

Technical Report from Mastering ROS

Chapter 1 – Introduction to ROS



Oleh:

Nama : Alifia Mutiara Rahma

NIM : 1103200025

PROGRAM STUDI TEKNIK KOMPUTER

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

2023

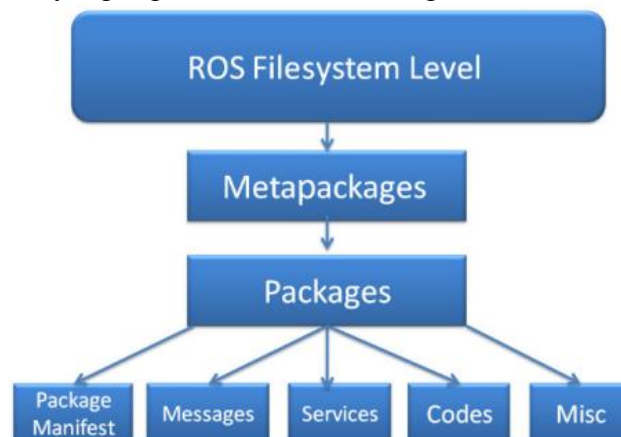
I. *Why should we learn ROS?*

Sistem Operasi Robot (ROS) adalah kerangka kerja yang fleksibel, menyediakan alat dan perpustakaan untuk menulis perangkat lunak robot, dengan tujuan membentuk cara standar memprogram robot dan menawarkan komponen perangkat lunak siap pakai yang dapat dengan mudah diintegrasikan dengan aplikasi robotika kustom, serta mendukung pengembangan fitur tinggi dan berbagai sensor dan aktuator canggih tanpa kesulitan. Dalam ROS, banyak alat sumber terbuka untuk debugging, visualisasi, dan simulasi, dan ekosistemnya terus berkembang pesat dengan banyak pengguna dan pengembang di seluruh dunia. Tren adaptasi perangkat lunak ROS juga terlihat di industri robotika, di mana perusahaan beralih dari aplikasi robotika proprietary ke ROS.

Belajar ROS dianggap penting karena memberikan fleksibilitas dan menyediakan alat yang diperlukan untuk mengembangkan perangkat lunak robotika, dengan fungsionalitas siap pakai, dukungan untuk sensor dan aktuator canggih, serta operabilitas antar-platform. Keberadaan ekosistem ROS yang kaya dengan alat debugging, visualisasi, dan simulasi membuatnya dianggap sebagai pilihan yang langka dan sangat berguna. Risiko kegagalan sistem secara menyeluruh dapat diatasi melalui modularitasnya, sementara penanganan sumber daya konkuren memungkinkan pengembangan aplikasi yang kompleks tanpa kesulitan. Relevansi ROS dalam dunia pengembangan robotika modern ditegaskan oleh pertumbuhan pesat komunitas ROS dan adaptasinya oleh perusahaan robotika tingkat tinggi dan industri.

II. *Understanding the ROS filesystem level*

ROS dianggap lebih dari sekadar kerangka pengembangan sekaligus dapat disebut sebagai meta-OS, mengingat tidak hanya menyediakan alat dan perpustakaan, tetapi juga fungsi mirip sistem operasi, termasuk abstraksi perangkat keras, manajemen paket, dan rangkaian alat pengembang. Seperti sistem operasi yang sesungguhnya, file-file ROS diatur di hard disk dengan cara tertentu, seperti yang digambarkan dalam diagram berikut.



- ROS2 Packages

Berikut beberapa ROS2 Packages:

- config: Semua file konfigurasi yang digunakan dalam paket ROS ini disimpan di dalam folder ini. Folder ini dibuat oleh pengguna, dan merupakan praktik umum untuk memberi nama folder ini config karena di sinilah kita menyimpan file konfigurasi.
- include/package_name: Folder ini berisi header dan pustaka yang perlu kita gunakan di dalam paket.

