

Dokumentasi Lecture 3

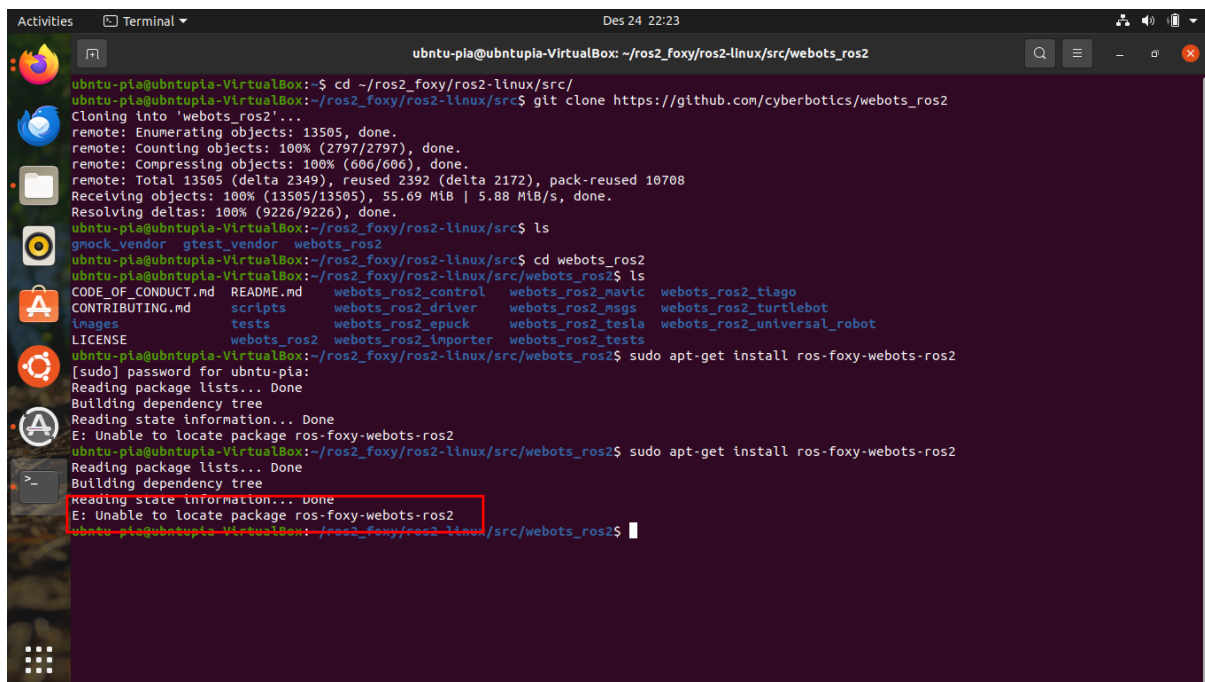
Playlist Video:

https://www.youtube.com/watch?v=mUmOwr-U_68&list=PLt69C9MnPehkP0ZXZOqmlGRTOch8o9GiQ&index=6

Video 1 - Webots ROS2 tutorial series : Setting Up ROS2 and Webots | Ubuntu 20.04 | [Tutorial 1]

Pada percobaan pertama berhasil menginstall ros2 tapi dalam proses instalasi webots tidak dapat dilanjutkan karena memori tidak cukup (dokumentasi terhapus). Usaha yang dilakukan adalah menambah memori tapi masih gagal. Sehingga membuat vm baru dengan memori yang lebih besar. Proses percobaan dengan vm dilakukan pada percobaan kedua. Pada percobaan pertama sempat mengabaikan pesan eror bahwa memori sudah tidak cukup sehingga terdapat beberapa file yang tidak berhasil diinstall. Namun, yang terjadi adalah saat akan mengerjakan tutorial video kedua, tidak berhasil dijalankan, karena file tidak ada.

Pada percobaan kedua, tutorial dalam video berhasil dalam instalasi ROS2nya. Namun, untuk instalasi webots gagal, dan muncul pesan bahwa directory tidak ada.



```

ubuntu-pia@ubuntu-pia-VirtualBox: ~/ros2_foxy/ros2-linux/src/webots_ros2
ubuntu-pia@ubuntu-pia-VirtualBox:~$ cd ~/ros2_foxy/ros2-linux/src/
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src$ git clone https://github.com/cyberbotics/webots_ros2
Cloning into 'webots_ros2'...
remote: Enumerating objects: 13505, done.
remote: Counting objects: 100% (2797/2797), done.
remote: Compressing objects: 100% (606/606), done.
remote: Total 13505 (delta 2349), reused 2392 (delta 2172), pack-reused 10708
Receiving objects: 100% (13505/13505), 55.69 MiB | 5.88 MiB/s, done.
Resolving deltas: 100% (9226/9226), done.
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src$ ls
gmock_vendor  gtest_vendor  webots_ros2
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src$ cd webots_ros2
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src/webots_ros2$ ls
CODE_OF_CONDUCT.md  README.md      webots_ros2_control  webots_ros2_navic  webots_ros2_tiago
CONTRIBUTING.md    scripts        webots_ros2_driver   webots_ros2_msgs   webots_ros2_turtlebot
images              tests          webots_ros2_epuck    webots_ros2_tesla   webots_ros2_universal_robot
LICENSE             webots_ros2    webots_ros2_importer webots_ros2_tests
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src/webots_ros2$ sudo apt-get install ros-foxy-webots-ros2
[sudo] password for ubuntu-pia:
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package ros-foxy-webots-ros2
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src/webots_ros2$ sudo apt-get install ros-foxy-webots-ros2
Reading package lists... Done
Building dependency tree
Reading state information... Done
E: Unable to locate package ros-foxy-webots-ros2
ubuntu-pia@ubuntu-pia-VirtualBox:~/ros2_foxy/ros2-linux/src/webots_ros2$
  
