UTS Semester Ganjil 2023/2024

Nama : Daffa Asyqar Ahmad Khalisheka
NIM : 1103200034
Kelas : TK-44-G7
Matkul : Robotika dan Sistem Cerdas

Technical Report Webots Tutorial Project Wall follower robot using e-puck | Controller code in Python

- GCTronic' e-puck

1. Overview of the Robot
   E-puck dirancang untuk memenuhi beberapa persyaratan berikut:
   - Desain elegan dari struktur mekanikal yang sederhana, elektronika, dan perangkat lunak e-puck merupakan contoh sistem yang bersih dan modern.
   - Kelebihan e-puck terletak pada fleksibilitasnya yang mencakup berbagai kegiatan edukatif, menawarkan banyak kemungkinan dengan sensor-sensor, daya pemrosesan, dan ekstensi yang dimilikinya.
   - Perangkat lunak simulasi Webots telah diintegrasikan dengan robot e-puck untuk memudahkan pemrograman, simulasi, dan pengendalian jarak jauh dari robot fisik tersebut.
   - E-puck adalah perangkat kecil yang mudah diatur di atas meja sebelah komputer. Tidak memerlukan kabel apa pun, memberikan kenyamanan kerja optimal.
   - Keandalan dan Pemeliharaan: e-puck tangguh digunakan oleh mahasiswa dan mudah untuk diperbaiki.
   - Affordable: harga dari e-puck ramah terhadap anggaran universitas.

2. E-puck features

Feature	Description
Size	7.4 cm in diameter, 4.5 cm high
Weight	150 g
Battery	about 3 hours with the provided 5Wh LiION rechargeable battery
Processor	Microchip dsPIC 30F6014A @ 60MHz (about 15 MIPS)
Motors	2 stepper motors with 20 steps per revolution and a 50:1 reduction gear
IR sensors	8 infra-red sensors measuring ambient light and proximity of obstacles in a 4 cm range

Camera	color camera with a maximum resolution of 640x480 (typical use: 52x39 or 640x1)
Microphones	3 omni-directional microphones for sound localization
Accelerometer	3D accelerometer along the X, Y and Z axes
Gyroscope	3D gyroscope along the X, Y and Z axes
LEDs	8 red LEDs on the ring and one green LED on the body
Speaker	on-board speaker capable of playing WAV or tone sounds
Switch	16 position rotating switch
Bluetooth	Bluetooth for robot-computer and robot-robot wireless communication
Remote Control	infra-red LED for receiving standard remote control commands
Expansion bus	expansion bus to add new possibilities to your robot
Programming	C programming with the GNU GCC compiler system
Simulation	Webots facilitates the programming of e-puck with a powerful simulation, remote control and cross-compilation system

- E-puck Model

1. E-puck characteristics

Characteristics	Values
Diameter	71 mm
Height	50 mm
Wheel radius	20.5 mm
Axle Length	52 mm
Weight	0.16 kg
Max. forward/backward speed	0.25 m/s
Max. rotation speed	6.28 rad/s

E-puck characteristics

Model standar dari e-puck disediakan dalam file PROTO "E-puck.proto" yang terletak di direktori "WEBOTS_HOME/projects/robots/gctronic/e-puck/protos" dari distribusi Webots (lihat juga file PROTO "E-puckDistanceSensor.proto" dan file PROTO "E-puckGroundSensors.proto"); Anda akan menemukan spesifikasi lengkap di dalamnya. Dua bidang PROTO, yaitu groundSensorSlot dan turretSlot, telah disertakan dalam model simulasi untuk mendukung modul ekstensi. Secara khusus, modul sensor tanah ekstensi dari robot e-puck nyata dimodelkan menggunakan PROTO "E-puckGroundSensors.proto" di Webots untuk menyediakan 3 sensor infra-merah opsional yang mengarah ke tanah di depan robot. Nama perangkat yang disimulasikan yang akan digunakan sebagai argumen dari fungsi wb_robot_get_device (lihat bagian Robot) disajikan dalam tabel di bawah ini.

2. Devices names

Device	Name
Motors	'left wheel motor' and 'right wheel motor'
Position sensors	'left wheel sensor' and 'right wheel sensor'
Proximity sensors	'ps0' to 'ps7'
Light sensors	'ls0' to 'ls7'
LEDs	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Accelerometer	'accelerometer'
Gyro	'gyro'
Ground sensors (extension)	'gs0', 'gs1' and 'gs2'
Speaker	'speaker'

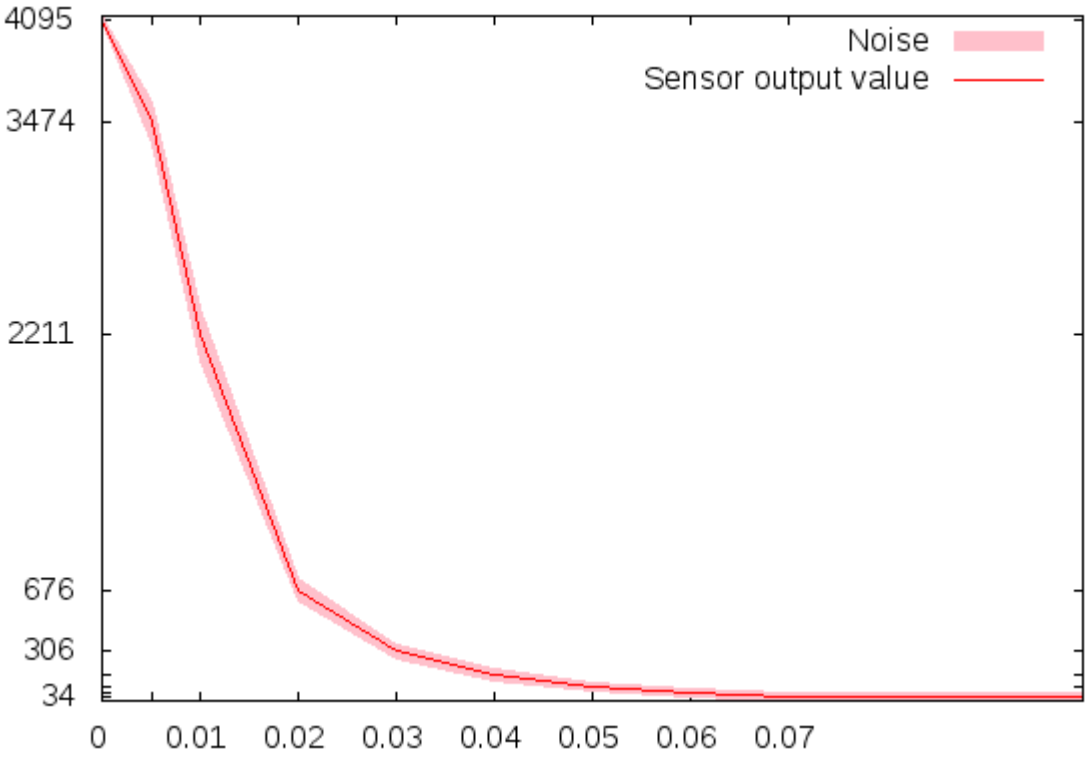
Devices names

Fungsi wb_motor_set_velocity dan wb_position_sensor_get_value digunakan untuk mengatur kecepatan robot dan membaca nilai encoder-nya.

3. Devices orientations

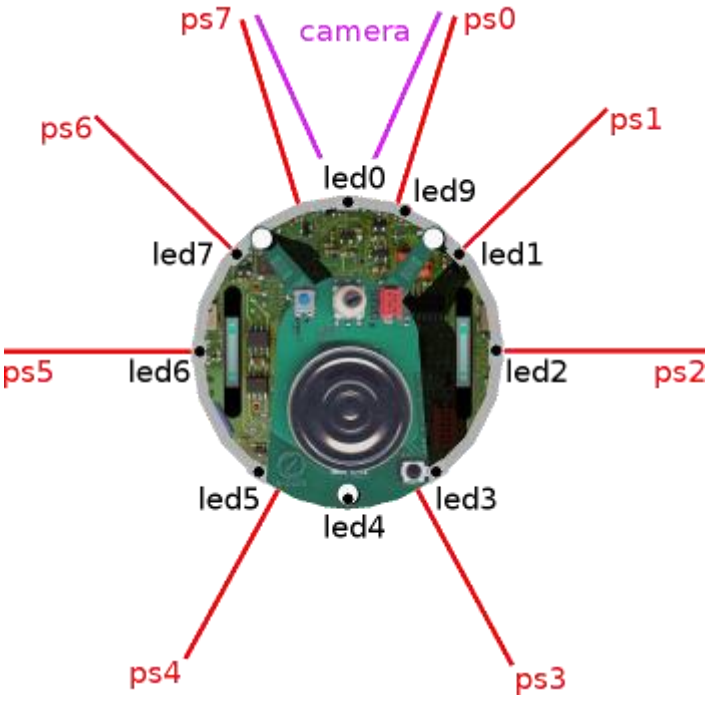
Device	x (m)	y (m)	z (m)	Orientation (rad)
ps0	0.010	0.033	-0.030	1.27
ps1	0.025	0.033	-0.022	0.77
ps2	0.031	0.033	0.00	0.00
ps3	0.015	0.033	0.030	5.21
ps4	-0.015	0.033	0.030	4.21
ps5	-0.031	0.033	0.00	3.14159
ps6	-0.025	0.033	-0.022	2.37
ps7	-0.010	0.033	-0.030	1.87
camera	0.000	0.028	-0.030	4.71239

Arah maju (forward) dari e-puck ditentukan oleh sumbu z negatif dari koordinat dunia. Ini juga merupakan arah yang diarahkan oleh mata kamera. Vektor arah kamera menghadap ke arah yang berlawanan, yaitu arah sumbu z positif. Arah sumbu roda ditentukan oleh sumbu x positif. Sensor kedekatan (proximity sensors), sensor cahaya (light sensors), dan LED dinomori searah jarum jam. Lokasi dan orientasi mereka ditunjukkan dalam gambar ini. Kolom terakhir pada gambar tersebut mencantumkan sudut antara sumbu x negatif dan arah perangkat, dengan bidang zOx diorientasikan berlawanan arah jarum jam. Perlu dicatat bahwa sensor kedekatan dan sensor cahaya sebenarnya adalah perangkat yang sama pada robot nyata yang digunakan dalam mode yang berbeda, sehingga arahnya bersamaan. Respon sensor kedekatan disimulasikan sesuai dengan tabel pencarian dalam gambar ini; tabel ini hasil dari kalibrasi yang dilakukan pada robot nyata.



Proximity sensor response against distance

Resolusi kamera terbatas pada 52x39 piksel, karena itu merupakan gambar persegi panjang maksimum dengan rasio 4:3 yang dapat diperoleh dari antarmuka kontrol jarak jauh dengan robot sesungguhnya.

