

TUGAS UAS
ROBOTIK



Telkom
University

disusun oleh :

Nama : Yusuf Sulle

Kelas : TK-43-01

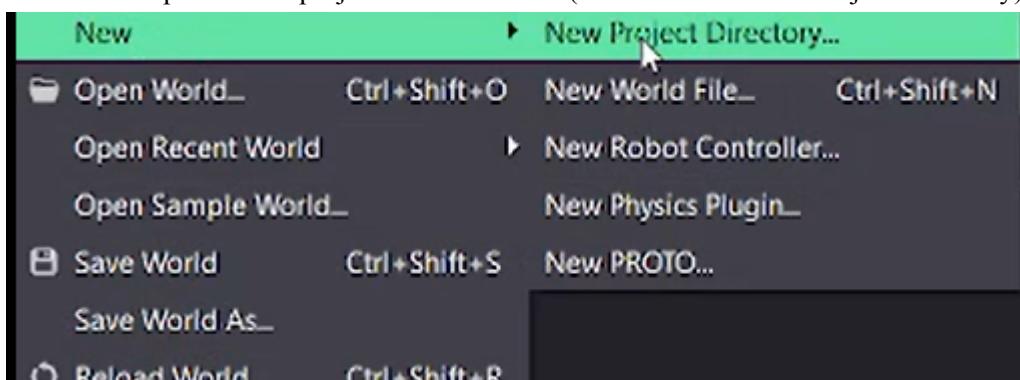
NIM : 1103194015

S1 Teknik Komputer
Fakultas Teknik Elektro
Universitas Telkom

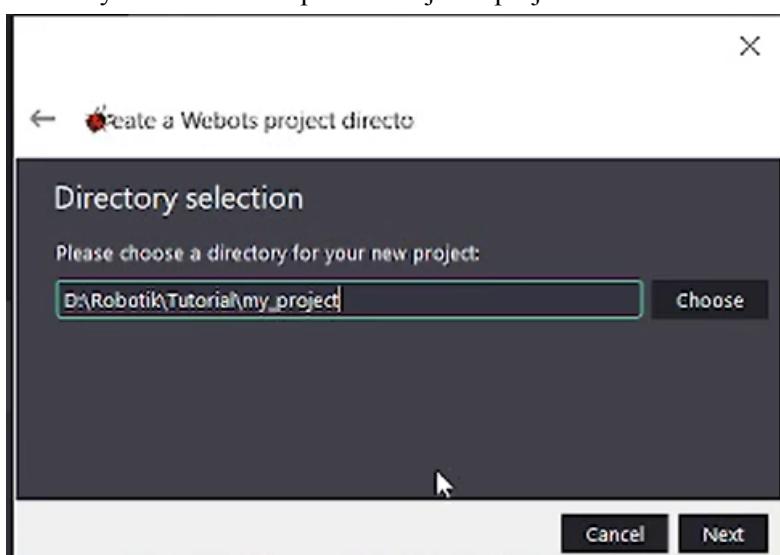
Webots Tutorial Documentation

1. First Simulation

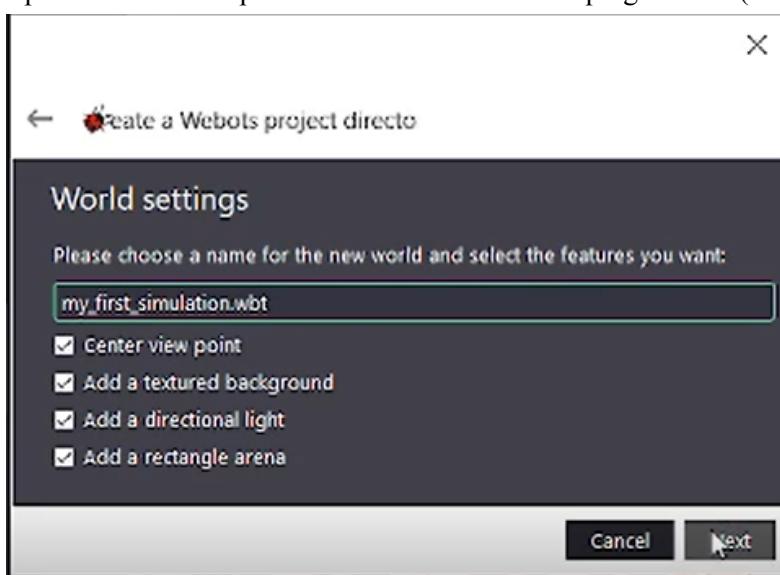
Lokasi untuk pembuatan project baru di webots (File > New > New Project Directory)



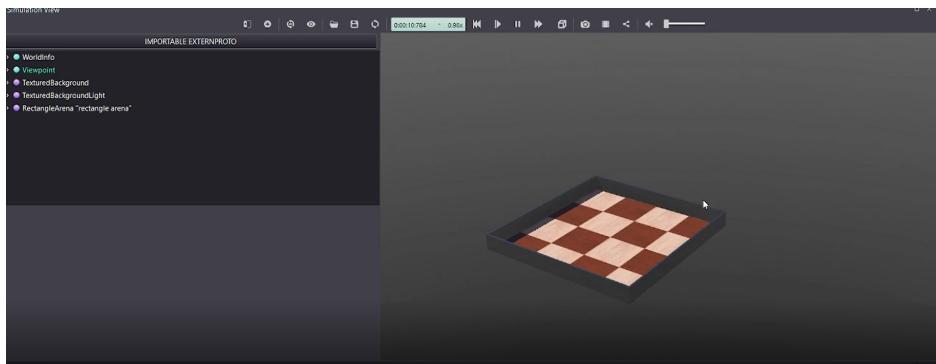
Directory Selection serta penamaan judul project