```

Gambar 1 pesan eror

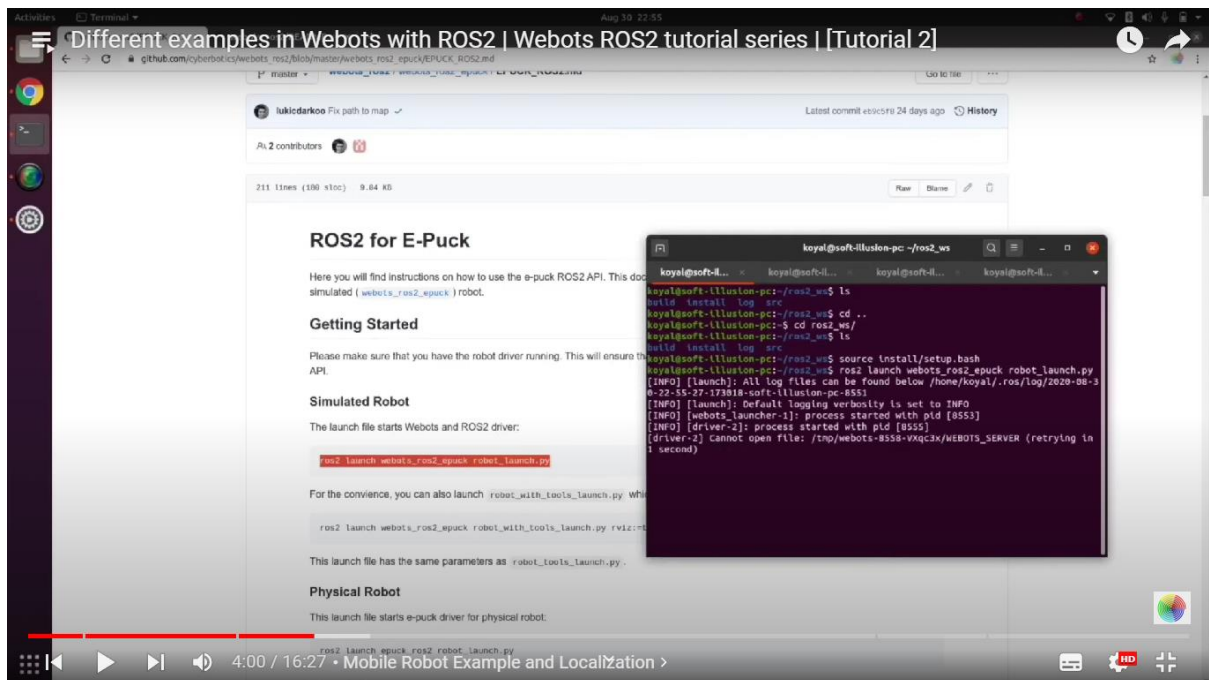
Dalam hal ini, video 2 akan dilanjutkan dengan dokumentasi.

Video 2 - Different examples in Webots with ROS2 | Webots ROS2 tutorial series | [Tutorial 2]

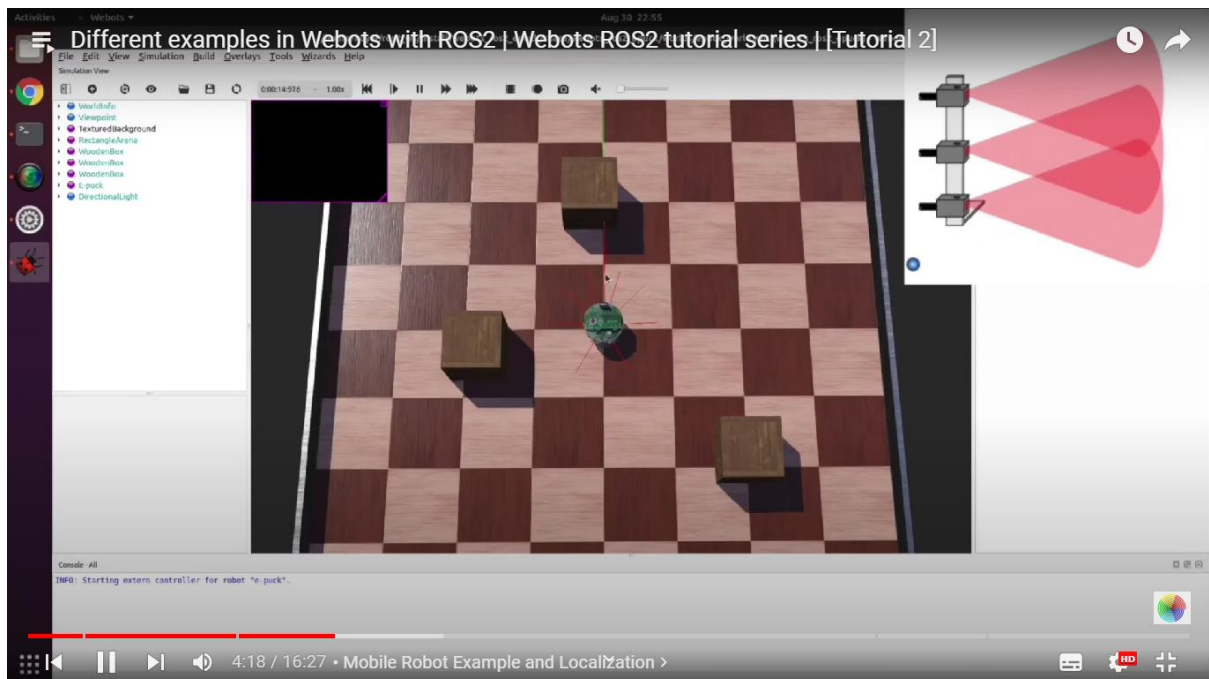
Video 2 terbagi dalam 4 sub, antara lain:

- Pengantar mengenai penjelasan singkat tentang berbagai jenis robot. Jenis robot yang dibahas yaitu robot stasioner dan robot bergerak. Robot stasioner adalah robot berbasis tetap, end-effector bergerak serta memiliki aplikasi industri dan dapat digunakan untuk robot bedah. Sedangkan robot bergerak adalah tentang pengenalan dan aplikasi robot bergerak dalam mobil otonom, otomasi gudang, dan lain-lain.

- Contoh robot bergerak dan lokalisasi. Terdapat tautan: <https://github.com/cyberbotics/webots> yang didalamnya mendeskripsikan secara rinci mengenai tata cara menjalankan E-Puck dengan paket ROS2. Tautan tersebut berisi workspace ROS2 yang dapat diluncurkan menggunakan langkah-langkah yang ditunjukkan dalam video.

