

Nama : Arfiq Rimeldo

NIM : 1103202102

Kelas : TK-44-G7

Technical Report ROS2 : Nodes Communication Using Services & Actions

A. ROS2

ROS 2, atau Robot Operating System 2, adalah platform perangkat lunak open-source yang dirancang khusus untuk pengembangan dan pengoperasian robot. ROS 2 merupakan evolusi dari ROS yang sebelumnya (ROS 1), dan dirancang untuk mengatasi beberapa keterbatasan yang ada pada versi sebelumnya.

ROS 2 dirancang dengan dukungan untuk komunikasi real-time, yang merupakan perbaikan signifikan dari ROS 1. Ini memungkinkan pengembang untuk mengimplementasikan sistem yang lebih responsif dan dapat diandalkan.

ROS 2 dirancang dengan mempertimbangkan skalabilitas, yang memungkinkan untuk digunakan dalam berbagai konteks, termasuk robot-robot yang membutuhkan tingkat kompleksitas dan keandalan yang berbeda.

ROS 2 diterapkan dengan desain modular yang memungkinkan pengembang untuk menggunakan komponen-komponen yang diperlukan dan meninggalkan yang tidak diperlukan. Hal ini meningkatkan reusabilitas dan mempermudah pengembangan aplikasi.

ROS 2 mendukung beberapa bahasa pemrograman, termasuk C++, Python, dan lainnya. Ini memberikan fleksibilitas lebih lanjut kepada pengembang untuk memilih bahasa yang paling sesuai dengan keahlian mereka.

Meskipun merupakan versi yang berbeda, ROS 2 dirancang dengan mempertimbangkan kompatibilitas dengan ROS 1. Ini berarti bahwa beberapa kode ROS 1 dapat diintegrasikan ke dalam proyek ROS 2.

B. Nodes Communication Using Services & Actions

Nodes itu sendiri merujuk pada unit dasar perangkat lunak yang menjalankan tugas tertentu dalam sistem robotika. Dalam konteks ROS, komunikasi antara node dapat difasilitasi menggunakan layanan (services) dan tindakan (actions).

1. Layanan (Services)

Layanan adalah pola komunikasi di mana satu node (klien) meminta tugas tertentu dari node lain (server). Ini adalah mekanisme komunikasi satu-ke-satu yang bersifat sinkron

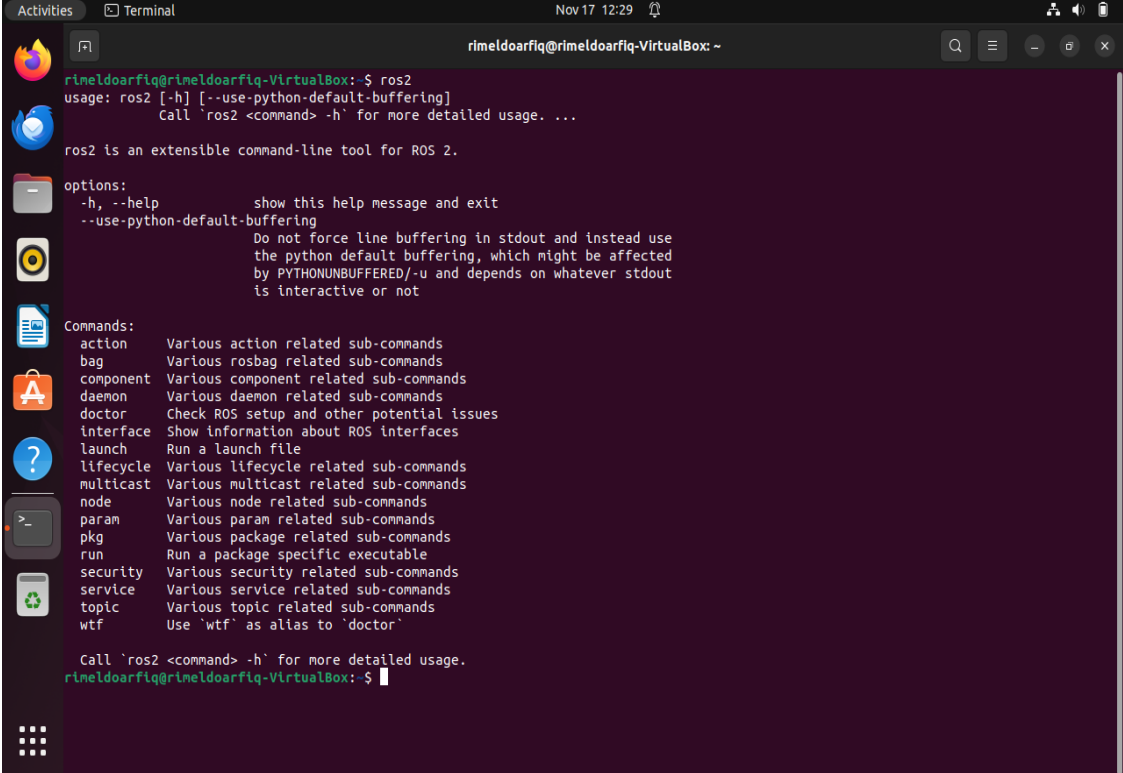
2. Tindakan (Actions)

