848805837]: Received request from client with message: Sending from Here
[ INFO] [1704301290.948729906]: Received request from client with message: Sending from Here
[ INFO] [1704301291.048604903]: Received request from client with message: Sending from Here
[ INFO] [1704301291.146363653]: Received request from client with message: Sending from Here
[ INFO] [1704301291.248828056]: Received request from client with message: Sending from Here
[ INFO] [1704301291.346661846]: Received request from client with message: Sending from Here
[ INFO] [1704301291.448662105]: Received request from client with message: Sending from Here
[ INFO] [1704301291.548520783]: Received request from client with message: Sending from Here
[ INFO] [1704301291.648717950]: Received request from client with message: Sending from Here
[ INFO] [1704301291.748464888]: Received request from client with message: Sending from Here
[ INFO] [1704301291.848651106]: Received request from client with message: Sending from Here
[ INFO] [1704301291.947556855]: Received request from client with message: Sending from Here
[ INFO] [1704301292.048726856]: Received request from client with message: Sending from Here
[ INFO] [1704301292.148266197]: Received request from client with message: Sending from Here
[ INFO] [1704301292.247119466]: Received request from client with message: Sending from Here
[ INFO] [1704301292.348103315]: Received request from client with message: Sending from Here
[ INFO] [1704301292.446618937]: Received request from client with message: Sending from Here
[ INFO] [1704301292.546775651]: Received request from client with message: Sending from Here
```

- Working with ROS actionlib

Pertama buat file Demo_action.action pada /action

```
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# |
```

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

Kemudian buat Action Server pada /src

```
#include <ros/ros.h>
#include <actionlib/server/simple_action_server.h>
#include <mastering_ros_demo_pkg/Demo_actionAction.h> // Gantilah 'your_package_name' dengan nama paket Anda

class DemoActionServer {
protected:
    ros::NodeHandle nh_;
    actionlib::SimpleActionServer<mastering_ros_demo_pkg::Demo_actionAction> as_; // Action server
    std::string action_name_;
    mastering_ros_demo_pkg::Demo_actionFeedback feedback_;
    mastering_ros_demo_pkg::Demo_actionResult result_;

public:
    DemoActionServer(const std::string &name) :
        as_(nh_, name, boost::bind(&DemoActionServer::executeCB, this, _1), false),
        action_name_(name) {
        as_.start(); // Memulai action server
    }

    ~DemoActionServer(void) {}

    // Callback function yang akan dijalankan saat action server menerima goal
    void executeCB(const mastering_ros_demo_pkg::Demo_actionGoalConstPtr &goal) {
        ros::Rate r(1); // Timer dengan delay 1 detik
        bool success = true;

        ROS_INFO("%s: Menerima goal %d", action_name_.c_str(), goal->count);

        for (int i = 0; i <= goal->count; i++) {

```

Buat juga action client pada direktori /src

```
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <actionlib/client/terminal_state.h>
#include <mastering_ros_demo_pkg/Demo_actionAction.h> // Gantilah 'your_package_name' dengan nama paket Anda

int main(int argc, char **argv) {
    ros::init(argc, argv, "demo_action_client"); // Inisialisasi node dengan nama 'demo_action_client'

    // Membuat instance dari SimpleActionClient dengan tipe action yang sesuai
    actionlib::SimpleActionClient<mastering_ros_demo_pkg::Demo_actionAction> ac("demo_action", true);

    ROS_INFO("Menunggu server...");
    ac.waitForServer(); // Menunggu hingga action server tersedia

    ROS_INFO("Server ditemukan, mengirim goal...");

    mastering_ros_demo_pkg::Demo_actionGoal goal; // Membuat goal
    goal.count = 10; // Contoh pengaturan nilai goal

    ac.sendGoal(goal); // Mengirim goal ke action server

    // Menunggu hingga goal selesai dengan timeout 30 detik
    bool finished_before_timeout = ac.waitForResult(ros::Duration(30.0));

    if (finished_before_timeout) {
        actionlib::SimpleClientGoalState state = ac.getState();
        ROS_INFO("Selesai: %s", state.toString().c_str());
    } else {
        ROS_INFO("Waktu habis, membatalkan goal...");
    }

    "demo_action_client.cpp" 35L, 1359C
```

4, 26

Setelah menulis kode untuk action server dan action client, pastikan Anda telah memasukkan kedua file ini ke dalam file CMakeLists.txt dan package.xml untuk kompilasi.

Setelah kompilasi selesai, Anda dapat menjalankan action server dan action client. Anda akan melihat bagaimana action client mengirimkan goal ke action server, dan action server memberikan feedback serta hasil kepada action client

- Building the ROS action server and client

Setelah action server dan client telah dibuat pada direktori src selanjutnya kita melakukan konfigurasi CMakeList.txt dan lakukan `catkin_make`

```

Scanning dependencies of target demo_msg_publisher
Scanning dependencies of target demo_msg_subscriber
Scanning dependencies of target demo_service_server
[ 79%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_action_server.dir/src/demo_action_ser
[ 82%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_service_server.dir/src/demo_service_s
[ 82%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_msg_subscriber.dir/src/demo_msg_subsc
[ 84%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_msg_publisher.dir/src/demo_msg_publis
Scanning dependencies of target demo_service_client
[ 85%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_action_client.dir/src/demo_action_cli
[ 87%] Building CXX object mastering_ros_demo_pkg/CMakeFiles/demo_service_client.dir/src/demo_service_c
[ 90%] Generating Python msg __init__.py for mastering_ros_demo_pkg
[ 90%] Generating Python srv __init__.py for mastering_ros_demo_pkg
[ 90%] Built target mastering_ros_demo_pkg_generate_messages_eus
[ 90%] Built target mastering_ros_demo_pkg_generate_messages_py
[ 90%] Built target mastering_ros_demo_pkg_generate_messages
[ 92%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_service_client
[ 93%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_msg_publisher
[ 93%] Built target demo_service_client
[ 93%] Built target demo_msg_publisher
[ 95%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_service_server
[ 96%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_msg_subscriber
[ 96%] Built target demo_service_server
[ 96%] Built target demo_msg_subscriber
[ 98%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_action_server
[ 98%] Built target demo_action_server
[100%] Linking CXX executable /root/catkin_ws/devel/lib/mastering_ros_demo_pkg/demo_action_client
[100%] Built target demo_action_client
root@6caed748117d:~/catkin_ws#

```

Setelah itu lakukan running pada 2 terminal untuk server dan client

```

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

C:\Users\byfish>docker exec -it 6caed748117d /bin/bash
root@6caed748117d:/# roslaunch mastering_ros_demo_pkg demo_action_client 10
[ INFO] [1704363080.556879753]: Menunggu server...
[ INFO] [1704363080.796819085]: Server ditemukan, mengirim goal...
[ INFO] [1704363091.799573995]: Selesai: SUCCEEDED
root@6caed748117d:/#

