

Technical Report Robotic

Nama: Sabilly A

Nim: 1103204057

CH 1- Introduction

Di chapter ini ada beberapa hal:

1. Why should we use ROS?
2. Understanding ROS filesystem
3. Understanding ROS compute
4. ROS Community

WHY should we use ROS?

Sistem Operasi Robot (Robot Operating System/ROS) adalah kerangka kerja yang fleksibel yang menyediakan berbagai tools dan perpustakaan untuk menulis perangkat lunak robotik. Ini menawarkan beberapa fitur yang powerful untuk membantu pengembang dalam tugas seperti pengiriman pesan, komputasi terdistribusi, penggunaan ulang kode, dan implementasi algoritma terkini untuk aplikasi robotik. Proyek ROS dimulai pada tahun 2007 oleh Morgan Quigley dan pengembangannya berlanjut di Willow Garage, laboratorium penelitian robotik untuk mengembangkan perangkat keras dan perangkat lunak sumber terbuka untuk robot.

Understanding ROS FileSystem

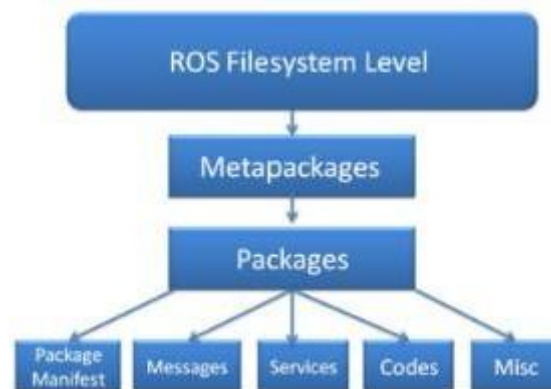


Figure 1.1 – ROS filesystem level

Paket: Paket-paket ROS merupakan elemen sentral dari perangkat lunak ROS. Mereka berisi satu atau lebih program ROS (node), perpustakaan, file konfigurasi, dan sebagainya, yang disusun bersama sebagai satu unit. Paket merupakan item pembangunan dan rilis atomik dalam perangkat lunak ROS.

- **Manifest paket:** File manifest paket ada di dalam sebuah paket dan berisi informasi tentang paket tersebut, pengarang, lisensi, dependensi, flag kompilasi, dan lain-lain. Berkas package.xml di dalam paket ROS adalah file manifest dari paket tersebut.

- **Metapaket:** Istilah metapaket merujuk pada satu atau lebih paket terkait yang dapat dielompokkan secara longgar. Pada prinsipnya, metapaket adalah paket virtual yang tidak mengandung kode sumber atau file-file tipikal yang biasanya ditemukan dalam paket.
- **Manifest metapaket:** Manifest metapaket mirip dengan manifest paket, namun perbedaannya adalah metapaket mungkin mencakup paket-paket di dalamnya sebagai dependensi runtime dan mendeklarasikan tag ekspor.
- **Pesan (.msg):** Kita dapat mendefinisikan pesan kustom di dalam folder msg di dalam sebuah paket (my_package/msg/MyMessageType.msg). Ekstensi file pesan adalah .msg.
- **Layanan (.srv):** Jenis data balasan dan permintaan dapat didefinisikan di dalam folder srv di dalam paket (my_package/srv/MyServiceType.srv).
- **Repositori:** Sebagian besar paket ROS dijaga menggunakan Sistem Kontrol Versi (Version Control System/VCS) seperti Git, Subversion (SVN), atau Mercurial (hg). Seperangkat file yang ditempatkan pada VCS mewakili sebuah repositori.