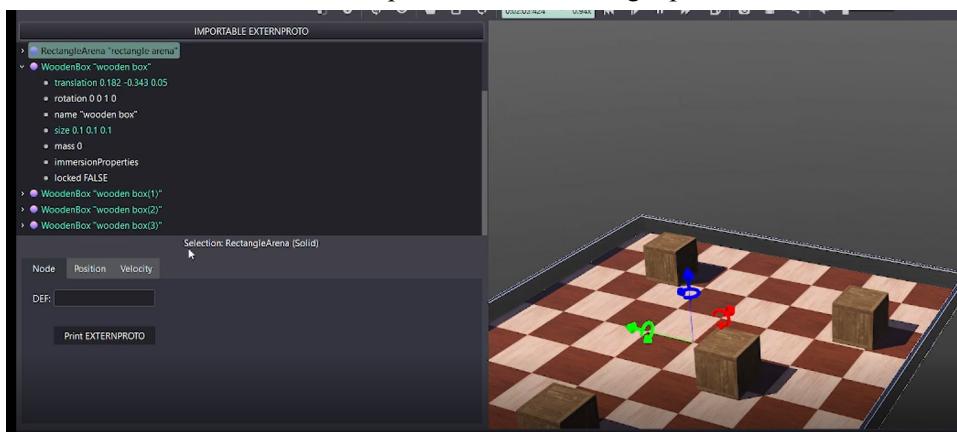
Pengaturan world sesuai dengan kebutuhan, pada kasus ini semua dicentang dan perlu diperhatikan bahwa penamaan world harus didampingi format (.wbt)



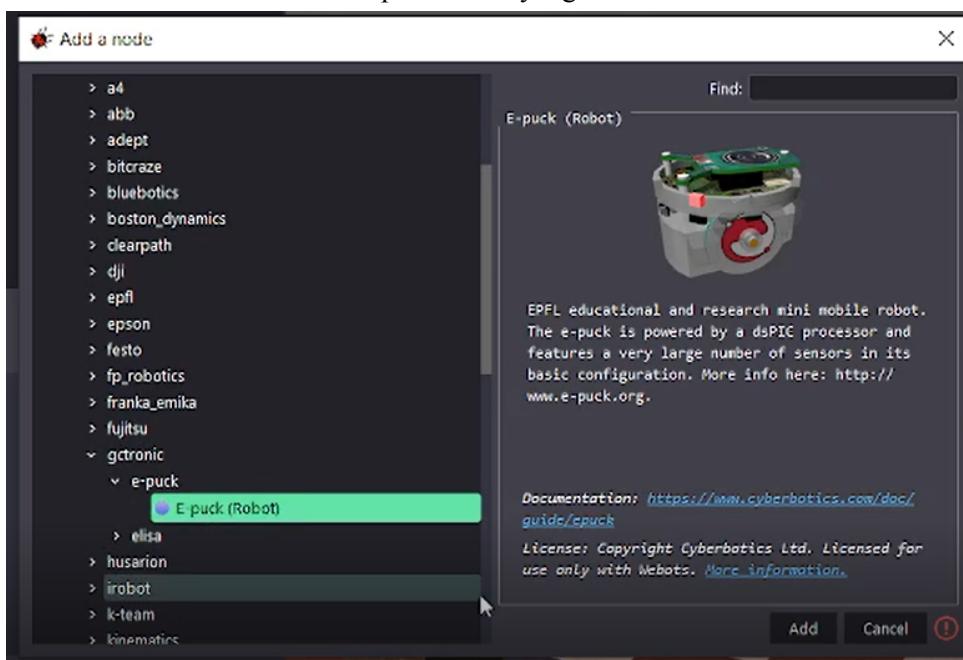
Berikut adalah tampilan awal project pada webots