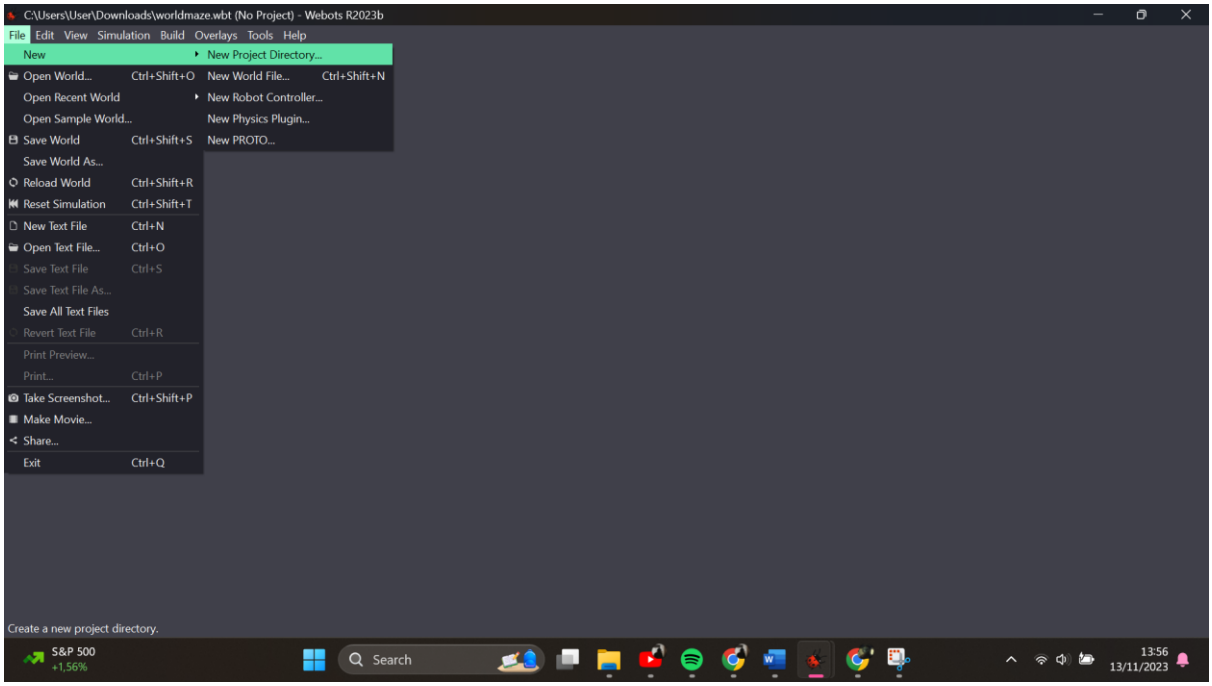


Sensors, LEDs and camera

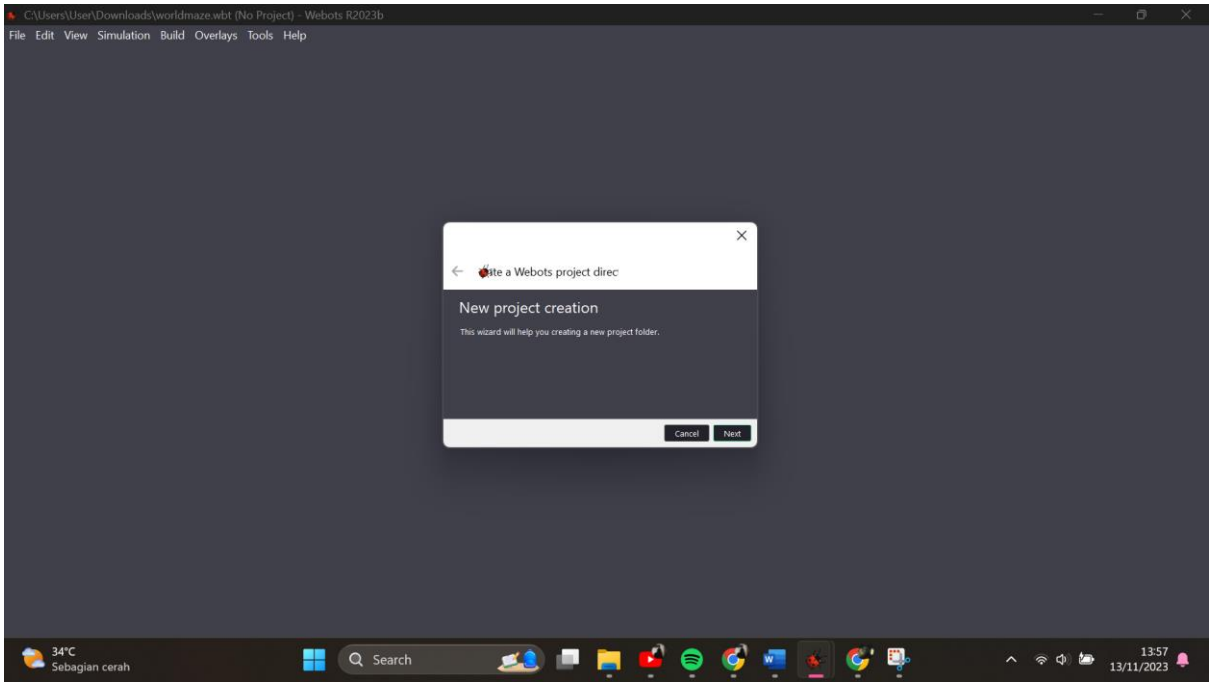
- Pembahasan Tutorial Webots Robot pengikut Project Wall menggunakan e-puck // Kode pengontrol dengan Python tiap langkah:

1. Part One: Project Setup
   Learn to create a new project in Webots, change the arena size and add a e-puck robot in Webots.

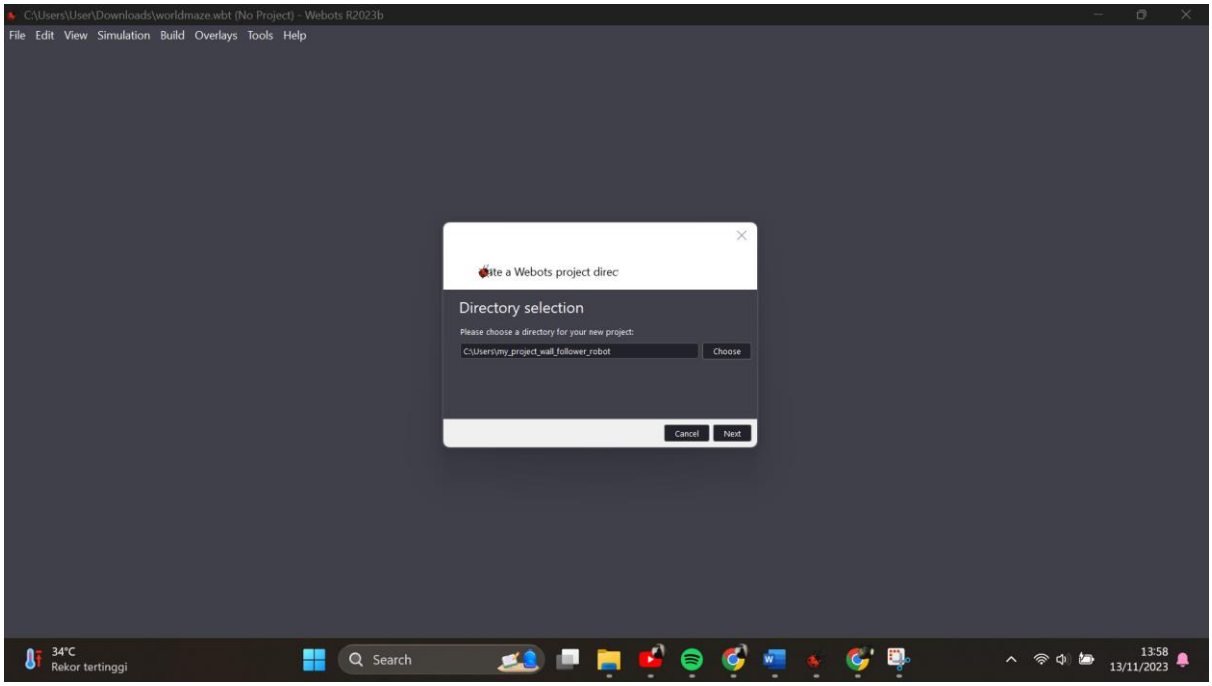
   Klik Menu Bar > File > New > New Project Directory



