UAS Robotika 2023/2024

Nama: Raihan Elsar Kusuma

NIM: 1103174185

Chapter 1: Introduction to ROS

Why should we use ROS?

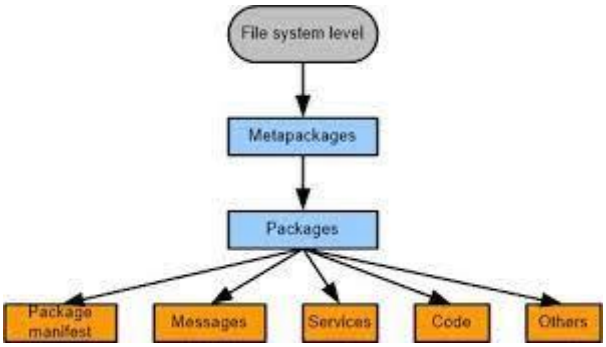
Robot Operating System (ROS) adalah kerangka kerja fleksibel yang menyediakan berbagai alat dan pustaka untuk menulis perangkat lunak robotik. ROS menawarkan beberapa fitur canggih untuk membantu pengembang dalam tugas-tugas seperti pengiriman pesan, komputasi terdistribusi, penggunaan kembali kode, dan implementasi algoritme canggih untuk aplikasi robotik. Proyek ROS dimulai pada tahun 2007 oleh Morgan Quigley dan pengembangannya dilanjutkan di Willow Garage, sebuah laboratorium untuk mengembangkan perangkat keras dan perangkat lunak sumber terbuka untuk robot. Tujuan dari ROS adalah untuk menetapkan cara standar untuk memprogram robot sambil menawarkan perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan dengan aplikasi robotik khusus. Ada banyak alasan untuk memilih ROS sebagai kerangka kerja pemrograman, dan beberapa di antaranya adalah sebagai berikut:

- Kemampuan kelas atas: ROS hadir dengan fungsi yang siap digunakan. Sebagai contoh, Pelokalan dan Pemetaan Simultan (SLAM) dan Adaptive Monte Carlo Localization (AMCL) di ROS dapat digunakan untuk memiliki otonom navigasi di robot seluler, sedangkan paket MoveIt dapat digunakan untuk gerakan perencanaan untuk manipulator robot. Kemampuan ini dapat langsung digunakan dalam perangkat lunak robot tanpa kerumitan. Dalam beberapa kasus, paket-paket ini cukup untuk memiliki tugas robotika inti pada platform yang berbeda. Juga, kemampuan ini sangat tinggi dapat dikonfigurasi; kita dapat menyempurnakan masing-masing menggunakan berbagai parameter.
- Banyak sekali alat: Ekosistem ROS dilengkapi dengan banyak sekali alat untuk melakukan debugging, memvisualisasikan, dan melakukan simulasi. Alat-alat tersebut, seperti rqt_gui, RViz, dan Gazebo, adalah beberapa alat open source terkuat untuk debugging, visualisasi, dan simulasi. Kerangka kerja perangkat lunak yang memiliki banyak alat ini sangat jarang.
- Dukungan untuk sensor dan aktuator kelas atas: ROS memungkinkan kita untuk menggunakan perangkat yang berbeda driver dan paket antarmuka berbagai sensor dan aktuator dalam robotika. Seperti sensor kelas atas termasuk LIDAR 3D, pemindai laser, sensor kedalaman, aktuator, dan banyak lagi. Kita dapat menghubungkan komponen□komponen ini dengan ROS tanpa kerumitan.
- Penanganan sumber daya secara bersamaan: Menangani sumber daya perangkat keras melalui lebih dari dua proses selalu memusingkan. Bayangkan kita ingin memproses sebuah gambar dari sebuah kamera untuk deteksi wajah dan deteksi gerakan; kita dapat menulis kode sebagai entitas tunggal yang dapat melakukan keduanya, atau kita dapat menulis kode berulir tunggal untuk konkurensi. Jika kita ingin menambahkan lebih dari dua fitur ke dalam sebuah thread, aplikasi-aplikasi akan menjadi kompleks

dan sulit untuk di-debug. Tetapi di ROS, kita dapat mengakses perangkat menggunakan topik ROS dari driver ROS. Sejumlah node ROS dapat berlangganan pesan gambar dari driver kamera ROS, dan setiap node bisa memiliki fungsi yang berbeda. Hal ini dapat mengurangi kompleksitas dalam komputasi dan juga meningkatkan kemampuan debugging seluruh sistem.