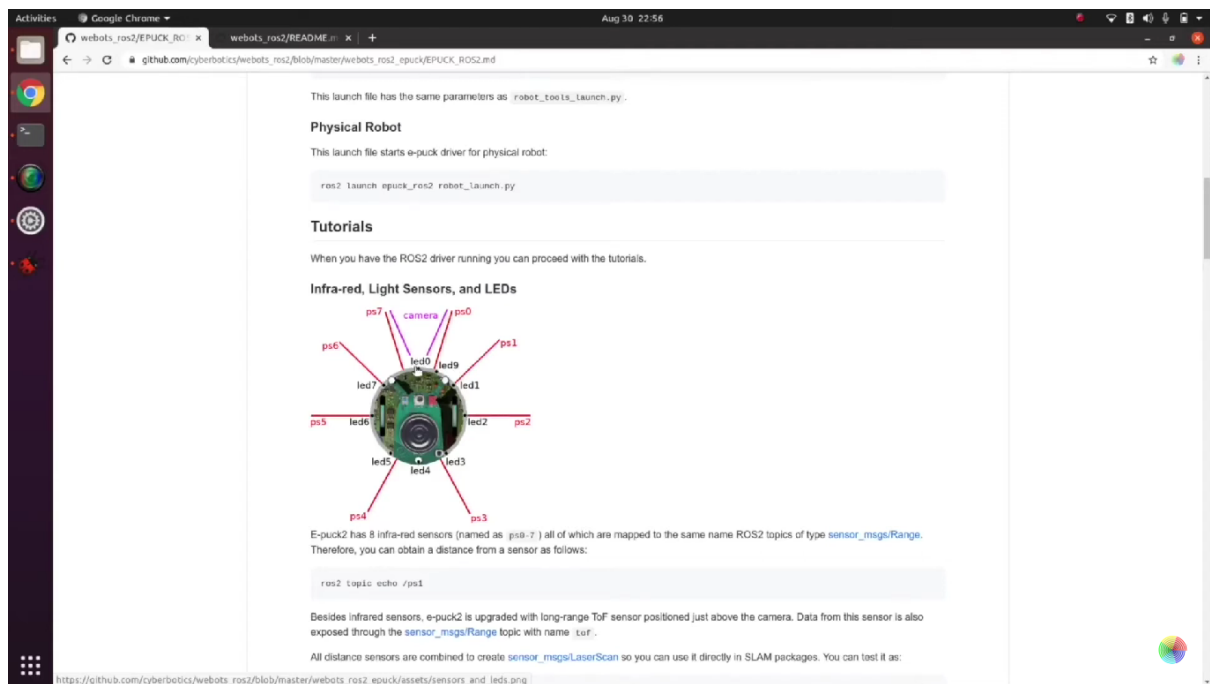


Gambar 2 launch ros2 for e-puck



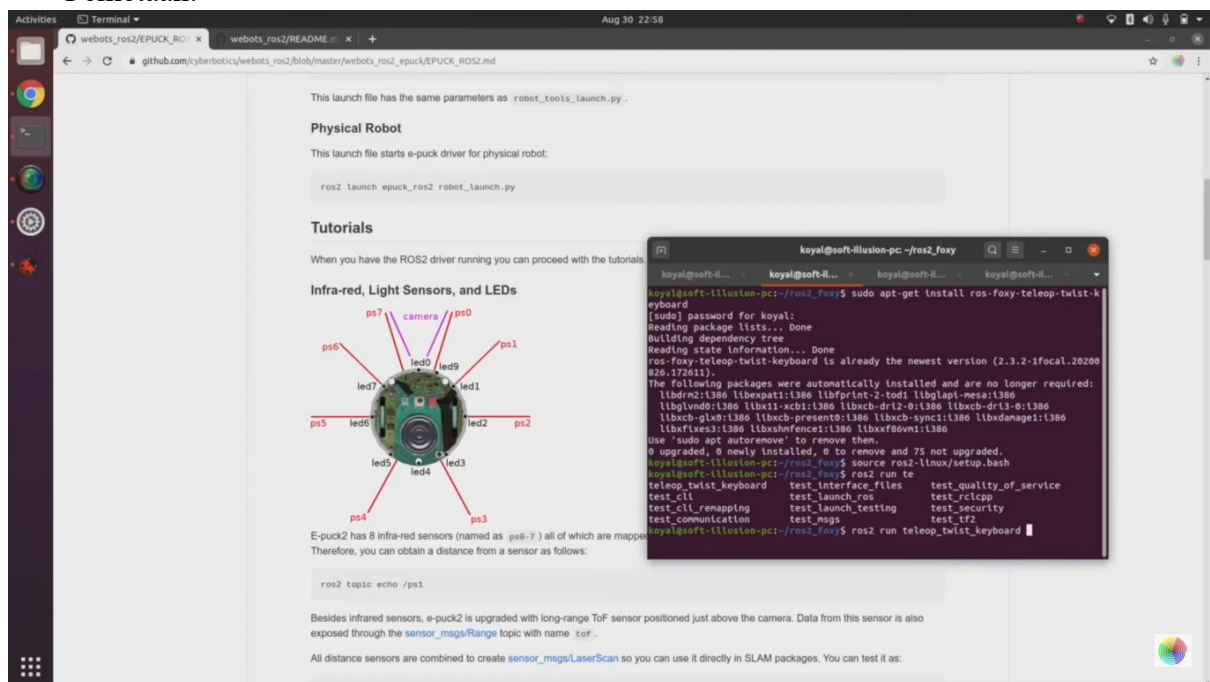
Gambar 3 berhasil launch e-puck

Selain itu, akan diperlihatkan E-puck dengan kamera dan 8 sensor jarak jauh. Ini akan digunakan untuk pemetaan pada langkah berikutnya.

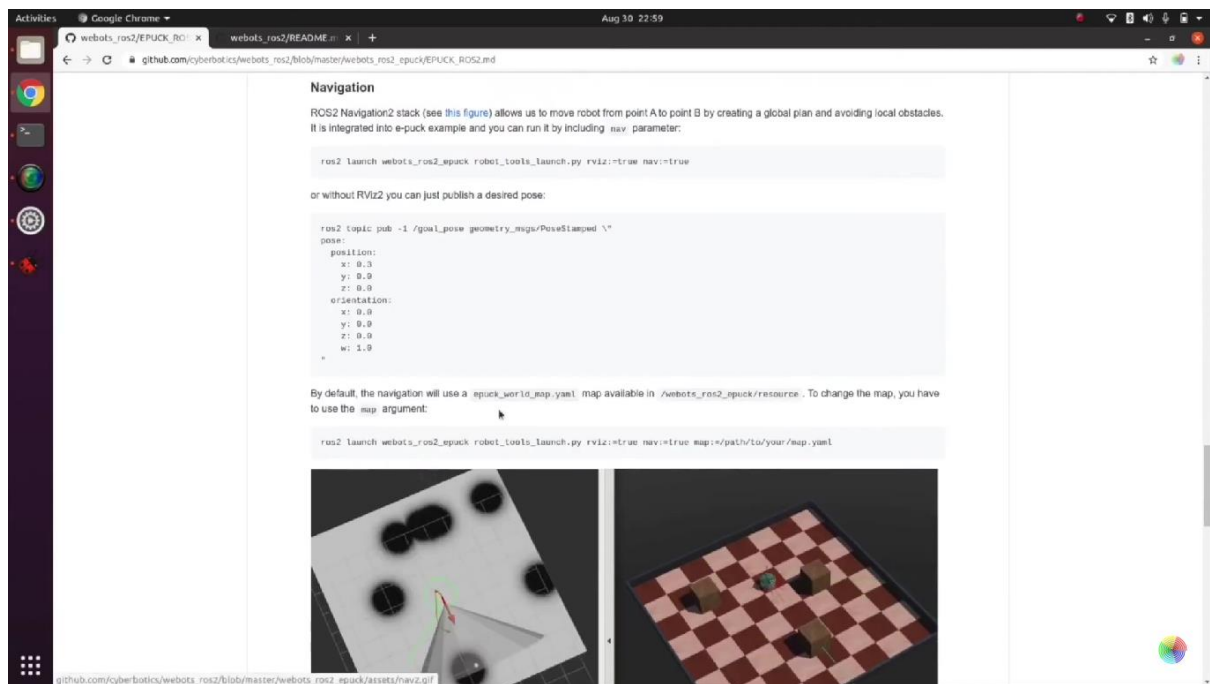


Gambar 4 E-puck

Di sini, paket teleop keyboard juga diinstal. Hal ini merupakan paket ROS bawaan yang dapat digunakan untuk mengontrol baik robot holonomik maupun non-holonomik. Pemetaan:

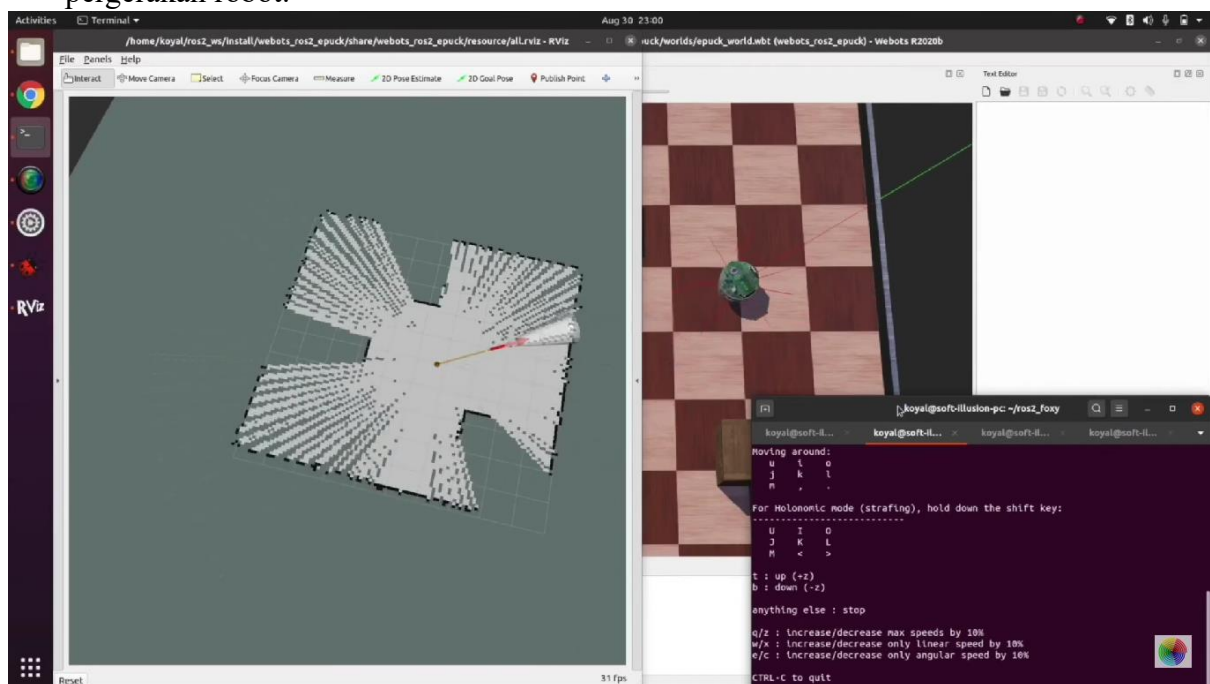


Gambar 5 teleop keyboard



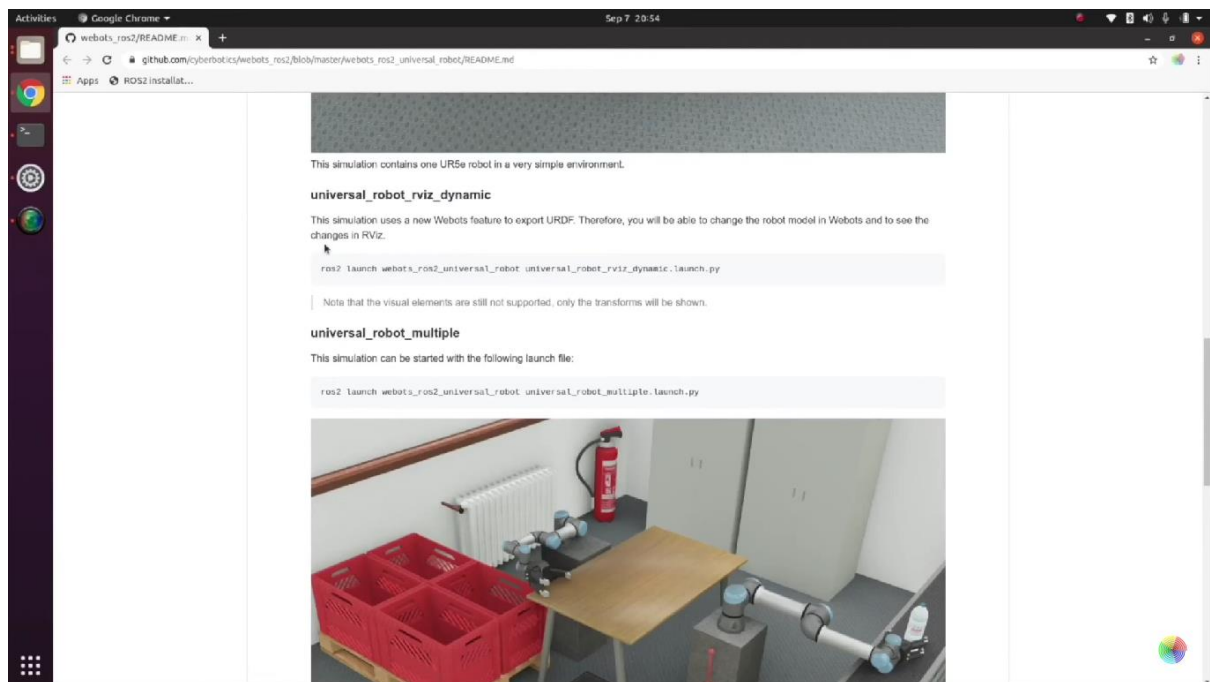
Gambar 6 navigation

Robot ditampilkan sebagai titik di Rviz. Ketika robot diputar, ia mulai membuat peta dengan mengambil titik-titik di dunia. Oleh karena itu, dengan memindahkan robot menggunakan keyboard, peta dengan semua rintangan dihasilkan menggunakan pergerakan robot.