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

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

```

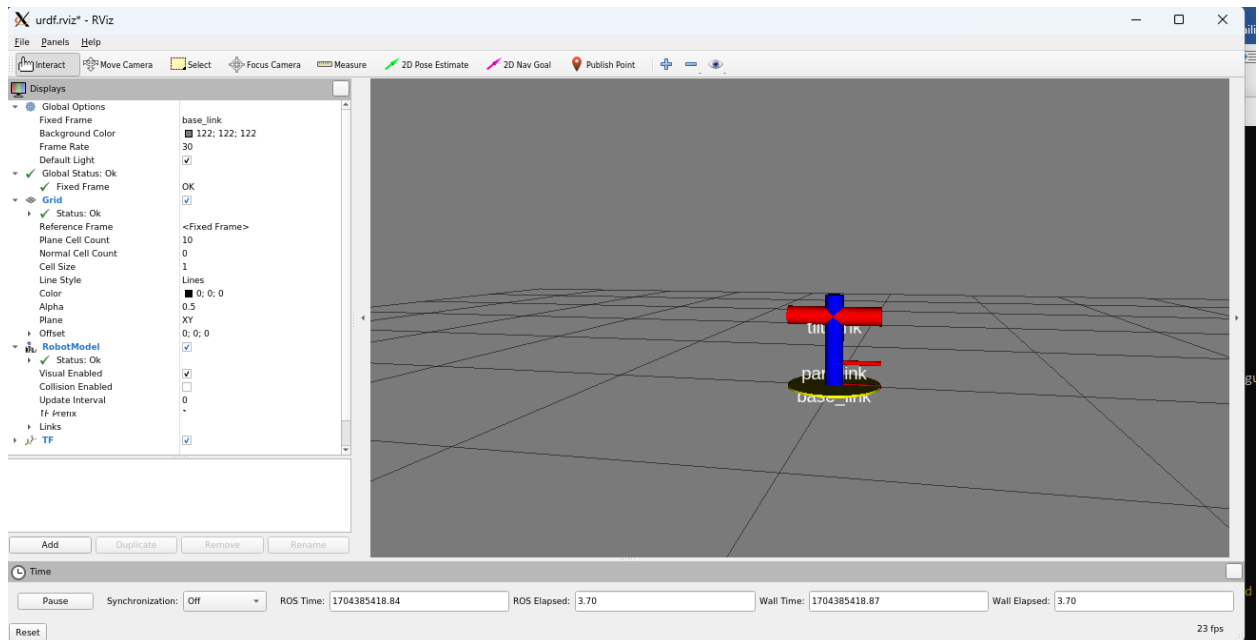
3. Working with ROS for 3D Modeling

- Creating the ROS package for the robot

Hal yang harus dilakukan pertama tama adalah melakukan gitclone dari file yang sudah disediakan oleh pemilik buku, berikut merupakan tampilan setelah git clone

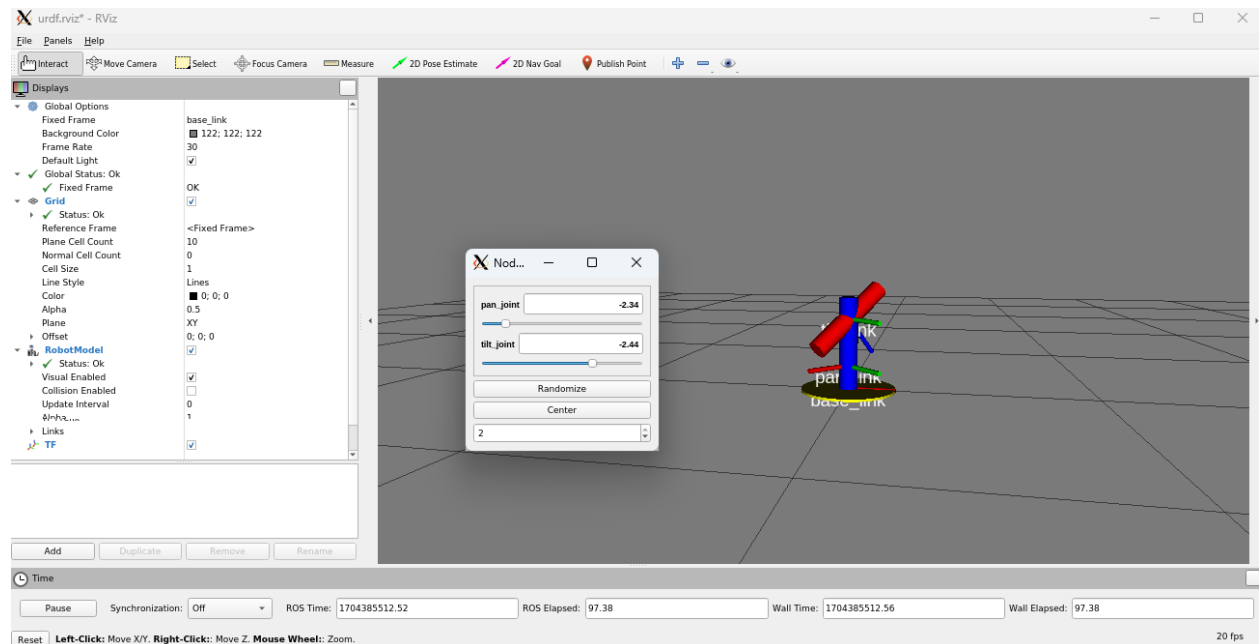
```
root@024a5ddf0613:~/catkin_ws# cd src/
root@024a5ddf0613:~/catkin_ws/src# ls
CMakeLists.txt  mastering_ros_robot_description_pkg
root@024a5ddf0613:~/catkin_ws/src# cd mastering_ros_robot_description_pkg/
root@024a5ddf0613:~/catkin_ws/src/mastering_ros_robot_description_pkg# ls -la
total 40
drwxr-xr-x 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#
```

Kemudian jalankan robot yang sudah ada di dalam file diatas



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 menggerakkan joint dengan bantuan GUI seperti gambar dibawah



- Converting xacro to URDF

Kita dapat melihat xacro dari robot pan-and-tilt dengan membuat file peluncuran, dan itu bisa diluncurkan menggunakan perintah seperti gambar dibawah ini

```
roslaunch mastering_ros_robot_description_pkg  
view_pan_tilt_xacro.launch
```

```
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://024a5ddf0613:41661/  
  
SUMMARY  
=====  
  
PARAMETERS  
* /robot_description: <?xml version="1...  
* /rostdistro: noetic  
* /rosversion: 1.16.0  
  
NODES  
/  
  joint_state_publisher_gui (joint_state_publisher_gui/joint_state_publisher_gui)  
  robot_state_publisher (robot_state_publisher/robot_state_publisher)  
  rviz (rviz/rviz)  
  
auto-starting new master  
process[master]: started with pid [11532]  
ROS_MASTER_URI=http://localhost:11311  
  
