LAPORAN UJIAN AKHIR SEMESTER ROBOTIKA

Chapter 1- Introduction to ROS

Disusun Oleh :

Rai Barokah Utari-1103200066

PROGRAM STUDI TEKNIK KOMPUTER

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

2023

TECHNICAL REPORT CHAPTER-1

1. *Why should we use ROS?*

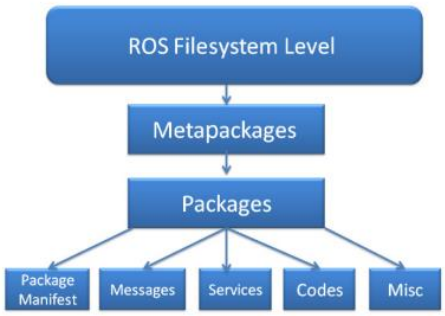
Sistem Operasi Robot adalah kerangka kerja fleksibel yang menyediakan berbagai alat dan perpustakaan untuk menulis perangkat lunak robot. ROS menawarkan beberapa fitur canggih yang membantu pengembang dengan tugas-tugas seperti pertukaran pesan, komputasi terdistribusi, penggunaan kembali kode, dan penerapan algoritma modern untuk aplikasi robot. Proyek ROS dimulai oleh Morgan Quigley pada tahun 2007 dan terus berkembang di Willow Garage, sebuah laboratorium penelitian robotika yang mengembangkan perangkat keras dan perangkat lunak open source untuk robot. Tujuan dari ROS adalah untuk menetapkan cara standar untuk memprogram robot sambil menyediakan komponen perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan ke dalam aplikasi robot khusus. Ada banyak alasan untuk memilih ROS sebagai kerangka pemrograman Anda. Hal ini mencakup fitur-fitur kelas atas, beragam alat yang tersedia, dukungan untuk sensor dan aktuator kelas atas, interoperabilitas antar platform, modularitas, penanganan sumber daya bersama, dan banyak lagi. ROS menyediakan fitur siap pakai seperti paket SLAM, AMCL, dan MoveIt yang dapat dengan mudah digunakan langsung di software robot. Ekosistem ROS memiliki beragam alat sumber terbuka yang jarang ditemukan di kerangka perangkat lunak lain, seperti rqt_gui, RViz, dan Gazebo.

ROS memungkinkan Anda menggunakan driver perangkat dan paket antarmuka untuk berbagai sensor dan aktuator kelas atas dalam robotika. Middleware pertukaran pesan ROS yang disebut Node memungkinkan komunikasi antara program yang berbeda dan dapat diprogram dalam berbagai bahasa seperti C++, C, Python, dan Java. Keuntungan lain dari ROS adalah modularitasnya. Hal ini memungkinkan Anda membuat node berbeda untuk setiap proses, memungkinkan sistem untuk terus berfungsi meskipun satu node gagal. Selain itu, ROS memungkinkan Anda mengurangi kompleksitas pemrosesan data dengan mengakses perangkat melalui driver kamera ROS melalui topik ROS, di mana setiap node memiliki fungsi yang berbeda. Komunitas ROS terus berkembang pesat di seluruh dunia, dan banyak pengguna serta pengembang dari perusahaan robotika kelas atas beralih ke ROS. Tren ini juga terlihat di industri robotika, dimana perusahaan beralih

dari aplikasi robot berpemilik ke ROS. Konsep inti ROS dibagi menjadi tiga tingkatan: tingkat sistem file, tingkat grafik komputer, dan tingkat komunitas.

2. *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 fungsi-fungsi 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. Paket-paket 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.

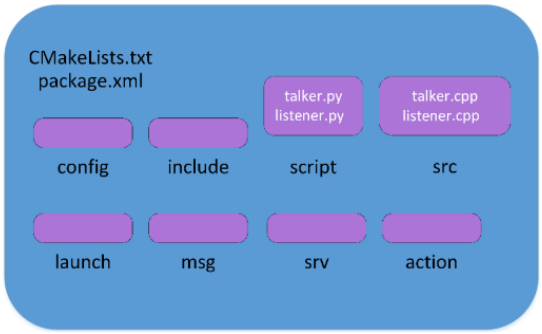
5. Pesan (.msg): Kita dapat mendefinisikan pesan kustom di dalam folder msg dalam sebuah paket (my_package/msg/MyMessageType.msg). Ekstensi berkas pesan adalah .msg.