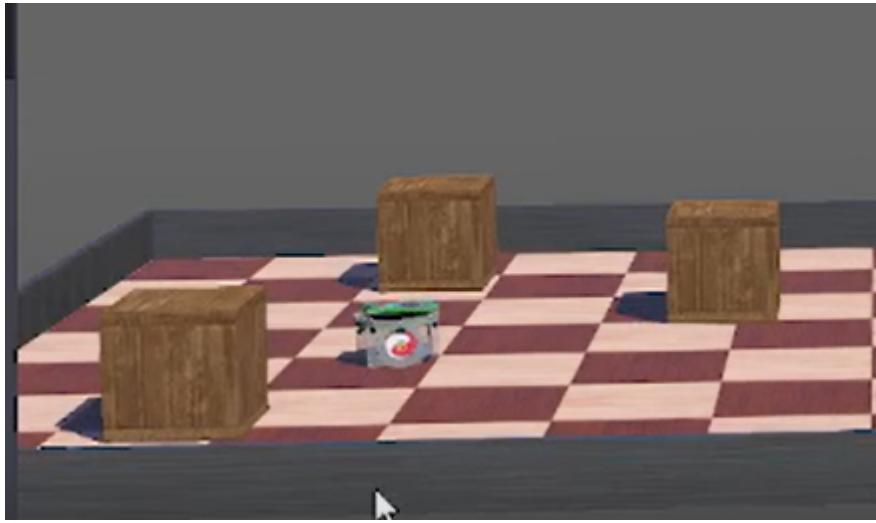


Kemudian menambahkan beberapa wooden box sebagai percobaan

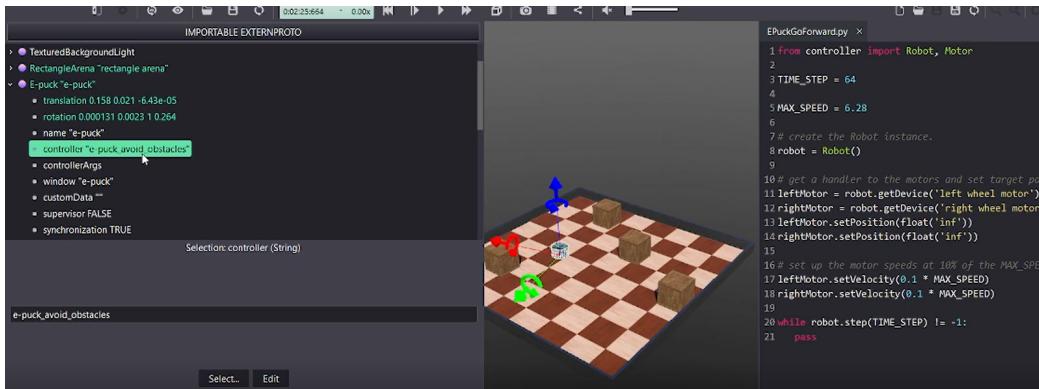


Setelah itu menambahkan robot pada world yang sudah dibuat



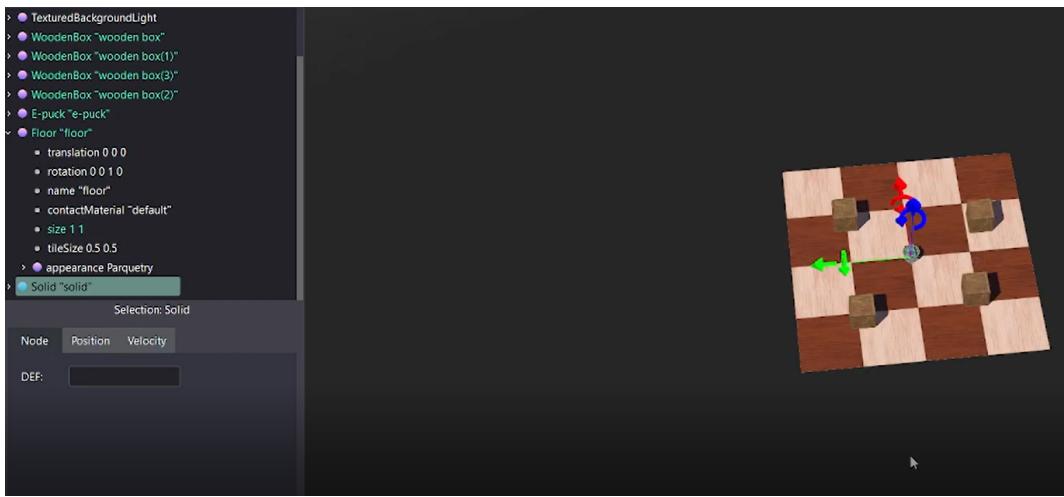


Kemudian kita dapat melakukan modifikasi pada controller untuk memberikan perintah sesuai dengan keinginan kita.

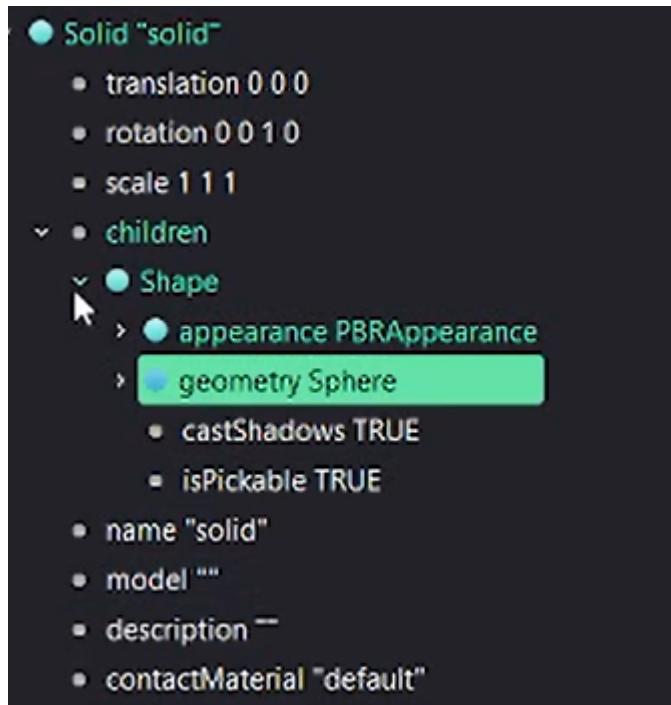


2. Modification of the environment

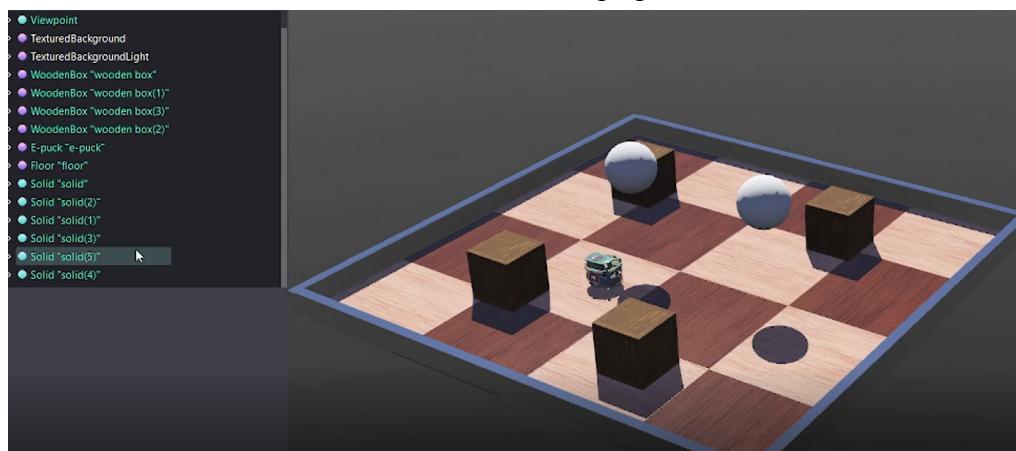
Pada tahap ini akan dilakukan modifikasi pada environment yang telah ada pada bagian sebelumnya.



Dengan menekan add button maka kita dapat menambahkan beberapa obstacles untuk mengisi world yang kita buat. Kemudian pada bagian ini merupakan salah satu contoh dalam pengeditan bentuk dari obstacle.



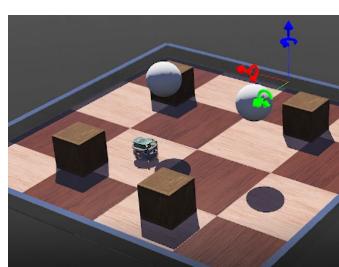
Berikut adalah hasil setelah melakukan beberapa penambahan obstacles



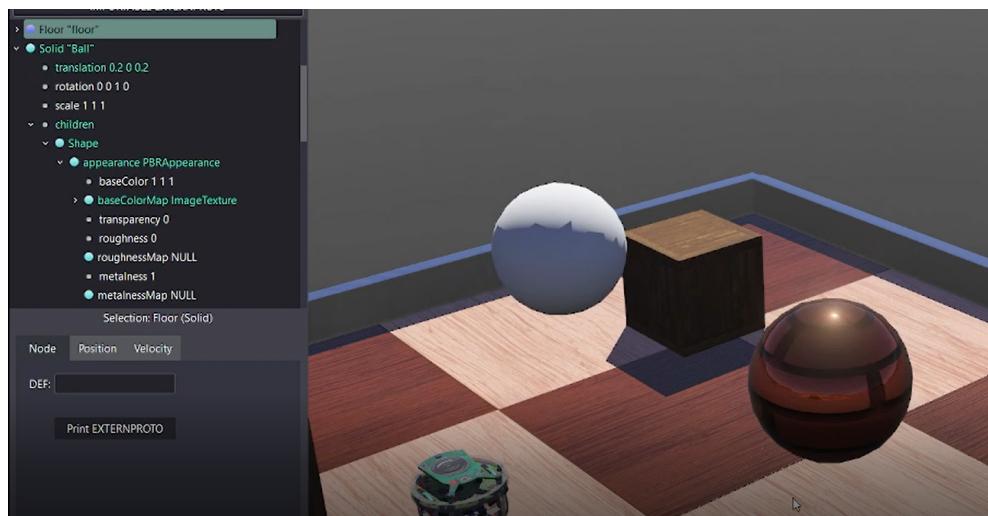
3. Appearance

Pada bagian ini akan berfokus pada cara untuk merubah warna dari obstacles yang telah dibuat. Berikut adalah contoh sebelum dan sesudah metode ini dilakukan

