

## **TECHNICAL REPORT UAS ROBOTIKA**

**ABYAN HAFIIZH – 1103202245 – TK44G7**

Pada technical report ini saya akan memaparkan tentang robotika, khususnya ROS, 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.

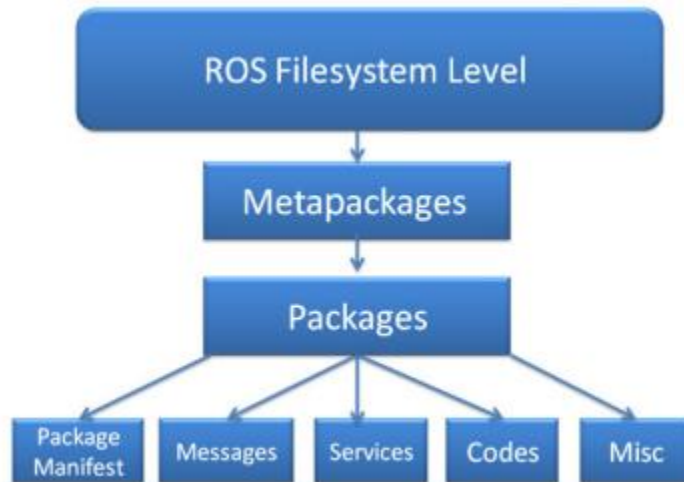
# **1. Introduction to ROS**

Dua bab pertama dari buku ini akan memperkenalkan konsep dasar ROS dan sistem manajemen paket ROS untuk mendekati pemrograman ROS. Dalam bab pertama ini, kita akan mengulas konsep-konsep ROS seperti ROS master, ROS nodes, ROS parameter server, dan pesan serta layanan ROS, sambil membahas apa yang kita perlukan untuk menginstal ROS dan bagaimana memulai dengan ROS master. Dalam bab ini, kita akan membahas topik-topik berikut:

- Mengapa kita perlu mempelajari ROS?
- Memahami level sistem file ROS.
- Memahami level grafik komputasi ROS.
- Tingkat komunitas ROS.

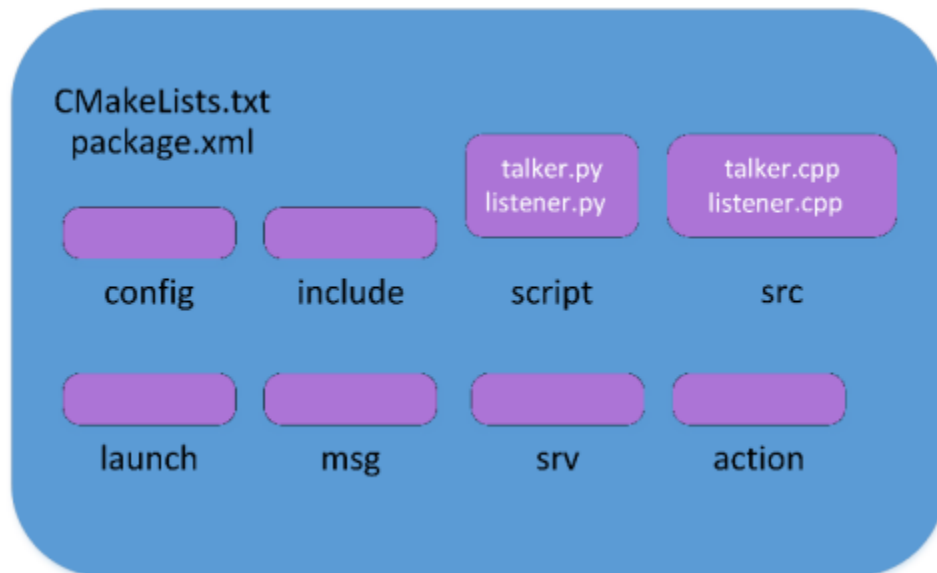
## Memahami Level Sistem File ROS

ROS lebih dari sekadar kerangka pengembangan. Kita dapat merujuk pada ROS sebagai meta-OS, karena tidak hanya menyediakan alat dan pustaka, tetapi juga fungsi mirip OS, seperti abstraksi perangkat keras, manajemen paket, dan alat pengembangan. Seperti sistem operasi nyata, file-file ROS diorganisasikan di hard disk dengan cara tertentu, seperti yang digambarkan dalam diagram berikut:



## ROS packages

Berikut adalah tipe tipe struktur ROS:



## Metapaket ROS

Metapaket adalah paket khusus yang hanya memerlukan satu file; yaitu file package.xml. Metapaket secara sederhana mengelompokkan serangkaian paket menjadi satu paket logis. Dalam file package.xml, metapaket mengandung tag ekspor, seperti yang ditunjukkan di bawah ini:

```
<export>  
  <metapackage/>  
</export>
```

Selain itu, dalam metapaket, tidak ada ketergantungan <buildtool\_depend> untuk catkin; hanya ada ketergantungan <run\_depend>, yang merupakan paket-paket yang dikelompokkan di dalam metapaket tersebut.

## 2. Getting Started with ROS Programming

Setelah membahas dasar-dasar ROS master, parameter server, dan roscore, kita sekarang dapat mulai membuat dan membangun paket ROS. Dalam bab ini, kita akan menciptakan berbagai node ROS dengan menerapkan sistem komunikasi ROS. Saat bekerja dengan paket ROS, kita juga akan menyegarkan pemahaman kita tentang konsep dasar node ROS, topik, pesan, layanan, dan actionlib.

Dalam bab ini, kita akan mengcover topik berikut:

- Membuat paket ROS
- Menambahkan file pesan dan layanan kustom
- Bekerja dengan layanan ROS
- Membuat file peluncuran (launch files)
- Aplikasi dari topik, layanan, dan actionlib

### Creating a ROS package

Persyaratan pertama untuk bekerja dengan paket ROS adalah membuat ruang kerja catkin ROS. Setelah menginstal ROS, kita dapat membuat dan membangun ruang kerja catkin yang disebut catkin\_ws:

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

Untuk mengkompilasi ruang kerja ini, kita harus menginisialisasi lingkungan ROS untuk mendapatkan akses ke fungsi-fungsi ROS:

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

Beralih ke direktori src yang kita buat sebelumnya:

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

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 dibawah merupakan hasil running perintah diatas;

```
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.
```

```

[ 0%] Built target _webots_ros_generate_messages_check_deps_field_set_float
[ 0%] Built target std_msgs_generate_messages_nodejs
[ 0%] Built target _webots_ros_generate_messages_check_deps_field_set_int32
[ 0%] Built target sensor_msgs_generate_messages_nodejs
[ 0%] Built target std_msgs_generate_messages_lisp
[ 0%] Built target sensor_msgs_generate_messages_lisp
[ 0%] Built target mastering_ros_robot_description_pkg_xacro_generated_to_devel_space_
[ 0%] Built target _webots_ros_generate_messages_check_deps_field_set_vec3f
[ 0%] Built target _webots_ros_generate_messages_check_deps_node_get_field
[ 0%] Built target _webots_ros_generate_messages_check_deps_get_float
[ 1%] Built target e_puck_manager
[ 64%] Built target webots_ros_generate_messages_cpp
[ 76%] Built target webots_ros_generate_messages_py
[ 76%] Built target webots_ros_generate_messages_eus
[ 79%] Built target webots_ros_generate_messages_lisp
[ 98%] Built target webots_ros_generate_messages_nodejs
[ 98%] Built target webots_ros_generate_messages
[ 98%] Built target keyboard_teleop
[ 98%] Built target catch_the_bird
[ 98%] Built target e_puck_line
[100%] Built target panoramic_view_recorder
[100%] Built target robot_information_parser
[100%] Built target complete_test
root@024a5ddf0613:~/catkin_ws#

```

## Membuat Node ROS

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

pertama tama kita buat `nodesdemo_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;
}

```

Kemudian kita buat nodes demo\_topic\_subscriber

```
#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;
}
~
~
~
~
~
~
~
~
```

Kemudian lakukan editing pada CMakeList.txt, agar nodes yang kita tambahkan dapat terdapat sebagai file 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)
```

Kemudian lakukan running code dari kedua code nodes diatas setelah melakukan beberapa konfigurasi lain seperti yang disampaikan pada buku, berikut merupakan output code

```
root@024d: ~# catkin_ws/mastering_ros_demo_pkg# cd msg/
root@024d: ~# catkin_ws/mastering_ros_demo_pkg/msg# ls
demo_msg.msg
root@024d: ~# catkin_ws/mastering_ros_demo_pkg/msg# |

[ INFO] [1704459209.184257497]: 16
[ INFO] [1704459209.284326476]: 17
[ INFO] [1704459209.384312223]: 18
[ INFO] [1704459209.484335896]: 19
[ INFO] [1704459209.584291744]: 20
[ INFO] [1704459209.684433102]: 21
[ INFO] [1704459209.784347444]: 22
[ INFO] [1704459209.884012881]: 23
[ INFO] [1704459209.984218554]: 24
[ INFO] [1704459210.084234446]: 25
[ INFO] [1704459210.184238132]: 26
[ INFO] [1704459210.284225642]: 27
[ INFO] [1704459210.384288707]: 28
[ INFO] [1704459210.484346815]: 29
[ INFO] [1704459210.58424333]: 30
[ INFO] [1704459210.684277299]: 31
[ INFO] [1704459210.784238295]: 32
[ INFO] [1704459210.884325935]: 33
[ INFO] [1704459210.984278182]: 34
[ INFO] [1704459211.084301789]: 35
[ INFO] [1704459211.184255599]: 36
[ INFO] [1704459211.284227747]: 37
[ INFO] [1704459211.384225626]: 38
[ INFO] [1704459211.484228655]: 39
[ INFO] [1704459211.584239643]: 40
[ INFO] [1704459211.684239331]: 41
[ INFO] [1704459211.784238707]: 42
[ INFO] [1704459211.884240282]: 43
[ INFO] [1704459211.984221450]: 44
[ INFO] [1704459212.084318759]: 45
[ INFO] [1704459212.184228921]: 46
[ INFO] [1704459212.284232657]: 47
[ INFO] [1704459212.384233692]: 48
[ INFO] [1704459212.484288531]: 49
[ INFO] [1704459212.584211769]: 50
[ INFO] [1704459212.684253376]: 51
[ INFO] [1704459212.784222575]: 52
[ INFO] [1704459212.884204116]: 53
[ INFO] [1704459212.984285644]: 54
[ INFO] [1704459213.084234727]: 55
[ INFO] [1704459213.184222920]: 56
[ INFO] [1704459213.284212683]: 57
[ INFO] [1704459213.384205381]: 58
[ INFO] [1704459213.484217298]: 59
[ INFO] [1704459213.584313853]: 60
[ INFO] [1704459213.684271126]: 61
[ INFO] [1704459213.784201199]: 62
[ INFO] [1704459213.884216414]: 63
[ INFO] [1704459213.984212947]: 64
[ INFO] [1704459214.08421312]: 65
[ INFO] [1704459214.184197713]: 66

[ INFO] [1704459224.184505152]: Received [166]
[ INFO] [1704459224.284503932]: Received [167]
[ INFO] [1704459224.384509077]: Received [168]
[ INFO] [1704459224.484714953]: Received [169]
[ INFO] [1704459224.585050345]: Received [170]
[ INFO] [1704459224.685149542]: Received [171]
[ INFO] [1704459224.785333466]: Received [172]
[ INFO] [1704459224.885332181]: Received [173]
[ INFO] [1704459224.984542347]: Received [174]
[ INFO] [1704459225.084675398]: Received [175]
[ INFO] [1704459225.185219524]: Received [176]
[ INFO] [1704459225.284591633]: Received [177]
[ INFO] [1704459225.384763076]: Received [178]
[ INFO] [1704459225.484570612]: Received [179]
[ INFO] [1704459225.584635166]: Received [180]
[ INFO] [1704459225.684674862]: Received [181]
[ INFO] [1704459225.784613742]: Received [182]
[ INFO] [1704459225.884556928]: Received [183]
[ INFO] [1704459225.984559272]: Received [184]
[ INFO] [1704459226.084551732]: Received [185]
[ INFO] [1704459226.184529201]: Received [186]
[ INFO] [1704459226.284527205]: Received [187]
[ INFO] [1704459226.384562146]: Received [188]
[ INFO] [1704459226.484679434]: Received [189]
[ INFO] [1704459226.584637440]: Received [190]
[ INFO] [1704459226.685022795]: Received [191]
[ INFO] [1704459226.784529861]: Received [192]
[ INFO] [1704459226.884868690]: Received [193]
[ INFO] [1704459226.984579959]: Received [194]
[ INFO] [1704459227.084098891]: Received [195]
[ INFO] [1704459227.184564701]: Received [196]
[ INFO] [1704459227.284665540]: 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] [1704459227.884808400]: Received [203]
[ INFO] [1704459227.984788971]: Received [204]
[ INFO] [1704459228.084932647]: Received [205]
[ INFO] [1704459228.185250154]: Received [206]
[ INFO] [1704459228.285616153]: Received [207]
[ INFO] [1704459228.385041458]: Received [208]
[ INFO] [1704459228.4850494190]: Received [209]
[ INFO] [1704459228.584084929]: Received [210]
[ INFO] [1704459228.685197022]: Received [211]
[ INFO] [1704459228.785232284]: Received [212]
[ INFO] [1704459228.885348861]: Received [213]
[ INFO] [1704459228.984582685]: Received [214]
[ INFO] [1704459229.084614110]: Received [215]
```