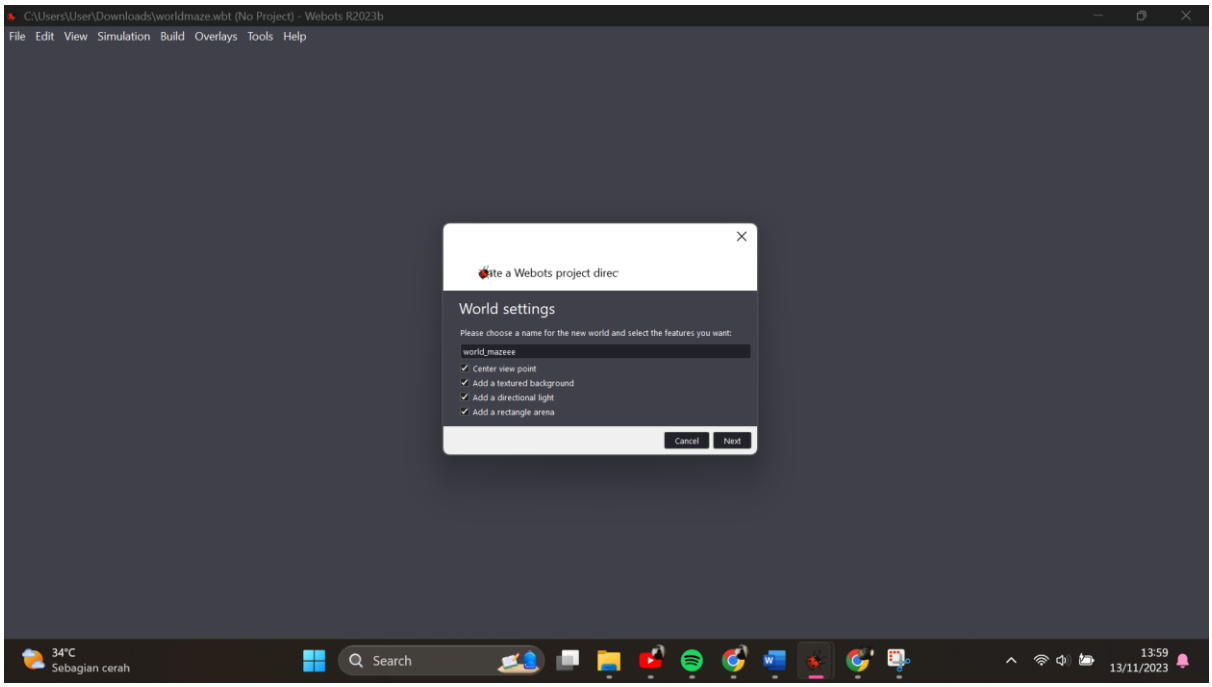
   Klik Next



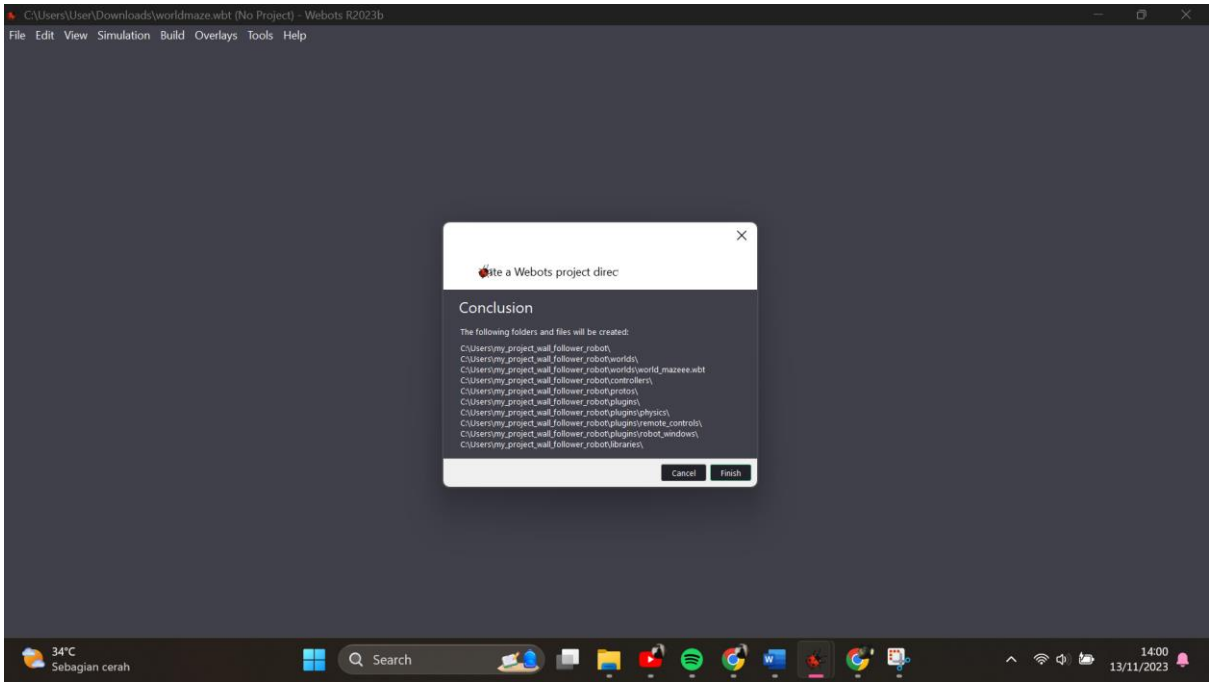
Name your project > C:\Users\my_project_wall_follower_robot > click next



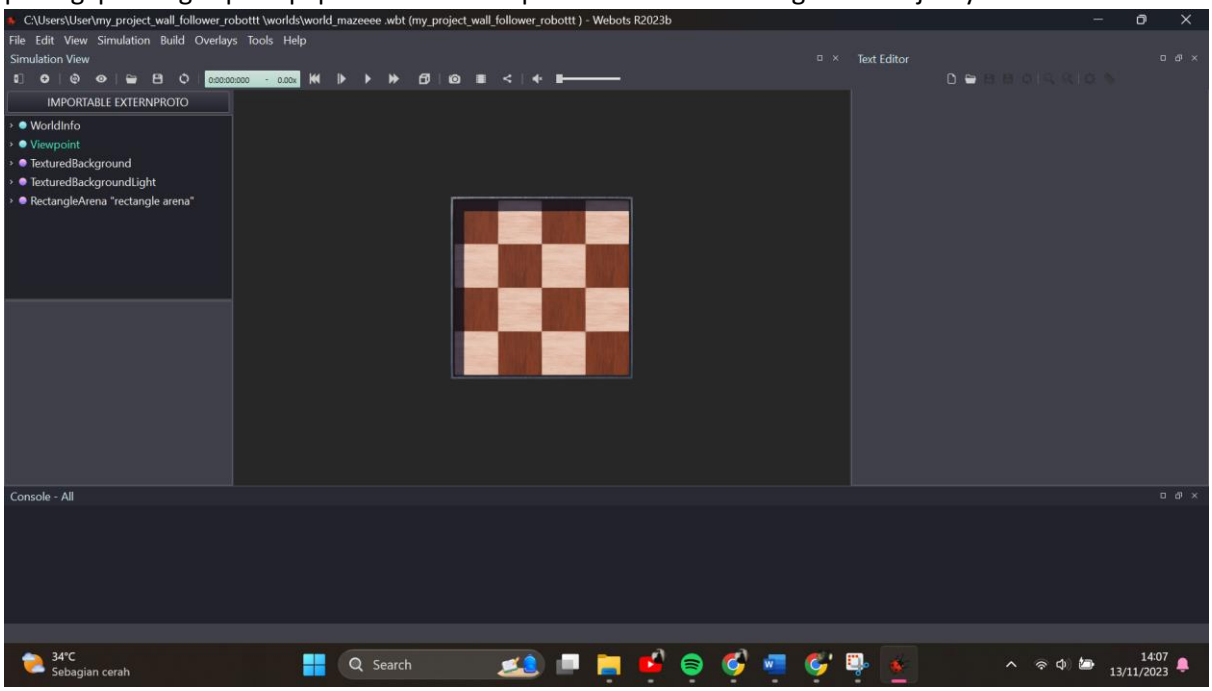
Name your world > world_mazeee > select “add a rectangle arena”



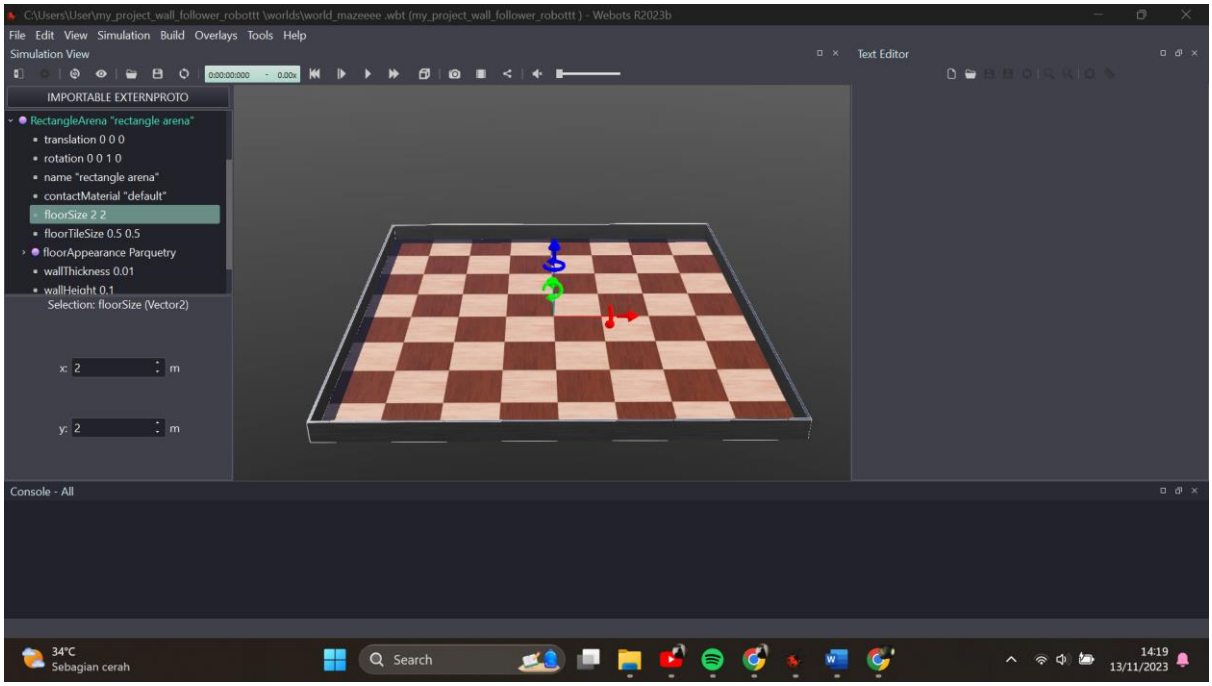
Click Finish



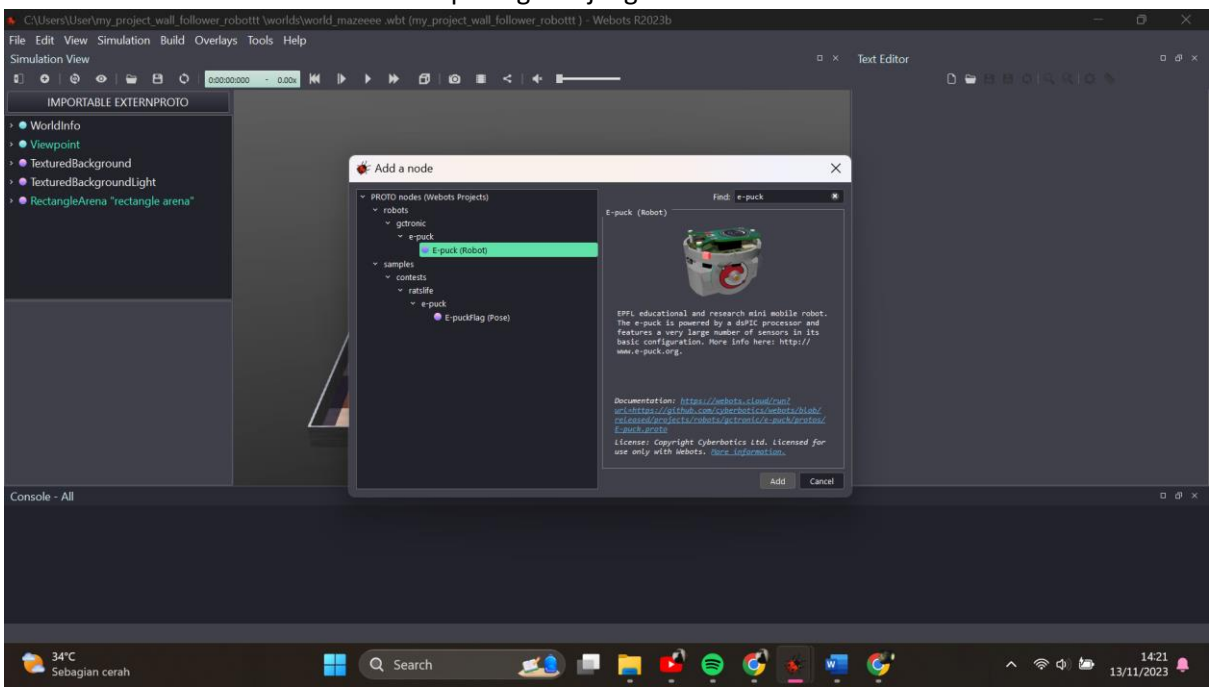
Sudah masuk ke dalam new project dan new world baru sudah dibuat dengan ada arena persegi paniang seperti papan catur dan siap untuk melakukan langkah selanjutnya