BEFORE



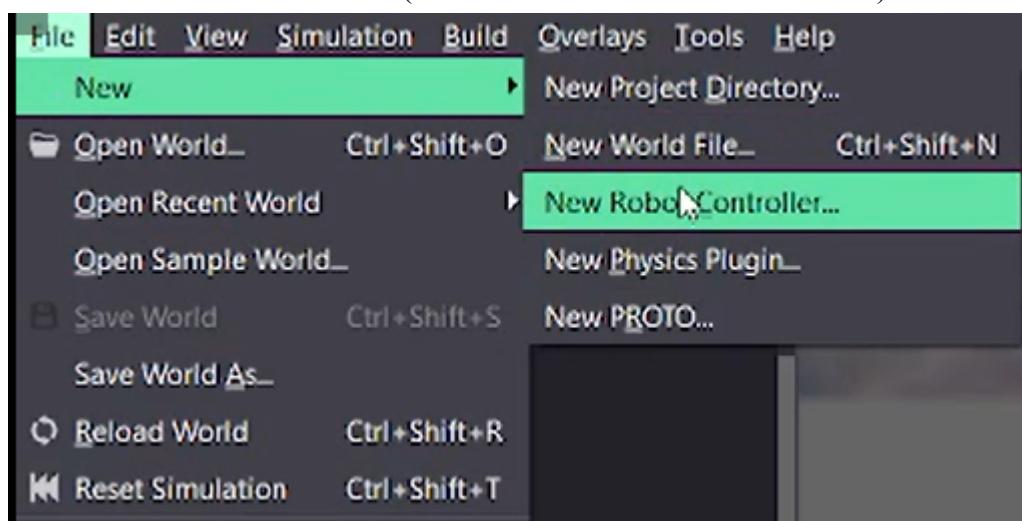
AFTER



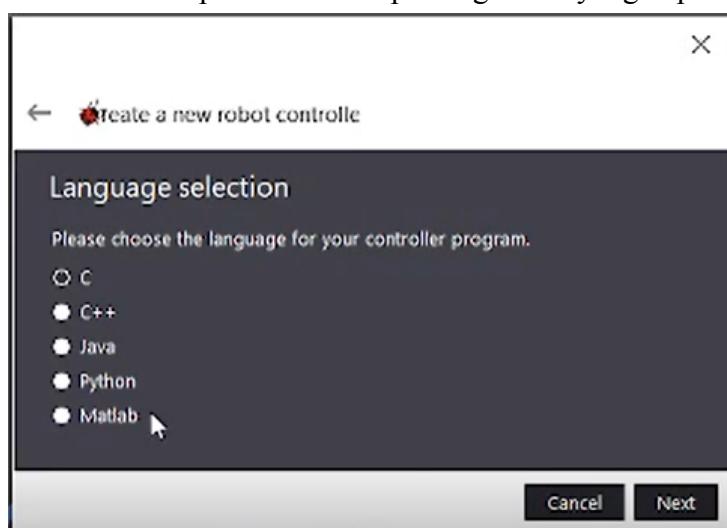
4. Controllers

Pada bagian ini, kita dapat belajar untuk melakukan perubahan pada source code dari robot yang telah kita buat atau robot yang kita gunakan pada world.

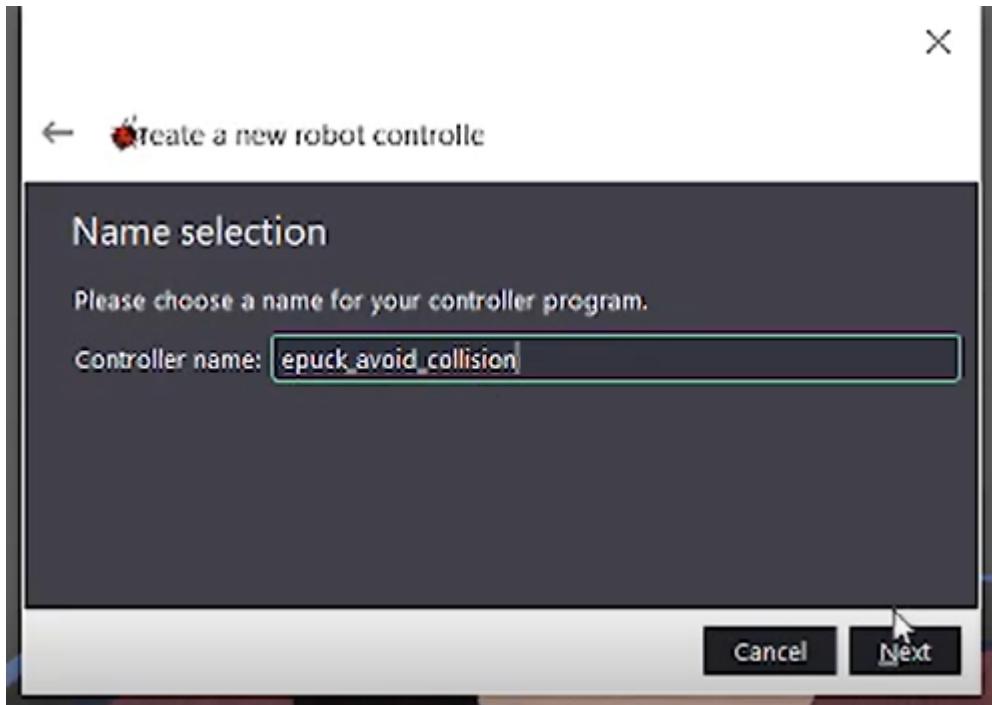
Pembuatan file controller baru (File > New > New Robot Controller)



Berikut adalah pilihan bahasa pemrograman yang dapat kita pakai



Kemudian kita dapat memberikan nama untuk file controller baru kita seperti gambar di bawah ini