- script: Folder ini berisi skrip Python yang dapat dieksekusi. Pada diagram blok, kita dapat melihat dua contoh skrip.
 - src: Folder ini menyimpan kode sumber C++.
 - launch: Folder ini berisi file peluncuran yang digunakan untuk menjalankan satu atau lebih node ROS.
 - msg: Folder ini berisi definisi pesan kustom.
 - srv: Folder ini berisi definisi layanan.
 - action: Folder ini berisi file aksi. Kita akan mempelajari lebih lanjut tentang jenis file ini di bab berikutnya.
 - package.xml: Ini adalah file manifes paket dari paket ini.
 - CMakeLists.txt: File ini berisi direktif untuk mengompilasi paket.
 - Kita perlu mengetahui beberapa perintah untuk membuat, memodifikasi, dan bekerja dengan paket ROS. Berikut beberapa perintah yang dapat digunakan untuk bekerja dengan paket ROS:
 - catkin_create_pkg: Perintah ini digunakan untuk membuat paket baru.
 - rospack: Perintah ini digunakan untuk mendapatkan informasi tentang paket dalam sistem file.
 - catkin_make: Perintah ini digunakan untuk membangun paket-paket dalam ruang kerja.
 - roscdep: Perintah ini akan menginstal dependensi sistem yang diperlukan untuk paket ini.
 - Untuk bekerja dengan paket, ROS menyediakan perintah mirip bash yang disebut rosbash (<http://wiki.ros.org/rosbash>), yang dapat digunakan untuk menavigasi dan memanipulasi paket ROS. Berikut beberapa perintah rosbash:
 - roscd: Perintah ini digunakan untuk mengubah direktori saat ini menggunakan nama paket, nama stack, atau lokasi khusus. Jika kita memberikan argumen berupa nama paket, itu akan beralih ke folder paket tersebut.
 - roscp: Perintah ini digunakan untuk menyalin file dari paket.
 - rosed: Perintah ini digunakan untuk mengedit file menggunakan editor vim.
 - rosrunc: Perintah ini digunakan untuk menjalankan executable di dalam paket.
- ROS metapackages

Metapackages adalah paket khusus yang memerlukan hanya satu file, yaitu file package.xml.

Metapackages secara sederhana mengelompokkan serangkaian paket menjadi satu paket logis tunggal. Dalam file package.xml, metapackages berisi tag ekspor. Dalam metapackages, tidak ada dependensi `<buildtool_depend>` untuk catkin; hanya ada dependensi `<run_depend>`, yang merupakan paket yang dikelompokkan di dalam metapackages. Stack navigasi ROS adalah contoh bagus dari tempat yang berisi metapackages.
 - ROS messages

Node ROS dapat menulis atau membaca data berbagai jenis. Berbagai jenis data 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 berbagai bahasa target.

Meskipun kerangka kerja ROS menyediakan sejumlah besar pesan yang khusus untuk robot yang telah diimplementasikan, pengembang dapat menentukan jenis pesan sendiri di dalam node mereka. Definisi pesan dapat terdiri dari dua jenis: bidang (fields) dan konstan (constants). Bidang dibagi menjadi jenis bidang dan nama bidang. Jenis bidang adalah tipe data dari pesan yang dikirim, sedangkan nama bidang adalah namanya. Berikut adalah contoh definisi pesan: int32 number, string name, float32 speed

Di sini, bagian pertama adalah jenis bidang dan yang kedua adalah nama bidang. Jenis bidang adalah tipe data, dan nama bidang dapat digunakan untuk mengakses nilai dari pesan. Sebagai contoh, kita dapat menggunakan `msg.number` untuk mengakses nilai dari nomor di dalam pesan.

ROS menyediakan serangkaian file pesan yang lebih kompleks dan terstruktur yang dirancang untuk mencakup kebutuhan aplikasi tertentu, seperti pertukaran informasi geometris umum (`geometry_msgs`) atau sensor (`sensor_msgs`). Pesan-pesan ini terdiri dari berbagai jenis primitif. Jenis khusus dari pesan ROS disebut sebagai header pesan. Header ini dapat membawa informasi seperti waktu, bingkai referensi, atau `frame_id`, dan nomor urut. Dengan menggunakan header, akan mendapatkan pesan berurutan dan kejelasan lebih tentang komponen mana yang mengirimkan pesan saat ini. Alat perintah `rosmmsg` dapat digunakan untuk memeriksa header pesan dan jenis bidang.

