

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

Nama : Al Ghifary Akmal Nasheeri

NIM : 1103201242

Kelas : TK-44-06

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

Kedua simulator robot ini bersifat multiplatform. CoppeliaSim dikembangkan oleh Coppelia Robotics dan menawarkan banyak model simulasi robot industri dan mobile yang populer, siap digunakan, serta berbagai fungsionalitas yang dapat dengan mudah diintegrasikan dan dikombinasikan melalui antarmuka pemrograman aplikasi (API) yang didedikasikan. Selain itu, CoppeliaSim dapat beroperasi dengan Robot Operating System (ROS) menggunakan antarmuka komunikasi yang memungkinkan kita mengendalikan adegan simulasi dan robot melalui topik dan layanan. Sama seperti Gazebo, CoppeliaSim dapat digunakan sebagai perangkat lunak mandiri, namun plugin eksternal harus diinstal untuk bekerja dengan ROS. Sedangkan Webots, adalah perangkat lunak sumber terbuka gratis yang digunakan untuk mensimulasikan robot 3D. Webots dikembangkan oleh Cyberbotics Ltd. dan sejak Desember 2018 telah dirilis di bawah lisensi sumber terbuka.

Seperti halnya dengan CoppeliaSim, Webots dapat dengan mudah dihubungkan dengan ROS. Dalam bab ini, kita akan mempelajari cara mengatur simulator ini dan menghubungkannya dengan jaringan ROS. Kami akan membahas beberapa kode awal untuk memahami cara kerja keduanya sebagai perangkat lunak mandiri dan bagaimana mereka dapat digunakan dengan layanan dan topik ROS.

### Setting up CoppeliaSim with ROS

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

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

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

Versi ini didukung oleh Ubuntu versi 20.04. Sebaiknya, ubah nama folder ini menjadi sesuatu yang lebih intuitif, misalnya:

```
mv CoppeliaSim_Edu_V4_2_0_Ubuntu20_04 CoppeliaSim
```

Untuk dengan mudah mengakses sumber daya CoppeliaSim, disarankan untuk mengatur variabel lingkungan COPPELIASIM\_ROOT yang menunjuk ke folder utama CoppeliaSim, seperti ini:

```
echo "export COPPELIASIM_ROOT=/path/to/CoppeliaSim/folder" >>
~/.bashrc
```

Di sini, **/path/to/CoppeliaSim/folder** adalah jalur absolut ke folder yang diekstrak.

CoppeliaSim menawarkan mode berikut untuk mengendalikan robot yang disimulasikan dari aplikasi eksternal:

- Remote application programming interface (API): API jarak jauh CoppeliaSim terdiri dari beberapa fungsi yang dapat dipanggil dari aplikasi eksternal yang dikembangkan dalam C/C++, Python, Lua, atau MATLAB. API jarak jauh berinteraksi dengan CoppeliaSim melalui jaringan menggunakan komunikasi soket. Dapat mengintegrasikan API jarak jauh pada node C++ atau Python Anda untuk menghubungkan ROS dengan adegan simulasi.
- RosInterface: Antarmuka saat ini untuk mengaktifkan komunikasi antara ROS dan CoppeliaSim. Dalam bab ini, kita akan membahas cara berinteraksi dengan CoppeliaSim menggunakan plugin RosInterface yang mereplikasi fungsionalitas API jarak jauh secara transparan. Melalui antarmuka ini, CoppeliaSim akan bertindak sebagai node ROS yang dapat berkomunikasi dengan node lain melalui layanan ROS, penerbit ROS, dan pelanggan ROS.

Untuk mengaktifkan antarmuka komunikasi ROS, jalankan perintah roscore di mesin Anda sebelum membuka simulator. Selanjutnya, untuk membuka CoppeliaSim, gunakan perintah berikut:

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

Selama startup, semua plugin yang diinstal di sistem akan dimuat. Untuk memastikan semuanya berfungsi dengan baik, Anda dapat memeriksa daftar node yang berjalan di sistem

Anda setelah meluncurkan CoppeliaSim. Sebagai contoh, jika melihat node-node ROS yang aktif, seperti `sim_ros_interface`, maka plugin `RosInterface` telah berhasil dimulai.

Untuk menjelajahi fungsionalitas plugin `RosInterface`, Anda dapat membuka skenario **`plugin_publisher_subscriber.ttt`** yang terletak di folder **`csim_demo_pkg/scene`** dari kode yang disediakan dengan buku ini. Setelah membuka skenario ini, jendela simulasi harus muncul, memperlihatkan robot yang dilengkapi dua kamera.