Tindakan adalah bentuk komunikasi yang lebih canggih, cocok untuk tugas yang memakan waktu lama atau memerlukan umpan balik selama eksekusi. Ini

adalah mekanisme komunikasi satu-ke-satu atau satu-ke-banyak yang bersifat asinkron.

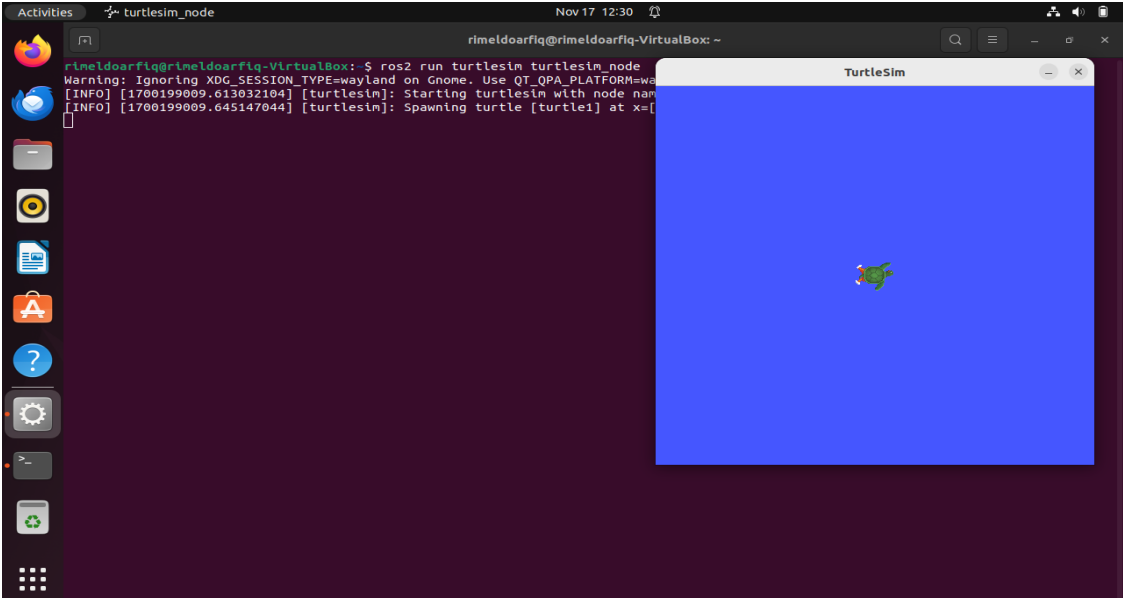
C. Langkah-Langkah Pengerjaan

1. Melakukan pengecekan pada VirtualBox apakah ROS2 sudah terinstall dan dapat dijalankan



```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~  
$ ros2  
usage: ros2 [-h] [--use-python-default-buffering]  
        Call 'ros2 <command> -h' for more detailed usage. ...  
  
ros2 is an extensible command-line tool for ROS 2.  
  
options:  
  -h, --help            show this help message and exit  
  --use-python-default-buffering  
                        Do not force line buffering in stdout and instead use  
                        the python default buffering, which might be affected  
                        by PYTHONUNBUFFERED/-u and depends on whatever stdout  
                        is interactive or not  
  
Commands:  
  action               Various action related sub-commands  
  bag                  Various rosbag related sub-commands  
  component            Various component related sub-commands  
  daemon              Various daemon related sub-commands  
  doctor               Check ROS setup and other potential issues  
  interface            Show information about ROS interfaces  
  launch              Run a launch file  
  lifecycle            Various lifecycle related sub-commands  
  multicast            Various multicast related sub-commands  
  node                 Various node related sub-commands  
  param               Various param related sub-commands  
  pkg                  Various package related sub-commands  
  run                  Run a package specific executable  
  security             Various security related sub-commands  
  service             Various service related sub-commands  
  topic               Various topic related sub-commands  
  wtf                  Use 'wtf' as alias to 'doctor'  
  
Call 'ros2 <command> -h' for more detailed usage.  
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~
```