Menambahkan Berkas .msg dan .srv Kustom

Dalam bagian ini, kita akan mempelajari cara membuat pesan kustom dan definisi layanan dalam paket saat ini. Definisi pesan disimpan dalam berkas .msg, sementara definisi layanan disimpan dalam berkas .srv. Definisi ini memberi tahu ROS tentang jenis data dan nama data yang akan ditransmisikan dari sebuah node ROS. Ketika sebuah pesan kustom ditambahkan, ROS akan mengonversi definisi tersebut menjadi kode C++ yang setara, yang dapat kita masukkan ke dalam node-node kita.

Kita akan memulai dengan definisi pesan. Definisi pesan harus ditulis dalam berkas .msg dan harus disimpan dalam folder msg, yang berada di dalam paket. Kita akan membuat berkas pesan yang disebut demo\_msg.msg dengan definisi berikut:

```
string greeting
int32 int
```

Sejauh ini, kita hanya bekerja dengan definisi pesan standar. Sekarang, kita telah menciptakan definisi kustom kita sendiri, yang berarti kita dapat mempelajari cara menggunakannya dalam kode kita.

Tambahkan folder msg dan buat file demo\_msg.msg

```
root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg# cd msg/
root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg/msg# ls
demo_msg.msg
root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg/msg# |
```

```
root@6caed748117d:~/catkin_ws/mastering_ros_demo_pkg/msg# cat demo_msg.msg
string greeting
int32 number

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

Lalu jalankan nodes message yang sudah kita buat, jangan lupa untuk melakukan update CMakeList dan package.xml

Mari kita bangun paket menggunakan `catkin\_make` dan uji node dengan mengikuti langkah-langkah berikut:

1. Jalankan roscore:

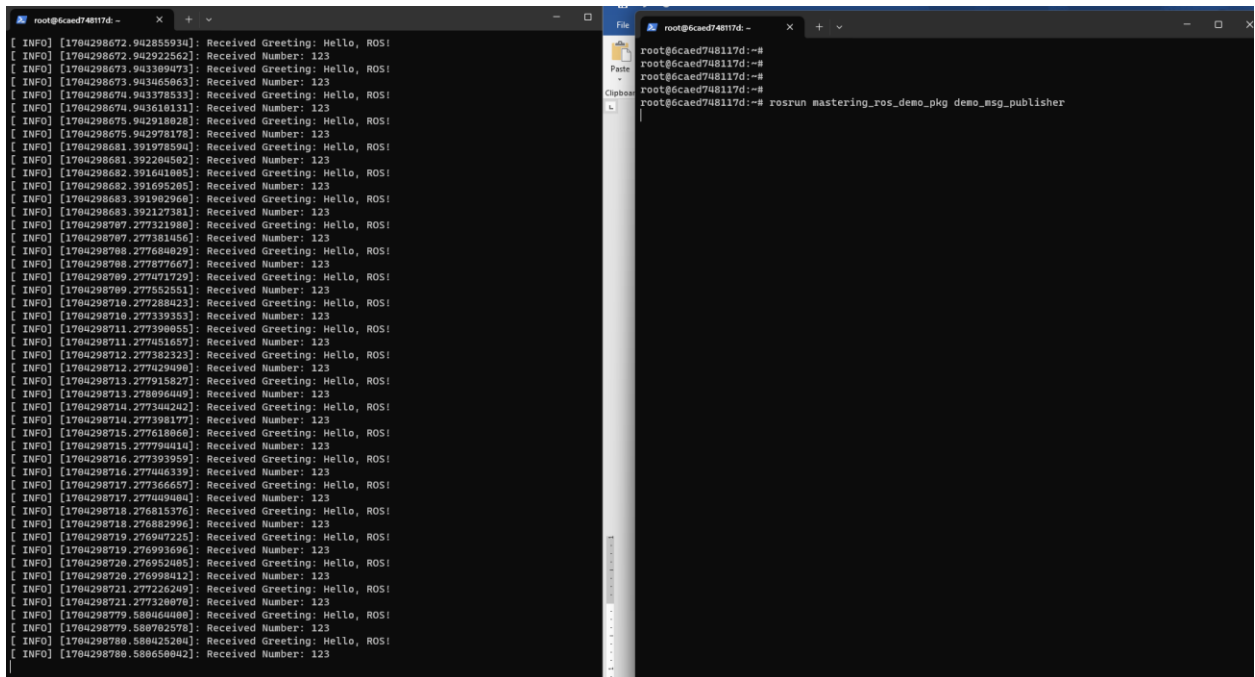
```
roscore
```

2. Mulai node penerbit pesan kustom:

```
roslaunch mastering_ros_demo_pkg demo_msg_publisher
```

3. Mulai node pelanggan pesan kustom:

```
roslaunch mastering_ros_demo_pkg demo_msg_subscriber
```



```
root@6caed748117d:~# roscore
[ INFO] [1704298672.942855934]: Received Greeting: Hello, ROS!
[ INFO] [1704298672.942922562]: Received Number: 123
[ INFO] [1704298673.943380493]: Received Greeting: Hello, ROS!
[ INFO] [1704298673.943455963]: 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] [1704298681.391978890]: Received Greeting: Hello, ROS!
[ INFO] [1704298681.392204582]: Received Number: 123
[ INFO] [1704298682.391641005]: Received Greeting: Hello, ROS!
[ INFO] [1704298682.391695205]: Received Number: 123
[ INFO] [1704298683.392122381]: Received Greeting: Hello, ROS!
[ INFO] [1704298687.277321980]: Received Greeting: Hello, ROS!
[ INFO] [1704298687.277381456]: Received Number: 123
[ INFO] [1704298688.277684029]: Received Greeting: Hello, ROS!
[ INFO] [1704298688.277877667]: Received Number: 123
[ INFO] [1704298689.277471729]: Received Greeting: Hello, ROS!
[ INFO] [1704298689.277552551]: Received Number: 123
[ INFO] [1704298710.277288423]: Received Greeting: Hello, ROS!
[ INFO] [1704298710.277339353]: Received Number: 123
[ INFO] [1704298711.277390055]: Received Greeting: Hello, ROS!
[ INFO] [1704298711.277451657]: Received Number: 123
[ INFO] [1704298712.277382323]: Received Greeting: Hello, ROS!
[ INFO] [1704298712.277429490]: Received Number: 123
[ INFO] [1704298713.277915827]: Received Greeting: Hello, ROS!
[ INFO] [1704298713.278096449]: Received Number: 123
[ INFO] [1704298714.277344242]: Received Greeting: Hello, ROS!
[ INFO] [1704298714.277393177]: Received Number: 123
[ INFO] [1704298715.277618060]: Received Greeting: Hello, ROS!
[ INFO] [1704298715.277794414]: Received Number: 123
[ INFO] [1704298716.277393959]: Received Greeting: Hello, ROS!
[ INFO] [1704298716.277446339]: Received Number: 123
[ INFO] [1704298717.277366557]: Received Greeting: Hello, ROS!
[ INFO] [1704298717.277409040]: Received Number: 123
[ INFO] [1704298718.276815376]: Received Greeting: Hello, ROS!
[ INFO] [1704298718.276882996]: Received Number: 123
[ INFO] [1704298719.276947225]: Received Greeting: Hello, ROS!
[ INFO] [1704298719.276991696]: Received Number: 123
[ INFO] [1704298720.276953046]: Received Greeting: Hello, ROS!
[ INFO] [1704298720.276998412]: Received Number: 123
[ INFO] [1704298721.277226249]: Received Greeting: Hello, ROS!
[ INFO] [1704298721.277328070]: Received Number: 123
[ INFO] [1704298779.588464000]: Received Greeting: Hello, ROS!
[ INFO] [1704298779.588703570]: Received Number: 123
[ INFO] [1704298780.588425204]: Received Greeting: Hello, ROS!
[ INFO] [1704298780.588658042]: Received Number: 123
```

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



## Bekerja dengan Layanan ROS

Dalam bagian ini, kita akan menciptakan node ROS yang dapat menggunakan definisi layanan yang sudah kita tentukan sebelumnya. Node layanan yang akan kita buat dapat mengirimkan pesan string sebagai permintaan ke server; kemudian, node server akan mengirimkan pesan lain sebagai respons.

Buka direktori `mastering_ros_demo_pkg/src` dan temukan node `demo_service_server.cpp` dan `demo_service_client.cpp`.

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