Dalam skenario ini, kamera pasif menampilkan gambar yang dipublikasikan dari kamera aktif, menerima data visual langsung dari kerangka ROS. Anda juga dapat memvisualisasikan aliran video yang dipublikasikan oleh CoppeliaSim menggunakan paket `image_view` dengan menjalankan perintah:

```
roslaunch image_view image:=/camera/image_raw
```

Kemudian, dapat dibahas cara menghubungkan CoppeliaSim dan ROS menggunakan plugin `RosInterface`.

## Understanding the `RosInterface` plugin

Plugin `RosInterface` adalah bagian dari kerangka kerja API CoppeliaSim. Meskipun plugin telah terinstal dengan benar di sistem Anda, operasi pemuatan akan gagal jika `roscore` tidak berjalan pada saat itu. Untuk mencegah perilaku yang tidak terduga, kita akan melihat cara memeriksa apakah plugin `RosInterface` berfungsi dengan baik. Selanjutnya, kita akan membahas cara berinteraksi dengan CoppeliaSim menggunakan topik ROS.

## Interacting with CoppeliaSim using ROS topics

Pada bagian ini, kita akan membahas cara menggunakan topik ROS untuk berkomunikasi dengan CoppeliaSim. Ini berguna ketika kita ingin mengirim informasi ke objek simulasi atau mengambil data yang dihasilkan oleh sensor atau aktuator robot.

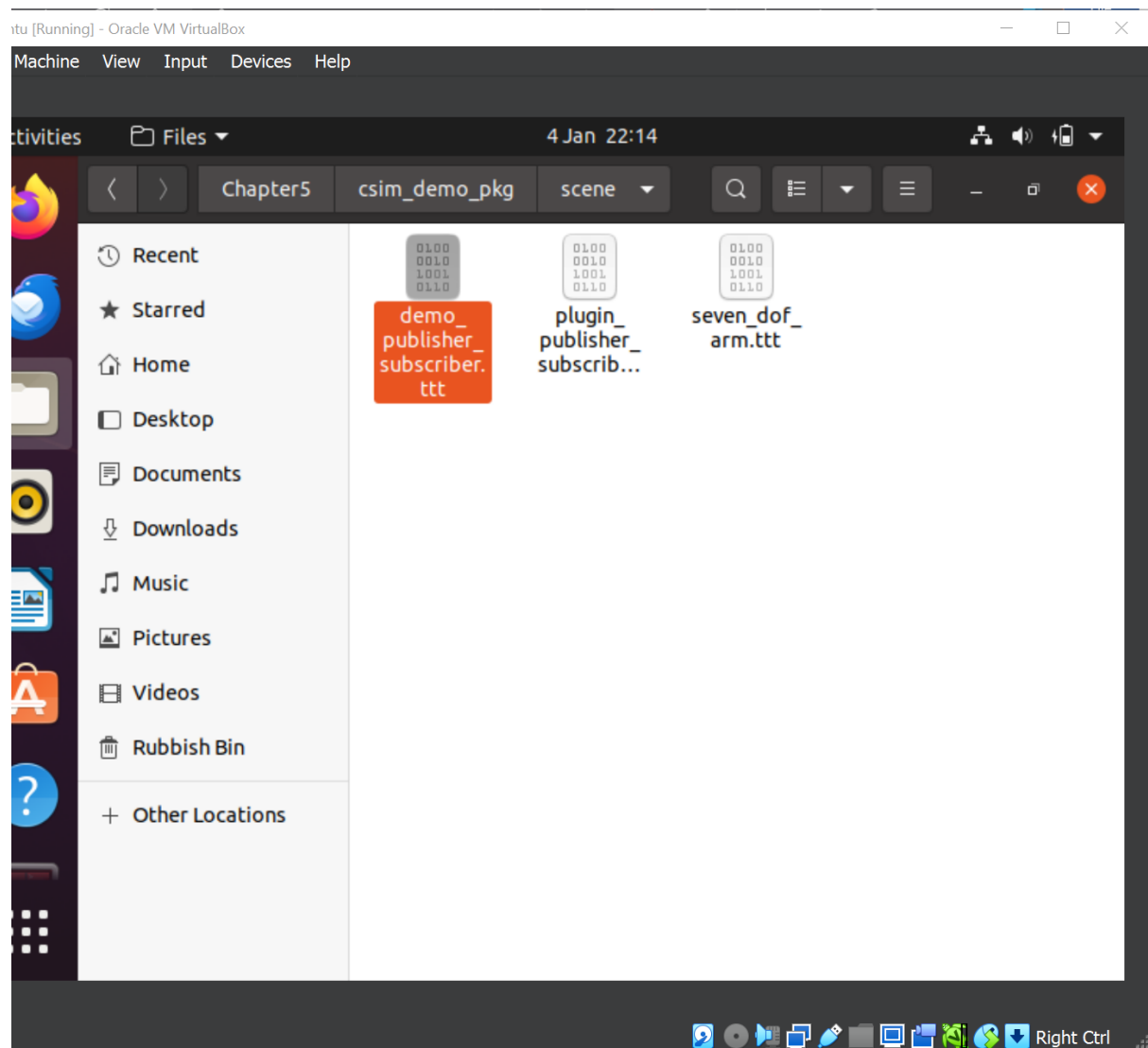
Cara paling umum untuk memprogram adegan simulasi di simulator ini adalah dengan menggunakan skrip Lua. Setiap objek dari adegan dapat dihubungkan dengan skrip yang secara otomatis dipanggil ketika simulasi dimulai dan dijalankan secara siklik selama waktu simulasi.