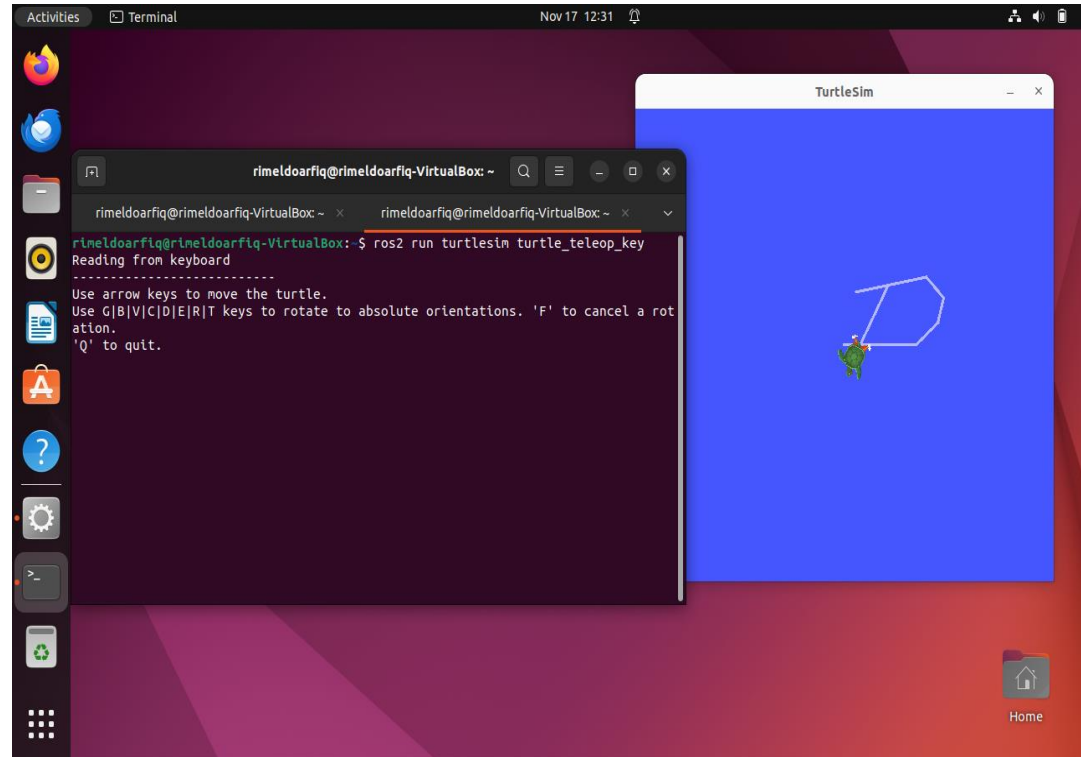
2. Menjalankan command untuk memunculkan turtlesim yang digunakan untuk komunikasi node nantinya



