Nama: Dilara Kynta Putri Raflita
NIM: 1103204059
Kelas: TK44G4

01 Introduction to ROS

Pada chapter ini memperkenalkan konsep dasar ROS dan sistem manajemen paket ROS untuk memulai pemrograman ROS.

Alasan mengapa harus menggunakan ROS?

ROS adalah kerangka kerja yang fleksibel yang menyediakan berbagai alat dan perpustakaan untuk menulis perangkat lunak robot. Proyek ROS dimulai pada tahun 2007 oleh Morgan Quigley dan pengembangannya berlanjut di Willow Garage, sebuah laboratorium penelitian robotika yang fokus pada pengembangan perangkat keras dan perangkat lunak sumber terbuka untuk robot.

Tujuan ROS adalah membentuk cara standar untuk memprogram robot dengan menawarkan komponen perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan dengan aplikasi robotika kustom.

ROS mirip dengan sistem operasi karena menyediakan fungsi-fungsi seperti abstraksi perangkat keras, manajemen paket, dan toolchain pengembang. Istilah "meta-OS" digunakan untuk menunjukkan bahwa ROS berfungsi sebagai kerangka kerja yang tidak hanya memfasilitasi pengembangan perangkat lunak robot, tetapi juga menyediakan layanan yang mirip dengan sistem operasi dalam mengelola sumber daya dan alur kerja pengembangan.

02 Getting Started With ROS Programming

Dalam tahap awal memulai pemrograman ROS, biasanya dipelajari beberapa konsep dasar dan keterampilan praktis yang diperlukan untuk mengembangkan perangkat lunak robotika menggunakan ROS. Beberapa dari konsep dan keterampilan tersebut termasuk:

1. Konsep Dasar ROS:
    - ROS Master: Memahami peran dan fungsi dari ROS Master, yang mengelola registrasi node, memfasilitasi komunikasi antar node, dan menyediakan layanan informasi tentang sistem.

- ROS Nodes: Mengetahui cara membuat, mengonfigurasi, dan menjalankan node ROS, yang merupakan unit dasar eksekusi dalam arsitektur ROS.

2. Manajemen Paket ROS:
    - ROS Packages: Membuat dan mengatur paket ROS untuk mengorganisir dan menyimpan kode robotika.
    - ROS Package Management: Memahami cara mengelola dependensi, menginstal, dan mengupdate paket ROS.
3. ROS Communication System:
    - ROS Topics: Menggunakan topik untuk mentransmisikan data antar node.
    - ROS Messages: Membuat dan menggunakan pesan ROS untuk menentukan format data yang dikirim melalui topik.
    - ROS Services: Memahami cara membuat dan menggunakan layanan untuk meminta atau memberikan data antar node.
    - Actionlib: Mengetahui penggunaan actionlib untuk tugas-tugas yang memerlukan umpan balik dan keluaran berkelanjutan.
4. Konfigurasi dan Pemrograman Node:
    - ROS Parameter Server: Memanfaatkan parameter server untuk menyimpan dan mengambil parameter konfigurasi.
    - Berkas Peluncuran (Launch Files): Membuat berkas peluncuran untuk mengorganisir dan menjalankan node dengan konfigurasi tertentu.
5. Praktik Pengembangan:
    - Pembuatan dan Implementasi Node ROS: Menciptakan node ROS yang dapat berkomunikasi dengan node lain melalui topik, layanan, atau actionlib.
    - Pembuatan dan Penggunaan Berkas Pesan dan Layanan Kustom: Menambahkan dan menggunakan pesan dan layanan kustom sesuai kebutuhan proyek.
6. Penerapan Konsep pada Kasus Nyata:
    - Aplikasi Topik, Layanan, dan Actionlib: Menerapkan konsep-konsep di atas dalam konteks kasus nyata untuk mengatasi tugas-tugas robotika yang lebih kompleks.

Dengan memahami hal-hal tersebut, pemrograman dapat membangun dasar yang kuat untuk mengembangkan perangkat lunak robotika dengan menggunakan ROS. Ini termasuk memahami cara mengorganisir proyek, berkomunikasi antar komponen, dan menggunakan alat dan konsep khusus yang disediakan oleh ROS.

03 Working with ROS for 3D Modelling

Dalam fase awal pembuatan robot, langkah pertama melibatkan proses desain dan pemodelan. Desain dan pemodelan robot dapat dilakukan menggunakan berbagai alat CAD (Computer-Aided Design) seperti Autodesk Fusion 360, SolidWorks, Blender, dan banyak lagi. Salah satu tujuan utama dari pemodelan robot adalah simulasi.

Alat simulasi robotik dapat memeriksa cacat kritis dalam desain robot dan dapat mengkonfirmasi bahwa robot akan berfungsi sebelum memasuki fase manufaktur. Dalam chapter ini akan membahas proses desain dua robot: manipulator dengan tujuh Derajat Kebebasan (Degrees-of-Freedom/DOF) dan robot penggerak diferensial.

04 Simulating Robots Using ROS and Gazebo

Setelah merancang model 3D sebuah robot, langkah berikutnya adalah mensimulasikannya. Simulasi robot memberikan gambaran tentang bagaimana robot beroperasi dalam lingkungan virtual.

Dalam hal ini, kita akan menggunakan simulator Gazebo (http://www.gazebosim.org/) untuk mensimulasikan manipulator dengan tujuh Derajat Kebebasan (DOF) dan robot bergerak diferensial. Gazebo adalah simulator multi-robot untuk simulasi robotik kompleks di dalam dan di luar ruangan. Dengan Gazebo, kita dapat mensimulasikan robot kompleks, sensor robot, dan berbagai objek 3D. Gazebo sudah memiliki model simulasi dari robot, sensor populer, dan berbagai objek 3D di repositorinya (https://bitbucket.org/osrf/gazebo_models/). Kita dapat langsung menggunakan model-model ini tanpa harus membuat yang baru.

Gazebo terintegrasi dengan baik dengan ROS berkat antarmuka ROS yang tepat, yang memberikan kontrol penuh terhadap Gazebo di dalam lingkungan ROS. Meskipun Gazebo dapat diinstal tanpa ROS, kita sebaiknya menginstal antarmuka ROS-Gazebo agar dapat berkomunikasi dari ROS ke Gazebo.

05 Simulating Robots Using ROS, Coppeliasim, and Webots

Seperti halnya dengan CoppeliaSium, simulator dapat dengan mudah dihubungkan dengan ROS. Dalam bab ini, akan mempelajari cara menyiapkan simulator ini dan menghubungannya dengan jaringan ROS. Beberapa kode awal dibahas untuk memahami bagaimana simulator ini bekerja baik sebagai perangkat lunak mandiri maupun bagaimana mereka digunakan dengan layanan dan topik ROS.

Topik-topik yang akan dibahas yaitu:

1. Menyiapkan CoppeliaSim dengan ROS
2. Mensimulasikan manipulator robot menggunakan CoppeliaSim dan ROS
3. Menyiapkan Webots dengan ROS
4. Menulis pengontrol pertama Anda
5. Menulis node teleoperasi menggunakan webots_ros