

Tugas UAS

Robotika

Nama : A. Muh. Faizhul Islam

Kelas : TK 45 01

NIM : 1103210074

1. Publisher dan Subscriber di ROS

Publisher adalah komponen yang bertugas untuk mengirimkan (memublikasikan) pesan ke topik tertentu di ROS. Topik ini merupakan saluran komunikasi tempat pesan dikirim dan diterima. Sebuah Publisher dapat diprogram untuk memublikasikan pesan secara periodik (misalnya, setiap detik) atau berdasarkan event tertentu. Setiap Publisher terkait dengan tipe pesan tertentu (misalnya `std_msgs/String`, `geometry_msgs/Twist`, dll). Tipe pesan ini mendefinisikan struktur data yang dipublikasikan.

Cara kerja Publisher:

- Publisher membuat komunikasi dengan topik tertentu.
- Publisher mengirimkan pesan (data) ke topik tersebut.
- Node lain yang tertarik pada topik tersebut dapat menerima pesan ini melalui Subscriber.

```
Select roscore http://0a9d27a51c73:11311/
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>docker pull ros:noetic
noetic: Pulling from library/ros
Digest: sha256:456b9a58b63e259c388fb59d8f6c1db6b17967f7b515ebb5aefff4eb2fba888
Status: Image is up to date for ros:noetic
docker.io/library/ros:noetic

C:\Users\ASUS>docker images -a
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
utrarobosoccer/noetic latest       2c24443e0978  4 years ago  1.31GB
ros                  noetic      456b9a58b63e  4 years ago  1.31GB

C:\Users\ASUS>docker run -it --rm ros:noetic
root@0a9d27a51c73:/# roscore
... logging to /root/.ros/log/028278ce-ca41-11ef-9377-0242ac110002/roslaunch-0a9d27a51c73-28.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://0a9d27a51c73:43523/
ros_comm version 1.17.0

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.17.0

NODES

auto-starting new master
process[master]: started with pid [36]
ROS_MASTER_URI=http://0a9d27a51c73:11311/

setting /run_id to 028278ce-ca41-11ef-9377-0242ac110002
process[roscout-1]: started with pid [46]
started core service [/roscout]
```

1. *docker pull ros:noetic*

- Perintah: `docker pull ros:noetic`
- Fungsi: Perintah ini digunakan untuk mengunduh (download) image Docker ROS versi Noetic dari Docker Hub ke komputer.

Penjelasan lebih lanjut:

`docker pull` adalah perintah untuk mengunduh image Docker dari registry (dalam hal ini Docker Hub).

- `ros:noetic` adalah nama image yang ingin diunduh, di mana `ros` adalah nama repositori dan `noetic` adalah tag untuk versi ROS (versi 1.15 dari ROS). Tag ini menunjukkan versi tertentu dari image ROS yang dibutuhkan.
- Tujuan: Setelah perintah ini dijalankan, image Docker ROS Noetic akan diunduh dan disimpan di sistem, dan siap untuk digunakan untuk menjalankan container ROS.

2. *docker images -a*

- Perintah: `docker images -a`
- Fungsi: Perintah ini digunakan untuk menampilkan semua image Docker yang ada di sistem, termasuk image yang tidak lagi digunakan (dangling images).

Penjelasan lebih lanjut:

- `docker images`: Perintah ini menampilkan daftar image Docker yang ada di sistem.
- `-a`: Opsi `-a` (atau `--all`) menunjukkan bahwa perintah ini akan menampilkan semua image Docker yang ada, termasuk image yang tidak terpakai atau dalam status "dangling".
- Tujuan: Perintah ini memungkinkan untuk melihat image yang sudah ada, termasuk ROS Noetic yang baru saja diunduh dengan perintah sebelumnya.

3. *docker run -it --rm ros:noetic*

- Perintah: `docker run -it --rm ros:noetic`
- Fungsi: Perintah ini digunakan untuk menjalankan container baru dari image ROS Noetic yang telah diunduh dan masuk ke dalam container tersebut dalam mode interaktif.

Penjelasan lebih lanjut:

- `docker run`: Perintah ini digunakan untuk membuat dan menjalankan container dari image Docker yang telah ada.
- `-it`: Ini adalah dua opsi yang digunakan bersama:
- `-i` (interactive): Menyediakan terminal interaktif untuk menjalankan perintah di dalam container.
- `-t` (tty): Memberikan terminal virtual yang memungkinkan Anda berinteraksi dengan container secara lebih nyaman (seperti terminal biasa).

- `--rm`: Opsi ini memastikan bahwa container akan dihapus secara otomatis setelah Anda keluar atau container dihentikan.
- `ros:noetic`: Ini adalah nama image yang digunakan untuk membuat container, yaitu image ROS Noetic yang telah Anda unduh sebelumnya.