setting /run_id to 7daa9bd2-ab1e-11ee-90eb-0242ac110002  
process[rosout-1]: started with pid [11542]  
started core service [/rosout]  
process[robot_state_publisher-2]: started with pid [11545]  
process[joint_state_publisher_gui-3]: started with pid [11546]  
process[rviz-4]: started with pid [11551]  
[ WARN ] [1704385794.493022777]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an extra dummy link to your URDF.  
qt.qpa.xcb: X server does not support XInput 2  
failed to get the current screen resources  
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'  
qt.qpa.xcb: QXcbConnection: XCB error: 1 (BadRequest), sequence: 164, resource id: 90, major code: 130 (Unknown), minor code: 47  
qt.qpa.xcb: QXcbConnection: XCB error: 170 (Unknown), sequence: 177, resource id: 90, major code: 146 (Unknown), minor code: 20  
qt.qpa.xcb: X server does not support XInput 2  
failed to get the current screen resources  
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'  
qt.qpa.xcb: QXcbConnection: XCB error: 1 (BadRequest), sequence: 164, resource id: 90, major code: 130 (Unknown), minor code: 47  
qt.qpa.xcb: QXcbConnection: XCB error: 170 (Unknown), sequence: 177, resource id: 90, major code: 146 (Unknown), minor code: 20
```

- Creating the robot description for a seven-DOF robot manipulator

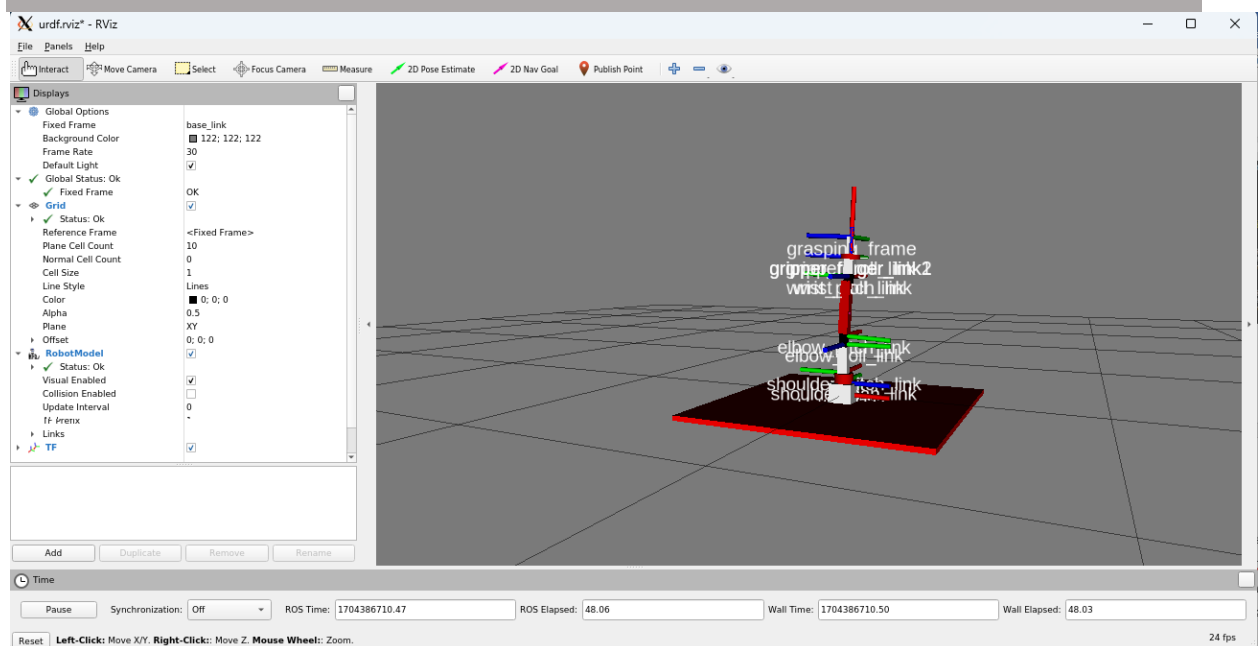
Kita akan memuat robot yang sedikit kompleks menggunakan URDF dan xacro. Robot pertama yang akan kita buat adalah seven-DOF arm yang redundansinya kinematik. Yang berarti robot ini memiliki lebih banyak persendian dan DOF yang lebih mudah untuk mendapatkan goal position dan orientasi.

Berikut merupakan spesifikasi dari Arm yang dimiliki seven DOF arm

- Degrees of freedom: 7
- Length of the arm: 50 cm
- Reach of the arm: 35 cm
- Number of links: 12
- Number of joints: 11

Dikarenakan kita sudah melakukan gitclone, untuk menjalankan robot ini kita tinggal menuliskan

```
roslaunch mastering_ros_robot_description_pkg view_arm.launch
```



saya mendapati error saat menjalankan perintah ini dan saya bingung bagaimana cara fix nya

- Creating a robot model for the differential drive mobile robot

robot roda differential (roda diferensial), sebuah jenis robot yang memiliki dua roda yang terhubung pada sisi-sisi yang berlawanan dari kerangka robot. Robot ini juga didukung oleh satu atau dua roda penumpu yang disebut roda caster.

Roda Differential: Robot ini memiliki dua roda yang diletakkan di sisi berlawanan dari kerangka robot. Kedua roda ini biasanya dioperasikan secara independen. Jika kedua motor yang menggerakkan roda tersebut berjalan dengan kecepatan yang sama, maka robot akan bergerak maju atau mundur. Namun,

jika satu roda bergerak lebih lambat dari yang lain, robot akan berbelok ke arah roda yang bergerak lebih lambat. Misalnya, untuk berbelok ke kiri, kecepatan roda kiri akan dikurangi.

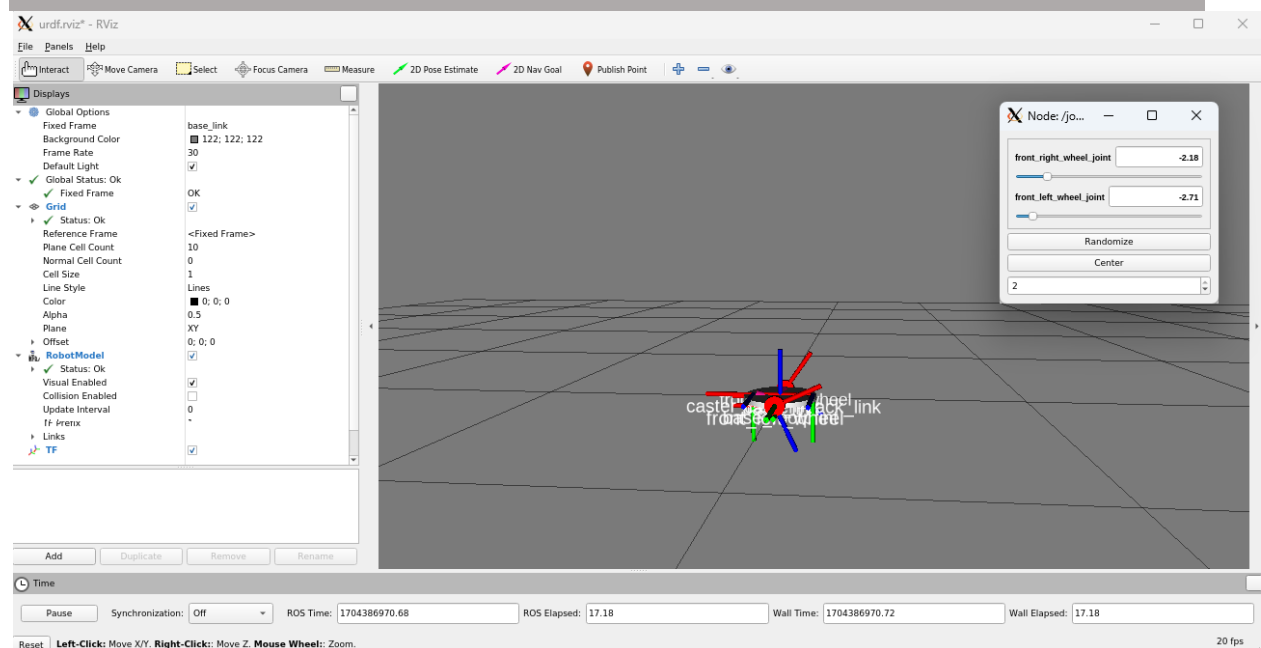
Kontrol Kecepatan: Kecepatan dan arah robot dikontrol dengan mengatur kecepatan masing-masing roda. Jika ingin robot bergerak ke arah tertentu atau berbelok, Anda dapat mengatur kecepatan roda satu sisi lebih lambat dari sisi lainnya.

Roda Caster: Selain dua roda utama, robot juga didukung oleh roda caster. Roda caster ini dirancang untuk berputar dengan bebas dan berfungsi sebagai penopang untuk menjaga keseimbangan robot. Roda caster akan mengikuti gerakan roda utama, memungkinkan robot untuk bergerak dengan lebih lancar dan stabil.

Secara keseluruhan, robot roda differential adalah desain yang efisien dan sederhana untuk mengontrol arah dan kecepatan robot dengan mengatur kecepatan roda yang berbeda di sisi berlawanan. Konsep ini sering digunakan dalam robot mobil otonom dan aplikasi robotik lainnya karena keandalan dan kemudahan pengendaliannya.

Berikut merupakan contoh robotnya, kita dapat run dengan mengetikan;