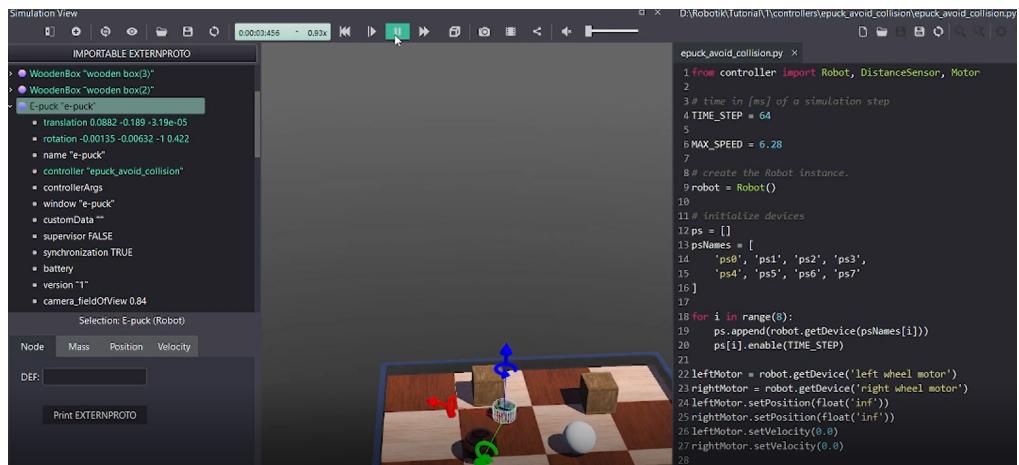
Setelah itu, maka akan muncul sebuah tab text editor pada sebelah kanan yang berisikan default code untuk controller baru kita

A screenshot of a software interface. On the left, there is a 3D simulation window showing a robot on a checkered floor with some obstacles. On the right, there is a text editor window with the file path "D:\Robottk\Tutorial\1\controllers\epuck_avoid_collision\epuck_avoid_collision.py". The code in the editor is as follows:

```
1 """epuck_avoid_collision controller."""
2
3 # You may need to import some classes of the controller module. Ex:
4 #   from controller import Robot, Motor, DistanceSensor
5 from controller import Robot
6
7 # create the Robot instance.
8 robot = Robot()
9
10 # get the time step of the current world.
11 timestep = int(robot.getBasicTimeStep())
12
13 # You should insert a getDevice-like function in order to get the
14 # instance of a device of the robot. Something like:
15 #   motor = robot.getDevice('motorname')
16 #   ds = robot.getDevice('dsname')
17 #   ds.enable(timestep)
18
19 # Main Loop:
20 # - perform simulation steps until Webots is stopping the controller
21 while robot.step(timestep) != -1:
22     # Read the sensors:
23     # Enter here functions to read sensor data, like:
24     #   val = ds.getValue()
25
26     # Process sensor data here.
27
28     # Enter here functions to send actuator commands, like:
29     #   motor.setPosition(10.0)
30     pass
31
32 # Enter here exit cleanup code.
33
```

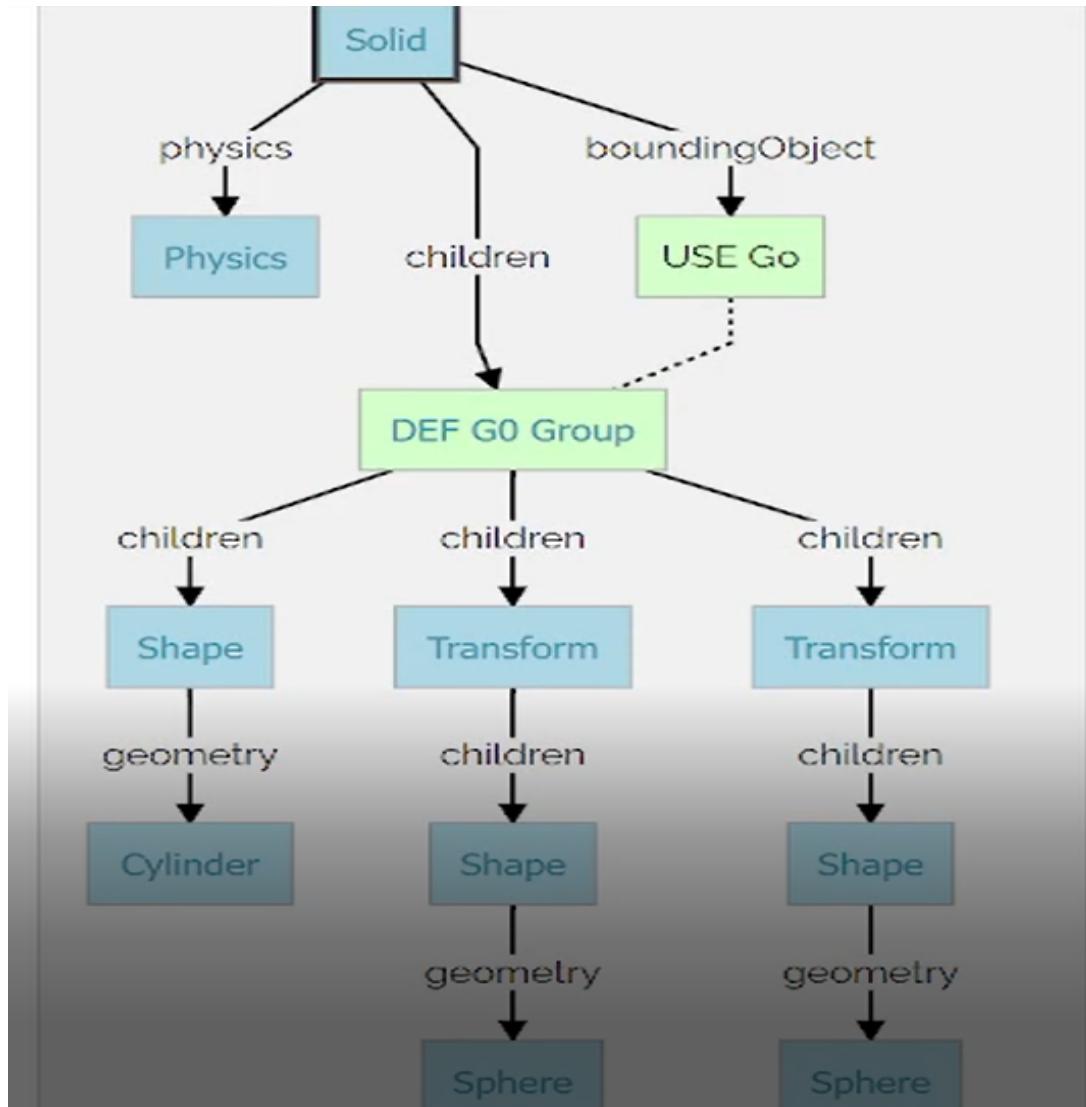
\

Berikut adalah contoh hasil implementasi edit controller dari beberapa tahap di atas