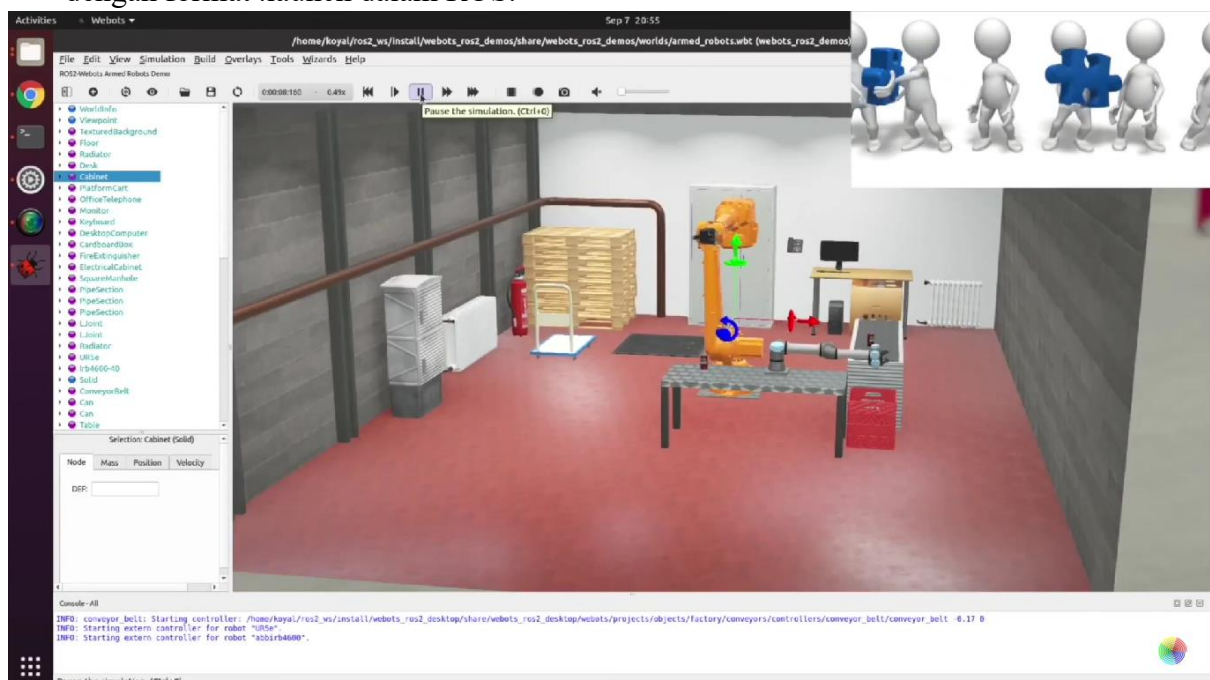
Gambar 7 robot ditampilkan sebagai titik di Rviz

- Hasil robot nyata diperlihatkan sebuah robot nyata berlayar dan outputnya terlihat di Rviz. Awan titik dibuat oleh sensor-sensor pada robot. Peta dunia nyata sesuai dengan apa yang kita dapatkan di layar. Move-base berjalan di dalam E-puck.

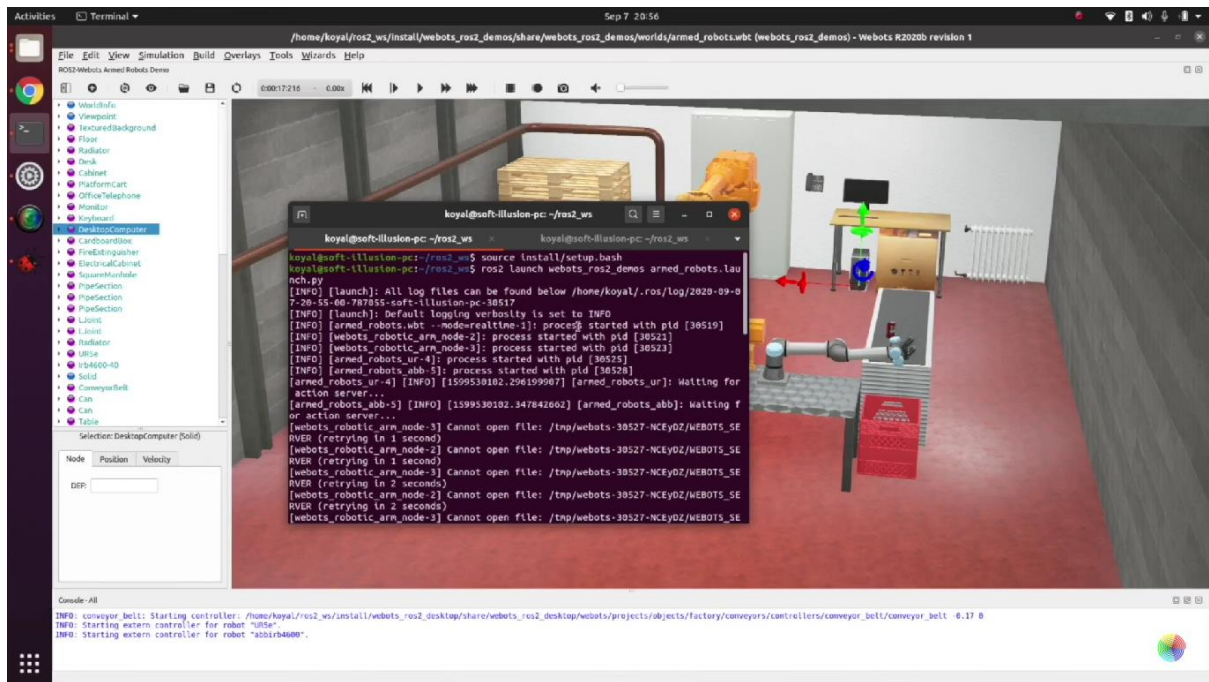


Gambar 8 rviz documentation

- Contoh robot stasioner (ABB & UR5), memiliki solusi langkah demi langkah yang baik dari beberapa implementasi robot stasioner dalam Webots. Sebuah contoh ditunjukkan dalam video. Dalam video, robot ABB dan robot Universal menyelesaikan tugas secara sinkronisasi. Di sini, file peluncuran dalam ROS2 dalam format python dibandingkan dengan format .launch dalam ROS.