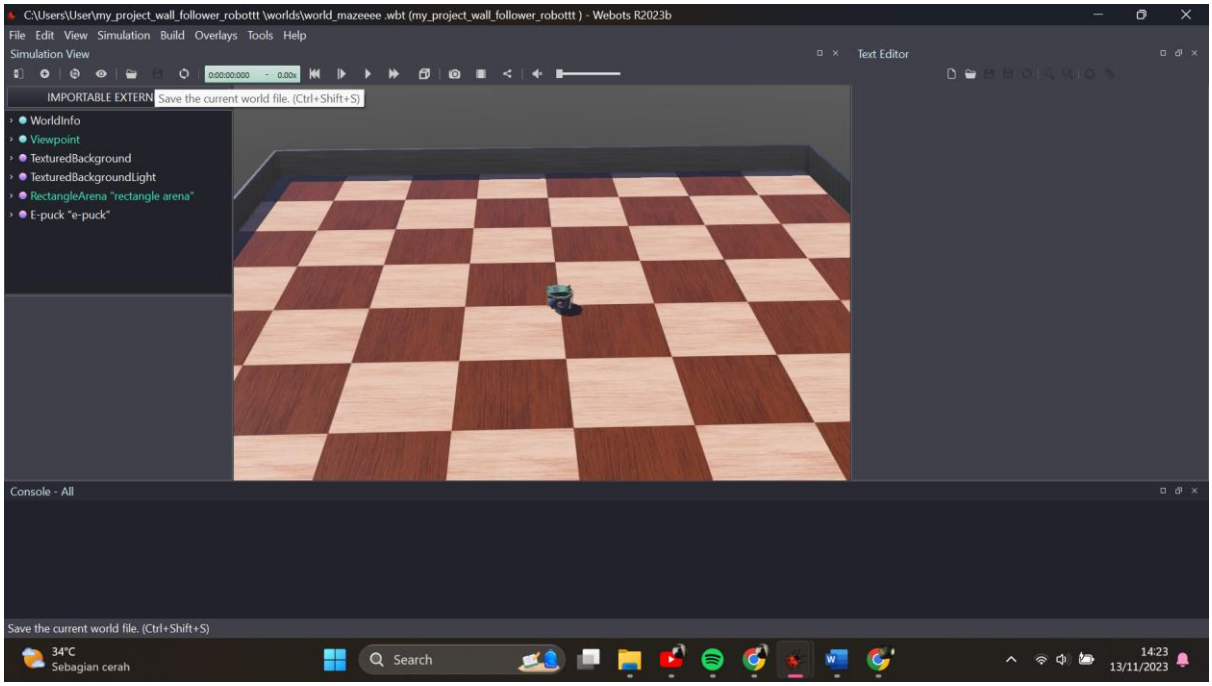
Klik arena Persegi Panjang > Klik floor sizenya > x = 2m & y = 2m



Click on + to add robot > type e-puck in the search bar > select e-puck robot > click add untuk menambahkan robot ke dalam arena persegi Panjang

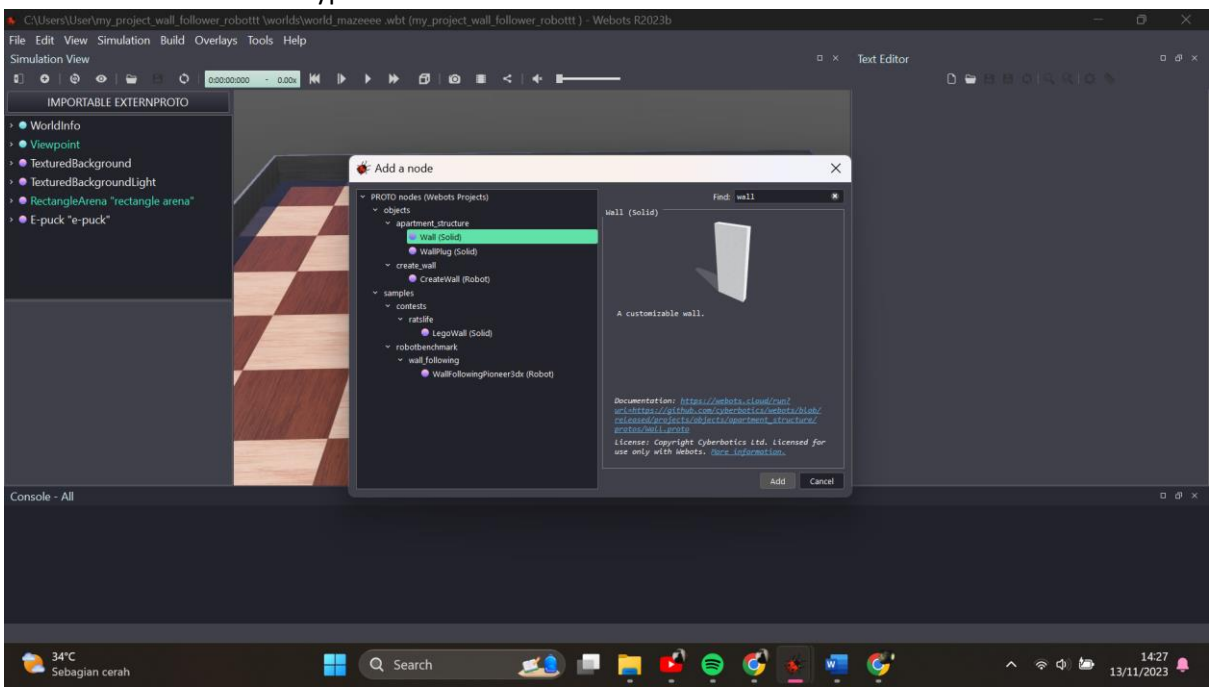


Robot e-puck sudah ditambahkan > click save untuk menyimpan world nya > ctrl + shift + s

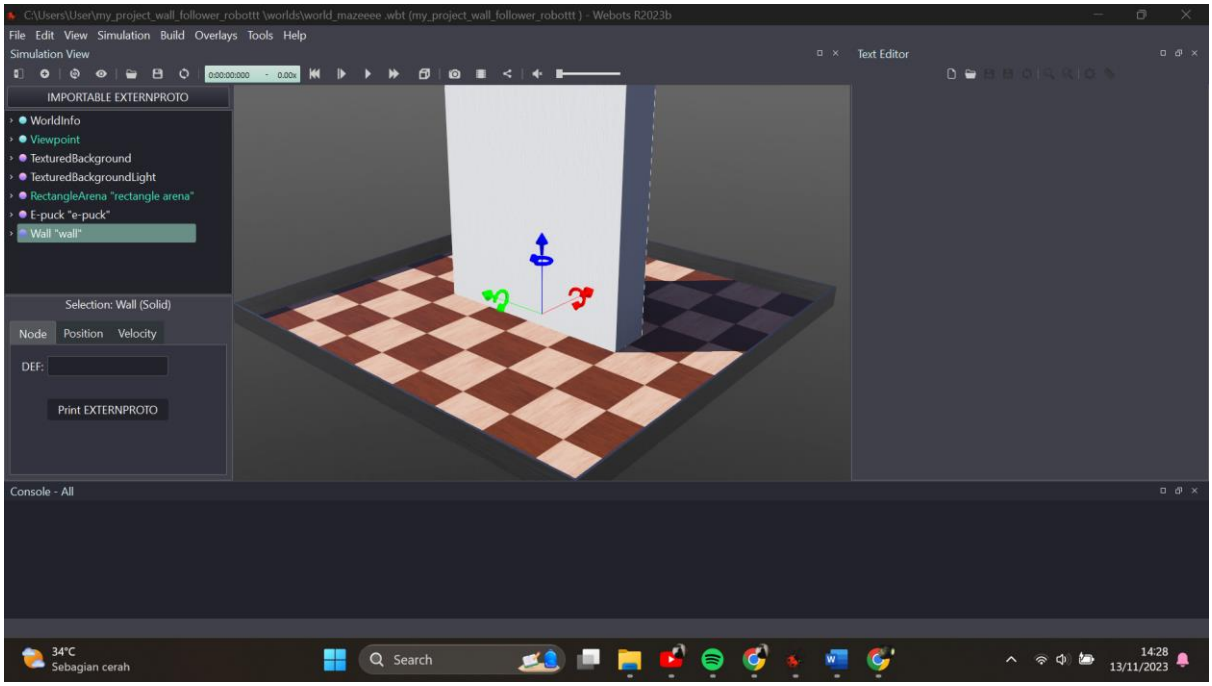


2. Part Two: Build a maze
   Learn to add objects like a wall in Webots. Then learn to translate and rotate objects in Webots. Duplicate the walls to create your own maze in Webots.

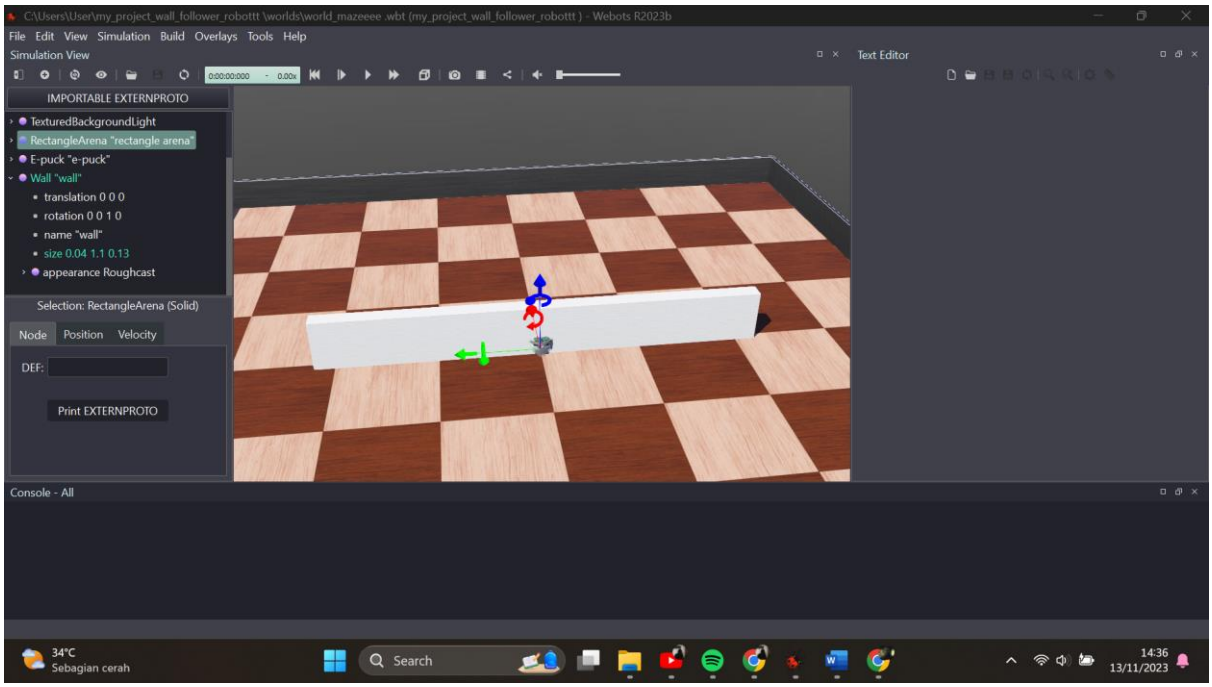
   Click on + to add a wall > type “wall” in the search bar > select “wall” > click add