Sebagai contoh, kita akan membuat adegan dengan dua objek. Satu akan diprogram untuk mempublikasikan data integer dari topik tertentu, sedangkan yang lainnya akan berlangganan topik ini, menampilkan data float di konsol CoppeliaSim.

Gunakan menu tarik-turun pada panel hirarki adegan, pilih **Entri Tambah | Dummy**. Kita dapat membuat dua objek, objek `dummy_publisher` dan objek `dummy_subscriber`, dan mengasosiasikan skrip dengan masing-masing dari mereka. Gunakan tombol kanan mouse pada objek yang dibuat, dan pilih **Entri Tambah | Anak skrip terkait | Non threaded**, seperti yang ditunjukkan dalam tangkapan layar.

Selain itu, kita dapat langsung memuat adegan simulasi dengan membuka file **`demo_publisher_subscriber.ttt`** yang terletak di folder **`csim_demo_pkg`** dari sumber kode buku ini di direktori `scene`.

## demo\_publisher\_subscriber.ttt

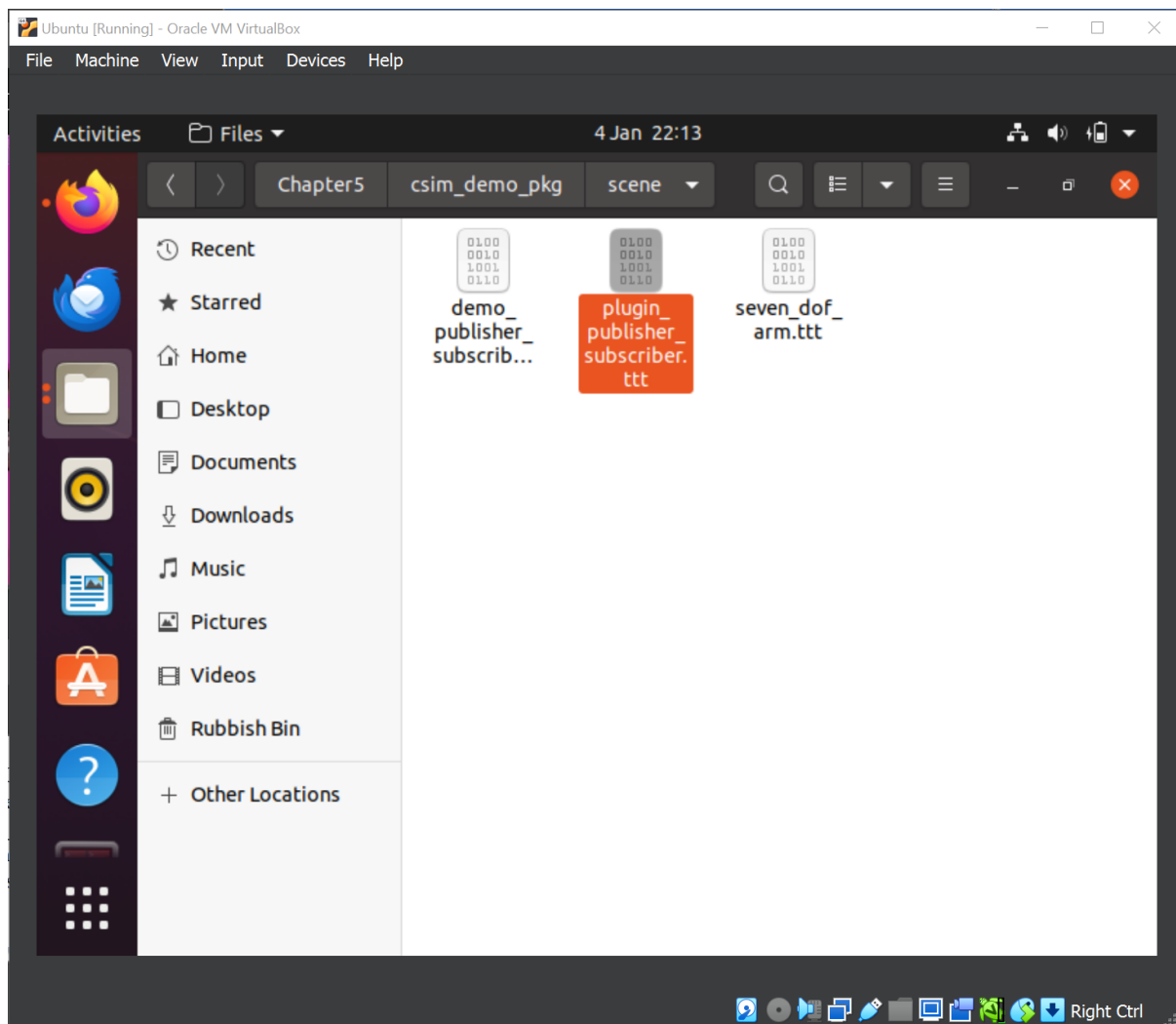


### Working with ROS messages

Dalam proses simulasi robot menggunakan CoppeliaSim, penting untuk memahami cara berinteraksi dengan ROS (Robot Operating System). Dalam contoh penerbit dummy\_publisher, kita mempublikasikan data integer ke topik ROS. Ini dilakukan dengan memahami struktur pesan yang akan dipublikasikan. Sebagai contoh, jika kita ingin mempublikasikan data integer, kita perlu membungkusnya dalam struktur pesan yang sesuai, sesuai dengan format yang diharapkan oleh ROS. Dalam hal ini, kita perlu memasukkan nilai ke dalam bidang "data" dari pesan.

Selanjutnya, kita melihat cara menyiarkan gambar melalui ROS yang diambil oleh sensor kamera di adegan simulasi. Proses ini melibatkan pengambilan gambar dan propertinya menggunakan metode tertentu. Selanjutnya, gambar tersebut disusun dalam struktur data yang sesuai dengan format pesan sensor\_msgs/Image yang diharapkan oleh ROS. Data ini kemudian dipublikasikan ke topik ROS terkait.

## plugin\_publisher\_subscriber.ttt



## Simulating a robotic arm using CoppeliaSim and ROS

Ketika kita ingin mensimulasikan robot lengan tujuh derajat kebebasan (DOF) menggunakan CoppeliaSim, langkah pertama adalah mengimpor model robot ke dalam adegan simulasi. Hal ini membutuhkan konversi model xacro ke dalam file URDF agar dapat diimpor oleh CoppeliaSim. Setelah diimpor, komponen robot seperti sendi dan link akan muncul dalam adegan. Namun, agar robot dapat dikendalikan, perlu mengaktifkan motor pada sendi-sendi tersebut. Pengaturan ini memastikan bahwa robot dapat bergerak selama simulasi dan mengikuti perintah kendali.