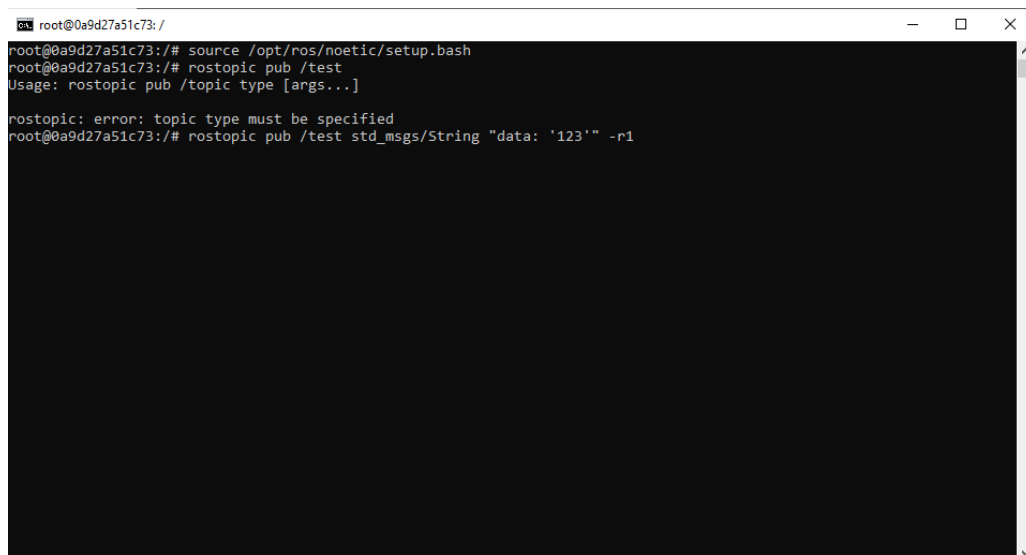
4. roscore

adalah perintah penting dalam ROS (Robot Operating System) yang berfungsi untuk menjalankan ROS Master, yang merupakan komponen pusat dalam sistem ROS. ROS Master memungkinkan komunikasi antar node di dalam sistem ROS dengan mengelola informasi tentang node dan topik.

Penjelasan tentang roscore:

Fungsi Utama:

- roscore bertugas untuk memulai dan menjalankan server utama dalam ROS yang disebut ROS Master.
- ROS Master mengelola komunikasi antara node, topik, layanan, dan parameter dalam sistem ROS.
- Tanpa menjalankan roscore, node-node di ROS tidak dapat berkomunikasi satu sama lain, karena ROS Master yang mengatur bagaimana node terhubung dan berkomunikasi melalui topik dan layanan.



```

root@0a9d27a51c73: /
root@0a9d27a51c73: /# source /opt/ros/noetic/setup.bash
root@0a9d27a51c73: /# rostopic pub /test
Usage: rostopic pub /topic type [args...]

rostopic: error: topic type must be specified
root@0a9d27a51c73: /# rostopic pub /test std_msgs/String "data: '123'" -r1

```

source /opt/ros/ro/noetic/setup.bash:

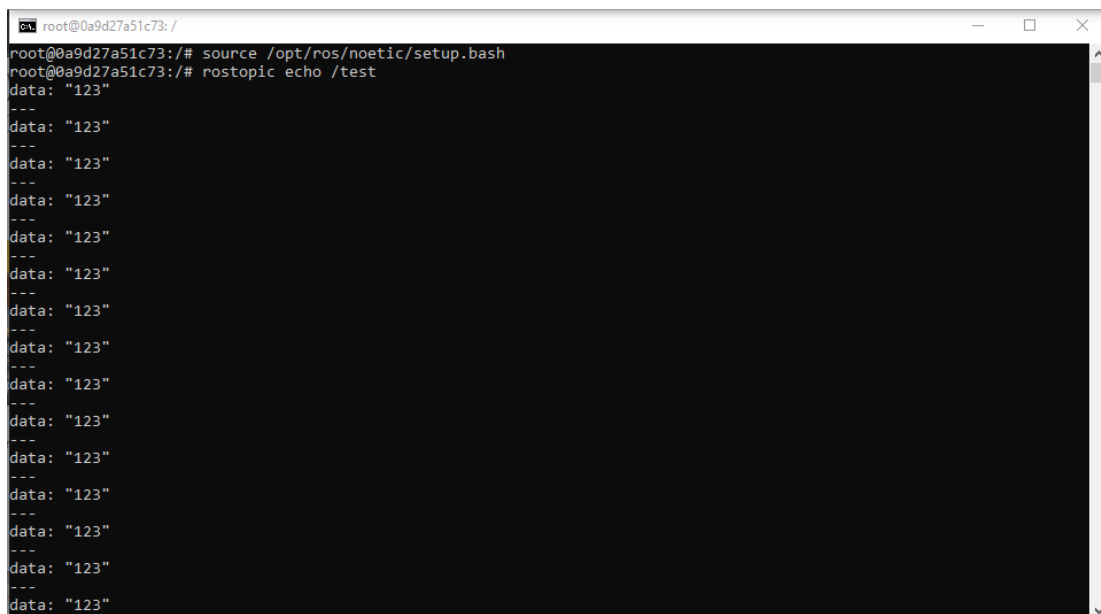
- Perintah ini digunakan untuk menginisialisasi lingkungan ROS (Robot Operating System) agar siap digunakan. ROS memiliki berbagai variabel lingkungan dan pustaka yang perlu diatur sebelum menjalankan perintah ROS lainnya. Perintah ini mengarah ke file `setup.bash` yang ada di instalasi ROS Noetic yang digunakan untuk menyiapkan lingkungan kerja.