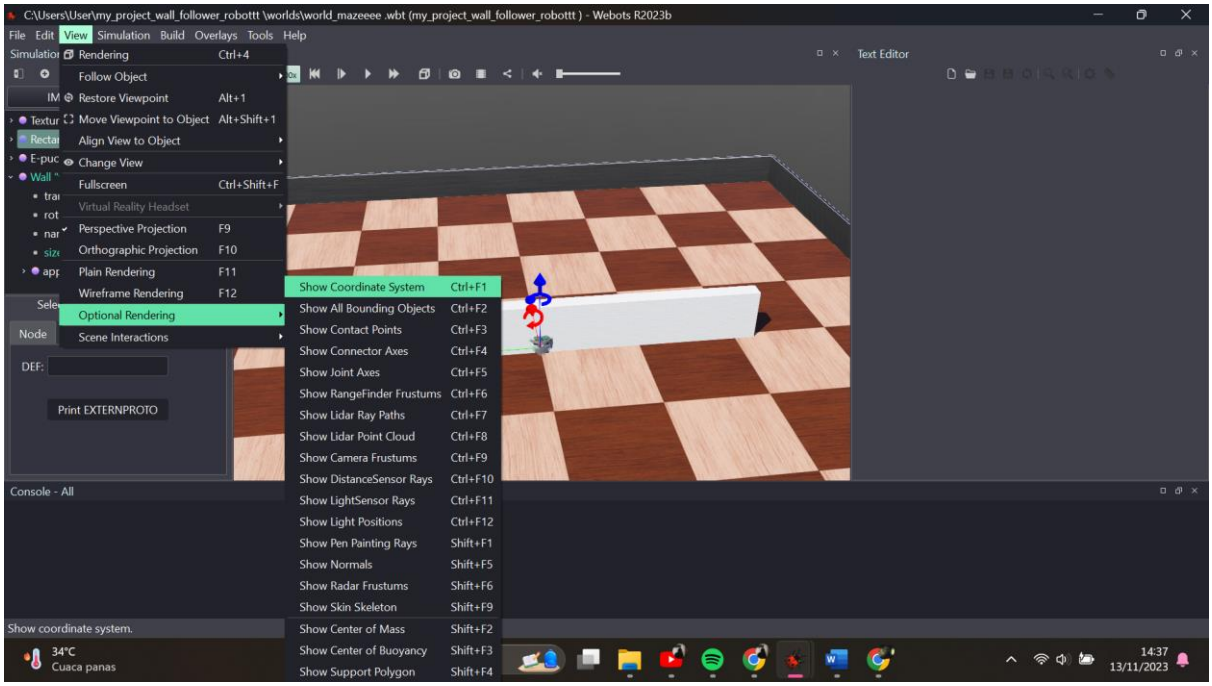
Tembok yang besar sudah ditambahkan ke dalam arena



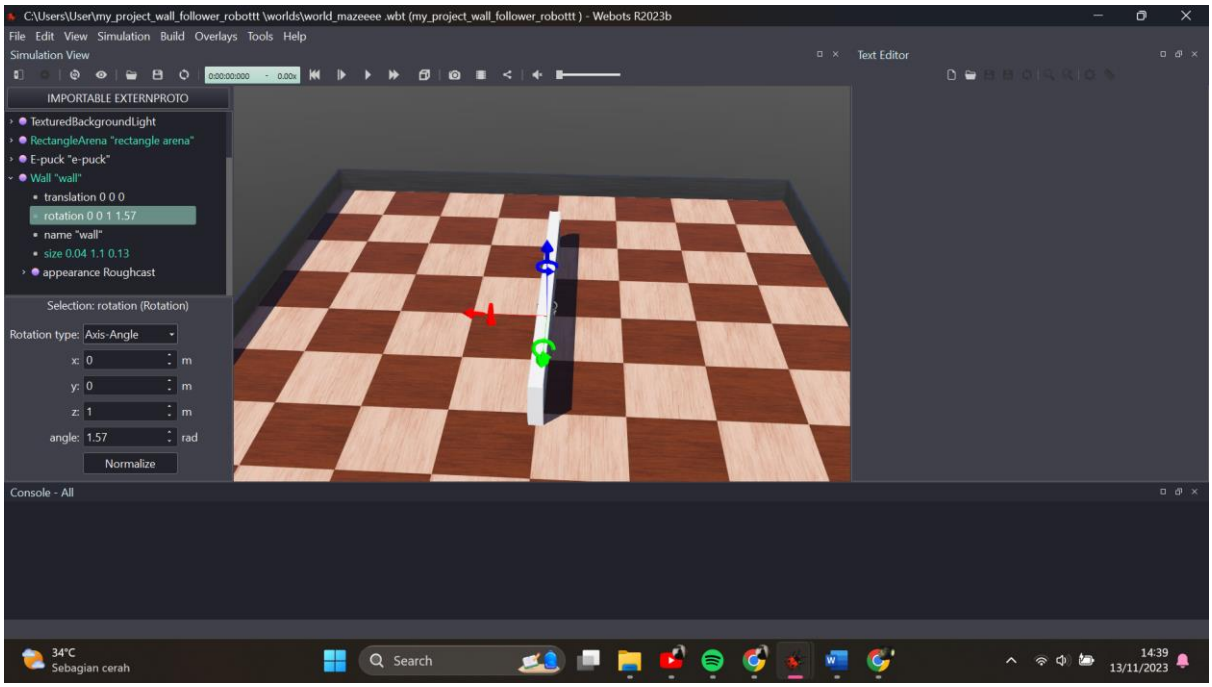
Click on wall & select size > ubah sesuai ukuran yang diperlukan



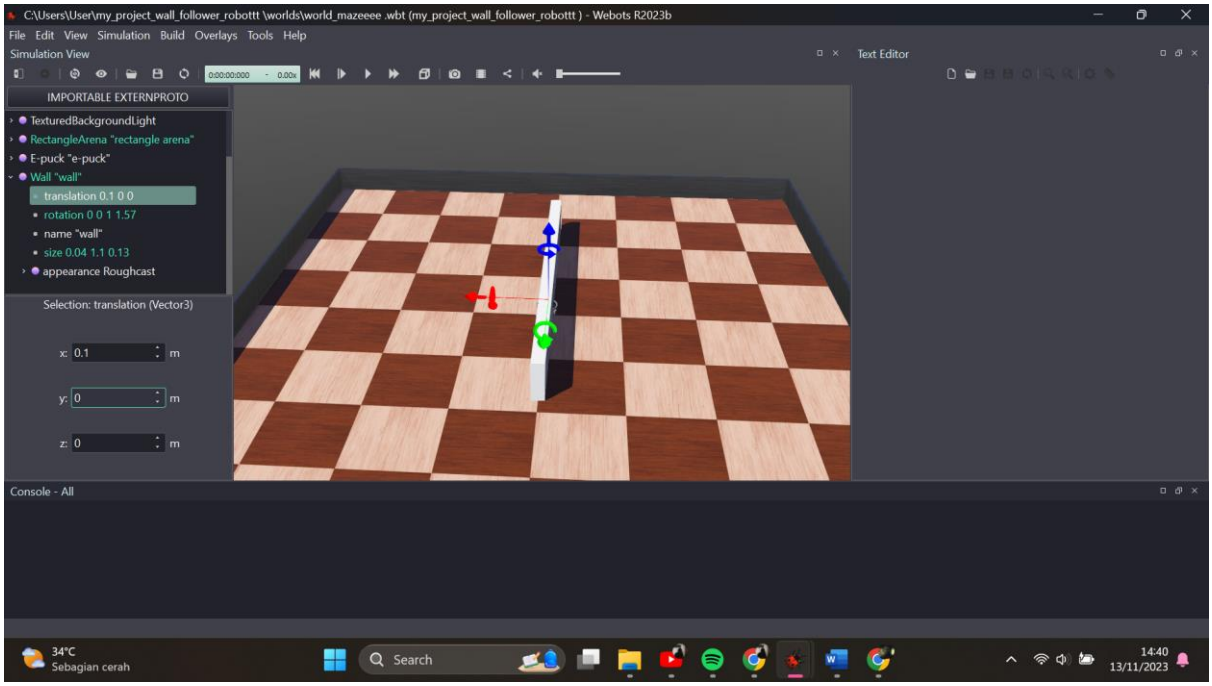
Click menu bar > view > optional rendering > show coordinate system



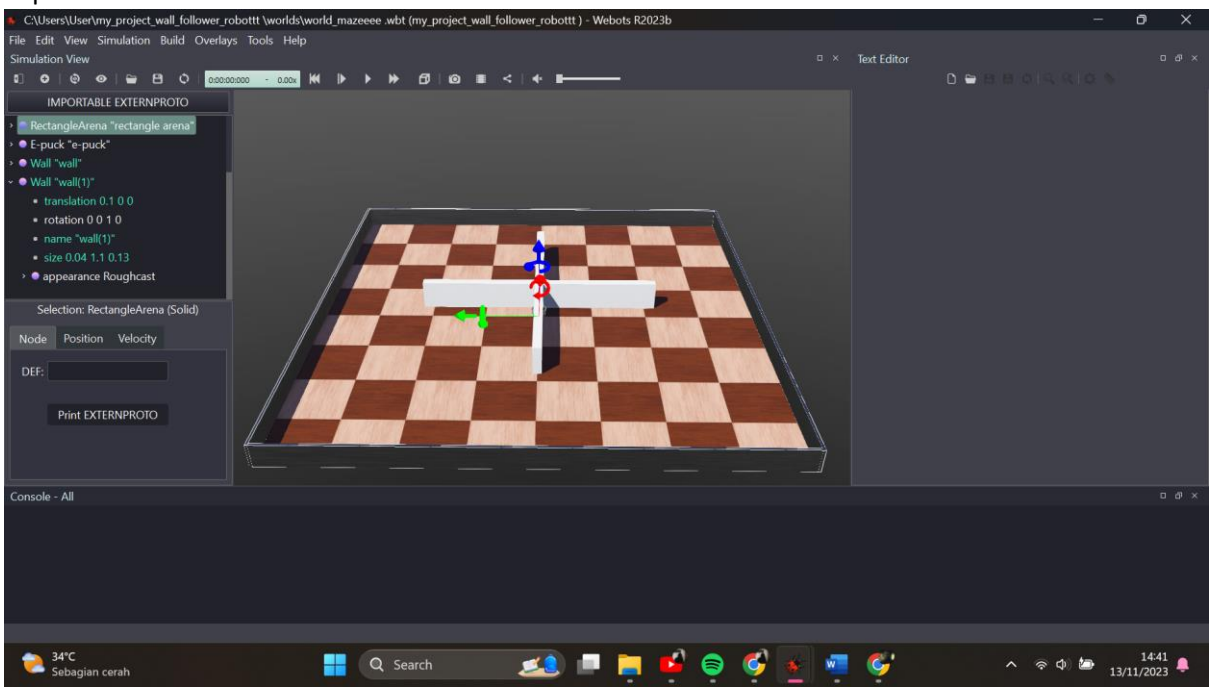
Click rotation > select rotation > atur sesuai keperluan > angle = 1.57 rad