Setelah impor dan konfigurasi selesai, langkah berikutnya adalah menghubungkan robot dengan ROS menggunakan plugin RosInterface. Ini melibatkan pembuatan skrip Lua yang memungkinkan pertukaran informasi antara CoppeliaSim dan ROS melalui topik-topik khusus. Dengan mengonfigurasi penerbit dan langganan pada setiap sendi robot, kita dapat mengontrol robot menggunakan perintah ROS. Sebaliknya, kita dapat memantau status sendi melalui topik ROS yang sesuai.

Melalui proses ini, kita berhasil mensimulasikan dan mengontrol robot lengan tujuh DOF menggunakan CoppeliaSim dan terhubung dengan ROS. Dengan ini, kita dapat terus mendalami simulasi robot dan mengeksplorasi penggunaan perangkat lunak simulasi robot lainnya seperti Webots.

### **Adding the ROS interface to CoppeliaSim joint controllers**

Pada bagian ini, kita akan mempelajari cara mengintegrasikan lengan robot tujuh derajat kebebasan (DOF) dengan plugin RosInterface pada CoppeliaSim untuk mentransmisikan status sendi dan menerima masukan kendali melalui topik ROS. Seperti yang telah terlihat pada contoh sebelumnya, kita memilih suatu komponen robot (misalnya, komponen `base_link_respondable`) dan membuat skrip Lua untuk mengelola komunikasi antara CoppeliaSim dan ROS. Dalam blok inisialisasi, kita mendapatkan handler untuk semua sendi robot, seperti `shoulder_pan`, `shoulder_pitch`, `elbow_roll`, dan lainnya. Selanjutnya, kita menetapkan penerbit sudut sendi menggunakan `simROS.advertise()` untuk setiap sendi model. Hal ini juga diulangi untuk pelanggan perintah sendi yang mengatur callback untuk memproses data masukan yang diterima. Sebagai contoh, kita menggunakan fungsi `setJointTargetPosition` untuk mengubah posisi suatu sendi berdasarkan data yang diterima. Setelah simulasi dimulai, kita dapat memindahkan sendi tertentu, misalnya, `elbow_pitch`, dengan menerbitkan nilai menggunakan perintah baris perintah ROS, dan sekaligus memantau posisi sendi tersebut melalui topik terkait. Dengan ini, kita siap mengimplementasikan algoritma kontrol untuk menggerakkan sendi-sendi lengan robot tujuh DOF. Dengan topik ini, kita menyelesaikan bagian pertama dari bab ini dan melanjutkan untuk membahas perangkat lunak simulasi robot lainnya, yaitu Webots.