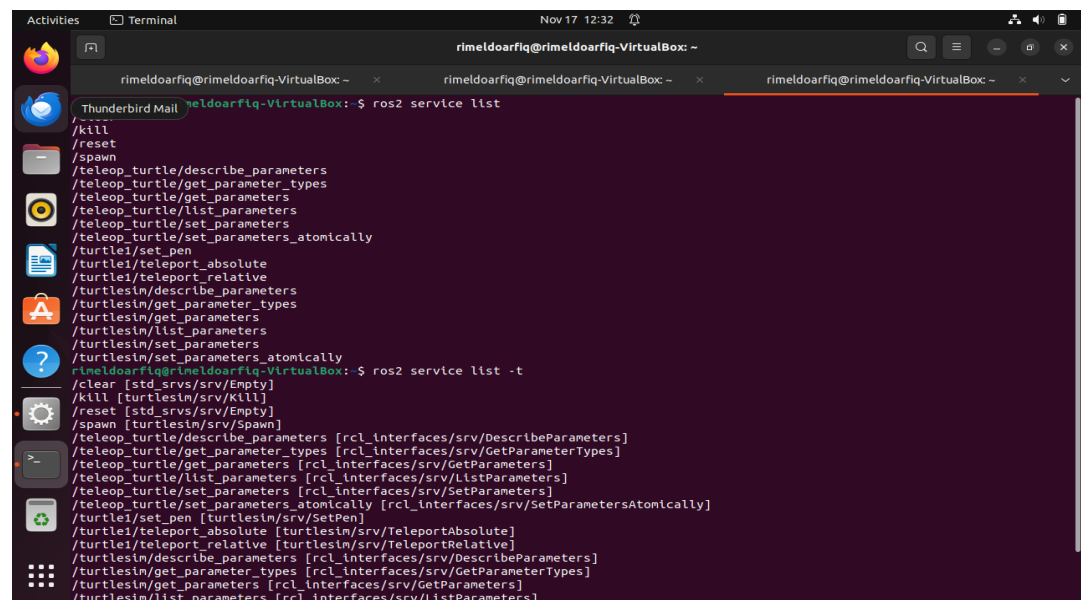
```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~  
$ ros2 run turtlesim turtlesim node  
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wa  
[INFO] [1700199009.613032104] [turtlesim]: Starting turtlesim with node na  
[INFO] [1700199009.645147044] [turtlesim]: Spawning turtle [turtle1] at x=[  
  
TurtleSim
```

The TurtleSim window displays a blue square environment with a small green turtle icon in the center.

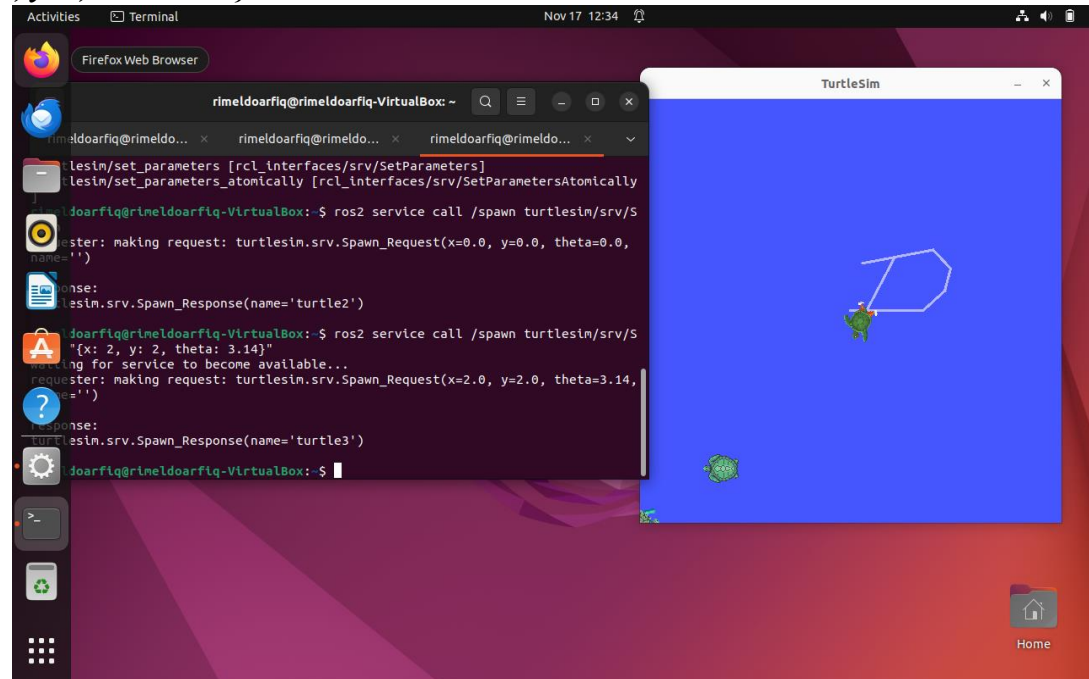
3. Menggunakan command ***ros2 run turtlesim turtle_teleop_key*** untuk menggerakkan kura-kura sesuai arah yang kita mau serta membuat kura-kura bergerak secara rotasi



4. Selanjutnya masuk ke bagian service, dimana dengan menggunakan command ***ros2 service list*** dan ***ros2 service list -t*** kita dapat melihat banyak service yang terdapat pada node untuk berkomunikasi dengan turtlesim tadi.