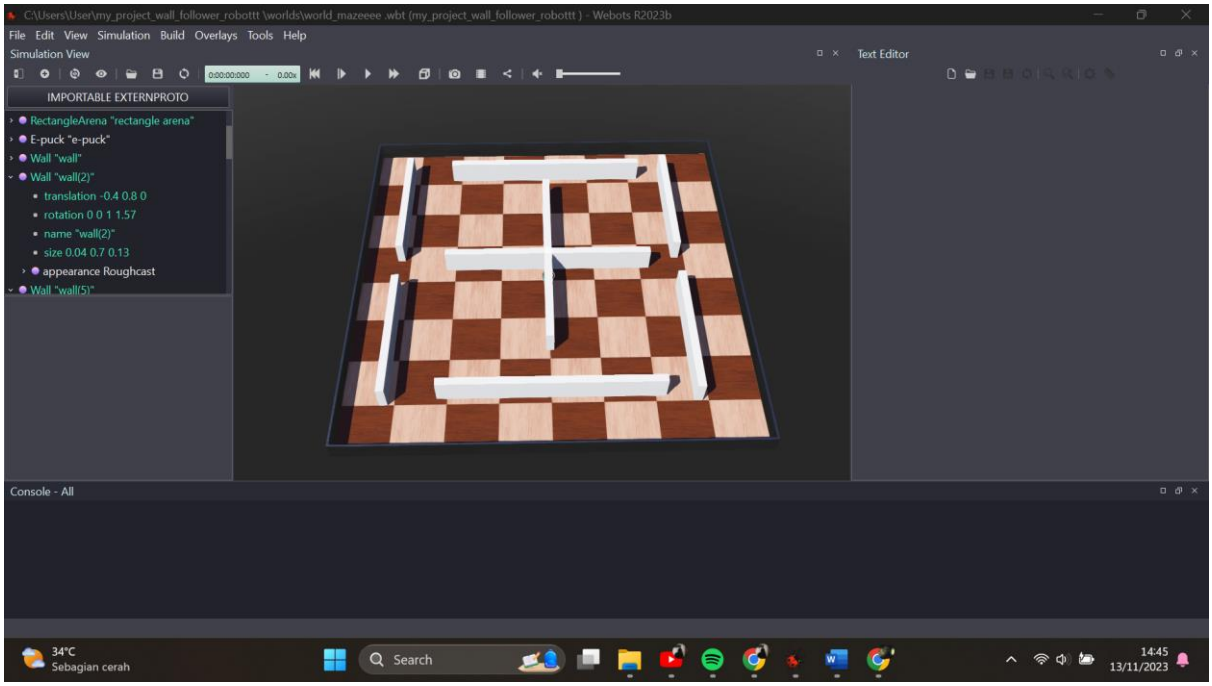
Click on wall > select translation > ubah sesuai keperluan untuk memindahkan wall nya



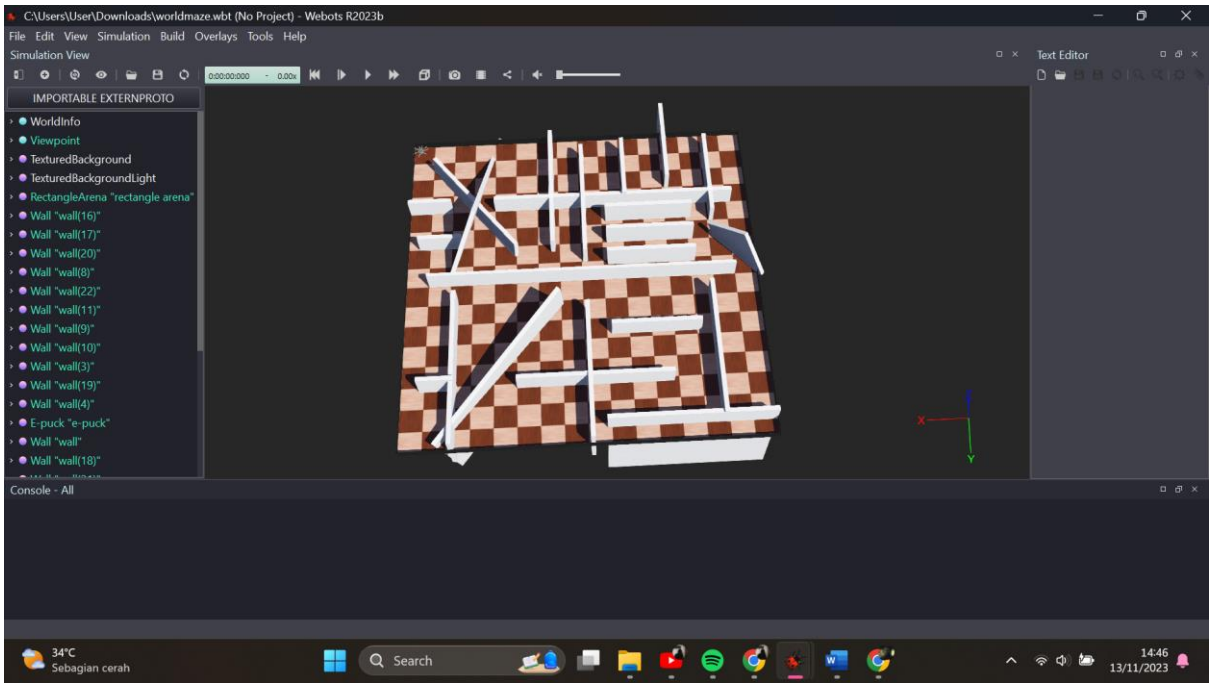
Menambahkan wall yg lain > click on wall + copy ctrl + c > paste ctrl + v > ubah rotation sesuai keperluan



Tambahkan beberapa wall lagi agar menjadi sebuah labirin

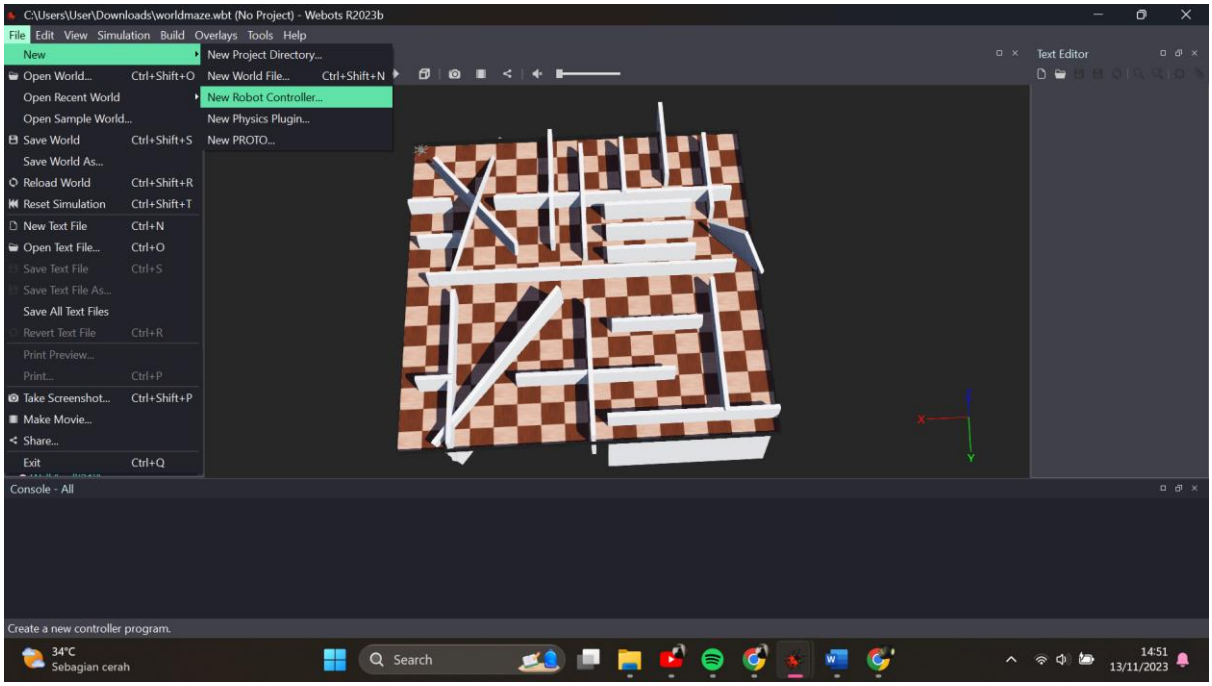


Jika sudah ditambahkan semua wall dan membentuk sebuah labirin hasilnya seperti ini

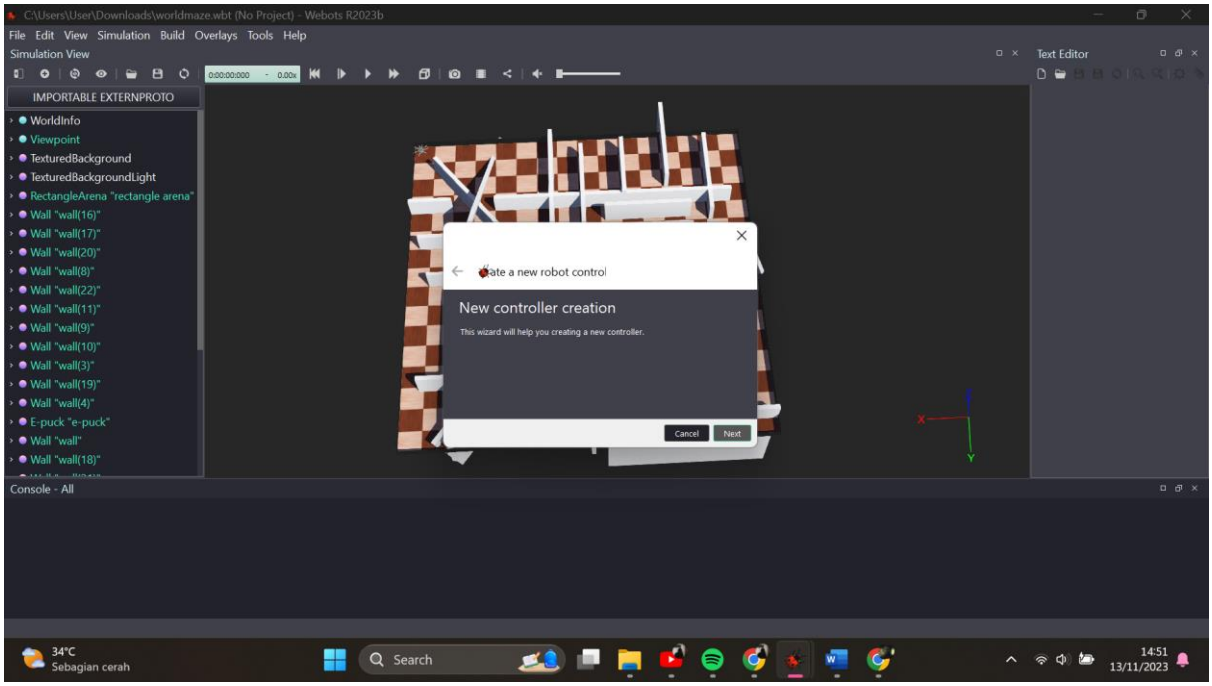


3. Part Three: Create a controller
   Learn to create a controller for your Webots project. Make sure to change the default controller in e-puck to use your controller.