- The ROS services

ROS services adalah jenis komunikasi permintaan/tanggapan antara node ROS. Salah satu node akan mengirimkan permintaan dan menunggu hingga mendapatkan tanggapan dari yang lain. Sama seperti definisi pesan saat menggunakan file `.msg`, harus mendefinisikan definisi layanan dalam file lain yang disebut `.srv`, yang harus disimpan di dalam subdirektori `srv` dari paket. Bagian pertama adalah tipe pesan permintaan, yang dipisahkan oleh `---`, sementara bagian berikutnya berisi tipe pesan tanggapan.

III. *Understanding ROS computation graph level*

Komputasi dalam ROS dilakukan menggunakan jaringan dari node-node ROS. Jaringan komputasi ini disebut sebagai grafik komputasi. Konsep utama dalam grafik komputasi melibatkan node-node ROS, master, parameter server, pesan, topik, layanan, dan bags (rekaman data). Setiap konsep dalam grafik ini memberikan kontribusi ke grafik ini dengan cara yang berbeda. Paket-paket terkait komunikasi ROS, termasuk perpustakaan klien inti, seperti `roscpp` dan `rospy`, dan 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 `roscall` untuk memeriksa konsep-konsep sebelumnya.

Beberapa elemen baru dari grafik ROS adalah sebagai berikut:

- Nodes: Nodes adalah proses yang memiliki komputasi. Setiap node ROS ditulis menggunakan perpustakaan klien ROS. Dengan menggunakan API perpustakaan klien, kita dapat mengimplementasikan berbagai fungsionalitas ROS, seperti metode komunikasi antara node, yang sangat berguna ketika node-node yang berbeda pada robot kita harus bertukar informasi antara mereka. Salah satu tujuan dari node ROS adalah membangun proses sederhana daripada proses besar dengan semua fungsionalitas yang diinginkan. Karena merupakan struktur sederhana, node ROS mudah untuk di-debug.

- Master: ROS master menyediakan proses registrasi dan pencarian nama untuk node-node lainnya. Node-node tidak akan dapat menemukan satu sama lain, bertukar pesan, atau memanggil layanan tanpa 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 kita menyimpan data di lokasi pusat. 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, kita dapat mengatakan bahwa node tersebut menerbitkan topik. Ketika sebuah node menerima pesan melalui topik, kita dapat mengatakan bahwa node tersebut berlangganan topik. Node yang menerbitkan dan berlangganan tidak menyadari keberadaan satu sama lain. Kita bahkan dapat berlangganan ke topik yang mungkin tidak memiliki penerbit. Singkatnya, produksi informasi dan konsumsinya terpisah. Setiap topik memiliki nama unik, dan setiap node dapat mengakses topik ini dan mengirimkan data melalui topik tersebut selama mereka memiliki tipe pesan yang sesuai.
 - Pencatatan: ROS menyediakan sistem pencatatan untuk menyimpan data, seperti data sensor, yang mungkin sulit dikumpulkan tetapi diperlukan untuk mengembangkan dan menguji algoritma robot. Ini dikenal sebagai bagfiles. Bagfiles adalah fitur yang sangat berguna ketika kita bekerja dengan mekanisme robot yang kompleks.
- ROS nodes

Node-node ROS melakukan komputasi menggunakan perpustakaan klien ROS seperti roscpp dan rospy. Sebuah robot mungkin terdiri dari banyak node; misalnya, satu node memproses gambar kamera, satu node menangani data serial dari robot, satu node dapat digunakan untuk menghitung odometri, dan sebagainya.

Menggunakan node dapat membuat sistem toleran terhadap kesalahan. Bahkan jika sebuah node mengalami kegagalan, seluruh sistem robot masih dapat berfungsi. Node-node juga mengurangi kompleksitas dan meningkatkan kemampuan debugging dibandingkan dengan kode monolitik karena setiap node hanya menangani satu fungsi.
 - ROS messages

Seperti yang telah dibahas sebelumnya, pesan-pesan merupakan struktur data sederhana yang berisi jenis bidang. Tipe data primitif standar dan larik dari tipe-tipe primitif didukung oleh pesan-pesan ROS. Definisi pesan dapat diakses menggunakan metode berikut. Sebagai contoh, untuk mengakses std_msgs/msg/String.msg ketika menggunakan klien roscpp, std_msgs/String.h harus disertakan untuk definisi pesan string. Selain tipe data pesan, perbandingan checksum MD5 digunakan oleh ROS untuk memastikan apakah penerbit dan pelanggan bertukar tipe data pesan yang sama. Alat bawaan ROS yang disebut rosmmsg digunakan untuk mengumpulkan informasi tentang pesan-pesan ROS. Beberapa parameter yang digunakan bersama dengan rosmmsg antara lain:

 - rosmmsg show [message_type]: Deskripsi pesan ditampilkan.
 - rosmmsg list: Daftar semua pesan ditampilkan.
 - rosmmsg md5 [message_type]: md5sum dari sebuah pesan ditampilkan.
 - rosmmsg package [package_name]: Daftar pesan dalam suatu paket ditampilkan.

- `rosmmsg packages [package_1] [package_2]`: Daftar semua paket yang berisi pesan ditampilkan.
- **ROS topics**
Topik-topik ROS dihubungkan oleh node-node menggunakan metode transport berbasis TCP/IP yang dikenal sebagai TCPROS. Metode ini merupakan metode transport default yang digunakan dalam ROS. Jenis komunikasi lainnya adalah UDPROS, yang memiliki latensi rendah dan transport yang longgar, dan hanya cocok untuk teleoperasi. Alat `rostopic` ROS dapat digunakan untuk mengumpulkan informasi tentang topik-topik ROS.
- **ROS services**
Dalam layanan ROS, satu node bertindak sebagai server ROS di mana klien layanan dapat meminta layanan dari server. Jika server menyelesaikan rutinitas layanan, ia akan mengirimkan hasilnya kepada klien layanan. Sebagai contoh, pertimbangkan sebuah node yang dapat memberikan jumlah dari dua angka yang diterima sebagai input sambil mengimplementasikan fungsionalitas ini melalui layanan ROS. Node-node lain dalam sistem kita mungkin meminta jumlah dari dua angka melalui layanan ini. Dalam situasi ini, topik-topik digunakan untuk mengalirkan aliran data yang kontinu.
- **ROS bagfiles**
Perintah `rosbag` digunakan untuk bekerja dengan file `rosbag` dalam ROS, yang berfungsi sebagai penyimpanan data pesan ROS yang mengalir melalui topik-topik. File bag menggunakan ekstensi `.bag` dan diciptakan dengan menggunakan perintah `rosbag record`. Perintah tersebut akan berlangganan ke satu atau lebih topik, dan menyimpan data pesan ke dalam file seiring dengan penerimaannya. File bag dapat memutar ulang topik yang sama dari mana data direkam, dan juga dapat melakukan remapping terhadap topik-topik yang sudah ada.
- **The ROS master**
ROS master mirip dengan server DNS, dalam artian bahwa ia mengaitkan nama unik dan ID ke elemen-elemen ROS yang aktif dalam sistem kita. Ketika suatu node mulai dalam sistem ROS, ia akan mencari ROS master dan mendaftarkan nama nodenya di dalamnya. Sehingga, ROS master memiliki rincian dari semua node yang saat ini berjalan di sistem ROS. Ketika rincian dari salah satu node berubah, itu akan menghasilkan pemanggilan kembali dan memperbarui node dengan rincian terbaru. Rincian node ini berguna untuk menghubungkan setiap node.

IV. *ROS community level*

Ini adalah sumber daya ROS yang memungkinkan komunitas baru dalam ROS untuk bertukar perangkat lunak dan pengetahuan. Beberapa sumber daya dalam komunitas ini antara lain:

- **Distribusi:** Mirip dengan distribusi Linux, distribusi ROS adalah kumpulan metapaket berjenis versi yang dapat kita instal. Distribusi ROS memungkinkan kita untuk dengan mudah menginstal dan mengumpulkan perangkat lunak ROS. Mereka juga menjaga konsistensi versi di sepanjang serangkaian perangkat lunak.