```
roslaunch mastering_ros_robot_description_pkg  
view_mobile_robot.launch
```



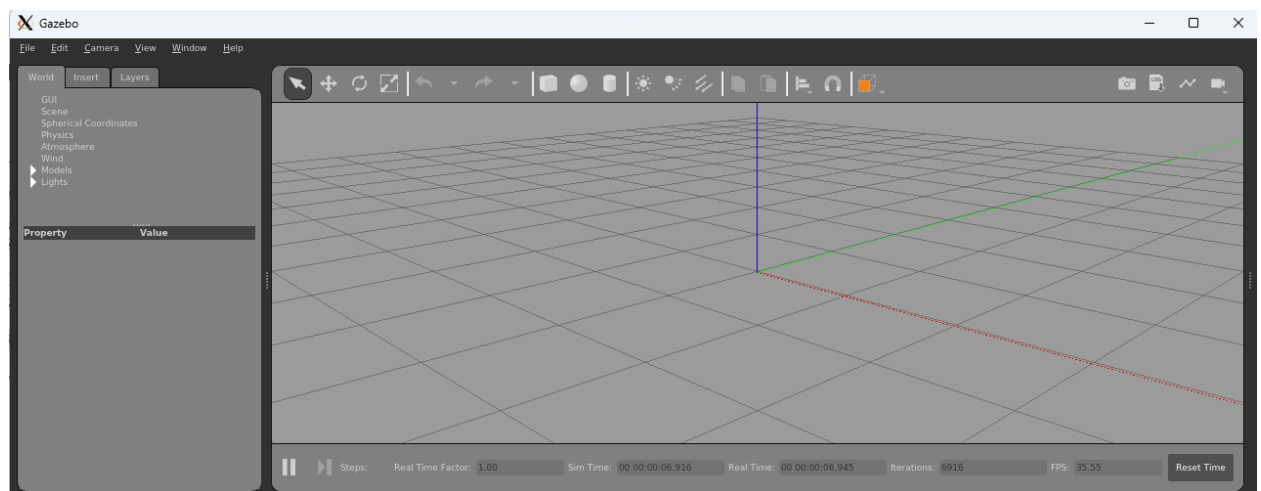
4. Simulating Robots Using ROS and Gazebo

- Simulating the robotic arm using Gazebo and ROS

Install gazebo dengan perintah yang disediakan pada ebook

```
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 libzvt0: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 libpcr2-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 instalasi selesai, cek apakah Gazebo terinstall dengan baik

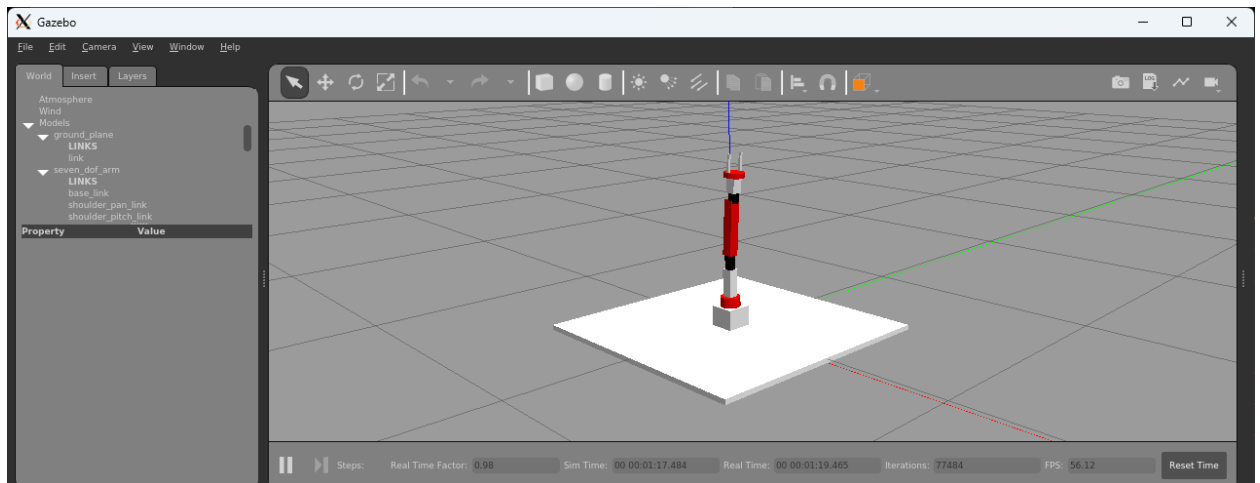


- Creating the robotic arm simulation model for Gazebo

Kita dapat membuat model simulasi untuk lengan robot dengan memperbarui robot yang ada deskripsi dengan menambahkan parameter simulasi. Kita dapat membuat paket yang diperlukan untuk mensimulasikan lengan robot menggunakan perintah berikut:

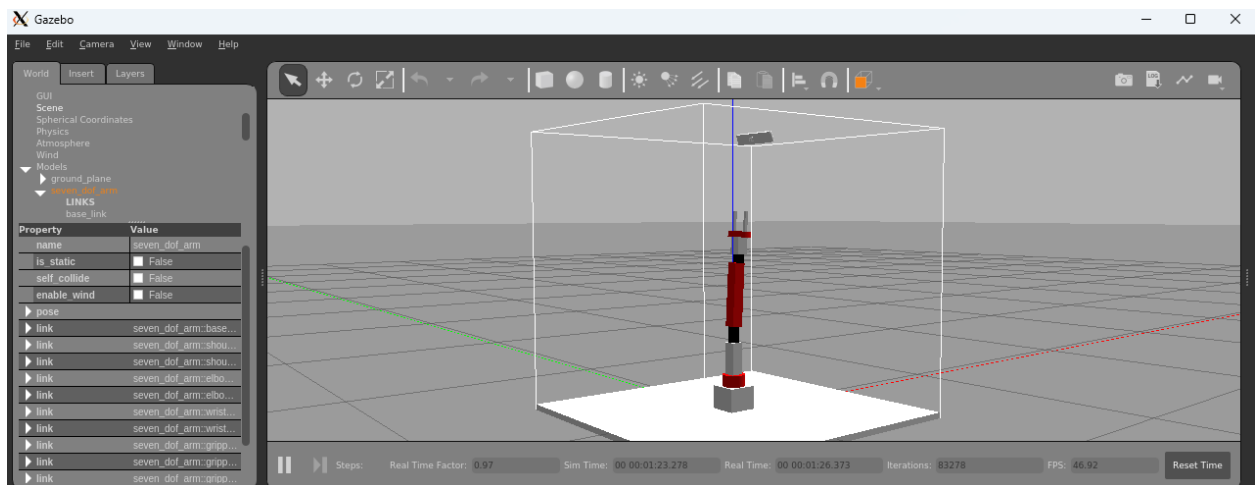
```
catkin_create_pkg seven_dof_arm_gazebo gazebo_msgs gazebo_plugins
gazebo_ros gazebo_ros_control mastering_ros_robot_description_pkg
```