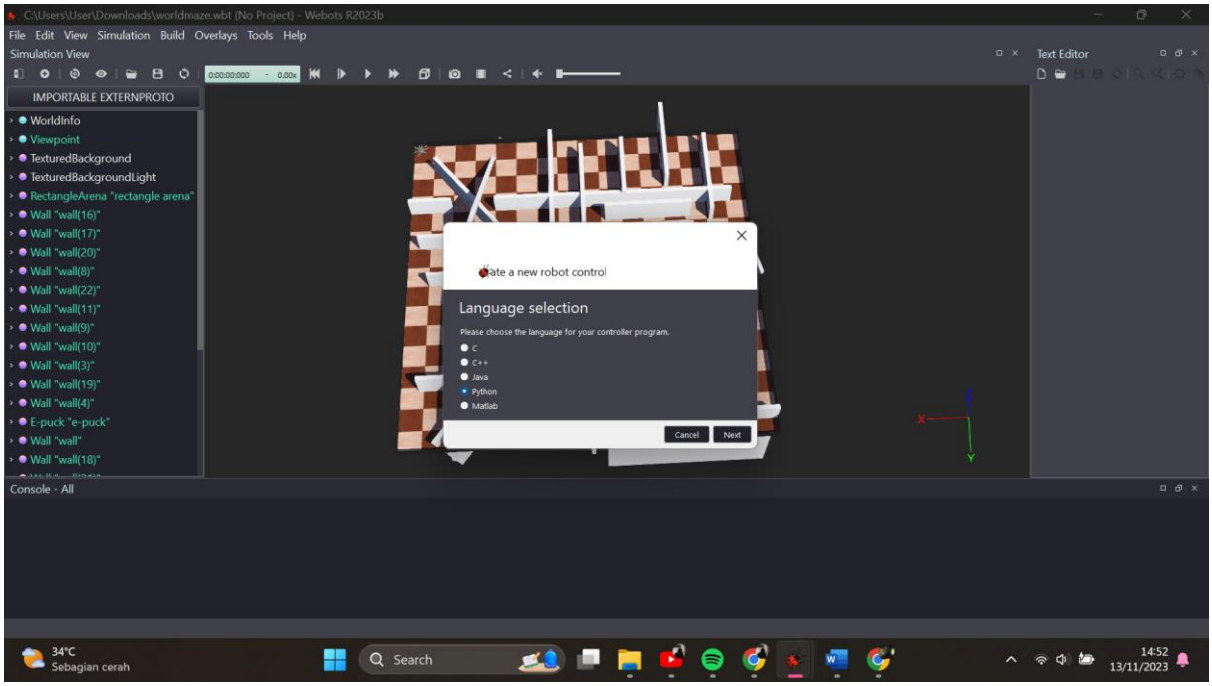
   Menu Bar > File > New > New Robot Controller



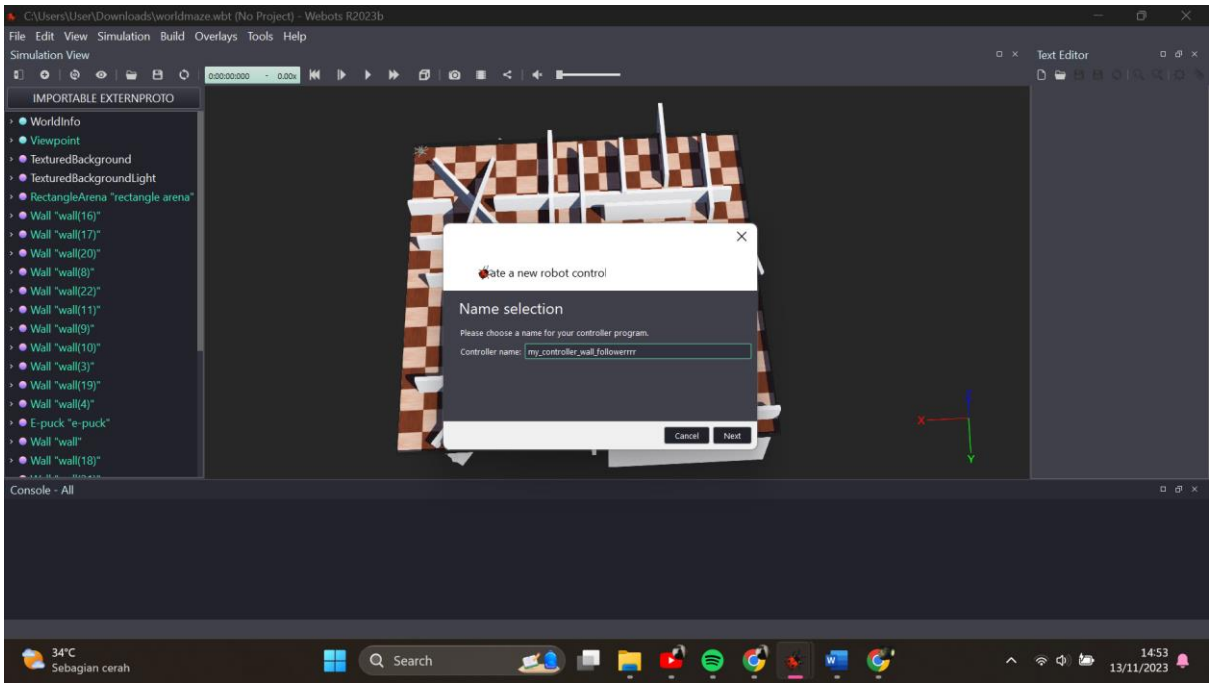
   Click Next



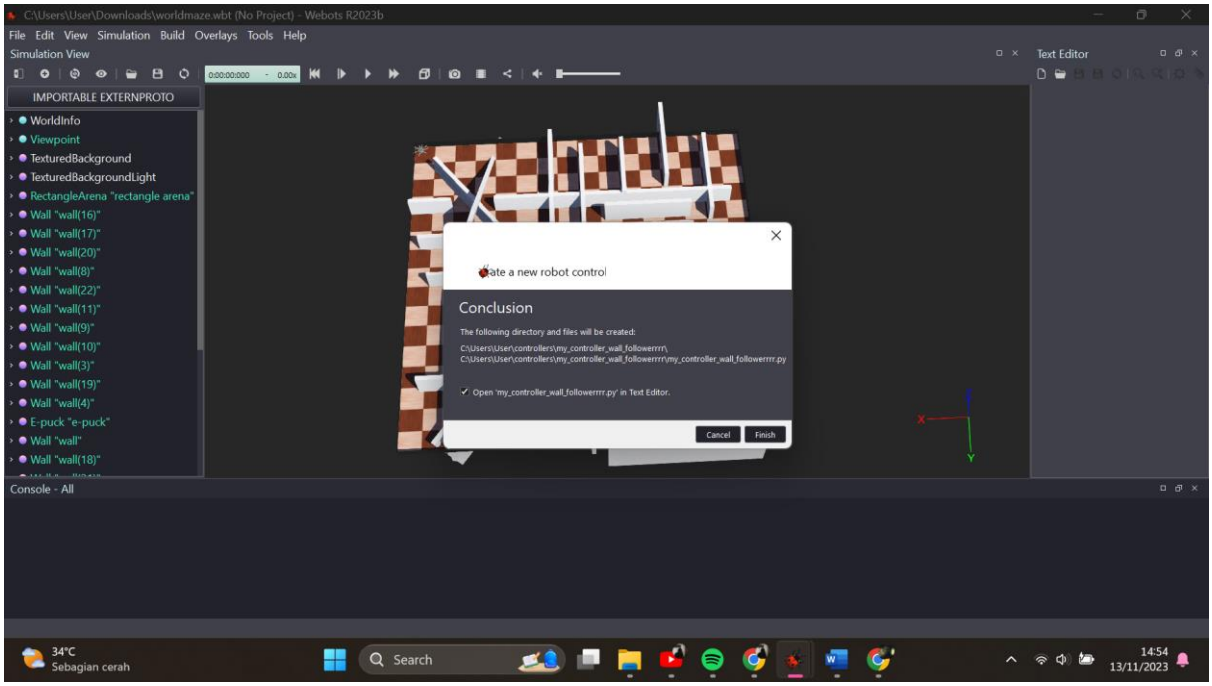
Select Python > Click Next



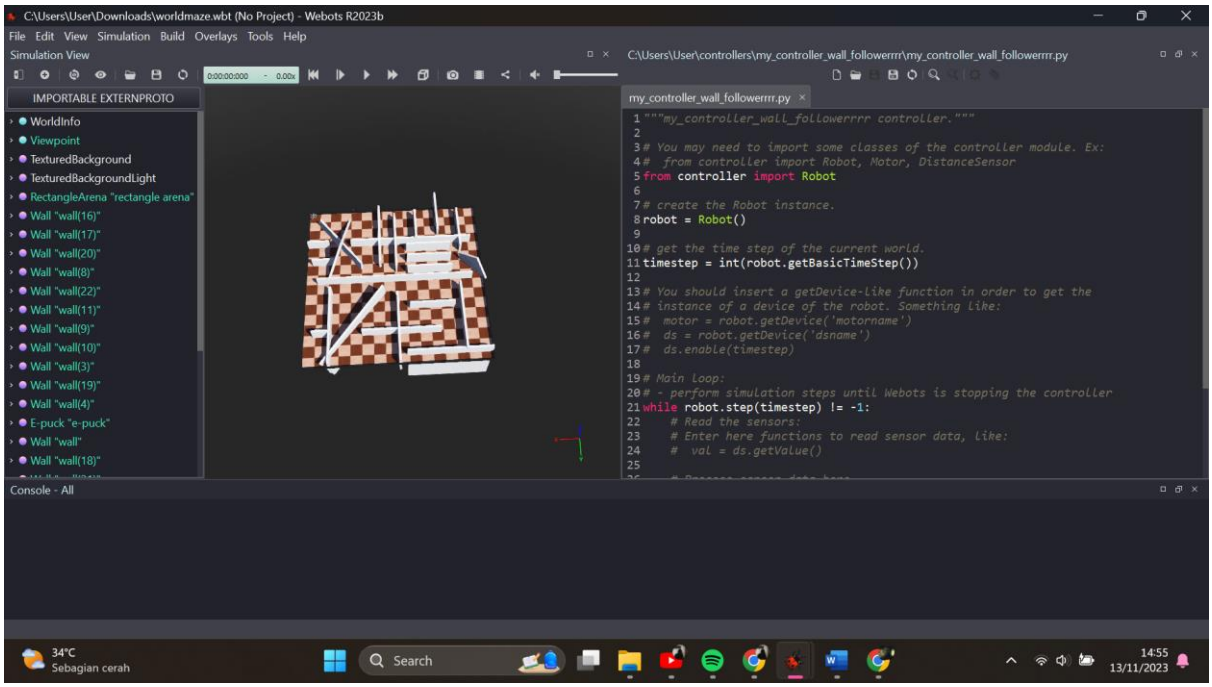
Name your controller > my_controller_wall_followerrrr > click next