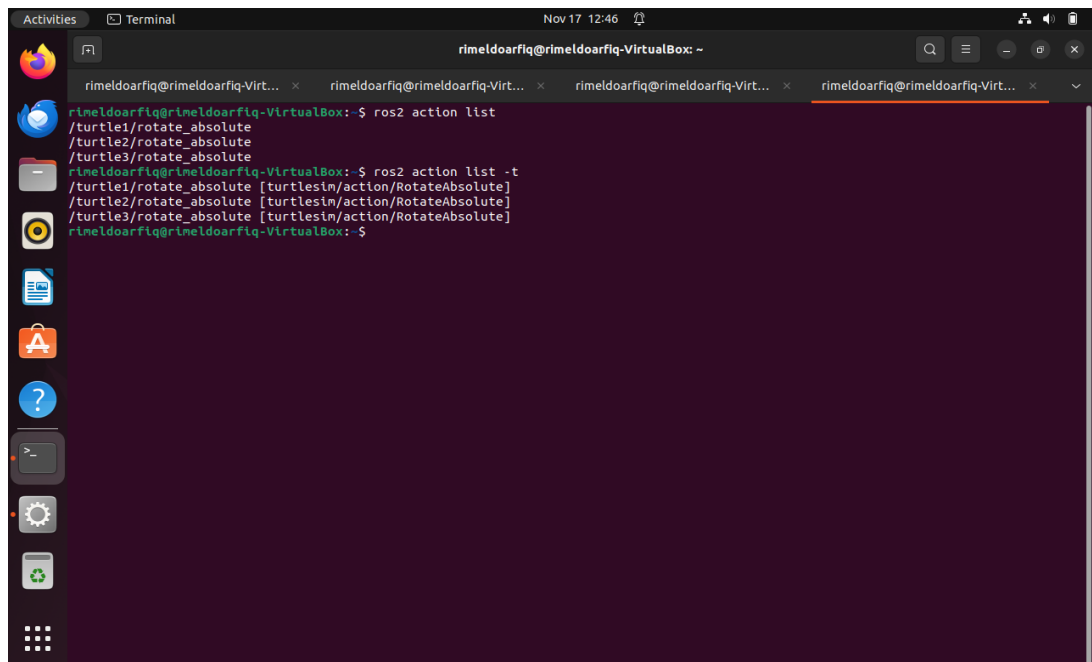


5. Selanjutnya kita memilih salah satu service dari service list tadi, yaitu `/spawn`. Dengan menggunakan command `ros2 service call /spawn turtlesim/srv/spawn "{x: 2, y: 2, theta: 3.14}"`