Understanding the ROS filesystem level

ROS bukan hanya sebuah kerangka pengembangan. Kita dapat menganggap ROS sebagai meta-OS, karena ROS tidak hanya menyediakan alat dan perpustakaan, tetapi juga fungsifungsi mirip sistem operasi, seperti abstraksi perangkat keras, manajemen paket, dan rangkaian alat pengembang. Seperti sistem operasi sungguhan, berkas-berkas ROS diorganisir di dalam hard disk dengan cara tertentu, seperti yang digambarkan dalam diagram berikut:



Gambar 1.1 *ROS Filesystem Level*

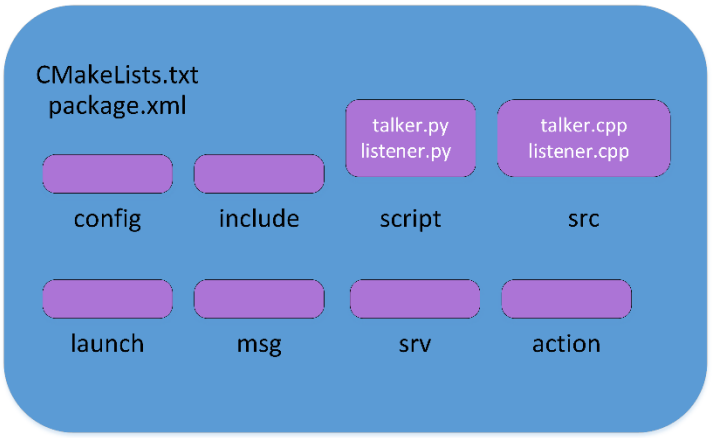
Berikut penjelasan untuk setiap blok dalam sistem berkas ROS:

1. Paket: Paket-paket ROS merupakan elemen sentral dari perangkat lunak ROS. Mereka berisi satu atau lebih program ROS (node), perpustakaan, berkas konfigurasi, dan sebagainya, yang diorganisir bersama sebagai satu unit. Paketpaket merupakan item pembangunan dan rilis atomik dalam perangkat lunak ROS.
2. Manifest Paket: Berkas manifest paket berada di dalam sebuah paket dan berisi informasi tentang paket, penulis, lisensi, dependensi, bendera kompilasi, dan sebagainya. Berkas package.xml di dalam paket ROS adalah berkas manifest dari paket tersebut.
3. Metapaket: Istilah metapaket merujuk pada satu atau lebih paket terkait yang dapat dielompokkan secara longgar. Pada prinsipnya, metapaket merupakan paket virtual yang tidak berisi kode sumber atau berkas-berkas khas yang biasanya ditemukan dalam paket-paket.
4. Manifest Metapaket: Manifest metapaket mirip dengan manifest paket, dengan perbedaan bahwa metapaket mungkin termasuk paket-paket di dalamnya sebagai dependensi runtime dan mendeklarasikan tag ekspor.
5. Pesan (.msg): Kita dapat mendefinisikan pesan kustom di dalam folder msg dalam sebuah paket (my_package/msg/MyMessageType.msg). Ekstensi berkas pesan adalah .msg.