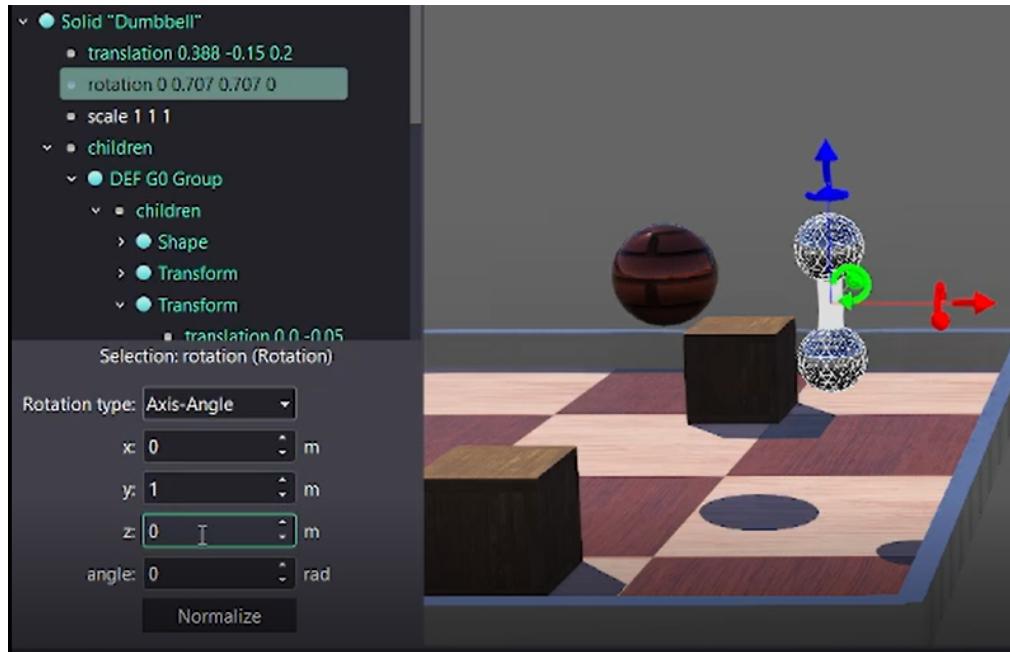


5. Compound Solid and Physics Attributes

Pada bagian ini, kita dapat melakukan beberapa pengaturan pada obstacles yang ada dan memberikan peranan masing-masing. Kemudian kita juga dapat menggabungkan beberapa obstacle menjadi satu dengan menggunakan cara grouping seperti flow di bawah ini :

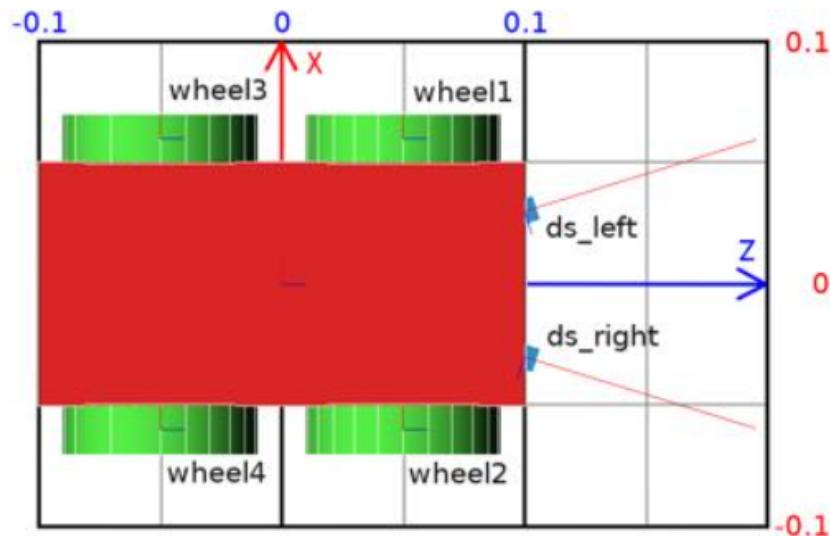


Berdasarkan dari flow di atas, maka kita dapat membuat seperti gambar di bawah ini yang merupakan gabungan dari obstacle 2 bola dan juga 1 silinder dengan memanfaatkan fungsi grouping.

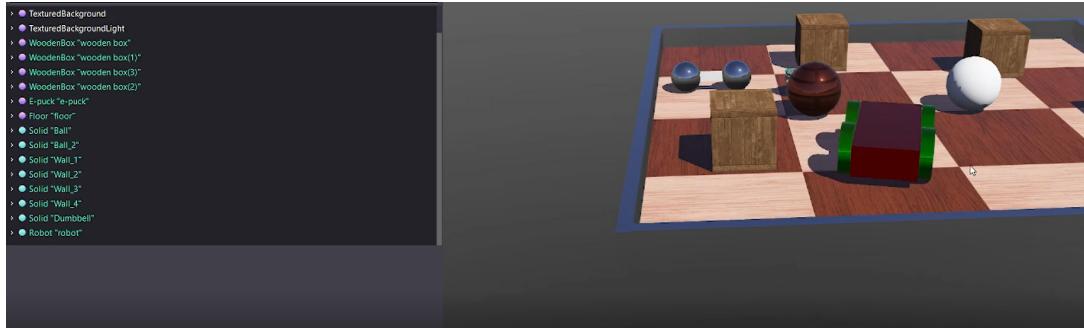


6. Wheels Robot

Pada bagian ini, merupakan sebuah latihan yang dimana kita diharuskan untuk mengkombinasikan beberapa materi yang sudah dipelajari pada bagian-bagian sebelumnya. Berikut adalah desain yang menjadi acuan dalam pembentukan robot secara manual



Kemudian pada bagian ini, penulis dapat mengimplementasikannya ke dalam world yang telah dibuat. Berikut adalah tampilan dari hasilnya :



Pada gambar di atas terdapat sebuah robot mobil yang dibuat sesuai dengan desain yang menjadi acuan dalam latihan ini.