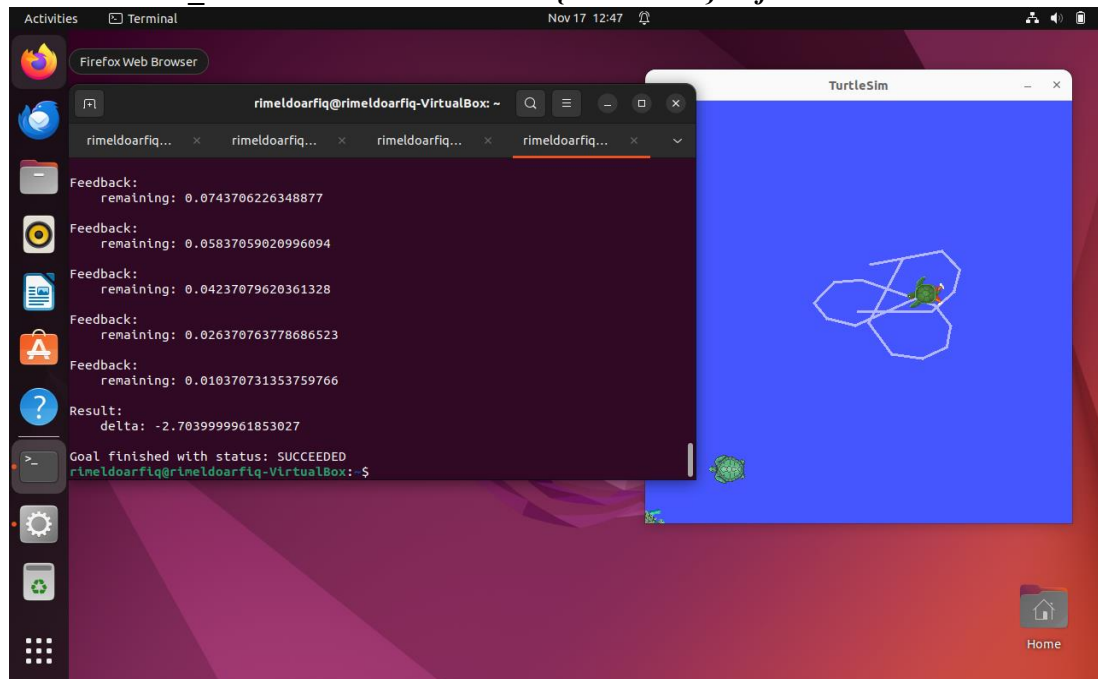


Dari gambar di atas dapat dilihat setelah menggunakan command spawn tadi, kita dapat memunculkan kura-kura baru sesuai dengan letak yang kita isi pada x, y, dan theta.

6. Selanjutnya masuk ke bagian action, dimana dengan menggunakan command `ros2 action list` dan `ros2 action list -t` kita dapat melihat banyak service yang terdapat pada node untuk berkomunikasi dengan turtlesim tadi.

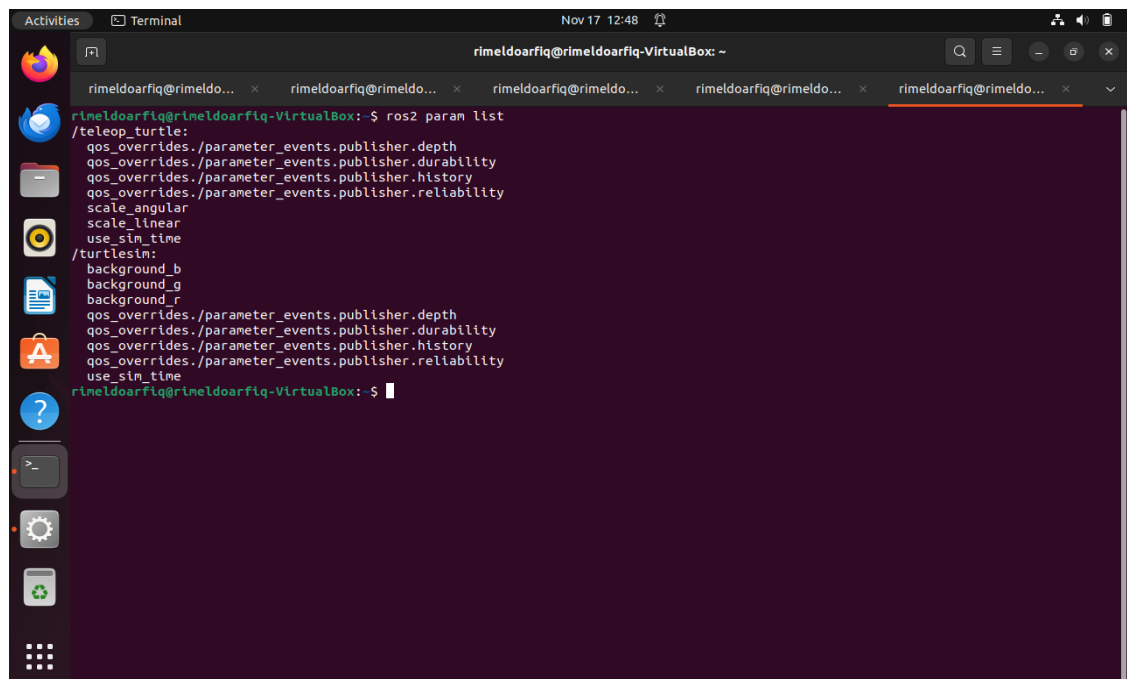


7. Selanjutnya kita akan mencoba action pada turtlebot1 yang merupakan turtlebot utama dengan menggunakan command `ros2 action send_goal /turtle1/rotate_absolute turtlesim/action "{theta: 3.14}" -feedback`.