6. Layanan (.srv): Tipe data permintaan dan balasan dapat didefinisikan di dalam folder srv dalam paket (my_package/srv/MyServiceType.srv).
7. Repositori: Sebagian besar paket-paket ROS dipelihara menggunakan Sistem Kontrol Versi (VCS) seperti Git, Subversion (SVN), atau Mercurial (hg). Seperangkat berkas yang ditempatkan pada VCS mewakili sebuah repositori.

ROS Packages

Paket-paket ROS adalah unit-unit perangkat lunak yang berfungsi sebagai komponen-komponen terpisah dalam kerangka kerja Robot Operating System(ROS).



Gambar 1.2 *ROS Packages*

ROS Metapackages

Metapackages adalah paket khusus yang hanya membutuhkan satu file; yaitu file package.xml file. Metapackages hanya mengelompokkan sekumpulan beberapa paket sebagai satu paket logis. Di dalam file package.xml, metapackage berisi tag ekspor, seperti yang ditunjukkan di sini:

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

Selain itu, dalam metapackages, tidak ada dependensi untuk catkin; hanya ada dependensi , yang merupakan paket-paket yang dikelompokkan yang dikelompokkan di dalam metapackage.

Tumpukan navigasi ROS adalah contoh yang baik untuk suatu tempat yang berisi metapaket. Jika ROS dan paket navigasinya telah terinstal, kita dapat mencoba menggunakan perintah berikut dengan beralih ke folder metapaket navigasi:

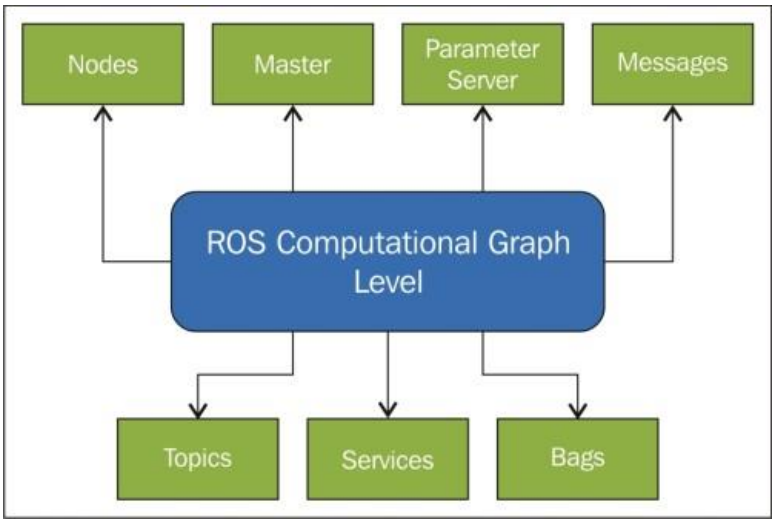
roscd navigation

Buka package.xml menggunakan editor teks favorit Anda (gedit, dalam kasus berikut):

gedit package.xml

Understanding the ROS computation graph level

Komputasi dalam ROS dilakukan menggunakan jaringan dari node-node ROS. Jaringan komputasi ini disebut sebagai grafik komputasi. Konsep-konsep utama dalam grafik komputasi ini adalah node-node ROS, master, parameter server, pesan-pesan, topik-topik, layanan-layanan, dan bags. Setiap konsep dalam grafik ini memberikan kontribusi yang berbeda terhadap grafik ini. Paket-paket terkait komunikasi ROS, termasuk perpustakaan klien inti, seperti roscpp dan rospython, serta implementasi konsep-konsep seperti topik, node, parameter, dan layanan, termasuk dalam satu tumpukan yang disebut ros_comm (http://wiki.ros.org/ros_comm). Tumpukan ini juga terdiri dari alat-alat seperti rostopic, rosparam, rosservice, dan rosnode untuk menyelidiki konsep-konsep sebelumnya. Tumpukan ros_comm mengandung paket-paket middleware komunikasi ROS, dan paketpaket ini secara kolektif disebut sebagai lapisan grafik ROS:



Gambar 1.3 *Structure of the ROS graph layer*

Elemen-elemen baru dalam grafik ROS:

- Node: Proses dengan perhitungan. Dibuat menggunakan perpustakaan klien ROS, memfasilitasi berbagai fungsionalitas dan komunikasi antar node.
- Master: Menyediakan registrasi nama dan koordinasi antar node. Vital dalam pertukaran pesan dan layanan dalam sistem terdistribusi.
- Parameter server: Tempat penyimpanan data terpusat yang dapat diakses dan dimodifikasi oleh semua node, merupakan bagian dari ROS master.
- Topik: Media untuk mengirim dan menerima pesan di ROS. Memungkinkan publikasi dan pelanggan tanpa kesadaran satu sama lain, memungkinkan komunikasi yang terpisah.
- Logging: Sistem penyimpanan data penting seperti data sensor, dikenal sebagai bagfiles, mendukung pengembangan algoritma robot yang kompleks.

1. ***ROS nodes***
   Node-node ROS melakukan komputasi menggunakan perpustakaan klien ROS seperti roscpp dan rospy. Sebuah robot mungkin memiliki banyak node; misalnya, satu node memproses gambar kamera, satu node menangani data serial dari robot, satu node dapat digunakan untuk menghitung odometri, dan seterusnya. Penggunaan node-node dapat membuat sistem toleran terhadap kegagalan. Meskipun sebuah node mengalami kegagalan, seluruh sistem robot masih dapat berfungsi. Node-node juga mengurangi kompleksitas dan meningkatkan kemampuan debug dibandingkan dengan kode monolitik karena setiap node hanya menangani satu fungsi. Semua node yang sedang berjalan sebaiknya memiliki nama yang ditetapkan untuk membantu kita mengidentifikasinya. Sebagai contoh, /camera_node bisa menjadi nama dari sebuah node yang menyiarkan gambar kamera.

2. ***ROS messages***
   Pesan dalam ROS adalah struktur data sederhana yang berisi tipe-tipe lapangan. Pesan ROS mendukung tipe data primitif standar dan array dari tipe-tipe primitif. Untuk mengakses definisi pesan, contohnya std_msgs/msg/String.msg saat menggunakan klien roscpp, kita harus menyertakan std_msgs/String.h untuk definisi pesan string.

   Prerequisites for starting with ROS

   Sebelum memulai dengan ROS, pastikan sistem operasi Anda adalah Ubuntu 20.04 LTS/Debian 10. ROS resmi didukung pada kedua sistem operasi tersebut, dengan rekomendasi menggunakan Ubuntu 20.04 sebagai versi LTS. Untuk memulai dengan ROS 23, lakukan instalasi desktop lengkap ROS Noetic. Versi ini adalah versi stabil terbaru. Temukan panduan instalasi untuk distribusi ROS terbaru di http://wiki.ros.org/noetic/Installation/Ubuntu. Pilih paket ros-noetic-desktop-full dari daftar repositori yang tersedia.

   ROS distributions
   ROS updates are aligned with new ROS distributions, comprising updated core software and a collection of new/updated ROS packages. Its release cycle mirrors that of Ubuntu Linux, with a new ROS version every 6 months. Usually, an LTS version of ROS is released for each Ubuntu LTS version, indicating long-term support (5 years for both ROS and Ubuntu).

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys (**Recommended**)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)

Gambar 1.4 *List of recent ROS Release*

Running the ROS master and the ROS parameter server

Sebelum menjalankan node-node ROS, roscore harus dijalankan untuk memulai ROS master, parameter server, dan node logging rosout. roscore adalah perintah yang menginisialisasi program-program tersebut. Node rosout mengumpulkan pesan log dari node-node ROS lainnya, menyimpannya, dan menyiarkannya kembali ke topik /rosout_agg yang berisi pesan log yang terkumpul. Menjalankan perintah roscore adalah prasyarat sebelum menggunakan node-node ROS, seperti yang ditunjukkan dalam tangkapan layar saat menjalankan roscore di Terminal.