Gambar 9 cara kerja robot stationer

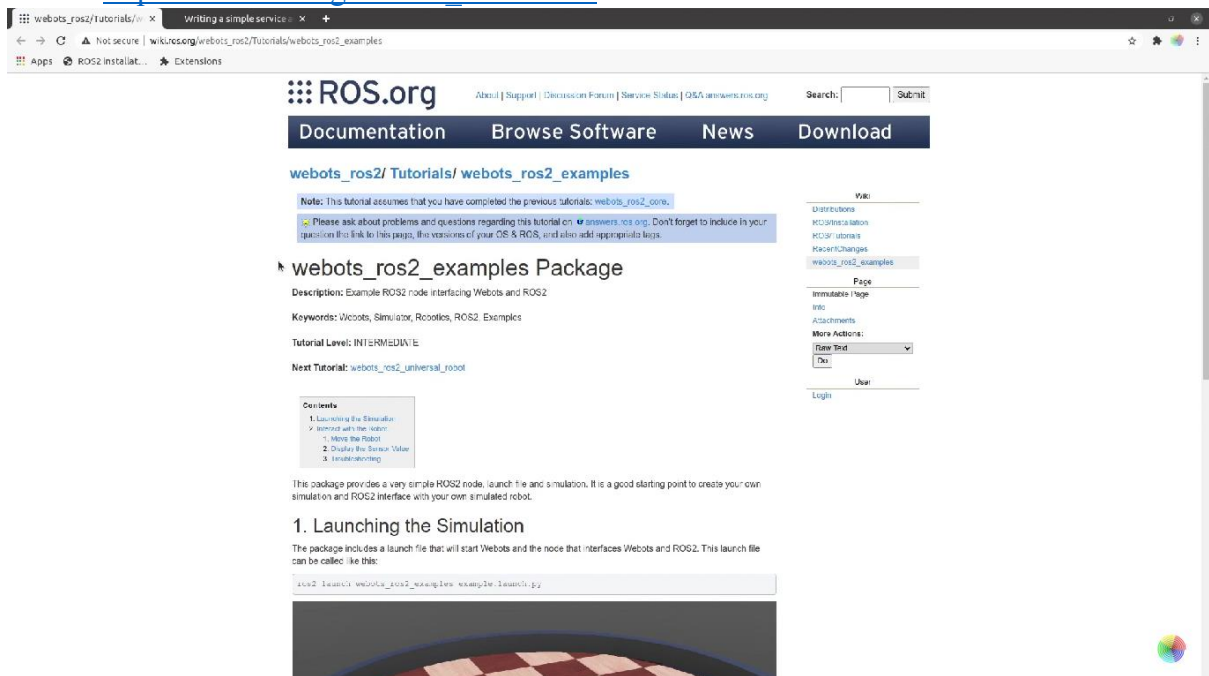


Gambar 10 launch robot stasioner

Video 3 - Using ROS2 Services to interact with Webots | Webots ROS2 tutorial series [Tutorial 3]

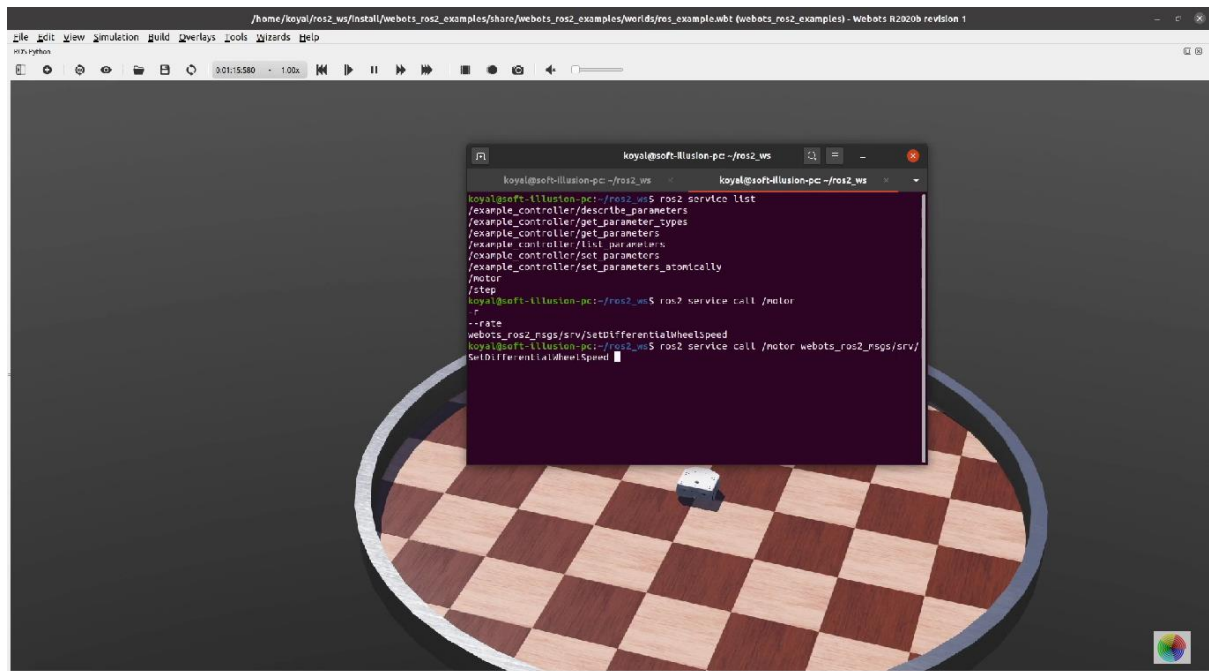
Video 2 terbagi dalam 4 sub, antara lain:

- *Using services with terminal*, terdapat tautan yang digunakan untuk referensi: https://wiki.ros.org/webots_ros2/Tutor



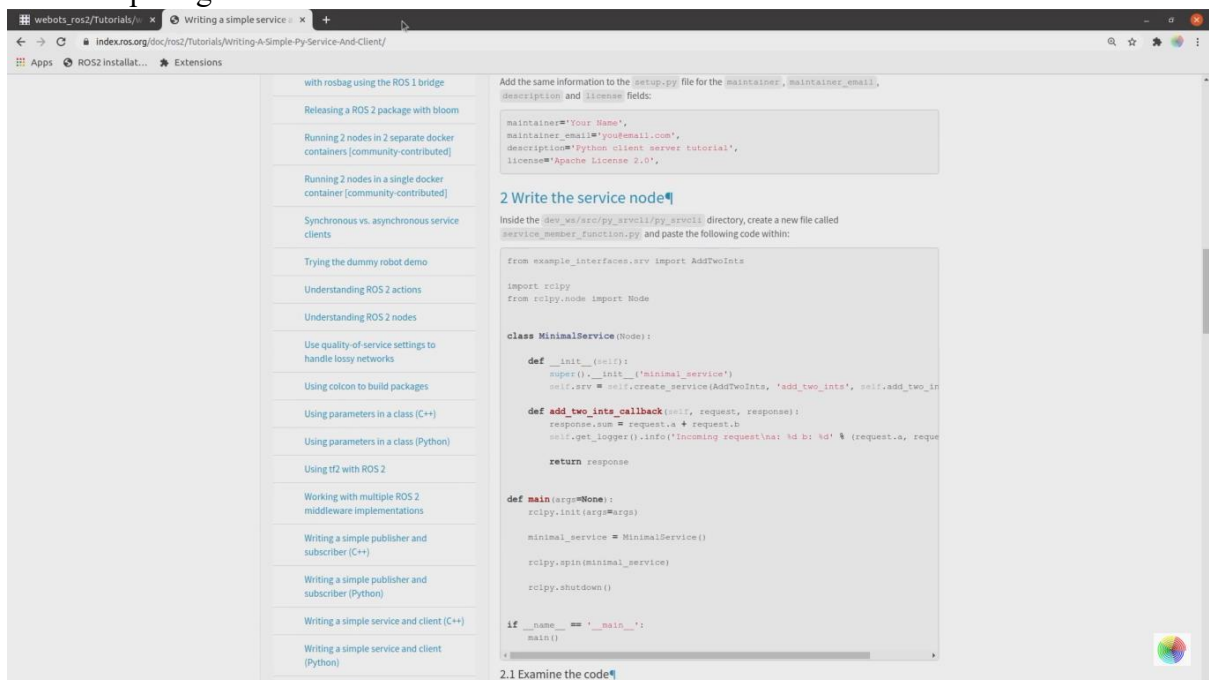
Gambar 11 isi dalam tautan

Dalam praktiknya dalam video diperlihatkan penggunaan perintah seperti `ros2 service list`, `ros2 service call`, `ros2 interface show`, dan lain-lain yang memungkinkan penggunaan layanan melalui terminal.