```
#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
```

Berikut merupakan code `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();

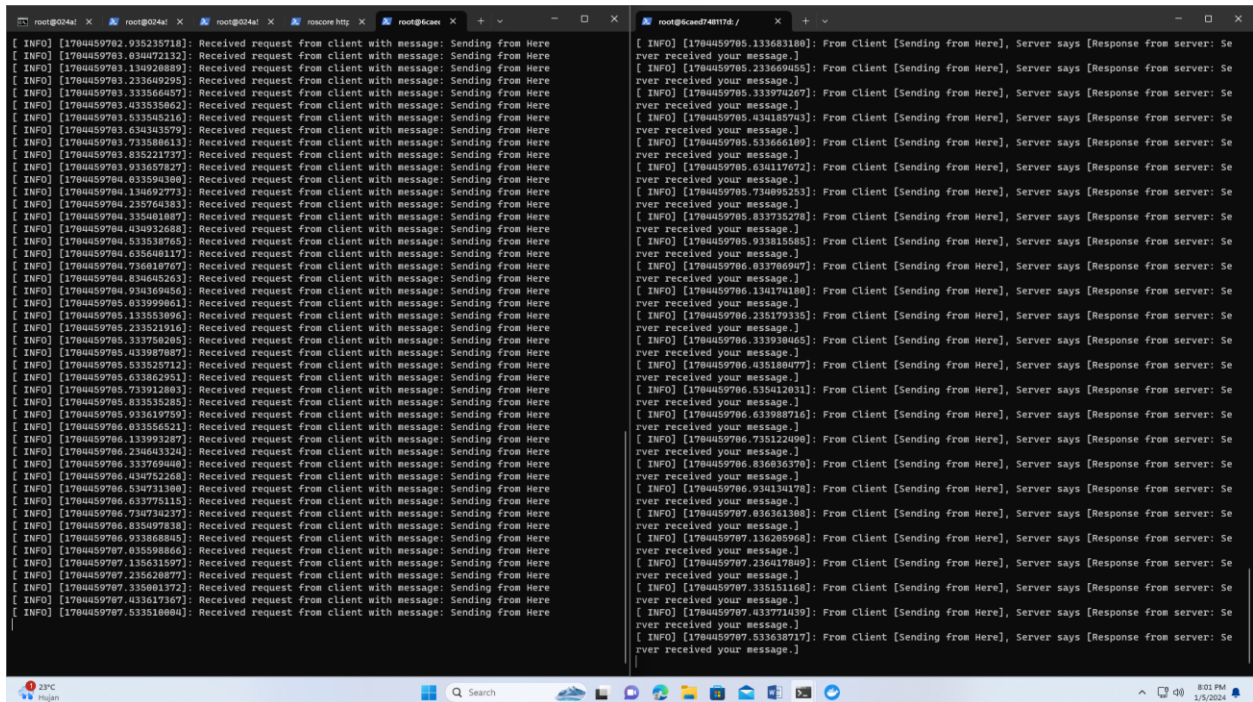
        if (client.call(srv)) {
            ROS_INFO("From Client [%s], Server says [%s]", srv.request.in.c_str(), srv.response.out.c_str());
        } else {
            ROS_ERROR("Failed to call service");
            return 1;
        }

        ros::spinOnce();
        loop_rate.sleep();
        ss.str(""); // Bersihkan stringstream setelah penggunaan
    }
}
"demo_service_client.cpp" 32L, 980C
```

Konfigurasi juga CMakeList.txt dan lakukan catkin\_make

```
####
### 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-bc68-0242ac10002/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.
```

Run program, dan ini merupakan outputnya



```
[ INFO] [1704459782.935235718]: Received request from client with message: Sending from Here
[ INFO] [1704459783.034072132]: Received request from client with message: Sending from Here
[ INFO] [1704459783.134020880]: Received request from client with message: Sending from Here
[ INFO] [1704459783.233604295]: Received request from client with message: Sending from Here
[ INFO] [1704459783.333664697]: Received request from client with message: Sending from Here
[ INFO] [1704459783.433535862]: Received request from client with message: Sending from Here
[ INFO] [1704459783.533545216]: Received request from client with message: Sending from Here
[ INFO] [1704459783.634343579]: Received request from client with message: Sending from Here
[ INFO] [1704459783.733580133]: Received request from client with message: Sending from Here
[ INFO] [1704459783.835221737]: Received request from client with message: Sending from Here
[ INFO] [1704459783.933578227]: Received request from client with message: Sending from Here
[ INFO] [1704459784.033594380]: Received request from client with message: Sending from Here
[ INFO] [1704459784.134092773]: Received request from client with message: Sending from Here
[ INFO] [1704459784.235764183]: Received request from client with message: Sending from Here
[ INFO] [1704459784.335401897]: Received request from client with message: Sending from Here
[ INFO] [1704459784.434932688]: Received request from client with message: Sending from Here
[ INFO] [1704459784.533538765]: Received request from client with message: Sending from Here
[ INFO] [1704459784.635604177]: Received request from client with message: Sending from Here
[ INFO] [1704459784.736019767]: Received request from client with message: Sending from Here
[ INFO] [1704459784.834645263]: Received request from client with message: Sending from Here
[ INFO] [1704459784.934369460]: Received request from client with message: Sending from Here
[ INFO] [1704459785.033999861]: Received request from client with message: Sending from Here
[ INFO] [1704459785.133583896]: Received request from client with message: Sending from Here
[ INFO] [1704459785.235219161]: Received request from client with message: Sending from Here
[ INFO] [1704459785.333750285]: Received request from client with message: Sending from Here
[ INFO] [1704459785.433987887]: Received request from client with message: Sending from Here
[ INFO] [1704459785.533252712]: Received request from client with message: Sending from Here
[ INFO] [1704459785.633629511]: Received request from client with message: Sending from Here
[ INFO] [1704459785.733912803]: Received request from client with message: Sending from Here
[ INFO] [1704459785.833535285]: Received request from client with message: Sending from Here
[ INFO] [1704459785.933619750]: Received request from client with message: Sending from Here
[ INFO] [1704459786.033565211]: Received request from client with message: Sending from Here
[ INFO] [1704459786.133993287]: Received request from client with message: Sending from Here
[ INFO] [1704459786.234643324]: Received request from client with message: Sending from Here
[ INFO] [1704459786.333769480]: Received request from client with message: Sending from Here
[ INFO] [1704459786.434762688]: Received request from client with message: Sending from Here
[ INFO] [1704459786.534731308]: Received request from client with message: Sending from Here
[ INFO] [1704459786.633775115]: Received request from client with message: Sending from Here
[ INFO] [1704459786.734734237]: Received request from client with message: Sending from Here
[ INFO] [1704459786.835097838]: Received request from client with message: Sending from Here
[ INFO] [1704459786.933688845]: Received request from client with message: Sending from Here
[ INFO] [1704459787.035988860]: Received request from client with message: Sending from Here
[ INFO] [1704459787.135631597]: Received request from client with message: Sending from Here
[ INFO] [1704459787.235620797]: Received request from client with message: Sending from Here
[ INFO] [1704459787.335011722]: Received request from client with message: Sending from Here
[ INFO] [1704459787.433617367]: Received request from client with message: Sending from Here
[ INFO] [1704459787.533518084]: Received request from client with message: Sending from Here
[ INFO] [1704459786.133683180]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.233669455]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.33374267]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.434185743]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.533664809]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.634117672]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.734090253]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.833735278]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459785.933815585]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.033786947]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.134174188]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.235179335]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.333930465]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.435180477]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.535412031]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.633989716]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.735122490]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.836036370]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459786.93434178]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.036361388]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.136285968]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.236417849]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.335151168]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.433771439]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459787.533630717]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
```

## Bekerja dengan actionlib ROS

Dalam layanan ROS, pengguna mengimplementasikan interaksi permintaan/balasan antara dua node, namun jika balasan membutuhkan waktu yang lama atau server belum menyelesaikan pekerjaan yang diberikan, kita harus menunggu hingga selesai. Hal ini akan menghambat aplikasi utama saat kita menunggu aksi yang diminta untuk selesai. Selain itu, klien yang memanggil dapat diimplementasikan untuk memantau eksekusi proses dari jarak jauh.

```
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# cd action/
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action# ls -la
total 12
drwxr-xr-x 2 root root 4096 Jan  4 09:52 .
drwxr-xr-x 7 root root 4096 Jan  5 12:52 ..
-rw-r--r-- 1 root root 105 Jan  4 09:32 Demo_action.action
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action#
```

Setelah `catkin\_make`, kita dapat menjalankan node-node tersebut dengan perintah berikut:

1. Jalankan roscore:

```
roscore
```

2. Mulai node server aksi:

```
roslaunch mastering_ros_demo_pkg demo_action_server
```

Membuat file peluncuran

3. Mulai node klien aksi:

```
roslaunch mastering_ros_demo_pkg demo_action_client 10 1
```

Keluaran dari proses-proses ini adalah sebagai berikut:

```
root@024a5dd8f0: x root@024a5dd8f0: x roscore http://kx x root@6caed748117d: x
[ INFO] [1704459789.833585695]: Received request from client with message: Sending from Here
[ INFO] [1704459789.933524172]: Received request from client with message: Sending from Here
[ INFO] [1704459710.833998409]: Received request from client with message: Sending from Here
[ INFO] [1704459710.133485983]: Received request from client with message: Sending from Here
[ INFO] [1704459710.233786410]: Received request from client with message: Sending from Here
[ INFO] [1704459710.333642724]: Received request from client with message: Sending from Here
[ INFO] [1704459710.434314108]: Received request from client with message: Sending from Here
[ INFO] [1704459710.533658391]: Received request from client with message: Sending from Here
[ INFO] [1704459710.633868997]: Received request from client with message: Sending from Here
[ INFO] [1704459710.734148018]: Received request from client with message: Sending from Here
[ INFO] [1704459710.835651074]: Received request from client with message: Sending from Here
[ INFO] [1704459710.935651556]: Received request from client with message: Sending from Here
[ INFO] [1704459711.035886030]: Received request from client with message: Sending from Here
[ INFO] [1704459711.133554713]: Received request from client with message: Sending from Here
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/src# cd -/catkin_ws/
root@6caed748117d:~/catkin_ws# cd src/
root@6caed748117d:~/catkin_ws/src# cd mastering_ros_demo_pkg/
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# ls
CMakeLists.txt action include msg package.xml src srv
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# cd action/
bash: cd: command not found
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg# cd action/
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action# ls -la
total 12
drwxr-xr-x 2 root root 4096 Jan  4 09:52 .
drwxr-xr-x 7 root root 4096 Jan  5 12:52 ..
-rw-r--r-- 1 root root 105 Jan  4 09:32 Demo_action.action
root@6caed748117d:~/catkin_ws/src/mastering_ros_demo_pkg/action# roslaunch mastering_ros_demo_pkg
[ INFO] [1704459892.717142997]: demo_action: Menerima goal 10
[ INFO] [1704459789.733674167]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459789.833758273]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459789.933712292]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.034238778]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.133619992]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.233999147]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.333775726]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.434704593]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.533749704]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.634184268]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.734434416]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.836273679]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459710.936229863]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459711.036358651]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
[ INFO] [1704459711.133717811]: From Client [Sending from Here], Server says [Response from server: Se
rver received your message.]
root@6caed748117d:/# roslaunch mastering_ros_demo_pkg demo_action_client 10 1
[ INFO] [1704459892.339977875]: Menunggu server...
[ INFO] [1704459892.716176735]: Server ditemukan, mengirim goal...
```

## **Membuat File Peluncuran**

File peluncuran dalam ROS sangat berguna untuk menjalankan lebih dari satu node. Pada contoh sebelumnya, kita melihat maksimal dua node ROS, namun bayangkan skenario di mana kita perlu meluncurkan 10 atau 20 node untuk sebuah robot. Tentunya akan sulit jika kita harus menjalankan setiap node secara berurutan satu per satu di terminal. Sebagai gantinya, kita dapat menuliskan semua node di dalam sebuah file berbasis XML yang disebut file peluncuran (launch file), dan dengan menggunakan perintah `roslaunch`, kita dapat mem-parsing file ini untuk menjalankan node-node tersebut.