rostopic pub /test:

- Ini adalah perintah untuk menerbitkan (publish) pesan ke suatu topik di ROS. /test adalah nama topik yang digunakan untuk komunikasi antara node di ROS. Biasanya, rostopic pub digunakan untuk mengirim pesan ke suatu topik dalam sistem ROS, dan perintah ini harus diikuti dengan tipe pesan yang akan dipublikasikan.

rostopic pub /test std_msgs/String "data: '123'" -r1:

- Perintah lengkap ini juga digunakan untuk mem-publish pesan ke topik /test, namun kali ini lebih spesifik.
- std_msgs/String adalah jenis pesan yang akan dipublikasikan, yang berarti pesan yang dikirim adalah tipe string.
- "data: '123'" adalah nilai yang akan dikirim dalam pesan. Dalam hal ini, pesan berisi string dengan nilai '123'.
- -r1 adalah parameter untuk mengatur rate atau laju publikasi pesan. Dalam hal ini, laju pengiriman pesan adalah 1 Hz, yang berarti pesan akan dipublikasikan setiap detik.



```
root@0a9d27a51c73: /  
root@0a9d27a51c73:/# source /opt/ros/noetic/setup.bash  
root@0a9d27a51c73:/# rostopic echo /test  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"  
---  
data: "123"
```

source /opt/ros/ros/noetic/setup.bash:

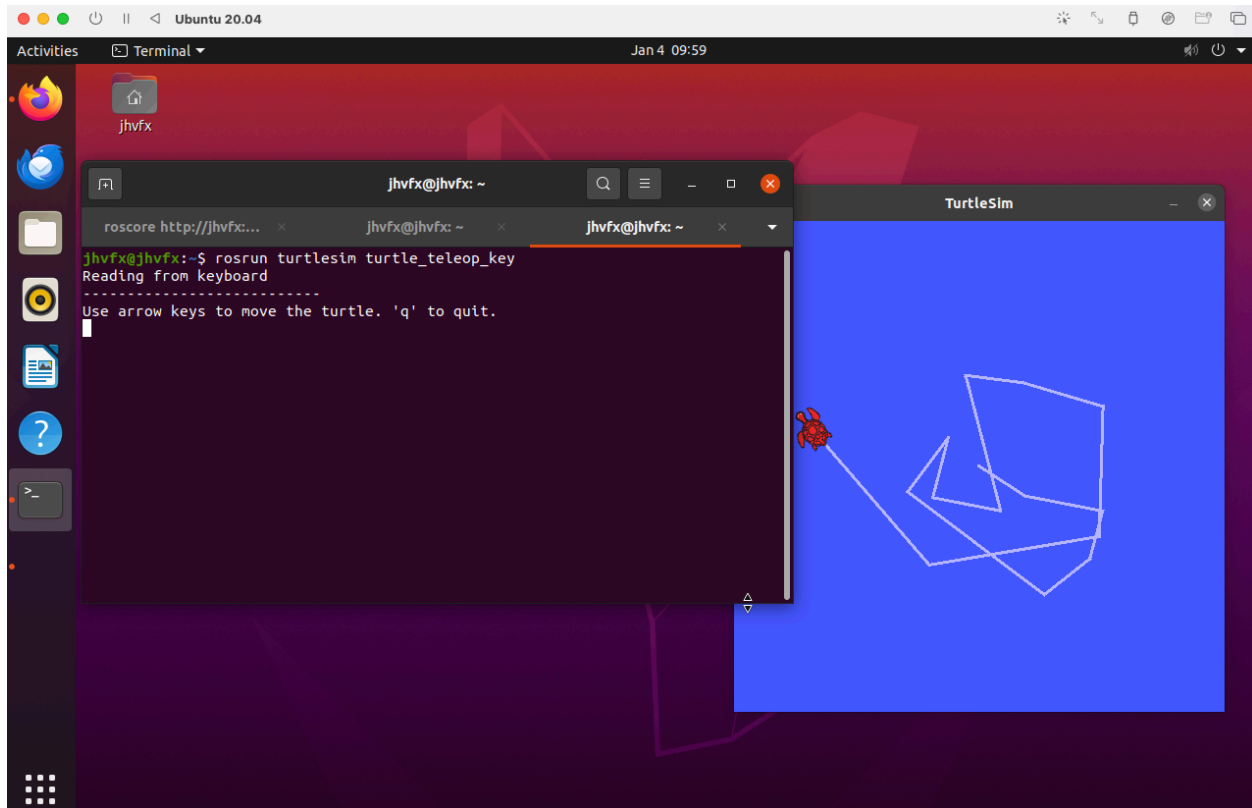
- perintah ini digunakan untuk menginisialisasi lingkungan ROS (Robot Operating System) dengan memuat konfigurasi yang ada di file setup.bash. Ini memastikan bahwa ROS dapat berjalan dengan baik dan dapat mengenali perintah-perintah yang digunakan dalam ROS, seperti rostopic dan lainnya.