Kemudian, karena kita sudah melakukan gitclone sebelumnya kita tinggal melakukan running pada file robot



- Simulating the robotic arm with Xtion Pro

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



File peluncuran memulai simulasi Gazebo lengan, memuat konfigurasi pengontrol, Muat pengontrol status bersama dan pengontrol posisi bersama, dan, akhirnya, jalankan robot penerbit negara, yang menerbitkan negara bersama dan transformasi (TF). Mari kita periksa topik pengontrol yang dihasilkan setelah menjalankan file peluncuran ini:

```

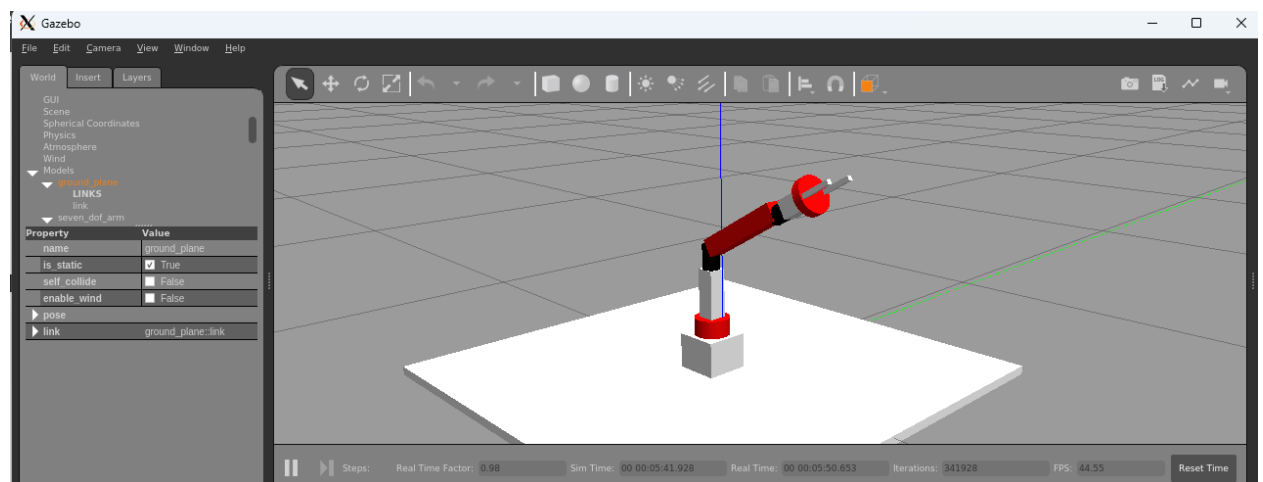
[INFO] [1704389771.071951, 0.395000000]: Loaded gazebo_ros_control.
[INFO] [1704389771.078084, 0.396000]: wait_for_service(/seven_dof_arm/controller_manager/load_controller)
able to contact [rospipe://024a5ddf0613:43219]
[INFO] [1704389771.079262, 0.397000]: Controller Spawner: Waiting for service controller_manager/switch_
[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

```

- Moving the robot joints

Setelah menyelesaikan topik sebelumnya, kita dapat mulai memerintahkan setiap sendi ke Posisi yang kita inginkan. Untuk memindahkan sendi robot di Gazebo, kita harus mempublikasikan nilai bersama yang diinginkan dengan ketik pesan `std_msgs/Float64` ke topik perintah pengontrol posisi gabungan.

Berikut adalah contoh memindahkan sendi keempat ke radian 1,0:



We can also view the joint states of the robot by using the following command:

```

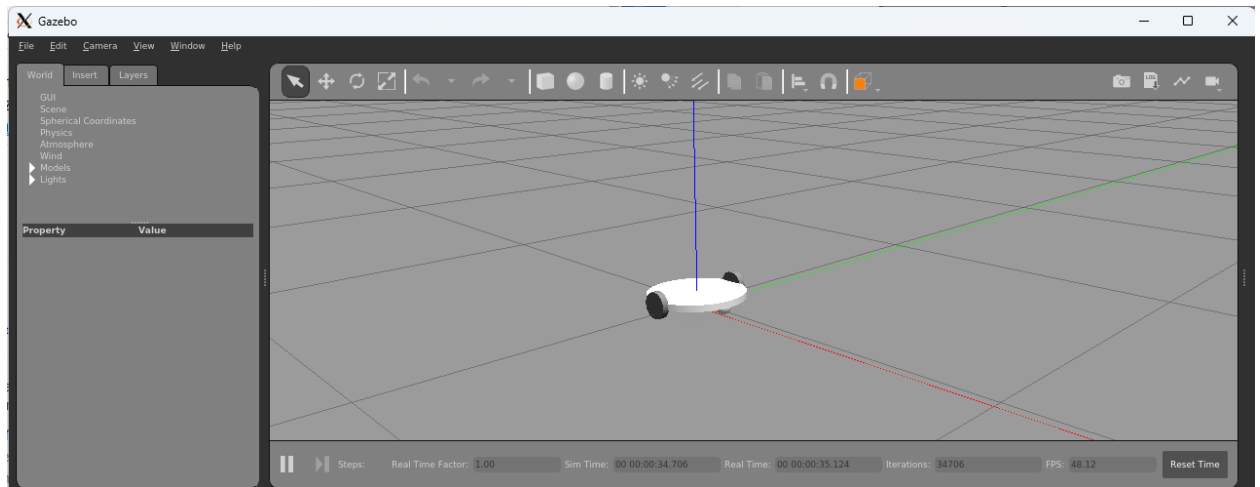
Publishing and latching message. Press ctrl-C to terminate
Crooot@024a5ddf0613:~/catkin_ws# rostopic echo /seven_dof_arm/joint_states
ARNING: no messages received and simulated time is active.
s /clock being published?

```

Now we can control all the joints of the seven DOF arm and, at the same time, we can read their values. In this way, we can implement custom robot control algorithms. In the next section, we will learn how to simulate the differential-drive robot.