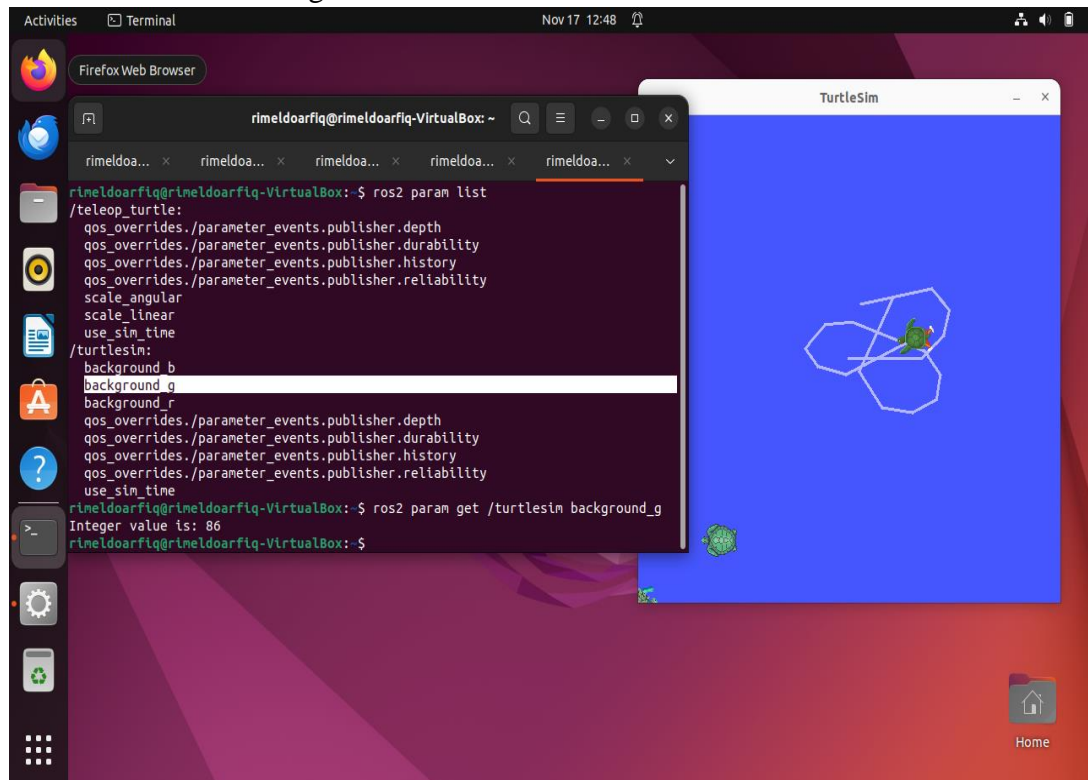


Dari gambar di atas dapat dilihat setelah running command tadi turtlebot1 akan bergerak memutar secara otomatis hingga nantinya berhenti sendiri. Karena menggunakan `-feedback` maka akan terlihat juga history atau log pergerakan turtlebot1 dari awal hingga succeeded.