rostopic echo /test:

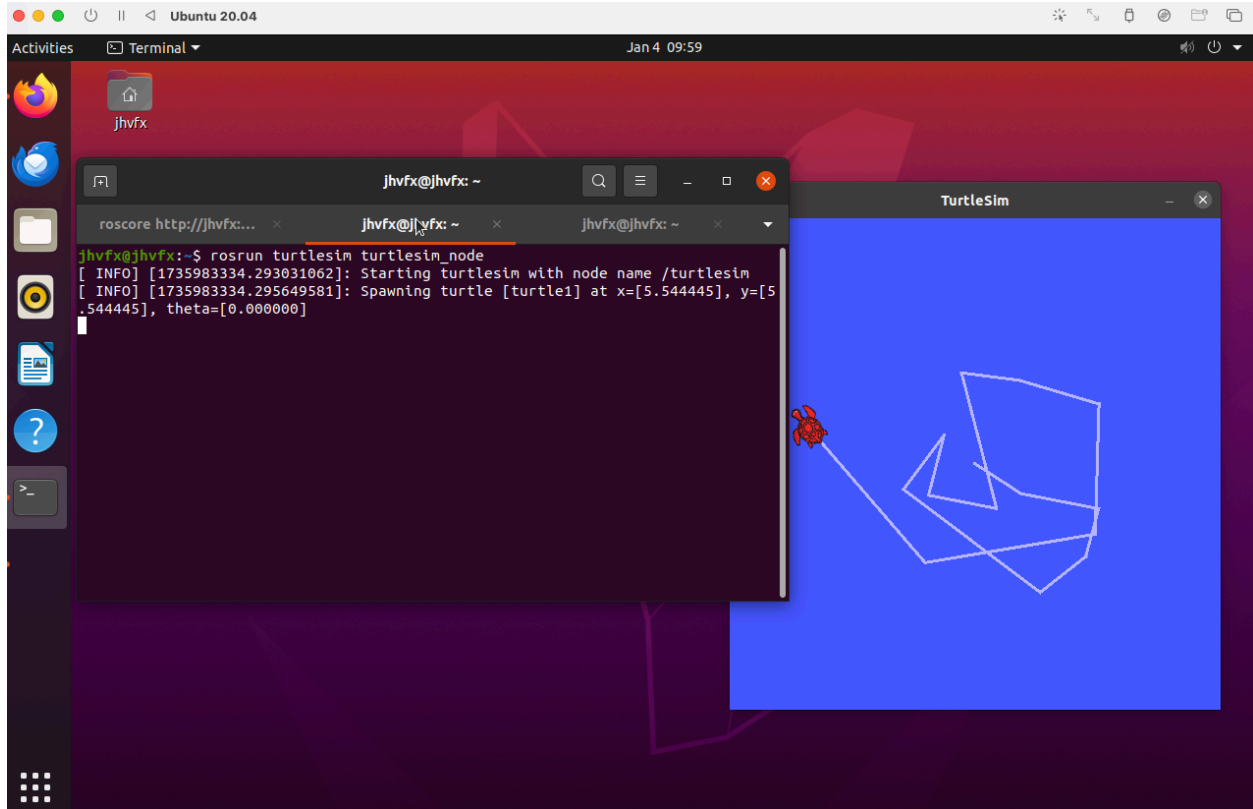
- Perintah ini digunakan untuk menjadi subscriber pada topik /test. rostopic echo akan menampilkan pesan yang diterima oleh topik tersebut ke terminal.

- Dalam hal ini, subscriber ini akan mendengarkan (subscribe) pada topik /test dan menampilkan pesan yang diterima di layar terminal.
- Jika ada pesan yang dipublikasikan ke topik /test (misalnya menggunakan perintah `rostopic pub /test std_msgs/String "data: '123'" -r1` seperti dalam kode sebelumnya), maka subscriber ini akan menampilkan pesan tersebut pada terminal.