Perintah `roslaunch` akan secara otomatis memulai ROS master dan parameter server. Dengan demikian, pada dasarnya tidak perlu menjalankan perintah `roscore` atau node-node secara individual; jika kita meluncurkan file peluncuran, semua operasi akan dilakukan dalam satu perintah. Perlu diperhatikan bahwa jika Anda memulai sebuah node menggunakan perintah `roslaunch`, menghentikan atau memulai ulang perintah ini akan memiliki efek yang sama dengan memulai ulang `roscore`.

### 3. Working with ROS for 3D Modeling

Fase pertama dari pembuatan robot melibatkan desain dan pemodelan. Kita dapat merancang dan memodelkan sebuah robot menggunakan perangkat CAD seperti Autodesk Fusion 360, SolidWorks, Blender, dan lainnya. Salah satu tujuan utama dari pemodelan robot adalah simulasi.

Alat simulasi robotik dapat memeriksa kelemahan kritis dalam desain robot dan memastikan bahwa robot akan berfungsi sebelum memasuki tahap manufaktur.

Dalam bab ini, kita akan membahas proses desain dua jenis robot. Satu adalah manipulator dengan tujuh Derajat Kebebasan (DOF), dan yang lainnya adalah robot penggerak diferensial. Pada bab-bab berikutnya, kita akan membahas simulasi, belajar cara membangun perangkat keras nyata, dan membahas tentang antarmuka dengan ROS.

Jika Anda berencana membuat model 3D dari robot dan mensimulasikannya menggunakan ROS, Anda perlu memahami beberapa paket ROS yang dapat membantu dalam desain robot. Membuat model untuk robot kita dalam ROS memiliki pentingnya beragam alasan. Sebagai contoh, Anda dapat menggunakan model ini untuk mensimulasikan dan mengendalikan robot, memvisualisasikannya, atau menggunakan alat-alat ROS untuk mendapatkan informasi mengenai struktur robotik dan kinematikanya.

#### Membuat Paket ROS untuk Deskripsi Robot

Sebelum membuat file URDF untuk robot, mari kita buat paket ROS di workspace catkin agar model robot dapat disimpan menggunakan perintah berikut:

```
catkin_create_pkg   mastering_ros_robot_description_pkg   roscpp   tf
geometry_msgs urdf rviz xacro
```

Paket ini terutama bergantung pada paket urdf dan xacro. Jika paket-paket ini belum terinstal di sistem Anda, Anda dapat menginstalnya menggunakan manajer paket:

```
sudo apt-get install ros-noetic-urdf
sudo apt-get install ros-noetic-xacro
```

Kita dapat membuat file URDF robot di dalam paket ini dan membuat file peluncuran untuk menampilkan file URDF yang telah dibuat di RViz. Paket lengkap tersedia di repositori Git berikut; Anda dapat mengklon repositori ini sebagai referensi untuk mengimplementasikan paket ini, atau Anda dapat mendapatkan paket dari kode sumber buku:

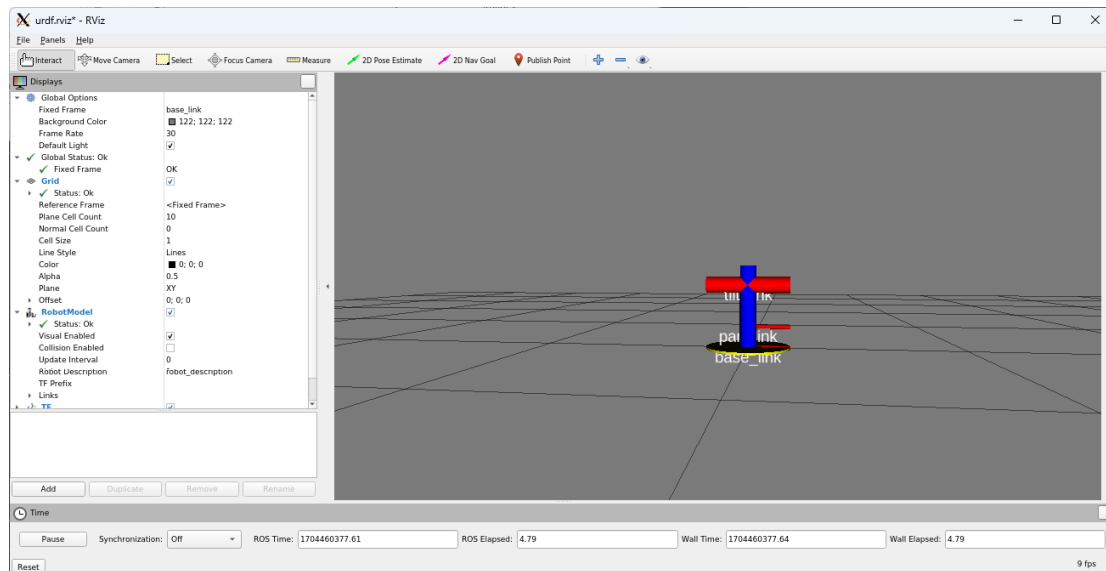
```
git clone
https://github.com/qboticslabs/mastering_ros_3rd_edition.git
cd mastering_ros_robot_description_pkg/
```

Sebelum membuat file URDF untuk robot ini, mari kita buat tiga folder bernama urdf, meshes, dan launch di dalam folder paket. Folder urdf dapat digunakan untuk menyimpan file URDF dan xacro yang akan kita buat. Folder meshes menyimpan mesh yang perlu kita sertakan dalam file URDF, dan folder launch menyimpan file peluncuran ROS.

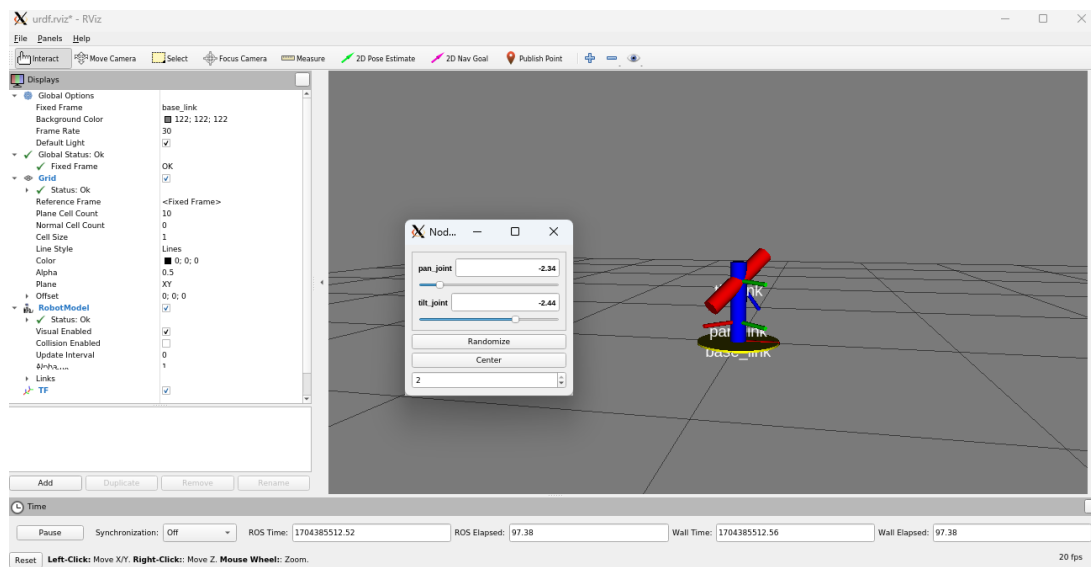
Berikut merupakan direktori hasil gitclone

```
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#
```

Dibawah ini merupakan hasil visualizing 3d models di Rviz



Kita juga dapat merubah posisi joint dengan bantuan GUI yang disediakan



## Mengonversi xacro ke URDF

Seperti yang telah disebutkan sebelumnya, file xacro dapat dikonversi menjadi file urdf setiap saat. Setelah merancang file xacro, Anda dapat menggunakan perintah berikut untuk mengonversinya menjadi file URDF:

```
roslaunch xacro pan_tilt.xacro > pan_tilt_generated.urdf
```