8. Bagian terakhir kita melihat parameter dari turtlebot ini menggunakan command `ros2 param list`

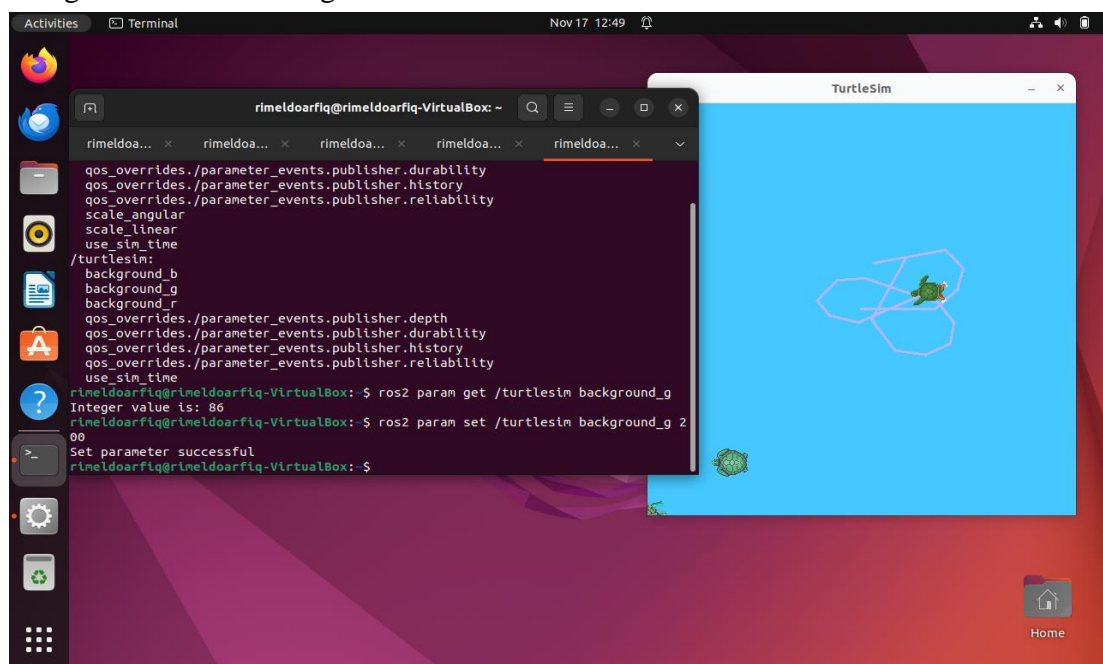


9. Selanjutnya running command ***ros2 param get /turtlesim background_g*** untuk melihat value dari background turtlebot



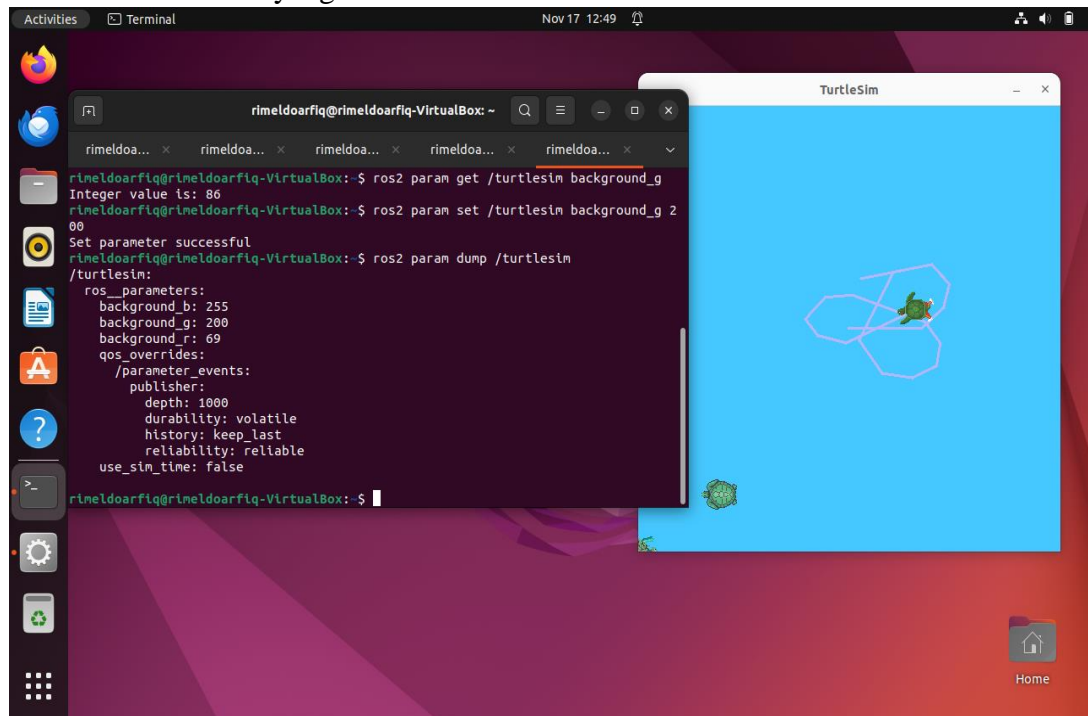
Dari gambar di atas dapat dilihat background valuenya 86 yang menyebabkan warna background berwarna biru tua.

10. Lanjutkan dengan command ***ros2 param set /turtlesim background_g 200*** untuk mengubah Dwarna background.



Dari gambar di atas dapat dilihat sesudah mengubah value menjadi 200 warna background berubah menjadi biru muda.

11. Terakhir, menggunakan command ***ros2 param dump /turtlesim*** untuk menyimpan editan turtlesim kita yang berwarna biru muda.



Dari gambar di atas dapat dilihat editan turtlesim berhasil disimpan dan kita dapat membuka turtlesim berwarna biru muda tersebut nantinya ketika running turtlesim bukan lagi turtlesim yang berwarna biru tua