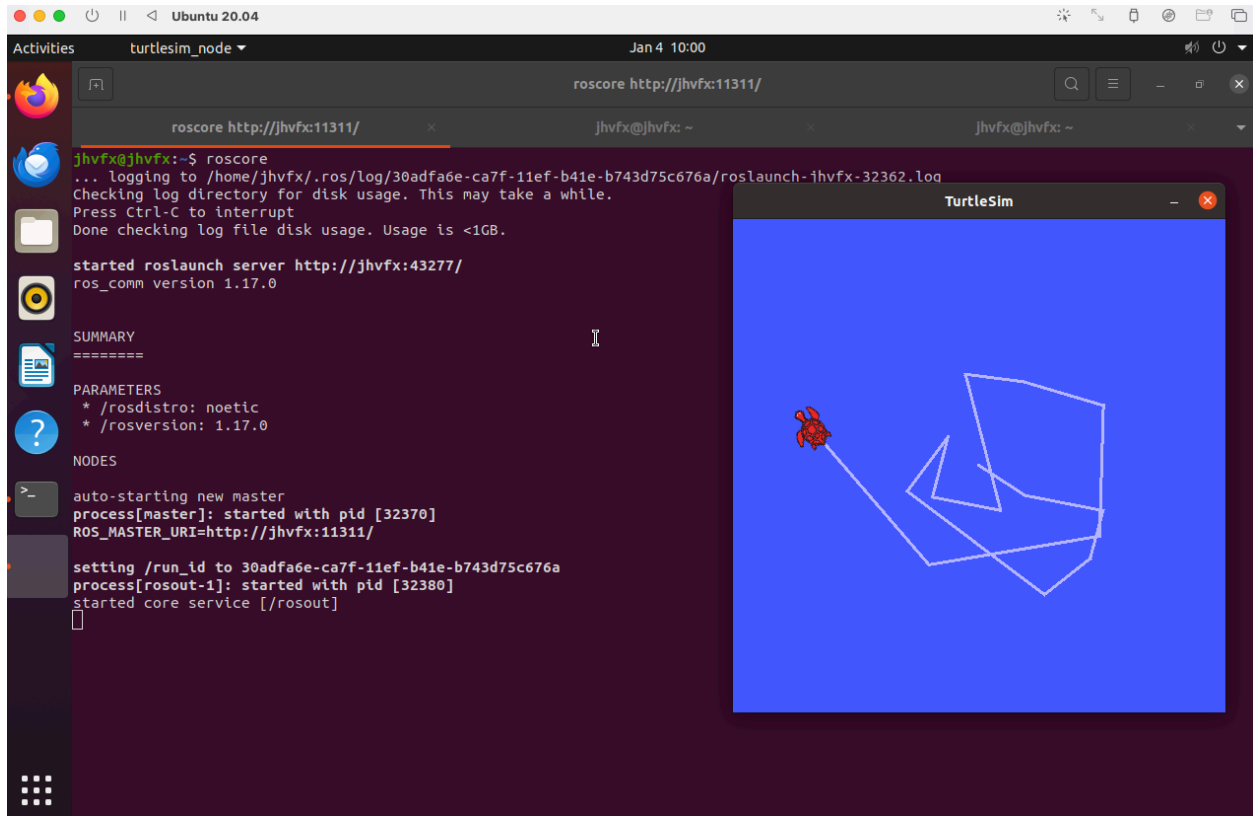
2. Turtlesim



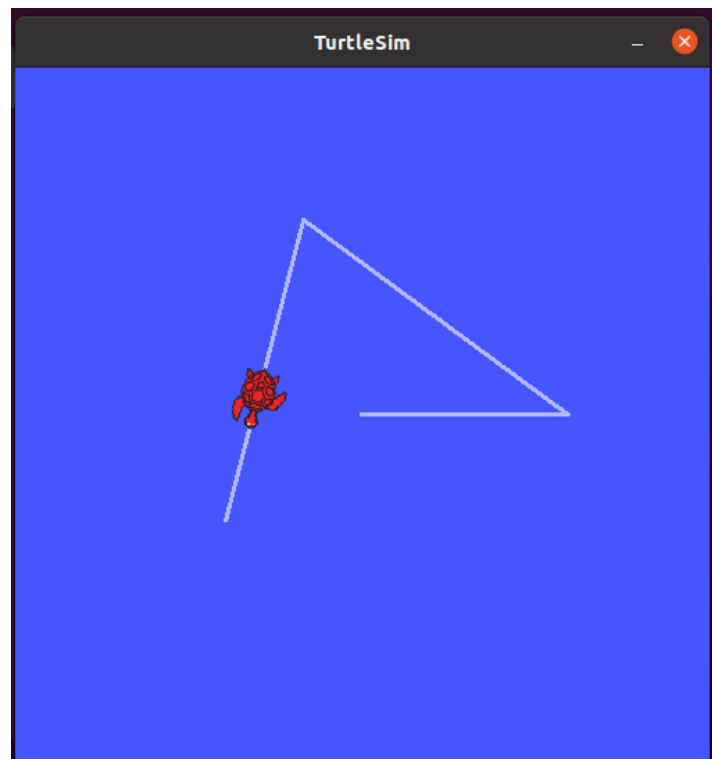
`roscore` adalah perintah yang digunakan untuk memulai sistem inti dari Robot Operating System (ROS). Ini adalah proses utama yang harus dijalankan sebelum menggunakan ROS, karena semua node ROS memerlukan layanan inti ini untuk berkomunikasi satu sama lain.



turtlesim_node adalah node utama dalam package turtlesim. Fungsinya adalah: Membuka jendela simulasi GUI dengan latar biru (mirip kolam). Menampilkan seekor kura-kura yang dapat digerakkan dengan perintah dari node lain. Node ini menyediakan topik (topics), layanan (services), dan parameter (parameters) untuk interaksi dengan kura-kura.