ROS Compute Graph

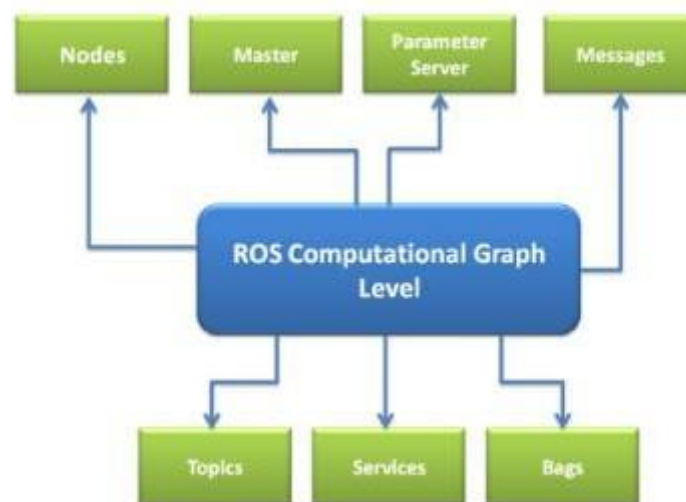


Figure 1.6 – Structure of the ROS graph layer

- **Node:** Node adalah proses yang melakukan komputasi. Setiap node ROS ditulis menggunakan perpustakaan klien ROS. Dengan menggunakan API perpustakaan klien, kita dapat mengimplementasikan berbagai fungsi ROS, seperti metode komunikasi antara node, yang sangat berguna ketika node-node yang berbeda pada robot harus bertukar informasi di antara mereka. Salah satu tujuan dari node ROS adalah membangun proses-proses yang sederhana daripada satu proses besar dengan semua fungsionalitas yang diinginkan. Karena merupakan struktur yang sederhana, node ROS mudah untuk dideteksi kesalahannya (debugging).
- **Master:** ROS master menyediakan proses pendaftaran nama dan pencarian untuk node-node lainnya. Node tidak akan dapat menemukan satu sama lain, bertukar pesan, atau

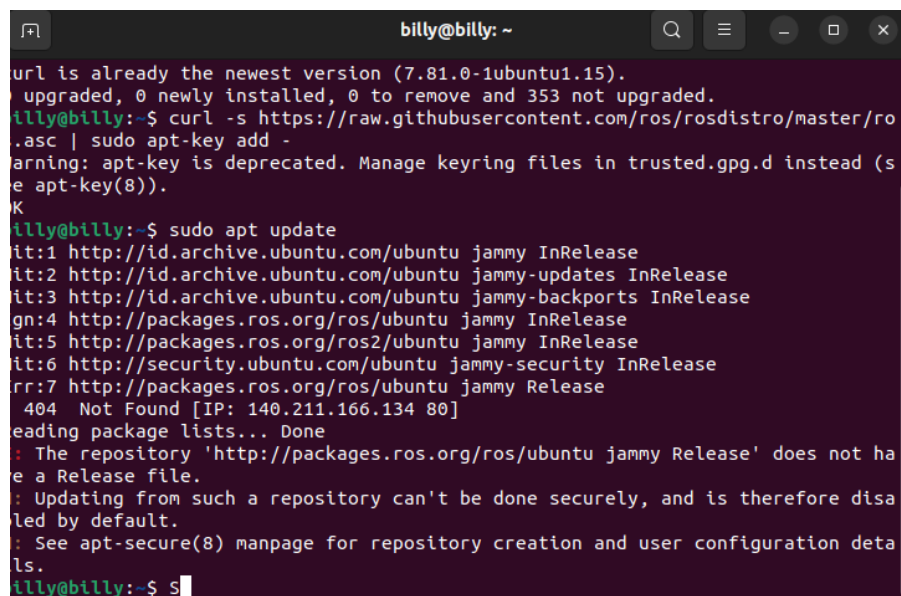
memanggil layanan tanpa adanya ROS master. Dalam sistem terdistribusi, kita harus menjalankan master pada satu komputer; kemudian, node-node remote lainnya dapat menemukan satu sama lain dengan berkomunikasi dengan master ini.