- Adding ROS Teleop Node

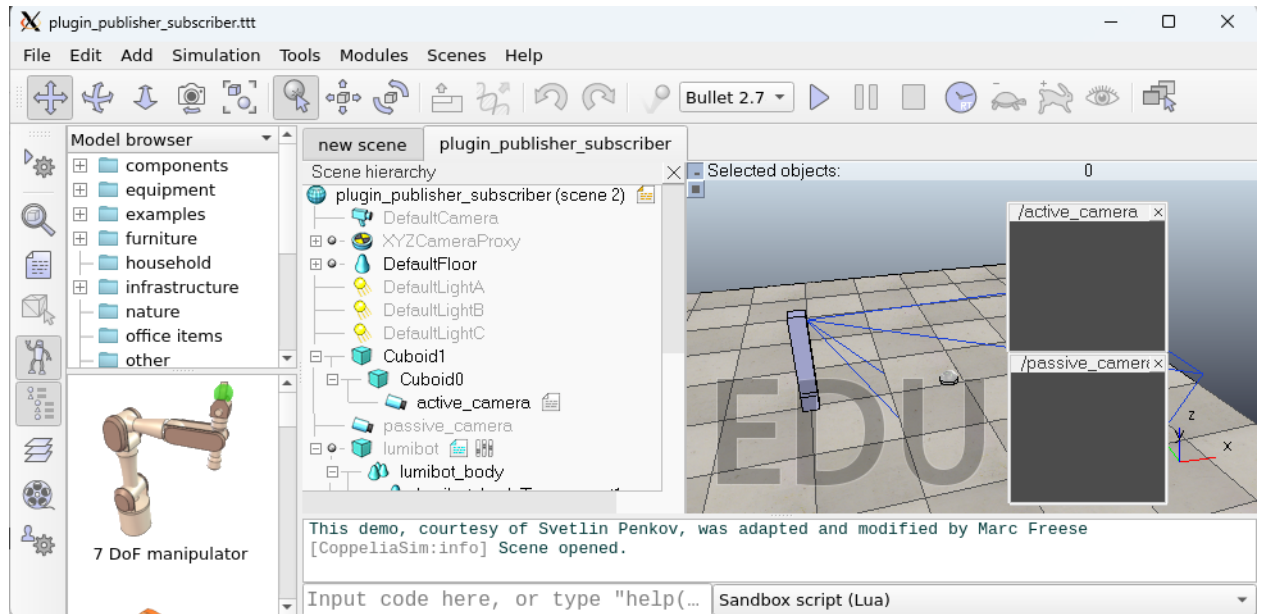
Anda akan melihat model robot berikut di Gazebo. Jika Anda mendapatkan model ini, Anda memiliki berhasil menyelesaikan tahap pertama simulasi:



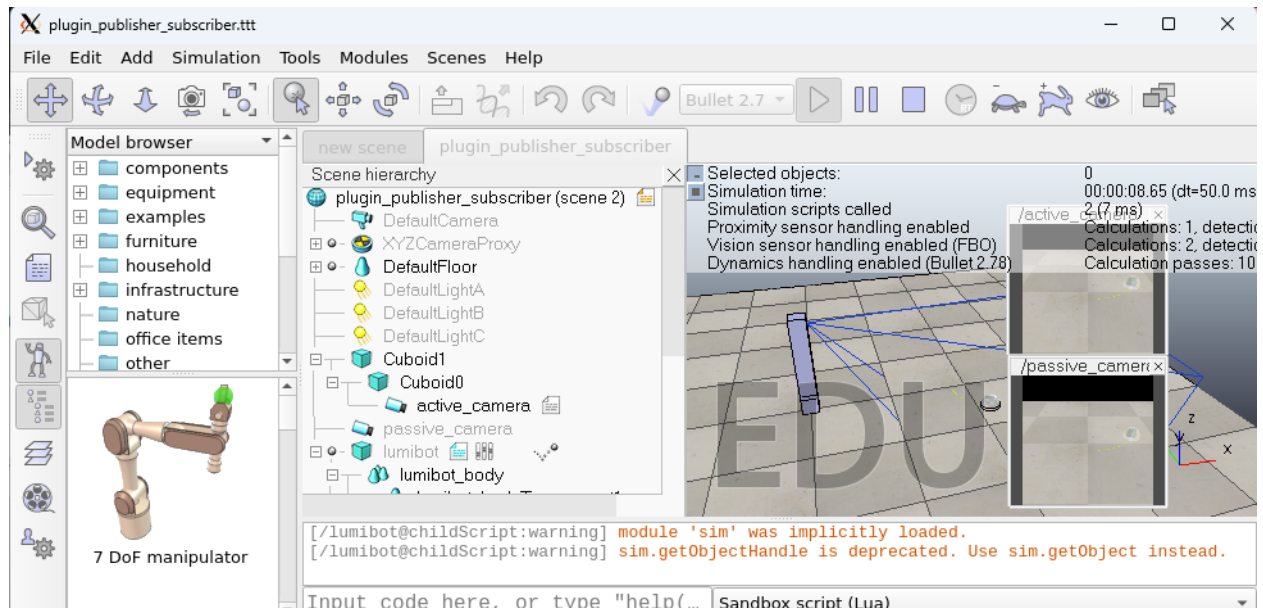
5. Simulating Robots Using ROS , Coppeliasim, and Webots

- Setting up Coppeliasim with ROS

Sebelumnya saya sudah melakukan download Coppeliasim pada docker saya dan saya sudah mengekstraknya menggunakan tar xfv, kemudian kita buka coppeliasim dan jalankan plugin_publisher_subscriber.ttt



Setelah kita lakukan running tampilan akan berubah seperti ini

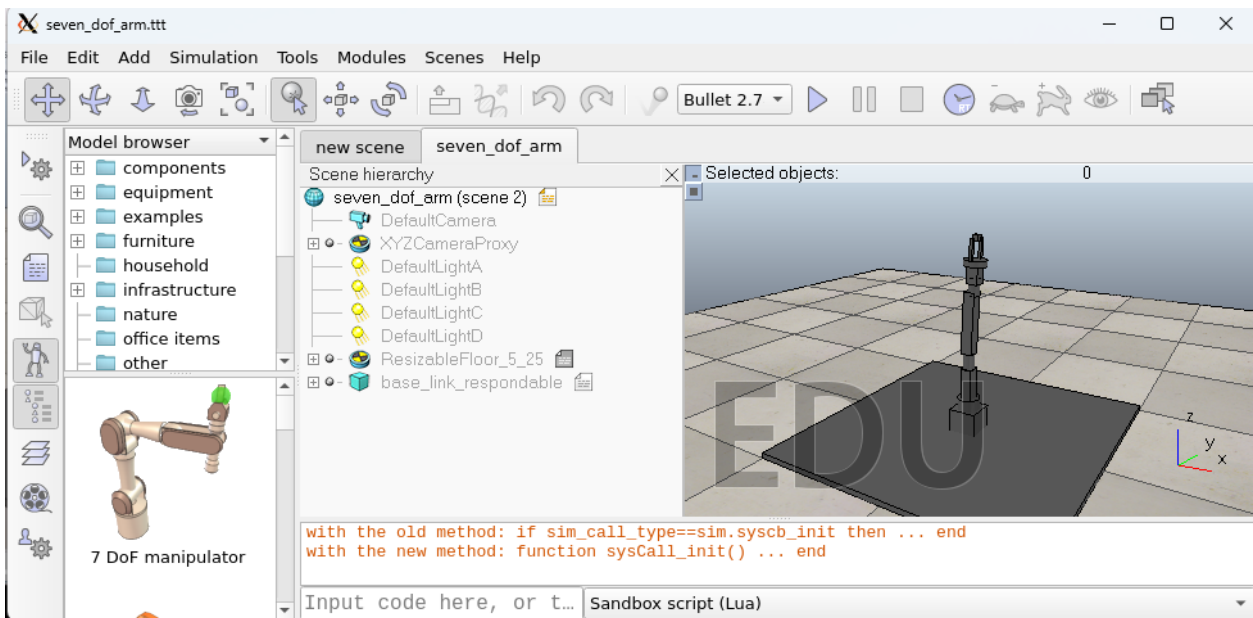


Kita juga bisa melakukan `rqt_image_view` untuk melihat tampilan kamera dari robot