`roslaunch turtlesim turtle_teleop_key` untuk Mengontrol gerakan kura-kura dengan perintah



Tampilan dari kura-kura yang kita perintahkan melalui terminal halaman *roslaunch turtle_teleop_key*

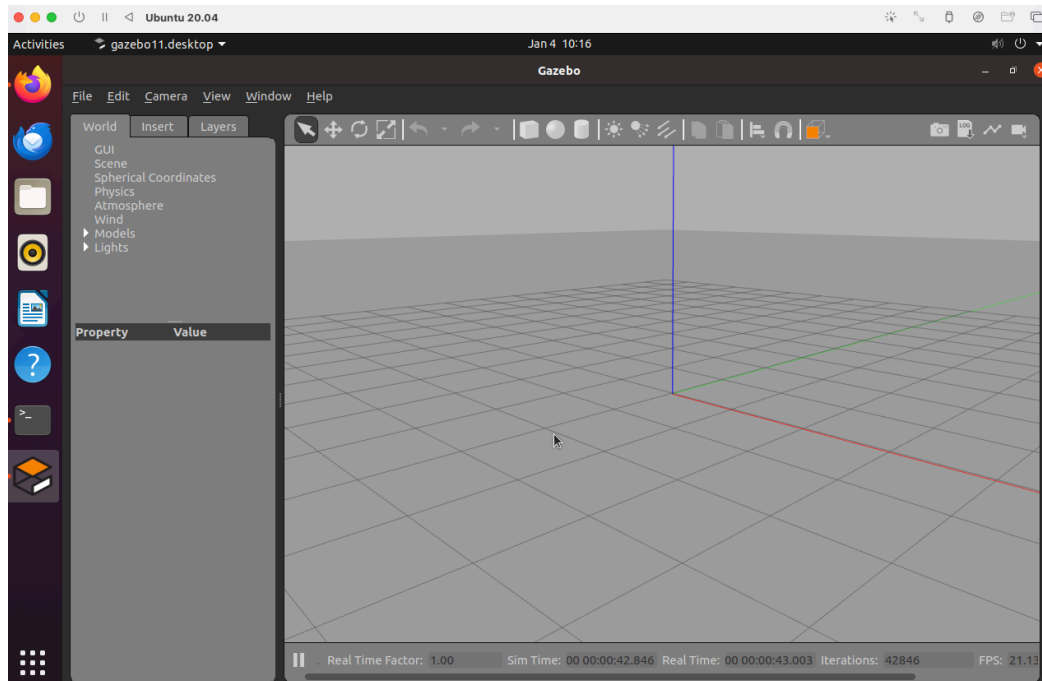
3. Gazebo

- **Pengertian:**

Gazebo adalah simulator robot open-source yang dirancang untuk memberikan simulasi fisika yang realistis dalam lingkungan 3D. Gazebo memungkinkan pengembang untuk menguji dan memvalidasi algoritma robotika dalam kondisi yang mendekati dunia nyata, tanpa memerlukan perangkat keras fisik. Gazebo terintegrasi secara erat dengan Robot Operating System (ROS), yang memungkinkan komunikasi dan kontrol yang efisien antara simulasi dan perangkat keras robot.

- **Cara Kerja:**

- **Arsitektur Modular:** Gazebo memiliki arsitektur modular yang memungkinkan pengguna untuk menambahkan plugin dan model sesuai kebutuhan. Ini memungkinkan fleksibilitas dalam menciptakan simulasi yang kompleks.
- **Simulasi Fisika:** Gazebo menggunakan berbagai engine fisika (seperti ODE, Bullet, dan DART) untuk mensimulasikan interaksi fisik antara objek. Ini mencakup efek gravitasi, tumbukan, dan dinamika gerakan, yang memberikan hasil simulasi yang realistis.
- **Modeling dan Lingkungan:** Pengguna dapat membuat model robot dan lingkungan menggunakan format file SDF (Simulation Description Format) atau URDF (Unified Robot Description Format). Model ini dapat mencakup berbagai elemen seperti sensor, aktuator, dan objek statis.
- **Integrasi dengan ROS:** Gazebo berfungsi sebagai simulator yang dapat berkomunikasi dengan node ROS. Data sensor dari simulasi dapat dikirim ke node ROS, dan perintah kontrol dari ROS dapat diterapkan pada robot dalam simulasi. Ini memungkinkan pengujian algoritma kontrol dan pemrosesan data sensor secara real-time.



- **Fungsi:**

- **Pengujian Algoritma:** Gazebo memungkinkan pengembang untuk menguji algoritma kontrol dan navigasi dalam simulasi sebelum menerapkannya pada robot fisik. Ini mengurangi risiko kesalahan yang dapat merusak perangkat keras.
- **Validasi Desain Robot:** Pengguna dapat memvalidasi desain robot dan sistem kontrol dalam berbagai skenario, termasuk situasi yang sulit atau berbahaya untuk diuji di dunia nyata.
- **Pelatihan dan Pendidikan:** Gazebo sering digunakan dalam pendidikan robotika untuk memberikan pengalaman praktis kepada siswa dalam pengembangan dan pengujian sistem robot.
- **Pengembangan Berbasis Komunitas:** Sebagai proyek open-source, Gazebo memiliki komunitas yang aktif yang berkontribusi pada pengembangan model, plugin, dan alat baru, memperluas fungsionalitas simulator.