- **Parameter server:** Parameter server memungkinkan penyimpanan data di lokasi sentral. Semua node dapat mengakses dan mengubah nilai-nilai ini. Parameter server adalah bagian dari ROS master.

- **Topik:** Setiap pesan dalam ROS diangkut menggunakan bus bernama topik. Ketika sebuah node mengirim pesan melalui topik, maka kita dapat mengatakan node tersebut mempublikasikan topik. Ketika sebuah node menerima pesan melalui topik, kita dapat mengatakan bahwa node tersebut berlangganan ke topik. Node penerbit dan node pelanggan tidak mengetahui keberadaan satu sama lain. Kita bahkan dapat berlangganan ke topik yang mungkin tidak memiliki penerbit. Singkatnya, produksi informasi dan konsumsinya adalah terpisah. Setiap topik memiliki nama unik, dan setiap node dapat mengakses topik ini dan mengirimkan data melalui topik tersebut asalkan mereka memiliki tipe pesan yang tepat.

- **Pencatatan:** ROS menyediakan sistem pencatatan untuk menyimpan data, seperti data sensor, yang mungkin sulit dikumpulkan namun penting untuk pengembangan dan pengujian algoritma robot. Ini dikenal sebagai berkas bag (bagfiles). Bagfiles adalah fitur yang sangat berguna ketika kita bekerja dengan mekanisme robot yang kompleks.

Pada Chapter 1 seharusnya menginstall ROS Noetic. Tetapi saat saya mencoba terjadi error karena ubuntu versi 22.04 tidak mendukung. Alhasil saat saya mencoba di 20.04 juga sama error.

A terminal window titled 'billy@billy: ~' showing the process of installing ROS. The user runs 'sudo apt update' and 'sudo apt install ros-ubuntu-jammy'. The output shows several repositories being updated, but then an error occurs: 'The repository 'http://packages.ros.org/ros/ubuntu jammy Release' does not have a Release file.' and 'Updating from such a repository can't be done securely, and is therefore disabled by default.' The user then runs 'sudo apt install ros-noetic' and gets a similar error: 'The repository 'http://packages.ros.org/ros2/ubuntu jammy Release' does not have a Release file.'

```
billy@billy: ~  
url is already the newest version (7.81.0-1ubuntu1.15).  
 upgraded, 0 newly installed, 0 to remove and 353 not upgraded.  
billy@billy:~$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/roskey.asc | sudo apt-key add -  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK  
billy@billy:~$ sudo apt update  
Hit:1 http://id.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:2 http://id.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:3 http://id.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Ign:4 http://packages.ros.org/ros/ubuntu jammy InRelease  
Hit:5 http://packages.ros.org/ros2/ubuntu jammy InRelease  
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Err:7 http://packages.ros.org/ros/ubuntu jammy Release  
  404 Not Found [IP: 140.211.166.134 80]  
Reading package lists... Done  
: The repository 'http://packages.ros.org/ros/ubuntu jammy Release' does not have a Release file.  
: Updating from such a repository can't be done securely, and is therefore disabled by default.  
: See apt-secure(8) manpage for repository creation and user configuration details.  
billy@billy:~$ sudo apt install ros-noetic  
Err:8 http://packages.ros.org/ros2/ubuntu jammy Release  
  404 Not Found [IP: 140.211.166.134 80]  
Reading package lists... Done  
: The repository 'http://packages.ros.org/ros2/ubuntu jammy Release' does not have a Release file.  
: Updating from such a repository can't be done securely, and is therefore disabled by default.  
: See apt-secure(8) manpage for repository creation and user configuration details.  
billy@billy:~$
```

Pertanyaan di buku chapter 1:

1. Mengapa kita harus menggunakan ROS?
ROS menyediakan kerangka kerja yang fleksibel bagi para pengembang untuk menulis perangkat lunak robotika dengan berbagai fitur yang kuat. Ia membantu dalam tugas-tugas seperti pertukaran pesan, komputasi terdistribusi, penggunaan ulang kode, dan implementasi algoritma-algoritma terkini untuk aplikasi-aplikasi robotika.
2. Apa elemen dasar dari kerangka kerja ROS?
Node: Proses yang melakukan komputasi, ditulis menggunakan perpustakaan klien ROS.
Master: Menyediakan proses pendaftaran nama dan pencarian untuk node lainnya.
Parameter Server: Tempat penyimpanan data yang dapat diakses dan diubah oleh semua node.
Topik: Cara ROS mengirimkan pesan antara node-node yang berbeda.
Layanan: Mekanisme yang memungkinkan node untuk berkomunikasi antara satu sama lain.
Pesan: Struktur data yang digunakan untuk komunikasi antar node.
3. Apa prasyarat yang diperlukan untuk pemrograman dengan ROS? Untuk memulai pemrograman dengan ROS, sebaiknya memiliki pemahaman dasar tentang bahasa pemrograman Python atau C++, serta pemahaman dasar tentang konsep robotika dan sistem operasi Linux.
4. Bagaimana kerja internal dari roscore? roscore adalah bagian inti dari ROS yang menyediakan beberapa layanan utama, termasuk Master dan Parameter Server. Ini juga menyediakan beberapa alat penting yang memungkinkan node-node ROS untuk berkomunikasi satu sama lain.

CH 02 – Getting Started with ROS

Creating a ROS package

Paket-paket ROS adalah unit dasar dari program-program ROS. Kita dapat membuat, membangun, dan merilis paket-paket ROS ke publik. Distribusi ROS yang saat ini kita gunakan adalah Noetic Ninjemys. Kami menggunakan sistem build catkin untuk membangun paket-paket ROS. Sistem build bertanggung jawab dalam menghasilkan target (eksekutabel/pustaka) dari kode sumber teks yang dapat digunakan oleh pengguna akhir. Pada distribusi yang lebih lama, seperti Electric dan Fuerte, rosbuilt adalah sistem build yang digunakan. Karena berbagai kekurangan yang dimiliki oleh rosbuilt, kemudian muncul catkin. Hal ini juga memungkinkan kita untuk mendekatkan sistem kompilasi ROS ke Cross Platform Make (CMake). Ini memiliki banyak keunggulan, seperti memporing paket ke sistem operasi lain, seperti Windows. Jika suatu sistem operasi mendukung CMake dan Python, paket-paket berbasis catkin dapat diporting ke dalamnya.

Pertanyaan di Buku CH 02

Protokol komunikasi yang didukung oleh ROS antara node-node meliputi:

1. **ROS Topics:** Menggunakan proses penerbitan (publishing) dan pen-subscribe-an (subscribing) pesan-pesan. Ini merupakan komunikasi yang berbasis pada pub-sub (publish-subscribe), di mana node-node mengirim dan menerima pesan-pesan melalui topik-topik tertentu.
2. **ROS Services:** Beroperasi menggunakan permintaan dan respon yang khusus, di mana sebuah node meminta layanan kepada node lain, dan node yang menerima permintaan akan memberikan respon sesuai.
3. **ROS Actions (actionlib):** Lebih kompleks daripada layanan (services) karena bisa mengelola tugas-tugas yang memerlukan waktu yang lama. Actions memungkinkan node untuk memonitor kemajuan dari tugas-tugas yang sedang berlangsung.

Perbedaan antara perintah **roslaunch** dan **roscpp** adalah sebagai berikut:

- **roscpp** digunakan untuk menjalankan satu paket perintah dalam ROS. Ini memerlukan nama paket dan nama program yang akan dijalankan.
- **roslaunch** digunakan untuk menjalankan beberapa node dari berbagai paket secara bersamaan dengan menggunakan file launch (berkas yang berisi konfigurasi untuk menjalankan beberapa node sekaligus).