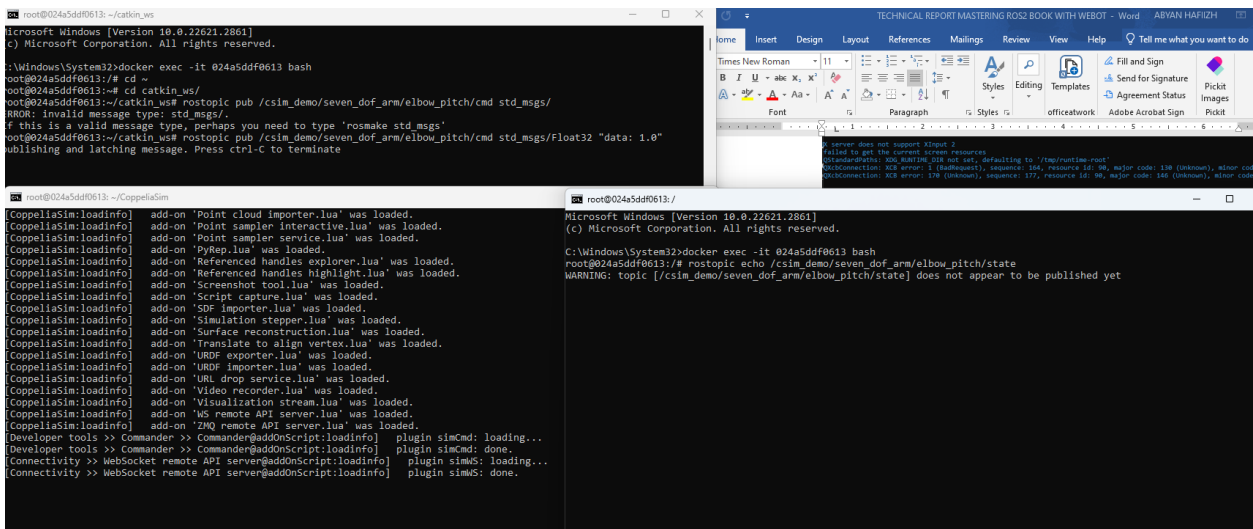
```
root@024a5ddf0613:~/catkin_ws# rqt_image_view
X server does not support XInput 2
failed to get the current screen resources
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
QXcbConnection: XCB error: 1 (BadRequest), sequence: 164, resource id: 90, major code: 130 (Unknown), minor code: 47
QXcbConnection: XCB error: 170 (Unknown), sequence: 177, resource id: 90, major code: 146 (Unknown), minor code: 20
```

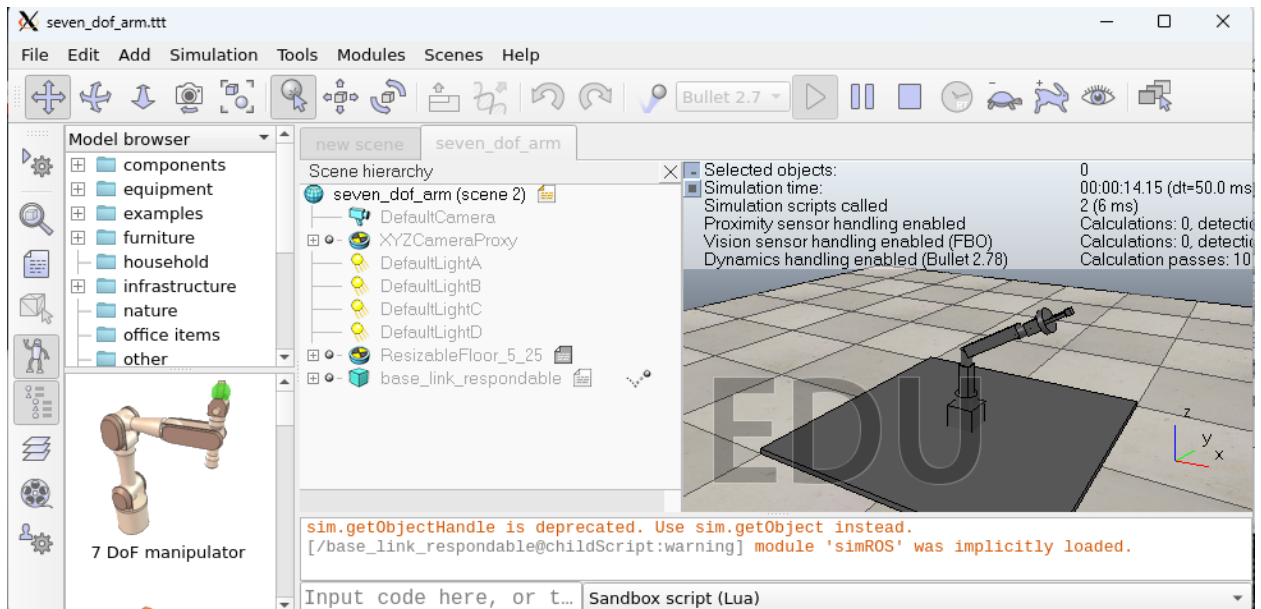
- Simulating a robotic arm using Coppeliasim and ROS

Kita akan melakukan simulasi pada seven doff arm, berikut merupakan tampak dari seven doff arm



Pada gambar dibawah ini saya mencoba melakukan publish message dan menangkapnya dengan terminal satunya, topic ini bertujuan untuk menggerakkan lengan robot





- Setting Up Webots with ROS

Berikut merupakan instruksi cara membuat sebuah simulasi scene dari awal menggunakan perangkat webots.

- Tujuan:

Membuat sebuah scene simulasi dari awal yang akan berisi objek-objek dan sebuah robot beroda yang dapat bergerak.

- Membuat Dunia Baru (World):

Untuk memulai, kita perlu menciptakan sebuah dunia (world) baru untuk simulasi ini.

Cara termudah untuk melakukannya adalah menggunakan opsi "wizard" atau penyihir yang disediakan oleh Webots untuk membantu pembuatan dunia baru.

- Paket Sumber Kode:

Dunia ini sudah tersedia dalam sumber kode buku dalam paket bernama webots_demo_pkg.

- Langkah-langkah Membuat Simulasi Baru:

Menu Utama: Buka menu utama di Webots, dan pilih opsi Wizards (Penyihir).

Buat Direktori Proyek Baru: Di dalam penyihir, pilih New Project Directory (Direktori Proyek Baru). Sebuah jendela atau applet akan muncul untuk membantu Anda dalam proses ini.

Langkah Selanjutnya: Klik Next (Selanjutnya) untuk memilih direktori proyek yang baru. Pada tahap ini, Anda harus menentukan path atau lokasi folder di mana proyek Anda akan disimpan.

Nama Direktori: Di bagian ini, masukkan path folder yang Anda inginkan dan beri nama folder tersebut sebagai `mobile_robot`.