- **Fitur Utama:**

- **Visualisasi 3D Realistik:** Gazebo menyediakan antarmuka grafis yang memungkinkan pengguna untuk melihat simulasi dalam 3D, termasuk pencahayaan, tekstur, dan efek visual lainnya.
- **Sensor dan Aktuator:** Gazebo mendukung berbagai jenis sensor (seperti kamera, lidar, dan IMU) dan aktuator, memungkinkan simulasi yang lebih mendalam dan realistis.

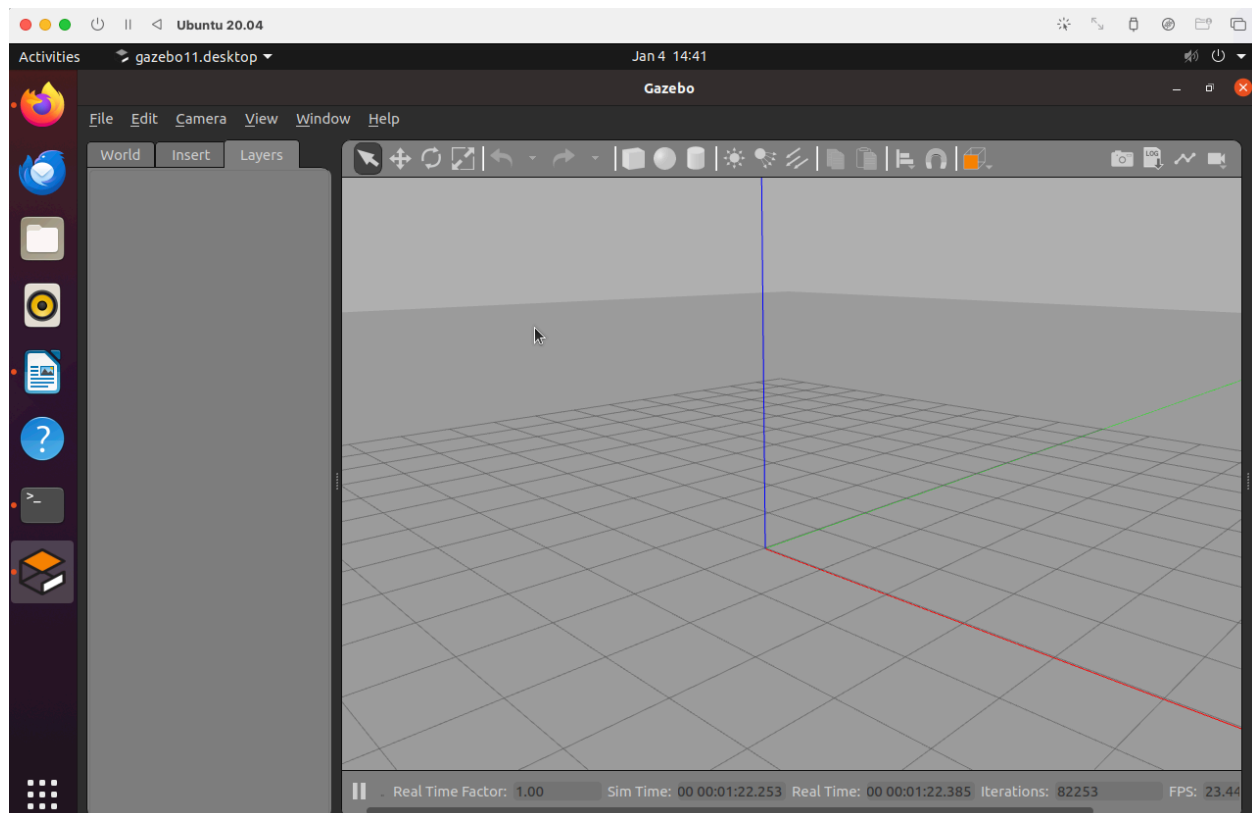
- **Plugin dan Ekstensi:** Pengguna dapat menambahkan plugin untuk memperluas fungsionalitas Gazebo, seperti menambahkan algoritma kontrol khusus atau integrasi dengan perangkat lunak lain.
- **Simulasi Multi-Robot:** Gazebo mendukung simulasi beberapa robot secara bersamaan, memungkinkan pengujian skenario kolaboratif dan interaksi antar robot.
- **Dukungan untuk Robot Berbasis ROS:** Gazebo terintegrasi dengan ROS, memungkinkan pengguna untuk memanfaatkan ekosistem ROS yang luas, termasuk pustaka dan alat yang telah ada.

```
jhvfx@jhvfx: ~
jhvfx@jhvfx:~$ sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
[sudo] password for jhvfx:
jhvfx@jhvfx:~$ wget https://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
--2025-01-04 14:39:14-- https://packages.osrfoundation.org/gazebo.key
Resolving packages.osrfoundation.org (packages.osrfoundation.org)... 52.52.171.7
Connecting to packages.osrfoundation.org (packages.osrfoundation.org)|52.52.171.73|:443... connected.
HTTP request sent, awaiting response... 200 OK
Content-Length: 1755 (1.7K) [application/octet-stream]
Saving to: 'STDOUT'

100%[=====] 1.71K --.-KB/s in 0s

2025-01-04 14:39:14 (502 MB/s) - written to stdout [1755/1755]
OK
jhvfx@jhvfx:~$ sudo apt update
Get:1 http://packages.osrfoundation.org/gazebo/ubuntu-stable focal InRelease [4,279 B]
Get:2 http://packages.osrfoundation.org/gazebo/ubuntu-stable focal/main arm64 Pa
```

```
jhvfx@jhvfx: ~  
libignition-fuel-tools4 libignition-fuel-tools4-dev libignition-msgs5  
libignition-msgs5-dev libignition-transport8 libignition-transport8-core-dev  
libignition-transport8-dev libignition-transport8-log  
libignition-transport8-log-dev libsdformat9 libsdformat9-dev sdformat9-sdf  
Suggested packages:  
gazebo11-doc  
The following packages will be upgraded:  
gazebo11 gazebo11-plugin-base libgazebo11 libgazebo11-dev  
libignition-common3 libignition-common3-av libignition-common3-av-dev  
libignition-common3-core-dev libignition-common3-dev  
libignition-common3-events libignition-common3-events-dev  
libignition-common3-graphics libignition-common3-graphics-dev  
libignition-common3-profiler libignition-common3-profiler-dev  
libignition-fuel-tools4 libignition-fuel-tools4-dev libignition-msgs5  
libignition-msgs5-dev libignition-transport8 libignition-transport8-core-dev  
libignition-transport8-dev libignition-transport8-log  
libignition-transport8-log-dev libsdformat9 libsdformat9-dev sdformat9-sdf  
27 upgraded, 0 newly installed, 0 to remove and 44 not upgraded.  
Need to get 18.0 MB of archives.  
After this operation, 707 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Abort.  
jhvfx@jhvfx:~$ gazebo
```