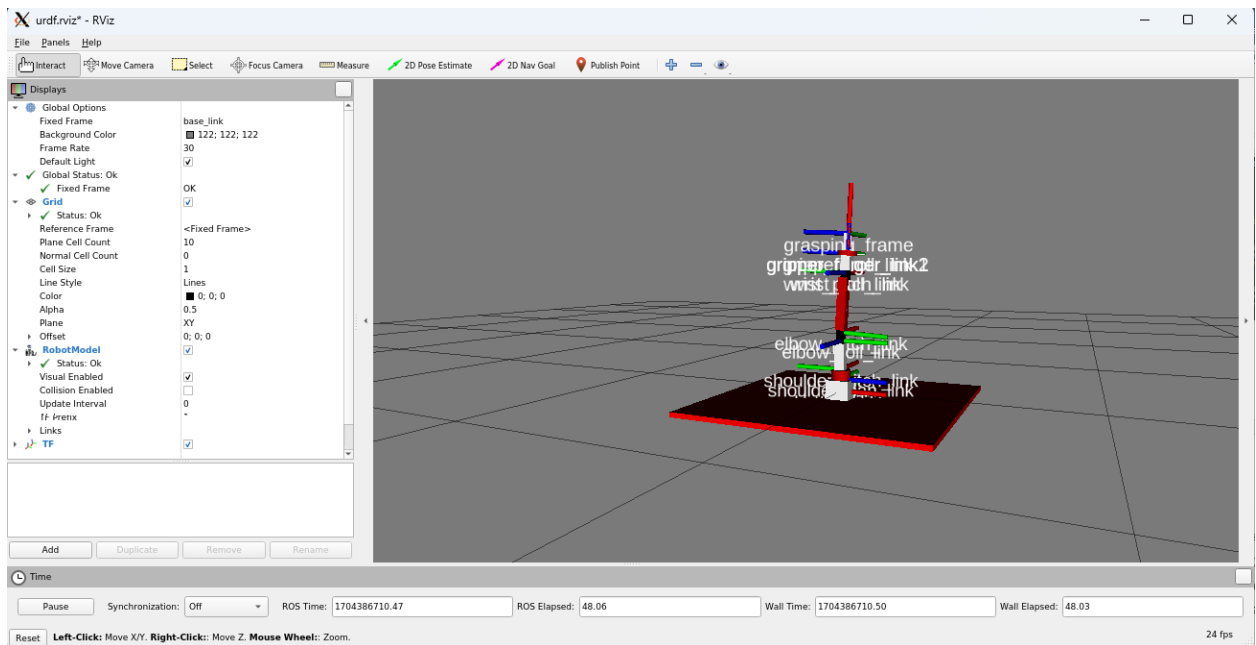
Anda dapat melihat file xacro dari robot pan-and-tilt dengan membuat file peluncuran, dan file tersebut dapat diluncurkan menggunakan perintah berikut:

```
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...  
* /roscpp: 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[roslaunch]: 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
```

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

```
roslaunch mastering_ros_robot_description_pkg view_arm.launch
```



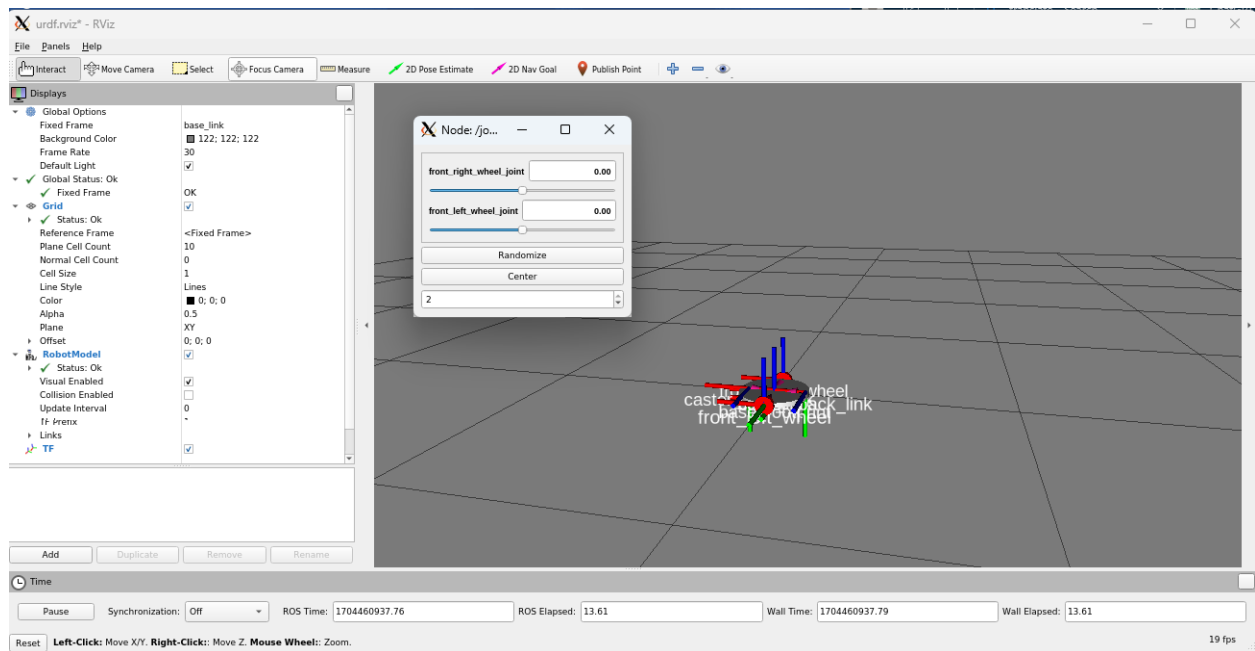
Paket penerbit negara bersama adalah salah satu paket ROS yang umum digunakan untuk berinteraksi dengan setiap sendi robot. Paket berisi joint\_state\_node penerbit, yang menemukan sambungan tidak tetap dari model URDF dan menerbitkan nilai status gabungan dari setiap sambungan dalam format pesan sensor\_msgs/JointState. Paket ini juga dapat digunakan bersamaan dengan robot\_state\_publisher paket untuk mempublikasikan posisi semua sendi. Sumber yang berbeda dapat digunakan untuk mengatur nilai setiap sambungan. Seperti yang telah kita lihat, salah satu caranya adalah dengan menggunakan slider GUI. Cara ini adalah terutama digunakan untuk pengujian. Jika tidak, topik JointState yang node berlangganan dapat digunakan.

### Membuat Model Robot untuk Robot Beroda Diferensial

Robot beroda diferensial memiliki dua roda yang terhubung di sisi berlawanan dari rangka robot, yang didukung oleh satu atau dua roda caster. Kecepatan robot dikontrol oleh roda dengan mengatur kecepatan masing-masing roda. Jika dua motor berjalan pada kecepatan yang sama, roda akan bergerak maju atau mundur. Jika satu roda berjalan lebih lambat daripada yang lain, robot akan berbelok ke arah roda dengan kecepatan lebih rendah. Jika kita ingin mengarahkan robot ke sisi kiri, kita mengurangi kecepatan roda kiri dan sebaliknya.

Setelah melakukan beberapa setting tambahan berikut merupakan tampilan dari robot





## 4. Simulating Robots Using ROS and Gazebo

Gazebo adalah simulator multi-robot untuk simulasi robotik kompleks baik di dalam ruangan maupun di luar ruangan. Dengan Gazebo, kita dapat mensimulasikan robot, sensor robot, dan berbagai objek 3D. Gazebo sudah menyediakan model simulasi dari robot, sensor, dan objek 3D populer dalam repositorinya ([https://bitbucket.org/osrf/gazebo\\_models/](https://bitbucket.org/osrf/gazebo_models/)). Kita dapat menggunakan model-model tersebut tanpa perlu membuat yang baru.

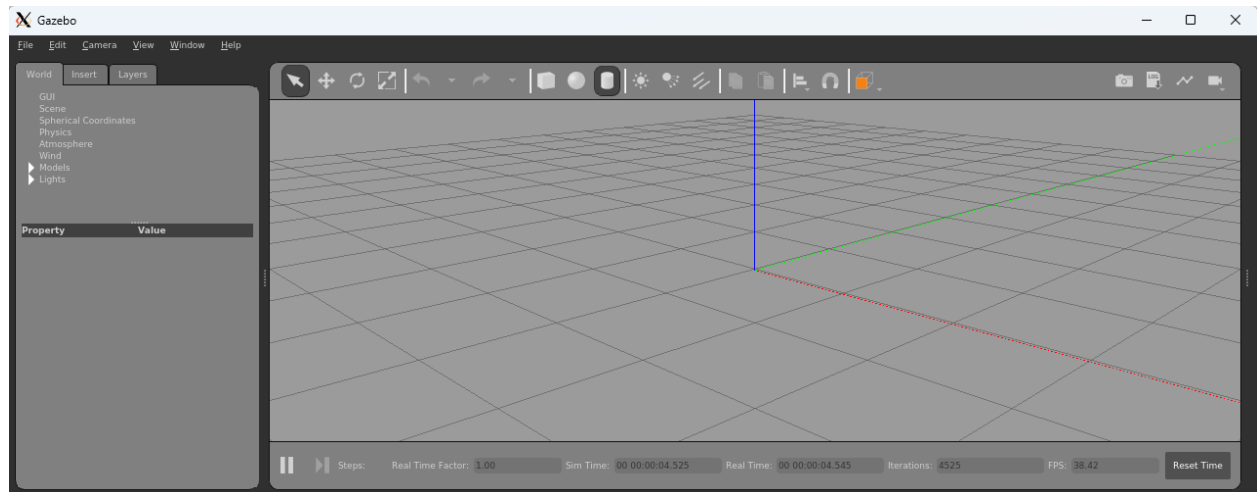
Gazebo terintegrasi dengan sempurna dengan ROS berkat antarmuka ROS yang sesuai, yang memungkinkan kontrol penuh atas Gazebo melalui ROS. Meskipun Gazebo dapat diinstal tanpa ROS, namun kita harus menginstal antarmuka ROS-Gazebo untuk berkomunikasi dari ROS ke Gazebo.

Mensimulasikan lengan robot menggunakan Gazebo dan ROS

Pada bab sebelumnya, kita telah merancang sebuah lengan robot dengan tujuh derajat kebebasan (DOF). Dalam bagian ini, kita akan mensimulasikan robot tersebut di Gazebo menggunakan ROS.