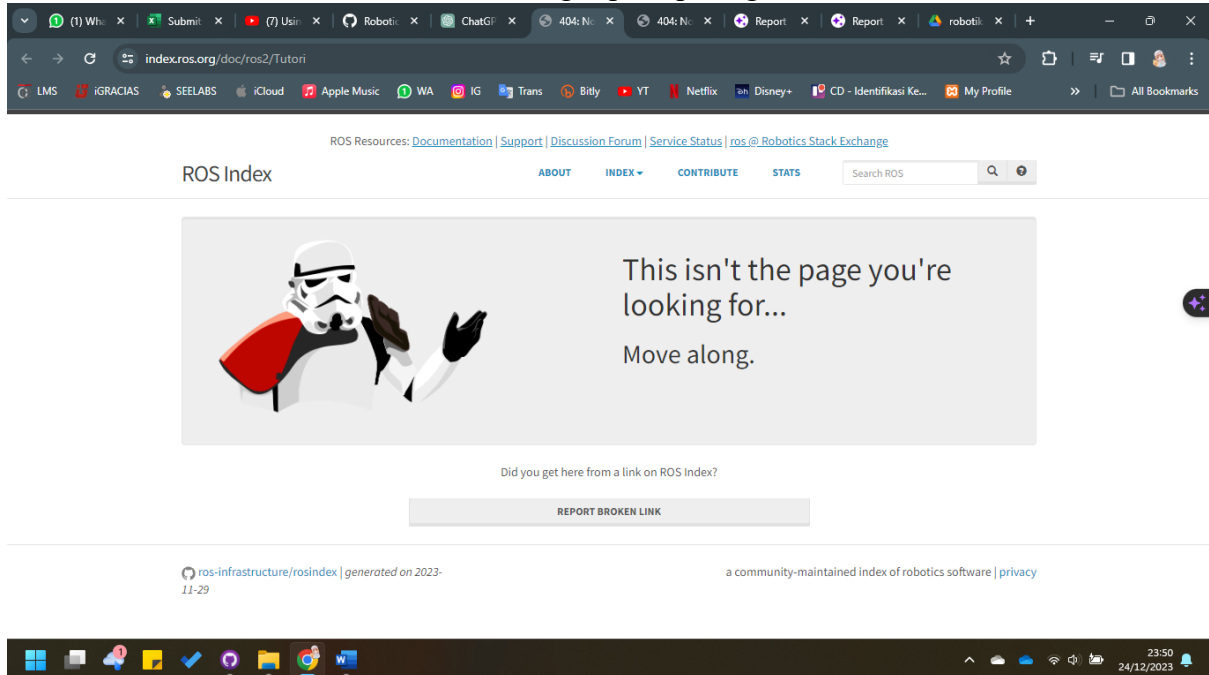
Gambar 12 ros2 services list

- *Using services by making a node*, terdapat tautan <https://index.ros.org/doc/ros2/Tutorials/writing-a-simple-py-service-and-client/> seperti gambar dibawah di dalam video:



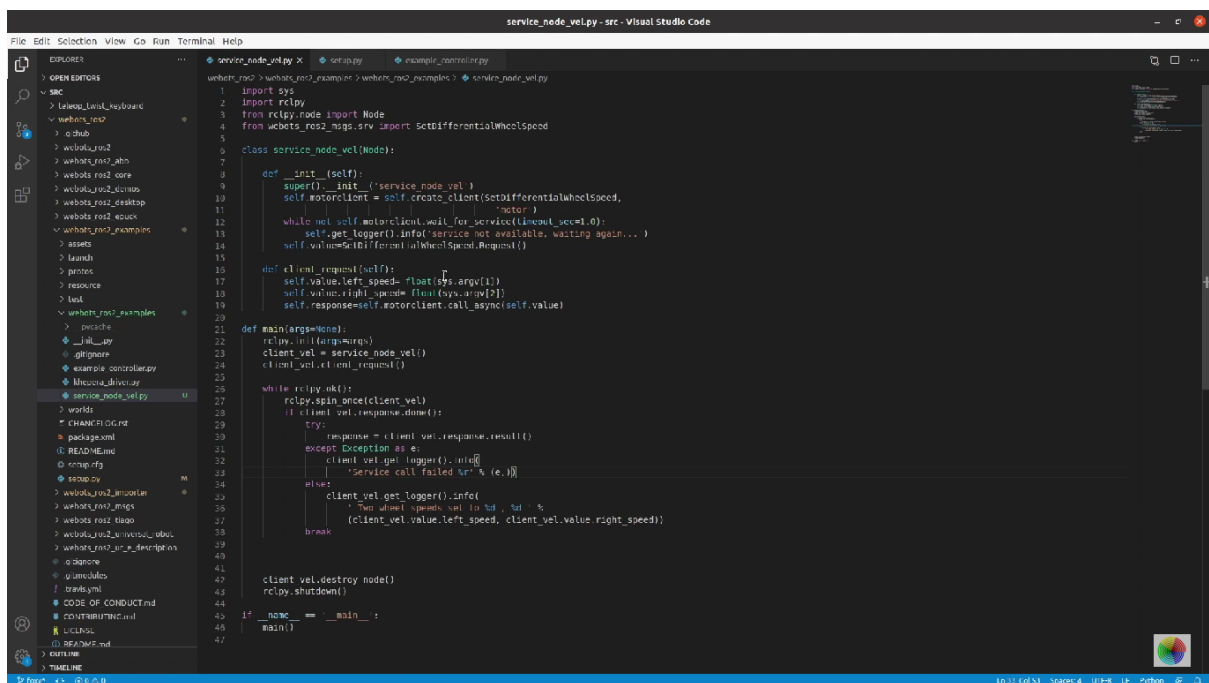
Gambar 13 isi dokumentasi dalam tautan di video

Namun, saat ini tautan tersebut sudah hilang seperti pada gambar:

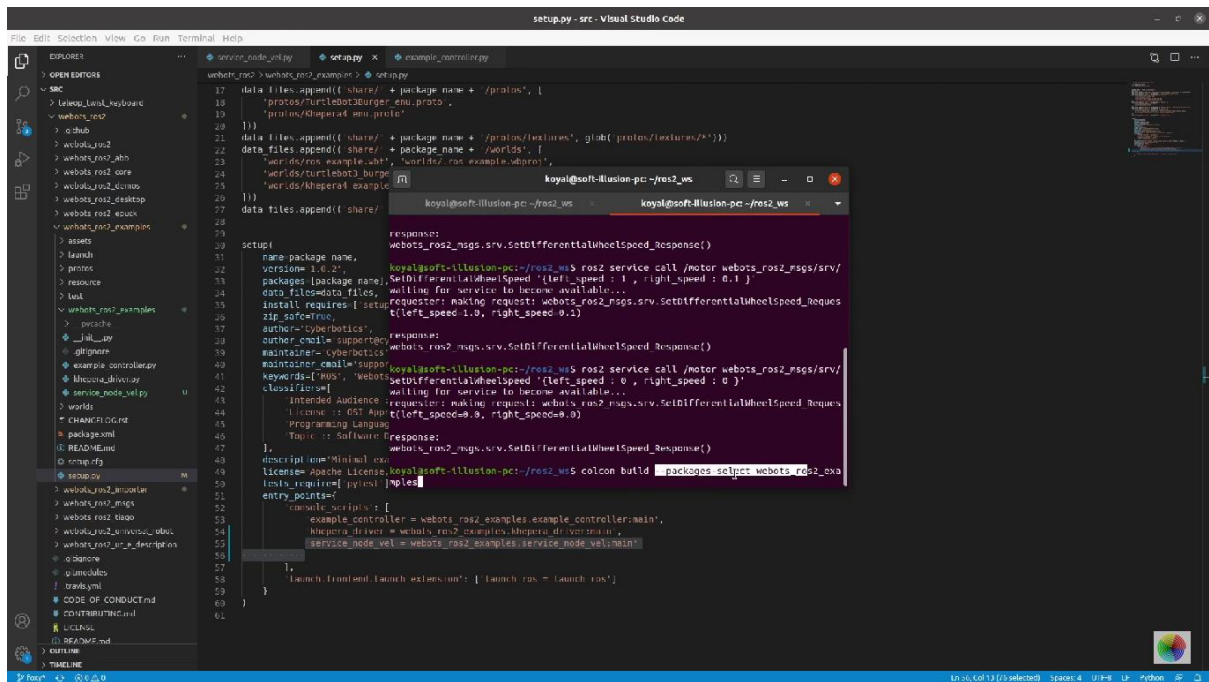


Gambar 14 saat tautan dibuka saat ini

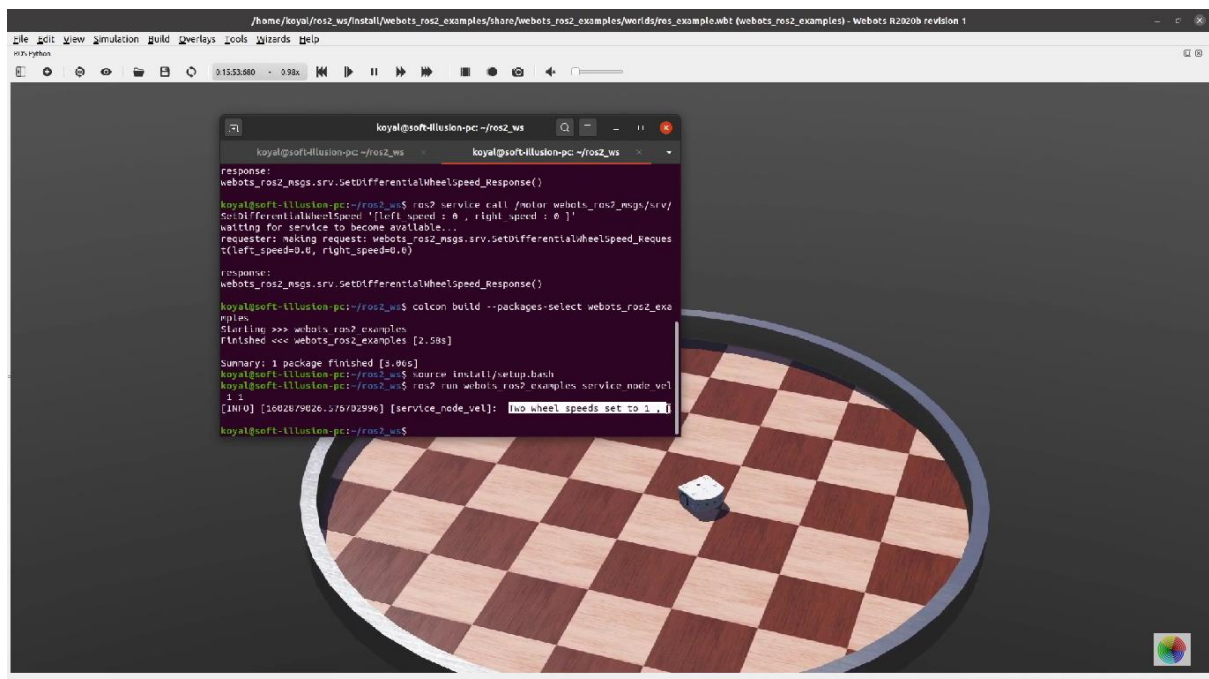
Kode didalam tautan tersebut dijelaskan dalam video beserta metode untuk menyesuaikan setup.py. Fungsi seperti client_request, pengaturan dasar node permintaan, dan tanggapan dalam sebuah kelas juga dijelaskan, beserta perbedaan antara menulis node dalam ROS1 dan ROS2. Robot dapat terlihat bergerak ke arah yang ditentukan menggunakan kecepatan yang diberikan.



Gambar 15 contoh source code controller untuk menggerakkan robot



Gambar 16 *build package*



Gambar 17 lauch dan menggerakan robot dengan node

- Perbedaan antara ROS dan ROS2

ROS 1 Services	VS	ROS 2 Services
In ROS 1 .msg and .srv files can have the same name but the generated code collides. The same is the case for the request and response parts of services.		In ROS 2 the generated code uses separate namespaces to guarantee it is collision-free.
If written in python need not initiate to run the service client or server.		We need to initialize service client and server in setup.py in order to build it and use it. Can do python3 <nodename>.py without adding it in setup.py.
In ROS1, services are synchronous. When your service client asks a request to the server, it is stuck until the server responds (or fails).		In ROS2, services are asynchronous. We need to set them to asynchronous mode .
rosservice and rosparam can not be tweaked by each other.		In ROS2, we also get parameters as services and those parameters can be tweaked by ros2 service and ros2 param.

Gambar 18 perbedaan

- Aplikasi dari layanan

Applications of services in ROS 2

- To set a parameter which we don't need to tweak more often.
- To reset the simulation or swapping the model .
- To change other parameters like time step or camera perspective of world.
- To enable and disable sensors in Webots.
- To change value of actuators position and velocity for their motion.
- Dynamically reconfiguration of world in webots.