Perbedaan utama antara ROS topics dan services dalam operasinya adalah sebagai berikut:

- **ROS Topics:** Memungkinkan beberapa node untuk berkomunikasi secara tidak langsung dengan mengirim dan menerima pesan pada topik-topik tertentu. Node-node yang berkomunikasi melalui topik tidak saling mengetahui satu sama lain secara eksplisit.
- **ROS Services:** Beroperasi dengan permintaan dan respon yang spesifik. Node yang meminta layanan akan mengirim permintaan ke node lain, dan node penerima akan memberikan respon sesuai dengan permintaan yang diterima.

Perbedaan antara ROS Services dan actionlib dalam operasi mereka adalah:

- **ROS Services:** Beroperasi dengan cara permintaan (request) dan respon (response). Setelah menerima permintaan, node yang memberikan layanan akan memberikan respon kembali ke node yang meminta layanan.
- **ROS Actionlib:** Lebih kompleks dan dirancang untuk tugas-tugas yang memerlukan waktu lama. Actionlib memungkinkan pemantauan kemajuan dari tugas yang sedang berlangsung dan juga dapat membatalkan tugas yang sudah berjalan.

CH 03 – Working with ROS for 3D model

URDF

Unified Robot Description Format (URDF) adalah format file XML (eXtensible Markup Language) yang digunakan dalam ROS (Robot Operating System) untuk menggambarkan robot secara kinematik dan geometris. URDF digunakan untuk mendefinisikan struktur

mekanik robot, seperti link (koneksi fisik antar bagian robot), joint (sambungan antar link), inersia dari setiap bagian robot, visualisasi geometris, dan informasi tambahan seperti sensor atau koordinat sistem.

URDF mendefinisikan robot dalam bentuk hirarkis, di mana setiap elemen memiliki relasi terhadap elemen-elemen lainnya. Beberapa komponen yang sering digunakan dalam file URDF adalah:

1. **Links:** Mewakili bagian-bagian fisik dari robot. Links dapat didefinisikan dengan atribut-atribut seperti geometri visual (dalam bentuk mesh atau primitif geometris), parameter inersia (massa, tensor inersia), dan nama link.
2. **Joints:** Merupakan sambungan antara dua link. Joints mendefinisikan hubungan kinematik antara link-link tersebut dengan menentukan jenis joint (seperti fixed, continuous, revolute, prismatic, dll.), parameter-parameter seperti batasan gerakan (misalnya, batas sudut untuk joint revolute), dan transformasi antara link-link.
3. **Sensors:** Dapat digunakan untuk mendefinisikan sensor-sensor yang dipasang pada robot seperti lidar, kamera, atau sensor lainnya. Informasi tentang jenis sensor, posisi, dan properti lainnya dapat dijelaskan dalam URDF.

URDF digunakan dalam ROS untuk visualisasi robot dalam simulasi 3D menggunakan alat seperti RViz (ROS Visualization) dan Gazebo. Ini memungkinkan pengguna untuk membuat model robot yang realistis dan menentukan perilaku kinematik dari robot tersebut.

Dengan menggunakan URDF, pengguna dapat menggambarkan robot secara terperinci dan dapat diintegrasikan dengan ROS untuk simulasi, perencanaan gerak, visualisasi, dan tugas-tugas lainnya dalam pengembangan robotika.

CH 04 – Simulating Robots using ROS and Gazebo

Gazebo adalah simulator robotika open-source yang digunakan untuk simulasi robot di lingkungan 3D yang realistis. Simulator ini dikembangkan oleh Open Source Robotics Foundation (OSRF) dan dirancang untuk mendukung pengembangan dan pengujian robotika di berbagai bidang, seperti robot industri, layanan, pendidikan, dan penelitian.