Sebelum memulai dengan Gazebo dan ROS, kita harus menginstal paket-paket berikut untuk bekerja dengan 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
```



Jika Gazebo terinstal dengan baik, maka tampilan akan seperti diatas

## Membuat model simulasi lengan robot untuk Gazebo

Kita dapat membuat model simulasi untuk lengan robot dengan memperbarui deskripsi robot yang ada dengan menambahkan parameter simulasi.

Kita dapat membuat paket yang diperlukan untuk mensimulasikan lengan robot dengan menggunakan perintah berikut:

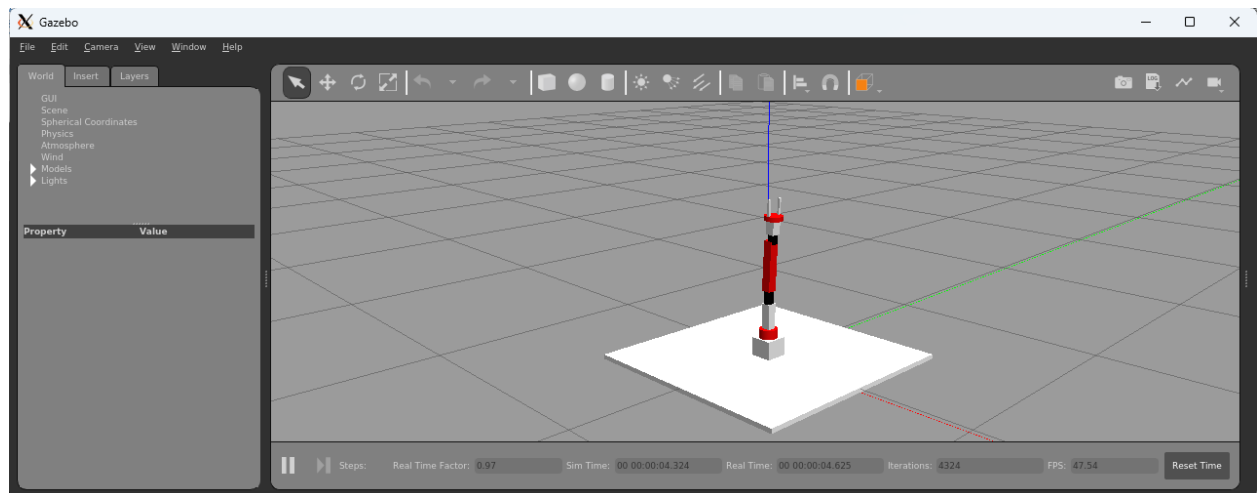
```
catkin_create_pkg    seven_dof_arm_gazebo    gazebo_msgs    gazebo_plugins    gazebo_ros
gazebo_ros_control mastering_ros_robot_description_pkg
```

Sebagai alternatif, paket lengkap tersedia di repositori Git berikut; Anda dapat mengkloning repositori ini sebagai referensi untuk mengimplementasikan paket ini, atau Anda dapat mendapatkan paket tersebut dari kode sumber buku:

```
git clone https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git
cd Chapter4/seven_dof_arm_gazebo
```

```
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#
```

Berikut merupakan tampilan dari Robot



Menambahkan warna dan tekstur pada model robot Gazebo

Dalam simulasi robot, kita dapat melihat bahwa setiap tautan memiliki warna dan tekstur yang berbeda.

Tag-tag berikut dalam file .xacro memberikan tekstur dan warna pada tautan robot:

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

```

</gazebo>

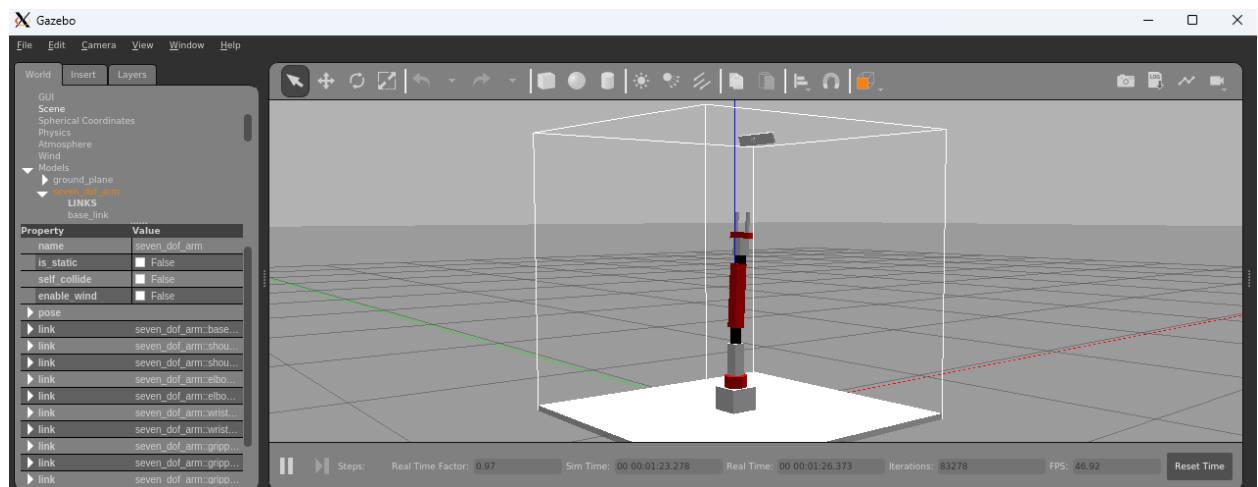
<gazebo reference="shoulder_pan_link">
<material>Gazebo/Red</material>

</gazebo>

```

Setiap tag gazebo merujuk pada tautan tertentu dari model robot.

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] [1704389771.107199, 0.395900000]: Loaded gazebo_ros_control.
[INFO] [1704389771.078084, 0.396000]: wait_for_service(/seven_dof_arm/controller_manager/load_controller
able to contact [rospic://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

```

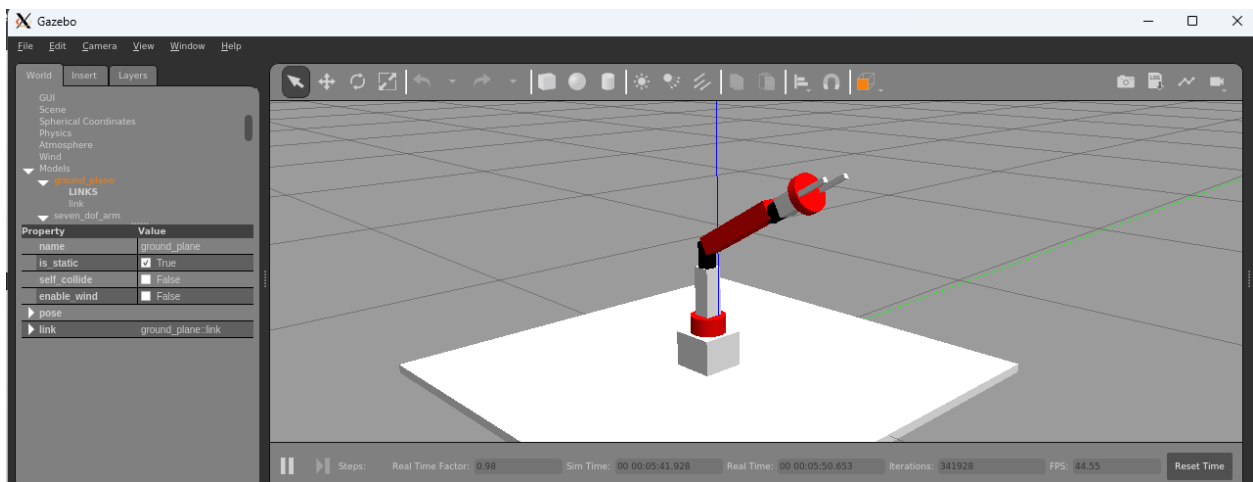
## Menggerakkan Sendi Robot

Setelah menyelesaikan topik sebelumnya, kita dapat mulai mengendalikan setiap sendi ke posisi yang diinginkan.

Untuk menggerakkan sendi robot di Gazebo, kita harus menerbitkan nilai sendi yang diinginkan dengan tipe pesan `std_msgs/Float64` ke topik perintah pengendali posisi sendi.

Berikut contoh menggerakkan sendi keempat ke 1.0 radian:

```
rostopic pub /seven_dof_arm/joint4_position_controller/command  
std_msgs/Float64 1.0
```



## Menambahkan Node Teleop ROS

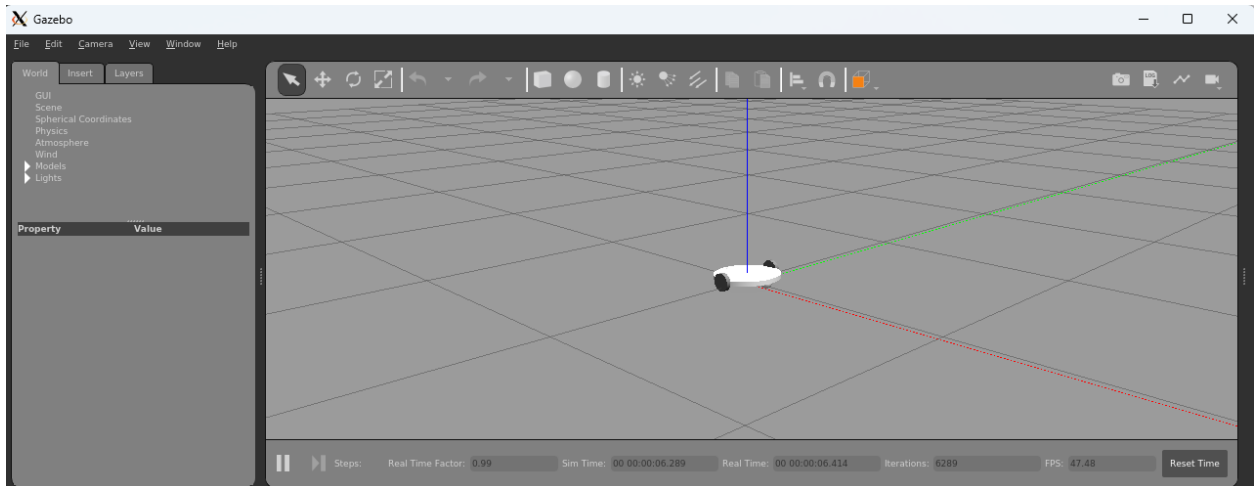
Node teleop ROS menerbitkan perintah ROS Twist dengan mengambil input keyboard. Dari node ini, kita dapat menghasilkan kecepatan linear dan angular, dan sudah ada implementasi node teleop standar yang tersedia; kita dapat dengan mudah menggunakan kembali node tersebut.