## **Setting up Webots with ROS**

Pertama-tama, kita perlu menginstal Webots di sistem kita sebelum mengonfigurasikannya dengan ROS, mirip dengan langkah-langkah yang dilakukan pada CoppeliaSim. Webots adalah perangkat lunak simulasi multiplatform yang didukung oleh Windows, Linux, dan macOS. Pengembangan awal dilakukan oleh Swiss Federal Institute of Technology, Lausanne (EPFL), dan saat ini dikelola oleh Cyberbotics dengan lisensi Apache 2 yang gratis dan sumber terbuka. Setelah instalasi, Webots menyediakan lingkungan pengembangan lengkap untuk memodelkan, memprogram, dan mensimulasikan robot, sering digunakan dalam industri, pendidikan, dan penelitian.

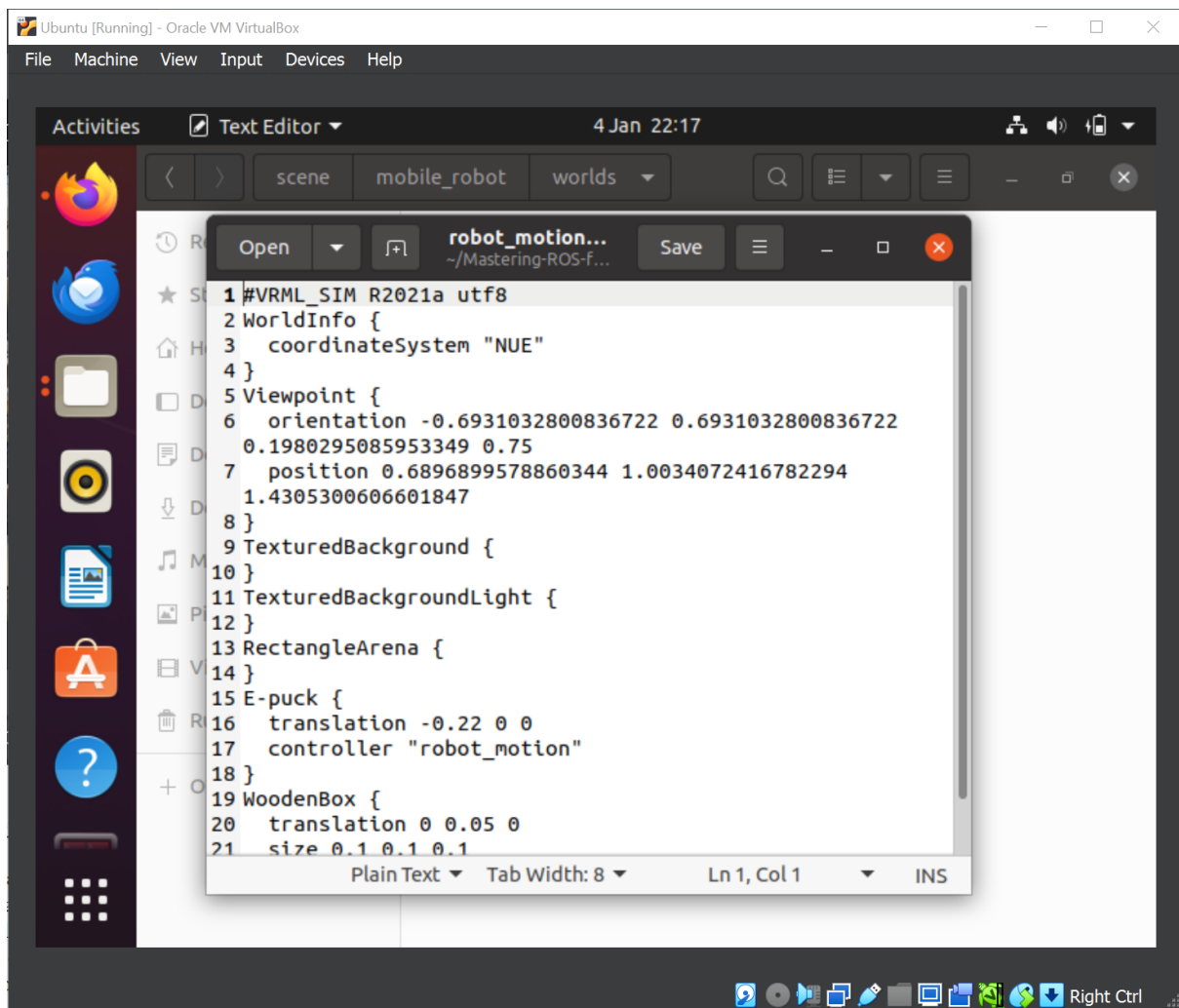
## **Introduction to the Webots simulator**