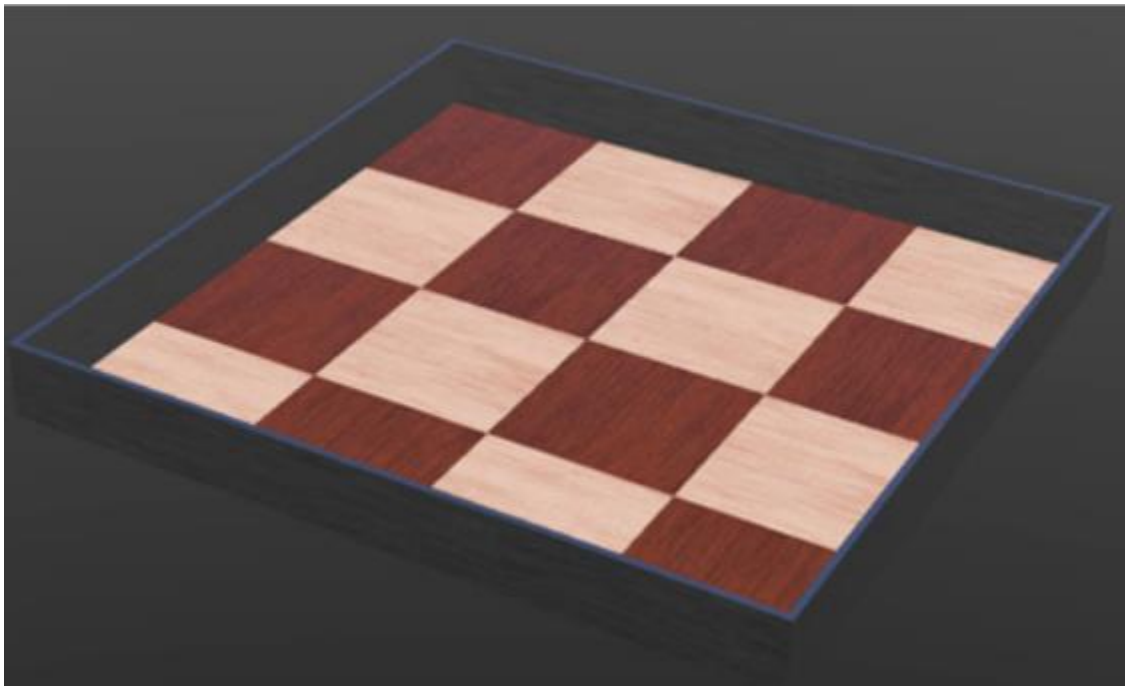
Nama Dunia: Selanjutnya, Anda akan diminta untuk memberi nama pada dunia yang akan Anda ciptakan. Masukkan nama `robot_motion_controller.wbt`.

Tambahan: Pastikan Anda mencentang opsi yang menyatakan "Add a rectangle area" (Tambahkan area berbentuk persegi panjang).

Selesai: Setelah semua pengaturan telah dilakukan, klik Finish (Selesai).

Tunggu dan Lihat Hasil: Setelah proses selesai, dunia simulasi yang baru Anda buat akan dimuat. Jika semua berjalan dengan benar, Anda seharusnya dapat melihat scene simulasi Anda, yang sesuai dengan tampilan yang ditunjukkan dalam screenshot yang disebutkan.

Jadi, teks di atas memberikan panduan langkah demi langkah tentang cara menciptakan dunia simulasi baru menggunakan Webots, mulai dari pengaturan awal hingga melihat hasil akhirnya.




```

/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_p
uck_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.

Berikut merupakan isi dari epuck manager

```

static float left_vel = 0.0;
static float right_vel = 0.0;
root@024sdd6013:~/catkin_ws/src/webots_demo_pkg/src#
static float wheel_radius = 0.205;
static float axes_length = 0.52;

void cmdVelCallback(const geometry_msgs::Twist::ConstPtr &vel) {
    left_vel = ( 1/wheel_radius)*(vel->linear.x-axes_length/2*vel->angular.z);
    right_vel = ( 1/wheel_radius)*(vel->linear.x+axes_length/2*vel->angular.z);
}

void modelNameCallback(const std_msgs::String::ConstPtr &name) {
    cnt++;
    strcpy(modelList[cnt], name->data.c_str());
    ROS_INFO("Model #%d: %s.", cnt, name->data.c_str());
}

int main(int argc, char** argv ) {
    ros::init(argc, argv, "e_puck_manager");
    ros::NodeHandle n;
    std::string modelName;

    // get the name of the robot
    ros::Subscriber nameSub = n.subscribe("model_name", 100, modelNameCallback);
    while (cnt == 0 ) {
        ros::spinOnce();
    }
    modelName = modelList[1];

    ros::Subscriber cmdVelSub = n.subscribe("cmd_vel", 1, cmdVelCallback);

    // set the motors to velocity control
    webots_ros::set_float wheelSrv;
    wheelSrv.request.value = INFINITY;
    ros::ServiceClient leftWheelPositionClient =
        n.serviceClient<webots_ros::set_float>(modelName + "/left_wheel_motor/set_position");
    leftWheelPositionClient.call(wheelSrv);
    ros::ServiceClient rightWheelPositionClient =
        n.serviceClient<webots_ros::set_float>(modelName + "/right_wheel_motor/set_position");
    rightWheelPositionClient.call(wheelSrv);

    wheelSrv.request.value = 0.0;
    ros::ServiceClient leftWheelVelocityClient =
        n.serviceClient<webots_ros::set_float>(modelName + "/left_wheel_motor/set_velocity");
    leftWheelVelocityClient.call(wheelSrv);
    ros::ServiceClient rightWheelVelocityClient =
        n.serviceClient<webots_ros::set_float>(modelName + "/right_wheel_motor/set_velocity");
    rightWheelVelocityClient.call(wheelSrv);

    ros::Rate r(10);
    while(ros::ok()) {
        wheelSrv.request.value = left_vel;

```

[illegible]

Berdasarkan pengerjaan UAS yang melibatkan praktek pembuatan robot dengan mengikuti langkah-langkah yang disajikan dalam buku "Mastering ROS", dapat disimpulkan bahwa pendekatan yang diadopsi dari buku tersebut memberikan fondasi yang kuat bagi pengembangan robotika. Langkah-langkah yang terstruktur dan didokumentasikan dengan baik dalam buku tersebut memfasilitasi proses pembuatan robot dengan efisiensi dan konsistensi. Selama pengerjaan UAS, penerapan prinsip-prinsip ROS seperti modularitas, integrasi komponen, dan pengujian simulasi dengan alat seperti Gazebo memberikan wawasan mendalam tentang kompleksitas dan kemungkinan yang ada dalam dunia robotika. Selain itu, metode yang diajarkan dalam buku ini memperkuat pemahaman tentang bagaimana mengoptimalkan fungsi, kinerja, dan interoperabilitas sistem robotika. Keseluruhan pengalaman ini tidak hanya memperkaya pengetahuan teoritis, tetapi juga memberikan keterampilan praktis yang esensial untuk menghadapi tantangan dalam pengembangan robotika di masa depan. Sebagai hasilnya, pengerjaan UAS ini membuktikan bahwa mengikuti pedoman dan best practices yang telah ditetapkan dalam "Mastering ROS" adalah pendekatan yang efektif dan relevan dalam mengembangkan solusi robotika yang inovatif dan berkinerja tinggi.

Referensi:

- [1] Joseph. L,& Cacace. J. (2021). *Mastering ROS for Robotics Programming Third Edition*. Packt Publishing