4. Introduitin ROS

Pengenalan ROS ROS (Robot Operating System) adalah kerangka kerja open-source yang dirancang untuk pengembangan perangkat lunak robotika. ROS mendukung komunikasi antarproses, manajemen paket, dan integrasi dengan berbagai alat serta pustaka. Sistem ini memungkinkan eksekusi proses terdistribusi melalui konsep Node, yang terhubung menggunakan layanan (RPC sinkron), topik (streaming data asinkron), dan Server Parameter.

Fitur Utama ROS:

- Dukungan komunikasi peer-to-peer.
- Skalabilitas untuk sistem besar.
- Dukungan multi-bahasa (Python, C++, Lisp).
- Integrasi dengan alat seperti OpenCV, Gazebo, dan OpenRAVE.
- Pengujian bawaan dengan rostest.

2. Persiapan Instalasi ROS

Langkah 1: Instal VirtualBox

1. Unduh VirtualBox dari situs resmi: <https://www.virtualbox.org/>
2. Instal VirtualBox sesuai dengan sistem operasi Anda.

Langkah 2: Instal Ubuntu di VirtualBox

1. Unduh ISO Ubuntu LTS dari <https://ubuntu.com/download>.
2. Buat mesin virtual di VirtualBox:
 - Pilih Ubuntu sebagai sistem operasi.
 - Atur RAM minimal 4GB dan penyimpanan minimal 20GB.
3. Boot mesin virtual dengan file ISO Ubuntu.
4. Ikuti panduan instalasi Ubuntu hingga selesai.

Langkah 3: Instal ROS di Ubuntu

1. Tambahkan Repository ROS:
2. `sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'`
3. `sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key 0xF42ED6FBAB17C654`

4. `sudo apt update`
5. Instal ROS Desktop Full:
6. `sudo apt install ros-noetic-desktop-full`
7. Inisialisasi rosdep:
8. `sudo rosdep init`
9. `rosdep update`
10. Setup Lingkungan ROS:
11. `echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc`
12. `source ~/.bashrc`
13. Verifikasi Instalasi:
14. `roscore`

3. Struktur Proyek ROS

- Node: Proses individu yang berkomunikasi dengan node lain.
- Topic: Saluran komunikasi antar node.
- Service: RPC untuk komunikasi sinkron.
- Parameter Server: Penyimpanan data konfigurasi global.

4. Pengujian Awal ROS

1. Buka dua terminal.
2. Terminal 1: Jalankan `roscore`.
3. Terminal 2: Jalankan node contoh:
4. `roslaunch turtlesim turtlesim_node`
5. Terminal lain: Kendalikan node:
6. `roslaunch turtlesim turtle_teleop_key`

5. Kesimpulan ROS adalah kerangka kerja fleksibel yang memungkinkan pengembangan robot yang kompleks dan terdistribusi dengan mudah. Dengan memahami konsep dasar dan langkah instalasi ini, Anda siap memulai eksplorasi lebih lanjut dalam pengembangan perangkat lunak robotika menggunakan ROS.