Simulasi Webots terdiri dari tiga elemen utama: file konfigurasi dunia, pengontrol, dan plugin fisik. File konfigurasi dunia (.wbt) digunakan untuk mengatur lingkungan simulasi. Pengontrol dapat diimplementasikan dalam berbagai bahasa pemrograman seperti C, C++, Python, atau Java. Webots juga mendukung skrip MATLAB. Plugin fisik digunakan untuk memodifikasi perilaku fisik simulasi. Komunikasi antara ROS dan Webots dapat diimplementasikan menggunakan pengontrol yang bertindak sebagai node ROS, menyediakan fungsi Webots sebagai layanan atau topik ke node ROS lainnya.

## **Simulating a mobile robot with Webots**

Tujuan dari bagian ini adalah membuat adegan simulasi dari awal yang berisi objek dan robot beroda. Langkah pertama adalah membuat dunia baru menggunakan opsi wizard. Dunia ini sudah tersedia dalam kode sumber buku di paket `webots_demo_pkg`. Setiap objek dalam adegan disusun secara hierarkis, seperti yang ditunjukkan pada panel pohon di UI. Sebuah dunia Webots terdiri dari file konfigurasi dunia, pengontrol, dan plugin fisik. Pengontrol dapat diimplementasikan dalam berbagai bahasa pemrograman dan digunakan untuk mengelola simulasi. Pada akhirnya, kita dapat memulai simulasi menggunakan tombol Start.

## robot\_motion\_controller.wbt



### Writing your first controller

Pada bagian ini, kita membuat kontroler pertama untuk robot beroda. Kita melihat bagaimana kontroler mengelola pergerakan robot dan reaksinya terhadap sensor. Mari ubah kontroler robot E-puck untuk bergerak ke arah tertentu. Kita dapat memilih bahasa pemrograman yang berbeda, tetapi kita akan menggunakan C++. Tujuan kontroler baru ini adalah mengendalikan kecepatan roda robot untuk menunjukkan struktur tipikal dari kontroler Webots.

Langkah pertama adalah mengubah kontroler yang terkait dengan robot beroda kita. Perlu dicatat bahwa setiap robot hanya dapat menggunakan satu kontroler pada satu waktu. Sebaliknya, kita dapat mengasosiasikan kontroler yang sama dengan robot yang berbeda. Langkah-langkah untuk menulis kontroler baru melibatkan:

1. Membuat file kontroler baru.
2. Menulis kontroler baru.
3. Mengompilasi kontroler baru.
4. Mengubah kontroler default robot dengan yang baru di panel properti robot.



Seperti yang telah dilakukan untuk pembuatan dunia, kita dapat menggunakan antarmuka wizard untuk menghasilkan kontroler baru. Kita memilih bahasa pemrograman C++ dan nama robot\_motion. Sejumlah kode sumber awal akan muncul di editor teks, yang kemudian dapat dikompilasi dengan tombol Build.

Kode kontroler menggunakan Webots API untuk mengendalikan kecepatan roda robot. Dalam loop utama, kecepatan setiap motor diatur sebagai 10 persen dari kecepatan maksimum, dan arah gerak motor kanan ditetapkan sebagai 1.0 untuk maju lurus dan -1.0 untuk berputar. Waktu yang sudah berlalu dipertimbangkan untuk mengatur kecepatan kontrol, dan fungsi step dipanggil untuk mengirim perintah ke motor.

Setelah mengompilasi kontroler, kita dapat menambahkannya ke robot dalam panel hirarki. Simulasi dapat dimulai untuk melihat hasil dari kontroler Webots pertama. Pada bagian berikutnya, akan dijelaskan integrasi ROS dan Webots.