Berikut adalah beberapa poin penting tentang Gazebo:

1. **Lingkungan Simulasi 3D:** Gazebo menyediakan lingkungan simulasi 3D yang kaya dan realistis. Pengguna dapat membuat lingkungan simulasi dengan berbagai fitur seperti tanah, bangunan, objek statis, dinamis, dan lainnya. Hal ini memungkinkan pengguna untuk mensimulasikan robot di berbagai situasi dan kondisi lingkungan yang berbeda.
2. **Fisika yang Realistis:** Gazebo dilengkapi dengan mesin fisika yang kuat yang memungkinkan simulasi yang realistis dari dinamika robot dan interaksi dengan lingkungan sekitarnya. Hal ini memungkinkan pengguna untuk memperkirakan perilaku robot di dunia nyata, termasuk respons terhadap gaya dan tindakan eksternal.

3. **Dukungan Multi-Robot:** Simulator ini mendukung simulasi beberapa robot secara simultan dalam lingkungan yang sama. Ini memungkinkan pengguna untuk menguji interaksi antara robot, misalnya, kolaborasi antara beberapa robot dalam suatu tugas.
4. **Pemodelan Sensor:** Gazebo memungkinkan pemodelan sensor seperti kamera, lidar, GPS, atau sensor lainnya. Hal ini memungkinkan pengguna untuk mensimulasikan persepsi robot dalam lingkungan virtual.
5. **Integrasi dengan ROS:** Gazebo memiliki integrasi yang erat dengan ROS, yang memungkinkan pengguna untuk menyambungkan dan mengendalikan robot ROS secara langsung dalam lingkungan simulasi Gazebo. Ini memfasilitasi pengembangan dan pengujian algoritma kontrol robot serta pengujian node ROS dalam simulasi yang realistis sebelum menerapkannya pada robot fisik.

Gazebo merupakan salah satu alat yang paling sering digunakan dalam pengembangan robotika untuk menguji dan mengembangkan algoritma kontrol, perencanaan gerak, navigasi, dan aplikasi lainnya sebelum diterapkan pada robot fisik secara nyata.

Pertanyaan di buku CH 04:

1. **Mengapa kita melakukan simulasi robotika?** Simulasi robotika memberikan cara yang aman dan efisien untuk menguji dan mengembangkan perangkat lunak robot sebelum diimplementasikan pada robot fisik di dunia nyata. Ini membantu dalam pengembangan algoritma kontrol, navigasi, perencanaan gerak, serta pengujian berbagai situasi tanpa risiko merusak peralatan atau menyebabkan cedera.
2. **Bagaimana cara menambahkan sensor ke simulasi Gazebo?** Sensor dapat ditambahkan ke simulasi Gazebo menggunakan plugin sensor. Anda bisa menambahkan sensor seperti kamera, lidar, GPS, dan sensor lainnya ke model robot atau lingkungan simulasi. Dengan menambahkan plugin sensor ke model robot atau lingkungan, Anda dapat mensimulasikan data sensor yang dihasilkan oleh sensor tersebut.
3. **Apa saja jenis controller dan antarmuka perangkat keras (hardware interfaces) ROS?** Di ROS, terdapat beberapa jenis controller dan antarmuka perangkat keras yang digunakan untuk mengontrol berbagai jenis robot. Beberapa di antaranya adalah:
 - **Joint State Controller:** Untuk membaca dan menerbitkan informasi tentang keadaan joint.
 - **Effort, Velocity, Position Controllers:** Untuk mengendalikan gerakan robot pada tingkat usaha, kecepatan, atau posisi.
 - **Diff Drive Controller:** Untuk mengontrol robot beroda differential drive.
 - **Robot Hardware Interface (HWI):** Antarmuka perangkat keras yang menghubungkan antara ROS dan perangkat keras fisik robot, memungkinkan pengendalian dan pengawasan langsung terhadap perangkat keras.