Teleop diimplementasikan dalam paket `diff_wheeled_robot_control`. Folder skrip berisi node `diff_wheeled_robot_key`, yang merupakan node teleop. Seperti biasanya, Anda dapat mengunduh paket ini dari repositori Git sebelumnya. Pada titik ini, Anda dapat mengakses paket ini dengan perintah berikut:

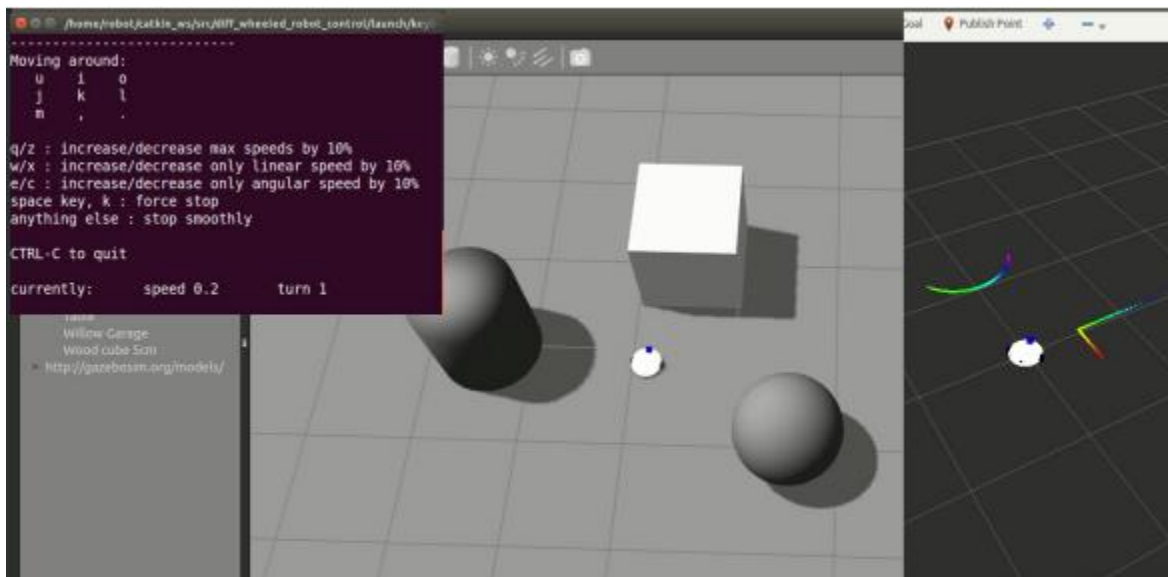
```
roscd diff_wheeled_robot_control
```

Untuk berhasil mengompilasi dan menggunakan paket ini, Anda mungkin perlu menginstal paket `joy_node`:

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



Node teleop ROS menerbitkan perintah ROS Twist dengan mengambil input keyboard. Dari simulasi ini, kita dapat menghasilkan kecepatan linier dan sudut, dan sudah ada implementasi node teleop standar tersedia; Kita cukup menggunakan kembali simulasi. Teleop diimplementasikan dalam paket `diff_wheeled_robot_control`. Naskah Folder berisi node `diff_wheeled_robot_key`, yang merupakan node teleop. Sesuai biasanya, Anda dapat mengunduh paket ini dari repositori Git sebelumnya. Pada titik ini, Anda Capai paket ini dengan perintah berikut:



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

Setelah mempelajari cara mensimulasikan robot dengan Gazebo, dalam bab ini kita akan membahas bagaimana menggunakan dua perangkat lunak simulasi robot lain yang sangat kuat: CoppeliaSim (<http://www.coppeliarobotics.com>) dan Webots (<https://cyberbotics.com/>).

Kedua simulator robot ini multiplatform. CoppeliaSim dikembangkan oleh Coppelia Robotics. Ini menawarkan banyak model simulasi robot industri dan mobile yang populer yang siap digunakan, serta berbagai fungsionalitas yang dapat dengan mudah diintegrasikan dan dikombinasikan melalui antarmuka pemrograman aplikasi (API) yang khusus. Selain itu, CoppeliaSim dapat beroperasi dengan Robot Operating System (ROS) menggunakan antarmuka komunikasi yang tepat, memungkinkan kita untuk mengontrol adegan simulasi dan robot melalui topik dan layanan. Seperti Gazebo, CoppeliaSim dapat digunakan sebagai perangkat lunak mandiri, sementara plugin eksternal harus diinstal untuk bekerja dengan ROS.

Sementara itu, Webots adalah perangkat lunak sumber terbuka gratis yang digunakan untuk mensimulasikan robot 3D. Ini dikembangkan oleh Cyberbotics Ltd. dan sejak Desember 2018 telah dirilis di bawah lisensi sumber terbuka dan gratis.

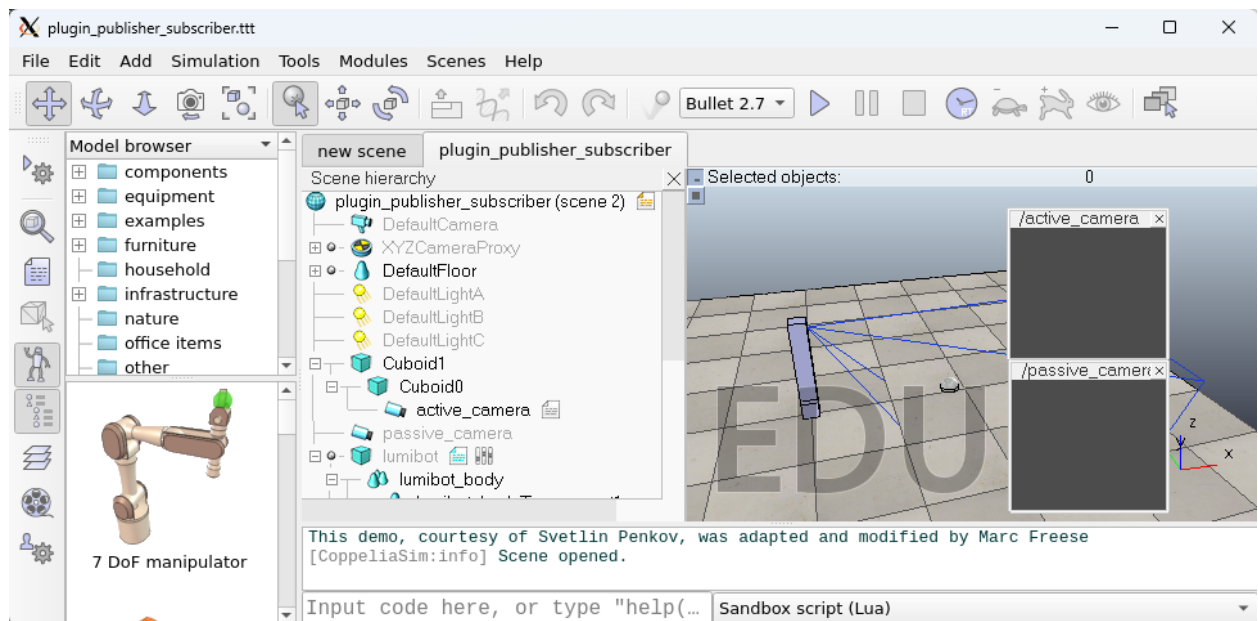
### Mempersiapkan CoppeliaSim dengan ROS

Sebelum memulai kerja dengan CoppeliaSim, kita perlu menginstalnya di sistem kita dan mengonfigurasi lingkungan kita untuk memulai jembatan komunikasi antara ROS dan adegan simulasi. CoppeliaSim adalah perangkat lunak lintas platform, tersedia untuk berbagai sistem operasi seperti Windows, macOS, dan Linux. Dikembangkan oleh Coppelia Robotics GmbH, perangkat lunak ini didistribusikan dengan lisensi pendidikan gratis serta lisensi komersial. Unduh versi terbaru dari simulator CoppeliaSim dari halaman unduhan Coppelia Robotics di <http://www.coppeliarobotics.com/downloads.html>, pilih versi edu untuk Linux. Dalam bab ini, kita akan merujuk pada versi CoppeliaSim 4.2.0.

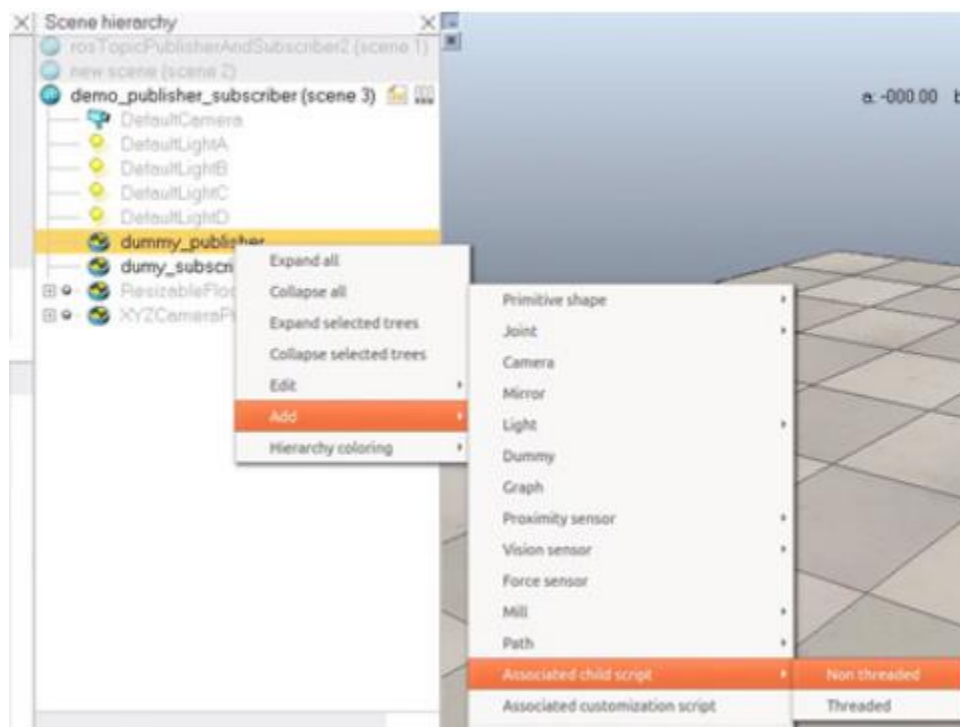
Setelah selesai mengunduh, ekstrak arsip tersebut. Pindah ke folder unduhan Anda dan gunakan perintah berikut:

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

berikut merupakan tampilan coppeliasim jika terinstall dengan benar



Sekarang kita akan membahas cara menggunakan topik ROS untuk berkomunikasi dengan CoppeliaSim. Ini adalah berguna ketika kita ingin mengirim informasi ke objek simulasi, atau mengambil data dihasilkan oleh sensor robot atau aktuator. Cara paling umum untuk memprogram adegan simulasi simulator ini adalah dengan menggunakan Lua Skrip. Setiap objek adegan dapat dikaitkan dengan skrip yang dipanggil secara otomatis ketika simulasi dimulai dan dijalankan secara siklis selama waktu simulasi





## Bekerja dengan Pesan ROS

Untuk mempublikasikan pesan ROS baru dalam skrip Lua, kita perlu membungkusnya dalam struktur data yang berisi bidang-bidang yang sama dengan pesan aslinya. Prosedur sebaliknya harus dilakukan untuk mengumpulkan informasi yang diterbitkan pada topik ROS. Mari kita analisis pekerjaan yang dilakukan pada contoh sebelumnya sebelum kita beralih ke sesuatu yang lebih kompleks. Pada contoh `dummy\_publisher`, tujuannya adalah untuk mempublikasikan data integer pada topik ROS. Kita dapat memeriksa struktur pesan integer menggunakan perintah ROS berikut:

```
rosmmsg show std_msgs/Int32
int32 data
```

```
root@024a5ddf0613:/# rosmmsg show std_msgs/Int32
int32 data

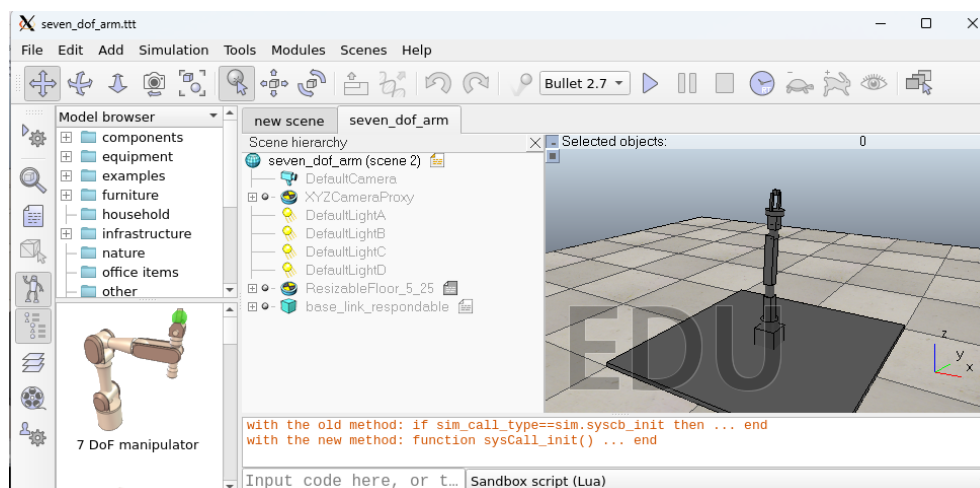
root@024a5ddf0613:/# |
```