7. Proto

Pada bagian ini, hal yang dilakukan adalah pembuatan jenis file PROTO yang berguna untuk memudahkan pengguna dalam menggunakan node yang berisikan pengaturan robot atau world yang telah dibuat. Hal ini dapat memudahkan pengguna apabila terdapat project baru yang membutuhkan file PROTO yang mempunyai pengaturan yang sama seperti project sebelumnya yang pernah dikerjakan.

Berikut adalah contoh implementasi penggunaan Proto dan cara pembuatannya :

```

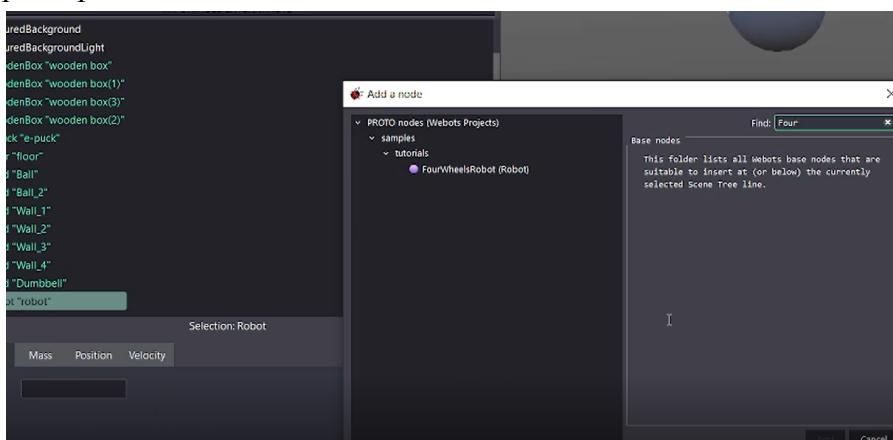
EXPLORER
DATA
controllers
libraries
plugins
protos
└ FourWheelsRobot.proto
worlds
└ 4_wheeled_robot.wbt
appearance.jpg
collision_avoidance.jpg
compound_solid.jpg
compound_solid.wbproj
first_simul.jpg
first_simul.wbproj
obstacles.jpg
red_stick_wall.jpg
└ 4_wheeled_robot.wbt
appearance.wbt
collision_avoidance.wbt
compound_solid.wbt
first_simul.wbt
obstacles.wbt
red_stick_wall.jpg

4_wheeled_robot.wbt
FourWheelsRobot.proto

10 basisTimestep 16
11 contactProperties [
12   ContactProperties {
13     material1 "dumbbell"
14     material2 "dumbbell"
15     coulombFriction [
16       0
17     ]
18   }
19 ]
20 viewpoint {
21   orientation 0.22308763069801116 -0.21374446880084462 -0.9510758177366905 4.619564916738813
22   position 0.07099796781414988 -0.998455534355468 0.45560600314463345
]
23
24 TexturedBackground {
25   I
26   TexturedBackgroundLight {
27   }
28   WoodenBox {
29     translation -0.4964389999201123 0.07439140007988782 0.04996076000000014
30     rotation -0.023074291492107938 0.6452444638382163 0.7636276310801663 3.843447935168992e-17
31     size 0.1 0.1 0.1
32     mass 0.2
33   }
34   WoodenBox {
35     translation 0.14397700007386244 -0.31078399992613753 0.04996076001835642
36     rotation 0 1 0 0
37     name "wooden box(1)"
38     size 0.1 0.1 0.1
39     mass 0.2
}

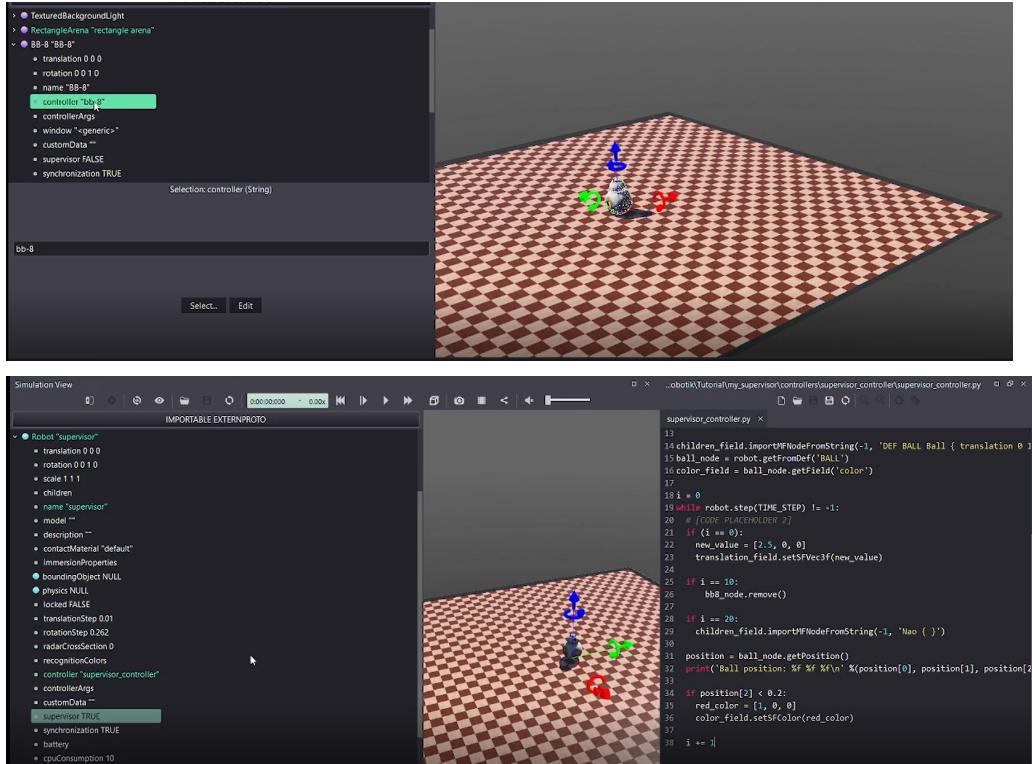
```

Pada gambar diatas, penulis memindahkan beberapa code dari file (.wbt) menuju file dengan format (.proto). Kemudian berikut adalah cara penggunaan dari file berjenis proto pada webots



8. The Supervisor

Pada bagian ini, kita dapat mencoba untuk mengimplementasikan sebuah robot dengan jenis Supervisor. Berikut adalah tampilan dari proses pengaturan robot supervisor.