### **Simulating the robotic arm using Webots and ROS**

Integrasi Webots-ROS melibatkan dua sisi: sisi ROS dan sisi Webots. Sisi ROS diimplementasikan melalui paket ROS webots\_ros, sementara Webots mendukung ROS secara alami berkat kontroler standar yang dapat ditambahkan ke model robot apa pun. Untuk menggunakan Webots dengan ROS, Anda perlu menginstal paket webots\_ros, yang dapat dilakukan menggunakan APT dengan perintah **sudo apt-get install ros-noetic-webots-ros**.

Setelah instalasi, kontroler yang sebelumnya dikembangkan harus diubah dengan kontroler bernama ros, seperti yang ditunjukkan dalam tangkapan layar. Setelah simulasi dimulai, kita dapat berinteraksi langsung dengan robot menggunakan sejumlah layanan yang mengimplementasikan fungsionalitas Webots di jaringan ROS sesuai konfigurasi sensor dan aktuator robot. Pastikan roscore aktif di jaringan ROS; jika tidak, akan muncul kesalahan di konsol Webots.

Webots hanya menerbitkan topik bernama /model\_name, yang berisi daftar model yang aktif dalam adegan simulasi. Informasi ini penting untuk menggunakan layanan Webots karena Webots menggunakan sintaksis khusus untuk mendeklarasikan layanannya atau topiknya di jaringan.

Pada tahap ini, layanan baru diterbitkan di jaringan ROS, mewakili gambar yang diambil oleh kamera. Layanan /camera/enable digunakan untuk mengaktifkan kamera, dan data kamera dapat dilihat menggunakan plugin image\_view.

Selain itu, kita dapat mengaktifkan dan membaca sensor lain, seperti sensor jarak, dan juga mengatur posisi, kecepatan, dan torsi sendi robot. Dalam kasus ini, kita ingin mengatur kecepatan roda. Integrasi dengan ROS membutuhkan pendekatan dari kedua sisi, dan di sisi ROS, kita dapat mengimplementasikan node baru menggunakan paket webots\_ros.

### **Writing a teleop node using webots\_ros**

Dalam bagian ini, kita akan mengimplementasikan sebuah node ROS untuk mengendalikan langsung kecepatan roda robot E-Puck berdasarkan pesan geometry\_msgs::Twist. Untuk

melakukannya, kita perlu menggunakan `webots_ros` sebagai dependensi. Langkah-langkahnya adalah sebagai berikut:

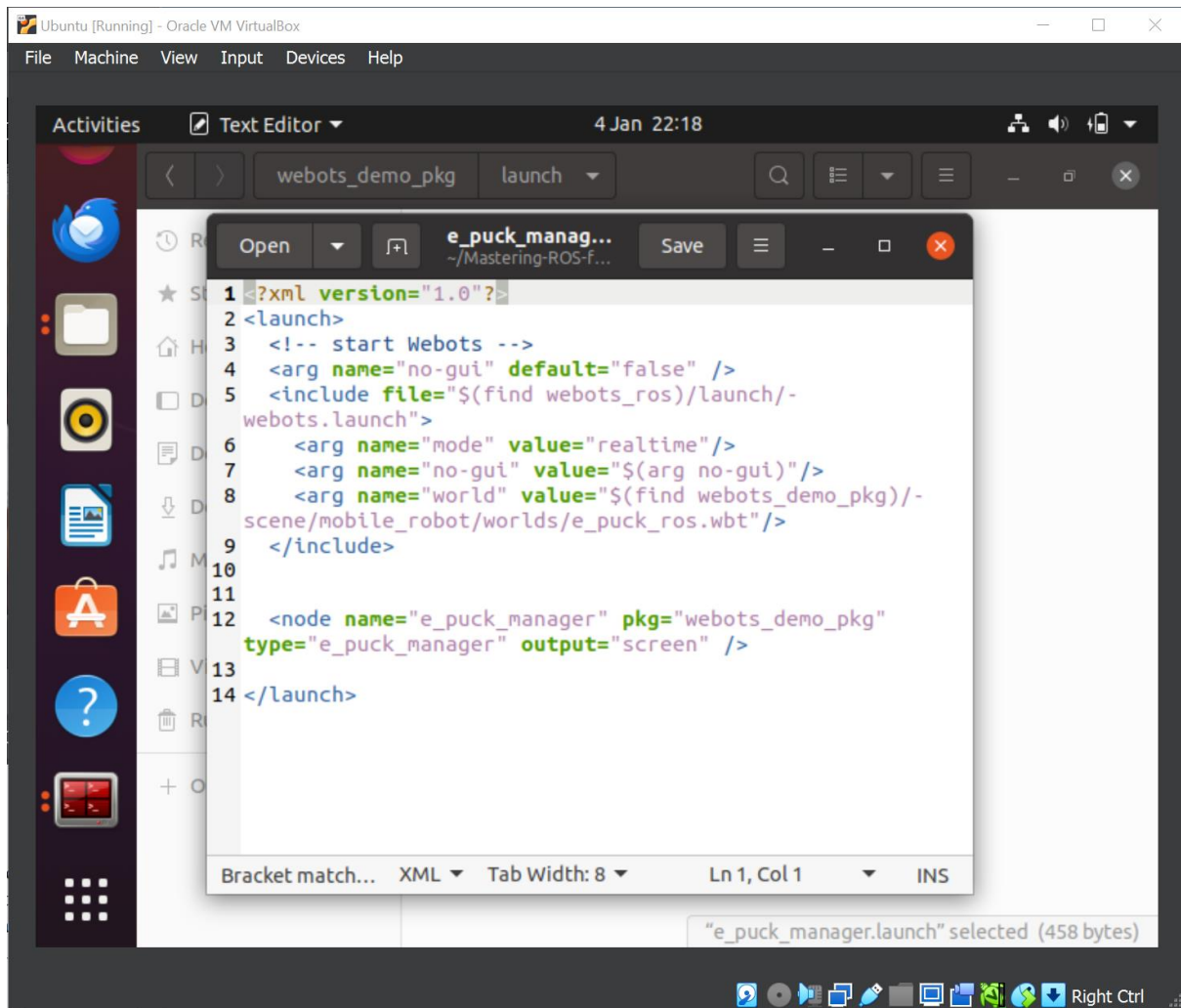
1. Membuat paket `webots_demo_pkg` dengan menyatakan `webots_ros` sebagai dependensinya menggunakan perintah **`catkin_create_pkg webots_demo_pkg roscpp webots_ros geometry_msgs`**.
2. Mendefinisikan beberapa file header yang diperlukan untuk mengimplementasikan pesan yang dibutuhkan untuk menggunakan layanan Webots.
3. Membuat variabel untuk menyimpan data yang diterima oleh panggilan kembali ROS, yaitu informasi tentang mode robot dan kecepatan yang akan diterapkan.
4. Mengimplementasikan dua panggilan kembali (callbacks) dalam node ini, salah satunya untuk membaca kecepatan linier dan angular yang diinginkan dan menetapkan kecepatan roda, dan yang lainnya untuk membaca nama model yang ditugaskan ke robot E-Puck.
5. Mengimplementasikan fungsi utama di mana semua yang diperlukan untuk mengendalikan robot diatur. Langkah-langkah meliputi inisialisasi node ROS, menunggu model robot disiarkan oleh Webots, dan mendefinisikan pelanggan (subscriber) untuk topik `/cmd_vel`.
6. Menggunakan layanan ROS untuk mengatur posisi roda menjadi INFINITY dan mengatur kecepatan menjadi 0.0.
7. Dalam loop utama, mengaplikasikan kecepatan yang dihitung dalam panggilan `geometry_msgs::Twist` ke roda kiri dan kanan menggunakan layanan ROS.

Dengan menggunakan node `keyboard_teleop` dari paket `diff_wheeled_robot_control`, kita dapat mengendalikan robot mobile di Webots menggunakan ROS. Simulasi dapat dimulai dan node dapat diluncurkan menggunakan beberapa perintah yang disebutkan di akhir bagian.

## Starting Webots with a launch file

Pada bagian terakhir ini, kita akan melihat bagaimana memulai Webots langsung menggunakan berkas *launch*. Hal ini dapat dilakukan berkat berkas *launch* yang sudah disediakan dalam paket *webots\_ros*. Untuk memulai dunia Webots yang diinginkan, kita perlu menyertakan berkas *launch* ini dan mengatur berkas *.wbt* untuk dimulai, seperti yang ditunjukkan dalam berkas *launch* *e\_puck\_manager.launch* di dalam direktori *webots\_demo\_package/launch*

### *e\_puck\_manager.launch*



The screenshot shows a text editor window titled "e\_puck\_manag..." with the following XML content:

```
1 <?xml version="1.0"?>
2 <launch>
3   <!-- start Webots -->
4   <arg name="no-gui" default="false" />
5   <include file="$(find webots_ros)/launch/-
  webots.launch">
6     <arg name="mode" value="realtime"/>
7     <arg name="no-gui" value="$(arg no-gui)"/>
8     <arg name="world" value="$(find webots_demo_pkg)/-
  scene/mobile_robot/worlds/e_puck_ros.wbt"/>
9   </include>
10
11   <node name="e_puck_manager" pkg="webots_demo_pkg"
12     type="e_puck_manager" output="screen" />
13
14 </launch>
```

The status bar at the bottom of the editor indicates "Bracket match..." and "XML". The system tray at the bottom of the VM shows various icons and the text "Right Ctrl".