4. **Bagaimana cara menggerakkan robot mobile dalam simulasi Gazebo?** Robot mobile dalam simulasi Gazebo dapat digerakkan dengan beberapa cara:
- **ROS Topics:** Anda dapat menerbitkan pesan gerakan ke topik yang sesuai dengan robot Anda, seperti **cmd_vel** untuk robot mobile.
 - **ROS Action Server:** Menggunakan ROS Actionlib untuk gerakan yang lebih kompleks atau yang memerlukan umpan balik.
 - **ROS Service:** Melalui layanan ROS untuk melakukan perintah tertentu seperti bergerak maju, mundur, belok, dan lainnya pada robot mobile.

CH 05 – Simulating robots using ROS, CoppeliaSim and Webots

CoppeliaSim, sebelumnya dikenal sebagai V-REP (Virtual Robot Experimentation Platform), adalah simulator robotika yang kuat dan serbaguna yang memungkinkan pengguna untuk membuat simulasi lingkungan robotik yang beragam.

CoppeliaSim, sebelumnya dikenal sebagai V-REP (Virtual Robot Experimentation Platform), adalah simulator robotika yang kuat dan serbaguna yang memungkinkan pengguna untuk membuat simulasi lingkungan robotik yang beragam.

Webots.

Webots adalah alat yang powerful dalam pengembangan dan pengujian robotika karena menyediakan lingkungan simulasi yang realistis dan memungkinkan pengguna untuk menguji dan memvalidasi berbagai aspek dari robot secara virtual sebelum penerapan pada dunia nyata.

Pertanyaan di buku:

• Bagaimana CoppeliaSim berkomunikasi dengan ROS?

CoppeliaSim memiliki plugin khusus yang disebut ROS Interface, yang memungkinkan komunikasi antara CoppeliaSim (V-REP pada masa lalu) dan ROS. Plugin ini memungkinkan pengiriman dan penerimaan pesan ROS ke dan dari simulasi CoppeliaSim, yang memungkinkan integrasi robot simulasi dengan node ROS.

• Bagaimana cara mengendalikan simulasi CoppeliaSim dengan ROS?

ROS Interface di CoppeliaSim memungkinkan ROS untuk mengendalikan robot dalam simulasi. Ini dapat dilakukan dengan menerbitkan pesan ROS ke topik tertentu yang terhubung ke objek di dalam simulasi CoppeliaSim, memungkinkan penggunaan topik ROS untuk mengontrol gerakan, sensor, atau operasi lainnya di simulasi.

• Bagaimana cara mengimpor model robot baru ke CoppeliaSim dan mengintegrasikannya dengan ROS?

Untuk mengimpor model robot baru ke CoppeliaSim, pengguna dapat membuat atau mengunduh model dalam format yang didukung oleh CoppeliaSim, seperti format COLLADA (.dae), dan kemudian memuatnya ke dalam simulasi. Setelah model dimuat, ROS Interface dapat digunakan untuk menghubungkan robot simulasi dengan ROS, memungkinkan kontrol dan komunikasi antara robot simulasi dan sistem ROS.

• Dapatkah Webots digunakan sebagai perangkat lunak mandiri?

Ya, Webots dapat digunakan sebagai perangkat lunak mandiri. Ini adalah simulator robotika

independen yang memiliki lingkungan simulasi terpadu dan pengguna dapat membuat, menguji, dan mengembangkan robot secara mandiri di dalam lingkungan Webots tanpa memerlukan integrasi dengan platform atau sistem lain.

- **Bagaimana cara ROS dan Webots berkomunikasi?**

Webots mendukung komunikasi dengan ROS melalui antarmuka yang disebut ROS-Bridge. ROS-Bridge memungkinkan Webots untuk berkomunikasi dengan node ROS. Ini memungkinkan pertukaran pesan dan layanan antara Webots dan sistem ROS, memungkinkan pengguna untuk mengontrol simulasi dan menerima data sensor menggunakan ROS. Dengan ROS-Bridge, pengguna dapat membuat node ROS untuk berinteraksi dengan robot simulasi Webots.