9. Using ROS

Pada bagian ini, kita harus dapat mengimplementasikan penggunaan ROS. Tetapi pada kasus yang penulis alami, OS yang digunakan adalah windows oleh karena itu kita harus menggunakan WLS2 sebagai subsystem

```
C:\>wsl --install
Installing: Virtual Machine Platform
Virtual Machine Platform has been installed.
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Downloading: WSL Kernel
Installing: WSL Kernel
WSL Kernel has been installed.
Downloading: Ubuntu
The requested operation is successful. Changes will not be effective until the system is rebooted.
```

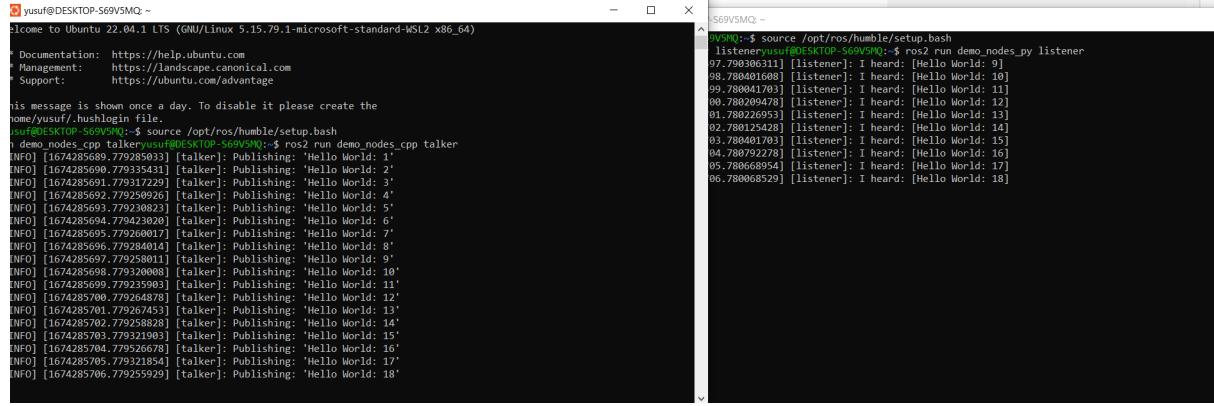
Setelah memiliki wsl maka kita dapat menginstall Ubuntu untuk melakukan running beberapa program terkait.

Untuk lebih penginstalan maka anda dapat mengikuti tutorial pada link di bawah ini :

- GITHUBURL

https://github.com/cyberbotics/webots_ros2/wiki/Windows-Installation-Guide

Setelah melakukan penginstalan ROS dan menghubungkannya dengan webots, maka kita dapat melakukan test dengan menggunakan Ubuntu OS. Hal ini berguna untuk memastikan bahwa ROS sudah bekerja dengan baik.



The screenshot shows two terminal windows. The left window is on a Windows 10 desktop (DESKTOP-S69V5MQ) and the right window is on an Ubuntu 22.04 LTS system (S69V5MQ). Both windows show the output of a ROS node named 'demo_nodes_cpp' running under the 'talker' and 'listener' namespaces respectively. The 'talker' node is publishing 'Hello World' messages at intervals, and the 'listener' node is printing each received message to the console.

```
yusuf@DESKTOP-S69V5MQ:~$ source /opt/ros/humble/setup.bash
[ros@DESKTOP-S69V5MQ:~]$ ros2 run demo_nodes_cpp talker
[INFO] [1674285696.779248033]: [talker]: Publishing: 'Hello World: 1'
[INFO] [1674285696.779253521]: [talker]: Publishing: 'Hello World: 2'
[INFO] [1674285691.779317239]: [talker]: Publishing: 'Hello World: 3'
[INFO] [1674285692.779320926]: [talker]: Publishing: 'Hello World: 4'
[INFO] [1674285693.779320823]: [talker]: Publishing: 'Hello World: 5'
[INFO] [1674285694.779423920]: [talker]: Publishing: 'Hello World: 6'
[INFO] [1674285695.779260817]: [talker]: Publishing: 'Hello World: 7'
[INFO] [1674285696.779284014]: [talker]: Publishing: 'Hello World: 8'
[INFO] [1674285697.779258011]: [talker]: Publishing: 'Hello World: 9'
[INFO] [1674285698.779320088]: [talker]: Publishing: 'Hello World: 10'
[INFO] [1674285698.779339093]: [talker]: Publishing: 'Hello World: 11'
[INFO] [1674285699.779257678]: [talker]: Publishing: 'Hello World: 12'
[INFO] [1674285700.779257683]: [talker]: Publishing: 'Hello World: 13'
[INFO] [1674285702.779258238]: [talker]: Publishing: 'Hello World: 14'
[INFO] [1674285703.779321903]: [talker]: Publishing: 'Hello World: 15'
[INFO] [1674285704.779526678]: [talker]: Publishing: 'Hello World: 16'
[INFO] [1674285705.779321854]: [talker]: Publishing: 'Hello World: 17'
[INFO] [1674285706.779255929]: [talker]: Publishing: 'Hello World: 18'

[ros@DESKTOP-S69V5MQ:~]$ ros2 run demo_nodes_py listener
[INFO] [1674285691.779303111]: [listener]: I heard: [Hello World: 9]
[INFO] [1674285698.779040168]: [listener]: I heard: [Hello World: 10]
[INFO] [1674285701.7790041703]: [listener]: I heard: [Hello World: 11]
[INFO] [1674285702.7790209478]: [listener]: I heard: [Hello World: 12]
[INFO] [1674285703.7790226953]: [listener]: I heard: [Hello World: 13]
[INFO] [1674285704.7790227053]: [listener]: I heard: [Hello World: 14]
[INFO] [1674285705.7790401703]: [listener]: I heard: [Hello World: 15]
[INFO] [1674285706.7790792278]: [listener]: I heard: [Hello World: 16]
[INFO] [1674285707.77908668054]: [listener]: I heard: [Hello World: 17]
[INFO] [1674285708.77908668529]: [listener]: I heard: [Hello World: 18]
```

Kemudian apabila proses di atas sudah berjalan dengan baik maka tahap selanjutnya kita dapat melakukan demo launch dengan menjalankan source code di bawah ini pada Ubuntu :

- source /opt/ros/\$ROS_DISTRO/setup.bash
- export WEBOTS_HOME=/mnt/c/Program\ Files/Webots
- ros2 launch webots_ros2_epuck robot_launch.py

maka dengan secara otomatis aplikasi webots akan terbuka ketika kita menjalankan code (ros2 launch webots_ros2_epuck robot_launch.py).