## Mensimulasikan Lengan Robot Menggunakan CoppeliaSim dan ROS

Pada bab sebelumnya, kita menggunakan Gazebo untuk mengimpor dan mensimulasikan lengan dengan tujuh derajat kebebasan (DOF) yang dirancang dalam Bab 3, Bekerja dengan ROS untuk Pemodelan 3D. Di sini, kita akan melakukan hal yang sama menggunakan CoppeliaSim. Langkah pertama untuk mensimulasikan lengan tujuh DOF kita adalah mengimpornya ke dalam adegan simulasi. CoppeliaSim memungkinkan Anda mengimpor robot baru menggunakan file URDF; untuk alasan ini, kita harus mengonversi model xacro dari lengan ke dalam file URDF, menyimpan file URDF yang dihasilkan di folder urdf dari paket csim\_demo\_pkg, seperti berikut:

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

Sekarang kita bisa mengimpor model robot, menggunakan plugin impor URDF. Pilih dari menu drop-down utama entri Plugins | URDF import dan tekan tombol Impor, memilih opsi impor default dari jendela dialog. Terakhir, pilih file yang diinginkan untuk diimpor, dan lengan tujuh DOF akan muncul di adegan, seperti yang diilustrasikan dalam tangkapan layar berikut:

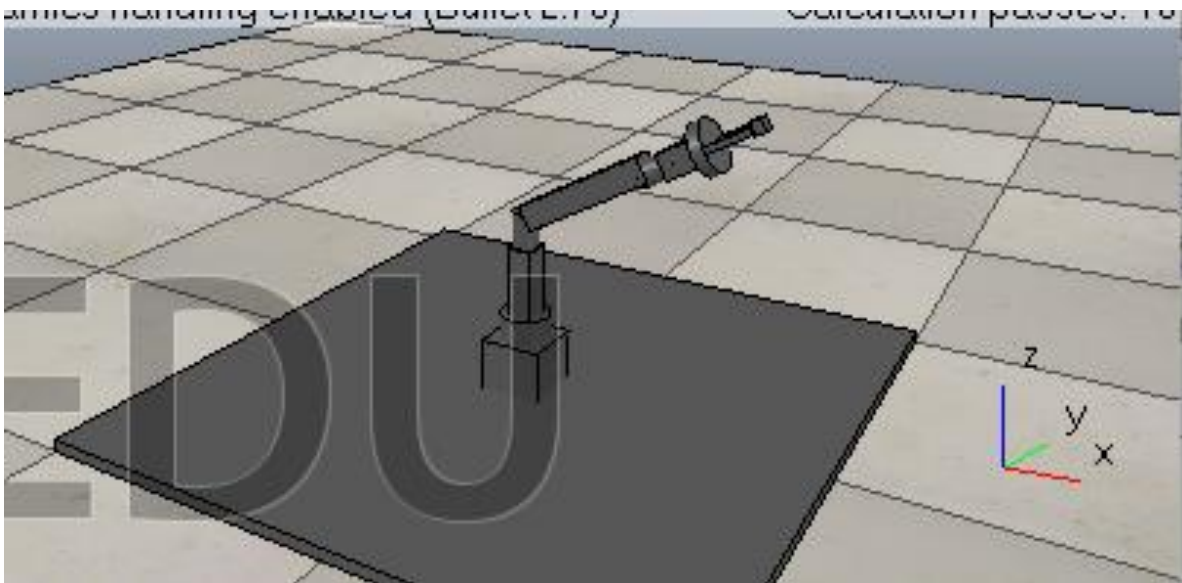


Di sini, kita menggunakan fungsi `setJointTargetPosition` untuk mengubah posisi dari sebuah joint tertentu. Argumen masukan dari fungsi-fungsi seperti ini adalah handler dari objek joint dan nilai yang akan ditetapkan. Setelah memulai simulasi, kita dapat memindahkan sebuah joint yang diinginkan, seperti joint `elbow_pitch`, dengan memublikasikan sebuah nilai menggunakan alat baris perintah, seperti yang diilustrasikan dalam potongan kode berikut:

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

Pada saat yang sama, kita dapat mendapatkan posisi dari joint dengan mendengarkan topik status, seperti berikut:

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



## Setting up Webots with ROS

Anda dapat memilih berbagai cara untuk menginstal simulator ini. Anda bisa mengunduh paket .deb dari halaman web Webots (<http://www.cyberbotics.com/#download>) atau menggunakan manajer paket Debian/Ubuntu Advanced Packaging Tool (APT). Dengan asumsi Anda menggunakan Ubuntu, mari kita mulai dengan mengotentikasi repositori Cyberbotics, seperti berikut:

```
wget -qO- https://cyberbotics.com/Cyberbotics.asc | sudo apt-key add -
```

Kemudian, Anda dapat mengonfigurasi manajer paket APT Anda dengan menambahkan repositori Cyberbotics, seperti berikut:

```
sudo apt-add-repository 'deb https://cyberbotics.com/debian/ binary-  
amd64/'  
  
sudo apt-get update
```

Setelah itu, lanjutkan dengan menginstal Webots menggunakan perintah berikut:

```
sudo apt-get install webots
```

Sekarang kita siap untuk memulai Webots dengan perintah berikut:

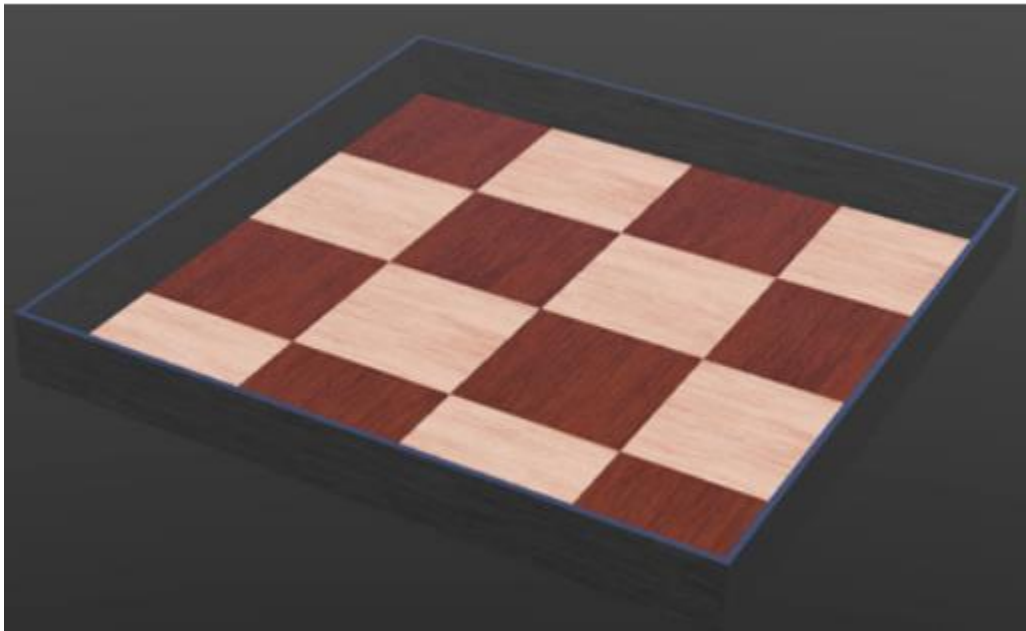
```
$ webots
```

Setelah perintah ini, antarmuka pengguna (UI) Webots akan terbuka. Dengan menggunakan menu simulasi di bagian atas jendela, Anda dapat mengontrol simulasi, memulai, menjeda, atau mempercepat eksekusinya.

Kini kita siap untuk memulai simulasi gerakan robot dan sensor. Sebelum kita membahas bagaimana memprogram robot dengan Webots, mari kita tinjau dasarnya terlebih dahulu.

## Mensimulasikan robot beroda dengan Webots

Tujuan dari bagian ini adalah untuk menciptakan sebuah adegan simulasi dari awal, yang berisi objek-objek dan sebuah robot beroda bergerak. Untuk melakukannya, kita perlu membuat dunia kosong baru. Anda dapat menggunakan opsi wizard untuk menciptakan adegan simulasi baru. Dunia ini sudah tersedia dalam kode sumber buku dalam paket `webots_demo_pkg`. Untuk membuat simulasi baru, gunakan menu bar atas dan pilih Wizards | New Project Directory. Sebuah aplikasi kecil akan membantu Anda menyiapkan semuanya. Klik Next untuk memilih direktori proyek. Masukkan jalur folder sesuai keinginan Anda, dan pilih nama folder sebagai `"mobile_robot"`. Anda juga dapat memilih nama dunia; masukkan `"robot_motion_controller.wbt"` dan pastikan untuk menandai opsi `"Add a rectangle area"`. Kemudian, klik Finish dan setelah adegan dimuat, itu seharusnya akan muncul seperti yang ditunjukkan dalam tangkapan layar berikut:



The code of the controller is listed in the following snippet:

```
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#define MAX_SPEED 6.28
//64 Milliseconds
#define TIME_STEP 64
using namespace webots;
int main(int argc, char **argv) {
  Robot *robot = new Robot();
  Motor *leftMotor = robot->getMotor("left wheel motor");
```

```
Motor *rightMotor = robot->getMotor("right wheel motor");
leftMotor->setPosition(INFINITY);
rightMotor->setPosition(INFINITY);
double t=0.0;
double r_direction=1.0;
while(true) {
leftMotor->setVelocity( MAX_SPEED*0.1);
rightMotor->setVelocity( r_direction*MAX_SPEED*0.1);
robot->step(TIME_STEP) ;
t+= TIME_STEP;
if ( t > 2000 ) {
r_direction*=-1.0;
}
if( t > 4000) {
r_direction = 1.0;
t = 0.0;
}
}
delete robot;
return 0;
}
```