- **Repositori:** ROS bergantung pada jaringan repositori kode yang terfederasi, di mana berbagai institusi dapat mengembangkan dan merilis komponen perangkat lunak robot mereka sendiri.
- **Wiki ROS:** Wiki komunitas ROS adalah forum utama untuk mendokumentasikan informasi tentang ROS. Siapa pun dapat mendaftar untuk akun dan menyumbangkan dokumentasi mereka sendiri, memberikan koreksi atau pembaruan, menulis tutorial, dan lainnya.
- **Sistem tiket bug:** Jika kita menemukan bug dalam perangkat lunak yang ada atau perlu menambahkan fitur baru, kita dapat menggunakan sumber daya ini.
- **Daftar milis:** Kita dapat menggunakan milis ROS-users untuk bertanya tentang perangkat lunak ROS dan membagikan masalah program dengan komunitas.
- **ROS Answers:** Sumber daya situs web ini membantu kita untuk bertanya tentang ROS. Jika kita memposting pertanyaan kita di situs ini, pengguna ROS lain dapat melihatnya dan memberikan solusi.
- **Blog:** Blog ROS diperbarui dengan berita, foto, dan video terkait komunitas ROS (<http://www.ros.org/news>).

ROS distributions

Pembaruan ROS dirilis bersama dengan distribusi ROS yang baru. Distribusi ROS yang baru terdiri dari versi terbaru perangkat lunak intinya dan sejumlah paket ROS baru/diperbarui. ROS mengikuti siklus rilis yang sama dengan distribusi Linux Ubuntu: versi baru ROS dirilis setiap 6 bulan. Umumnya, untuk setiap versi Ubuntu LTS, versi ROS LTS dirilis.

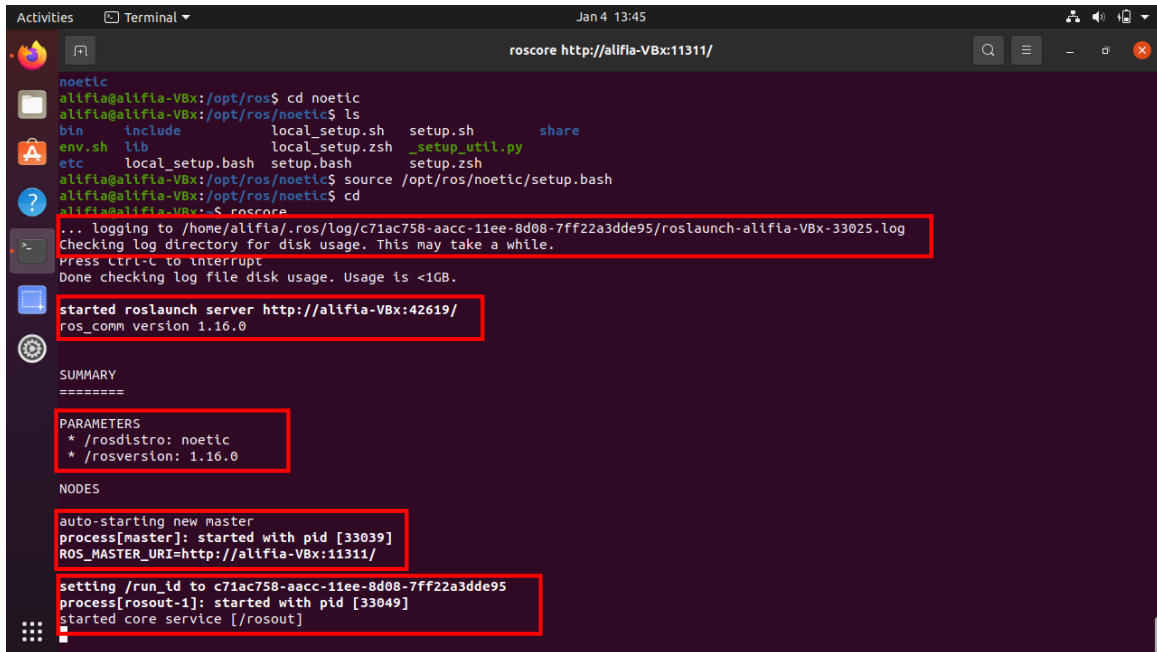
Running the ROS master and the ROS parameter server

Sebelum menjalankan node-node ROS apa pun, ROS master dan parameter server ROS harus diaktifkan. ROS master dan parameter server ROS dapat diaktifkan menggunakan satu perintah yang disebut `roscore`, yang akan memulai program-program berikut:

- ROS master
- Parameter server ROS
- Node logging `rosout`

Node `rosout` akan mengumpulkan pesan log dari node-node ROS lainnya dan menyimpannya dalam file log, serta meneruskan pesan log yang terkumpul ke topik lain. Topik `/rosout` diterbitkan oleh node-node ROS menggunakan perpustakaan klien ROS seperti `roscpp` dan `rospy`, dan topik ini disubscribe oleh node `rosout`, yang meneruskan pesan tersebut ke topik lain yang disebut `/rosout_agg`. Topik ini berisi aliran pesan log yang terkumpul. Perintah `roscore` harus dijalankan sebagai prasyarat sebelum menjalankan node-node ROS apa pun. Tangkapan layar berikut menunjukkan pesan-pesan yang dicetak ketika perintah `roscore` dijalankan dalam Terminal.

Roscore



```
noetic
alifia@alifia-VBx:/opt/ros$ cd noetic
alifia@alifia-VBx:/opt/ros/noetic$ ls
bin      include      local_setup.sh  setup.sh        share
env.sh   lib          local_setup.zsh _setup_util.py
etc      local_setup.bash  setup.bash      setup.zsh
alifia@alifia-VBx:/opt/ros/noetic$ source /opt/ros/noetic/setup.bash
alifia@alifia-VBx:/opt/ros/noetic$ cd
alifia@alifia-VBx:~$ roscore
... logging to /home/alifia/.ros/log/c71ac758-aacc-11ee-8d08-7ff22a3dde95/roslaunch-alifia-VBx-33025.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://alifia-VBx:42619/
ros_comm version 1.16.0

SUMMARY
=====
PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [33039]
ROS_MASTER_URI=http://alifia-VBx:11311/

setting /run_id to c71ac758-aacc-11ee-8d08-7ff22a3dde95
process[rosout-1]: started with pid [33049]
started core service [/rosout]
```

Berikut adalah penjelasan setiap bagian saat menjalankan perintah roscore di Terminal:

1. Pada bagian 1, dapat diamati bahwa sebuah file log dibuat di dalam folder `~/ros/log` untuk mengumpulkan log dari node-node ROS. File ini dapat digunakan untuk tujuan debugging.
2. Pada bagian 2, sebuah file peluncuran ROS yang disebut `roscore.xml` dimulai oleh perintah. Ketika sebuah file peluncuran dimulai, rosmaster dan server parameter ROS secara otomatis dimulai. Perintah `roslaunch` adalah skrip Python yang dapat memulai rosmaster dan server parameter ROS setiap kali mencoba menjalankan file peluncuran. Alamat server parameter ROS di dalam port ditunjukkan oleh bagian ini.
3. Pada bagian 3, parameter seperti `rostdistro` dan `rosversion` ditampilkan di Terminal. Parameter-parameter ini ditampilkan saat menjalankan `roscore.xml`. Rincian lebih lanjut tentang `roscore.xml` akan dibahas pada bagian berikutnya.
4. Pada bagian 4, terlihat bahwa node rosmaster dimulai dengan menggunakan `ROS_MASTER_URI`, yang telah ditentukan sebelumnya sebagai variabel lingkungan.
5. Pada bagian 5, terlihat bahwa node `rosout` dimulai, yang akan memulai berlangganan ke topik `/rosout` dan mengirimkannya kembali ke `/rosout_agg`.

V. Summary

Framework perangkat lunak ROS saat ini semakin populer di kalangan ahli robotika. Pengetahuan tentang ROS menjadi semakin penting bagi mereka yang berencana membangun karir sebagai insinyur robotika dalam beberapa tahun mendatang. Dalam bab ini, dasar-dasar ROS telah dibahas, terutama dengan tujuan menyegarkan pemahaman konsep jika sebelumnya Anda sudah familiar dengan ROS. Pentingnya mempelajari ROS dan keunggulannya di antara platform perangkat lunak robotika saat ini telah dibahas. Konsep dasar seperti ROS master dan parameter server telah dijelajahi, serta penjelasan tentang cara kerja roscore telah diberikan. Pada bab berikutnya, manajemen paket ROS akan diperkenalkan dan beberapa contoh praktis dari sistem komunikasi ROS akan dibahas.

VI. Daftar Pustaka

L. Joseph dan J. Cacace, Mastering ROS for Robotics Programming Third Edition, Birmingham: Packt Publishing Ltd, 2021.