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.

3. 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 Struktur C++ ROS package

4. ROS metapackages

Metapaket ROS adalah sekelompok paket terkait yang dikelompokkan secara longgar. Secara konseptual, metapaket merupakan entitas virtual yang tidak memiliki kode sumber atau berkas-berkas khas yang biasanya terdapat dalam paket-paket biasa. Metapaket biasanya digunakan untuk mengelompokkan atau mengatur berbagai paket terkait yang memiliki tujuan atau fungsi yang serupa di dalam ekosistem Robot Operating System (ROS).

5. *ROS messages*

Node-node ROS dapat menulis atau membaca data dari berbagai jenis. Jenis-jenis data yang berbeda ini dijelaskan menggunakan bahasa deskripsi pesan yang disederhanakan, juga disebut sebagai pesan ROS. Deskripsi jenis data ini dapat digunakan untuk menghasilkan kode sumber untuk jenis pesan yang sesuai dalam bahasa target yang berbeda. Meskipun kerangka kerja ROS menyediakan kumpulan besar pesan khusus robotika yang telah diimplementasikan, pengembang dapat mendefinisikan jenis pesan mereka sendiri di dalam node-node mereka. Definisi pesan dapat terdiri dari dua jenis: bidang dan konstan. Bidang terbagi menjadi jenis bidang dan nama bidang. Jenis bidang adalah tipe data pesan yang dikirim, sedangkan nama bidang adalah namanya. Berikut adalah tabel yang memperlihatkan beberapa tipe bidang bawaan yang dapat kita gunakan dalam ROS *messages.*

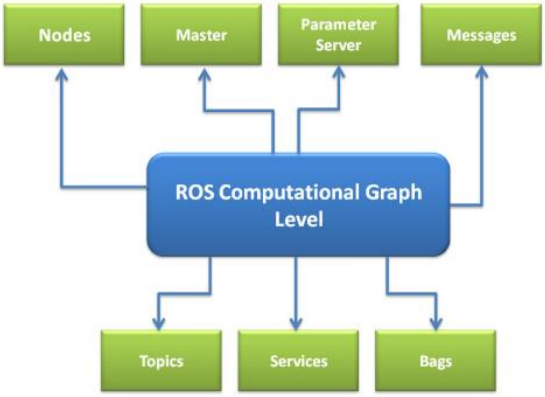
Primitive type	**Serialization**	**C++**	**Python**
`bool(1)`	Unsigned 8-bit int	`uint8_t(2)`	`bool`
`int8`	Signed 8-bit int	`int8_t`	`int`
`uint8`	Unsigned 8-bit int	`uint8_t`	`int (3)`
`int16`	Signed 16-bit int	`int16_t`	`int`
`uint16`	Unsigned 16-bit int	`uint16_t`	`int`
`int32`	Signed 32-bit int	`int32_t`	`int`
`uint32`	Unsigned 32-bit int	`uint32_t`	`int`
`int64`	Signed 64-bit int	`int64_t`	`long`
`uint64`	Unsigned 64-bit int	`uint64_t`	`long`
`float32`	32-bit IEEE float	`float`	`float`
`float64`	64-bit IEEE float	`double`	`float`
`string`	ascii string(4)	`std::string`	`string`
`time`	secs/nsecs unsigned 32-bit ints	`ros::Time`	`rospy.Time`
`duration`	secs/nsecs signed 32-bit ints	`ros::Duration`	`rospy.Duration`

6. *The ROS services*

Layanan ROS merupakan jenis komunikasi permintaan/respons antara node-node ROS. Satu node akan mengirim permintaan dan menunggu hingga mendapatkan respons dari node lainnya. Serupa dengan definisi pesan saat menggunakan file .msg, kita harus mendefinisikan definisi layanan dalam file lain yang disebut .srv, yang harus disimpan di dalam subdirektori srv dari paket tersebut.

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 paket-paket 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).

Gambar 1.4 *List of recent ROS releases*

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.