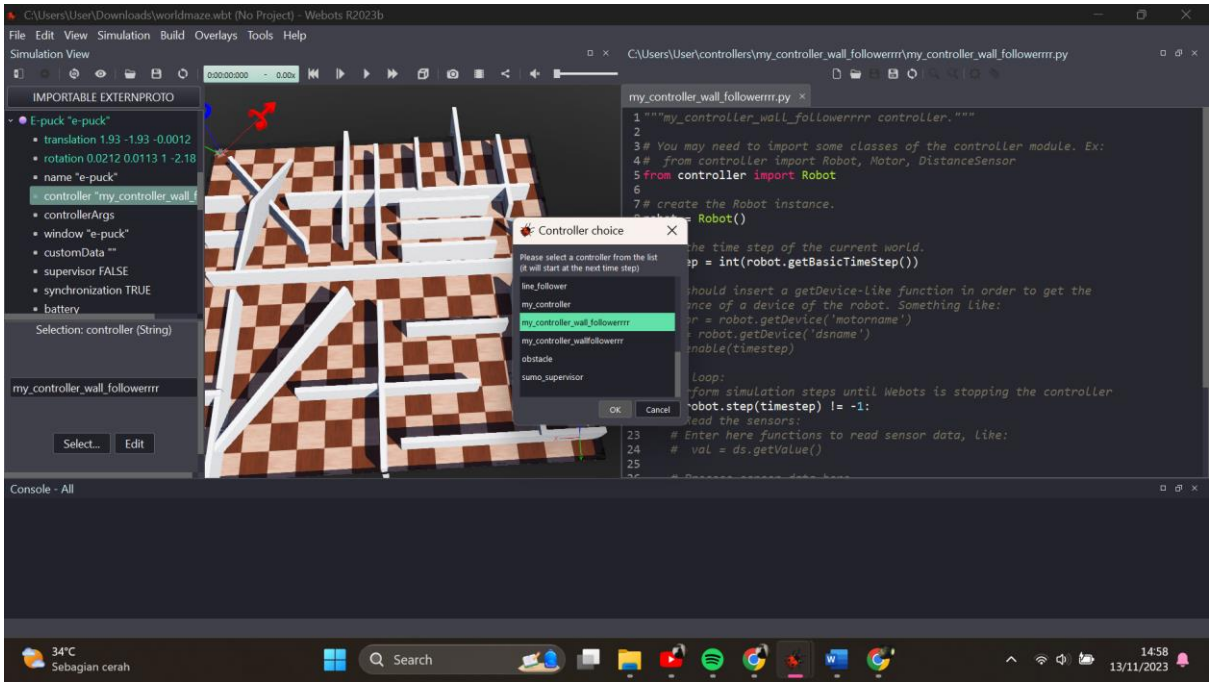
Click finish



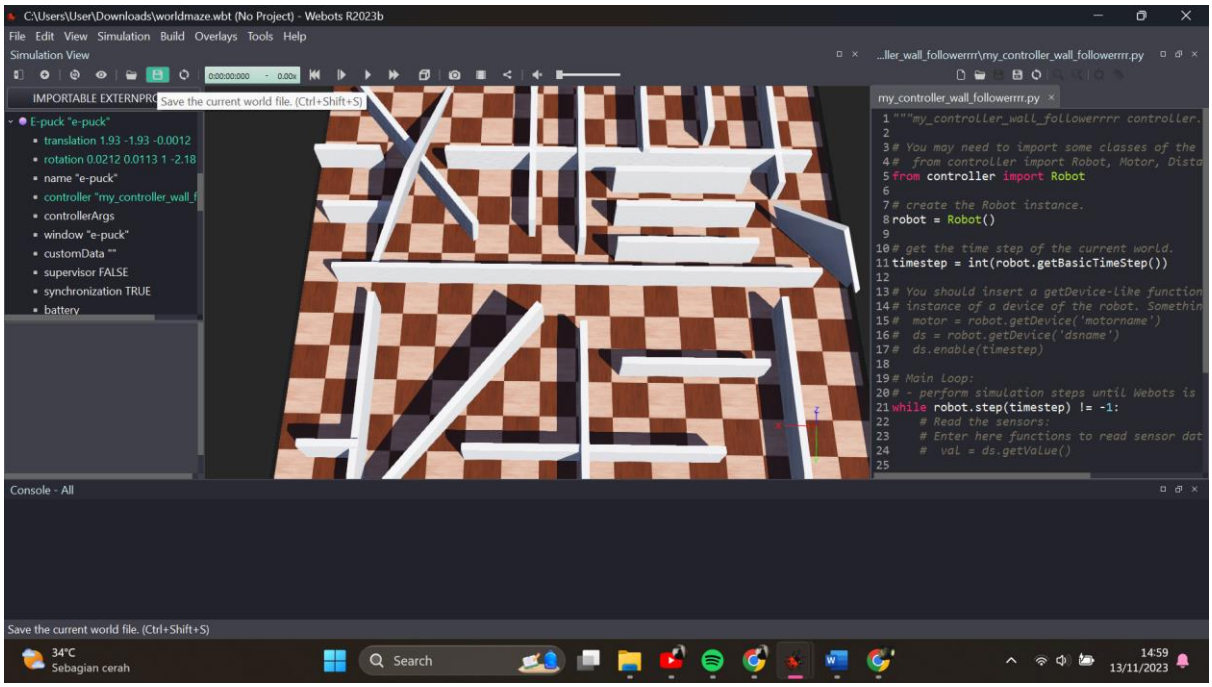
Text file untuk coding bahasa pemrograman python sudah muncul di bagian sebelah kanan webots



Go to webots scene tree > select epuck > click controller > click select > scroll and click my_controller_wall_followerrrrr > click ok



Then save the current file



4. Part Four: Controller code in Python 3
   In this section, learn to write a basic Python code for Webots. Learn to create robot instance, enable motors and distance /proximity sensors in Webots.

   – Mulai dengan membuat fungsi utama dan membuat instance robot

```
# Membuat instance dari robot dan menjalankan fungsi run_robot
if __name__ == "__main__":
    my_robot = Robot()
    run_robot(my_robot)
```

   – Selanjutnya membuat membuat fungsi bernama run robot untuk menulis wall following logic

```
–  # Fungsi untuk menjalankan logika robot mengikuti dinding
–  def run_robot(robot):
–      """Wall following robot"""
```

   – Aktifkan motor dan semua sensor dari robot e-puck

     Enable Motor: robot.getMotor()

     Enable proximity sensor: robot.getDistanceSensor()

```
# Mengaktifkan motor kiri dan kanan
    left_motor = robot.getMotor('left wheel motor')
    right_motor = robot.getMotor('right wheel motor')

    # Mengatur motor agar berputar tanpa batas dan memiliki kecepatan
awal 0
    left_motor.setPosition(float('inf'))
    left_motor.setVelocity(0.0)
    right_motor.setPosition(float('inf'))
    right_motor.setVelocity(0.0)
```

```
 # Mengaktifkan sensor jarak sebanyak 8 buah
    prox_sensors = []
    for ind in range(8):
        sensor_name = 'ps' + str(ind)
        prox_sensors.append(robot.getDistanceSensor(sensor_name))
        prox_sensors[ind].enable(timestep)
```

   – Tambahkan kecepatan maksimal e-puck sesuai dokumentasi epuck di website webots

```
–     # Mendapatkan nilai timestep dari lingkungan simulasi
–      timestep = int(robot.getBasicTimeStep())
–      max_speed = 6.28  # Kecepatan maksimum robot
```

- Bentuk code python yang sudah dibuat sementara ini:

```
# Mengimpor modul 'Robot' dari pustaka 'controller' dalam Webots
from controller import Robot

# Fungsi untuk menjalankan logika robot mengikuti dinding
def run_robot(robot):
    """Wall following robot"""

    # Mendapatkan nilai timestep dari lingkungan simulasi
    timestep = int(robot.getBasicTimeStep())
    max_speed = 6.28  # Kecepatan maksimum robot

    # Mengaktifkan motor kiri dan kanan
    left_motor = robot.getMotor('left wheel motor')
    right_motor = robot.getMotor('right wheel motor')

    # Mengatur motor agar berputar tanpa batas dan memiliki kecepatan awal 0
    left_motor.setPosition(float('inf'))
    left_motor.setVelocity(0.0)
    right_motor.setPosition(float('inf'))
    right_motor.setVelocity(0.0)

    # Mengaktifkan sensor jarak sebanyak 8 buah
    prox_sensors = []
    for ind in range(8):
        sensor_name = 'ps' + str(ind)
        prox_sensors.append(robot.getDistanceSensor(sensor_name))
        prox_sensors[ind].enable(timestep)

# Membuat instance dari robot dan menjalankan fungsi run_robot
if __name__ == "__main__":
    my_robot = Robot()
    run_robot(my_robot)
```

5. Part Five: Wall detection
   Learn to detect walls using proximity sensors (type of distance sensors) in Webots. Thus, also learn to read values from proximity sensors. Learn how to look at distance sensors rays for visualization in Webots.

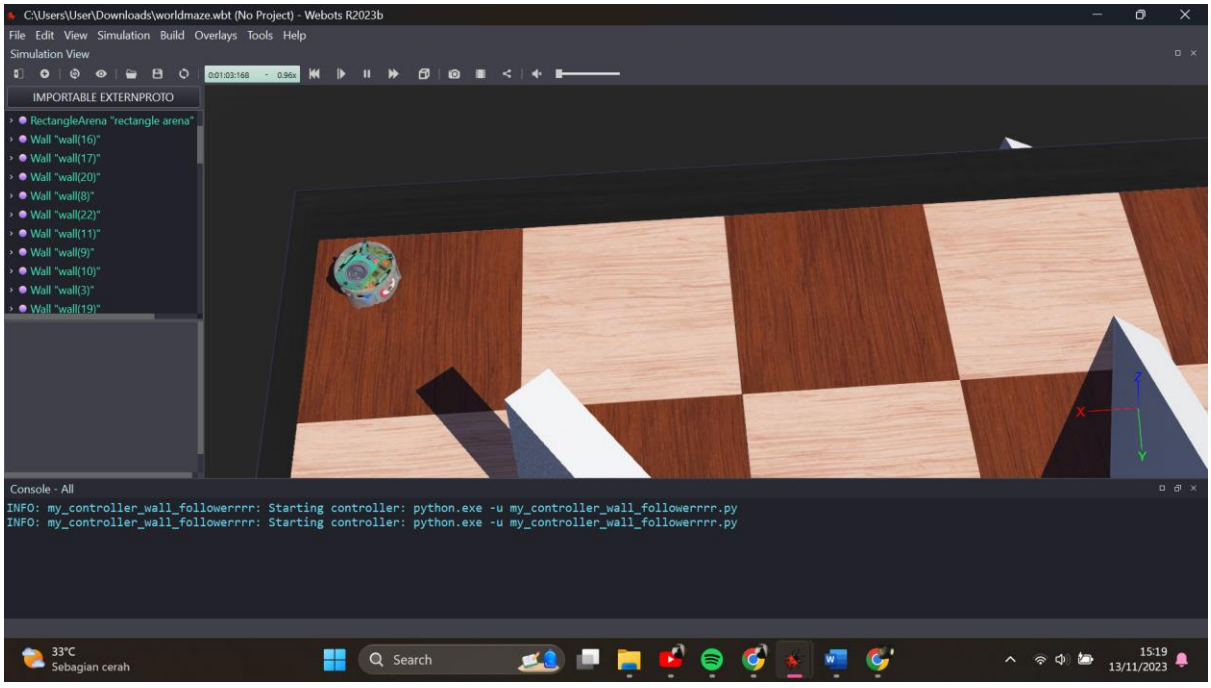
   - Baca dan cetak pembacaan dari sensor jarak yang ada di webots

```
# Membaca nilai sensor jarak dan mencetaknya ke konsol
        for ind in range(8):
            print("ind: {}, val: {}".format(ind,
prox_sensors[ind].getValue()))
```

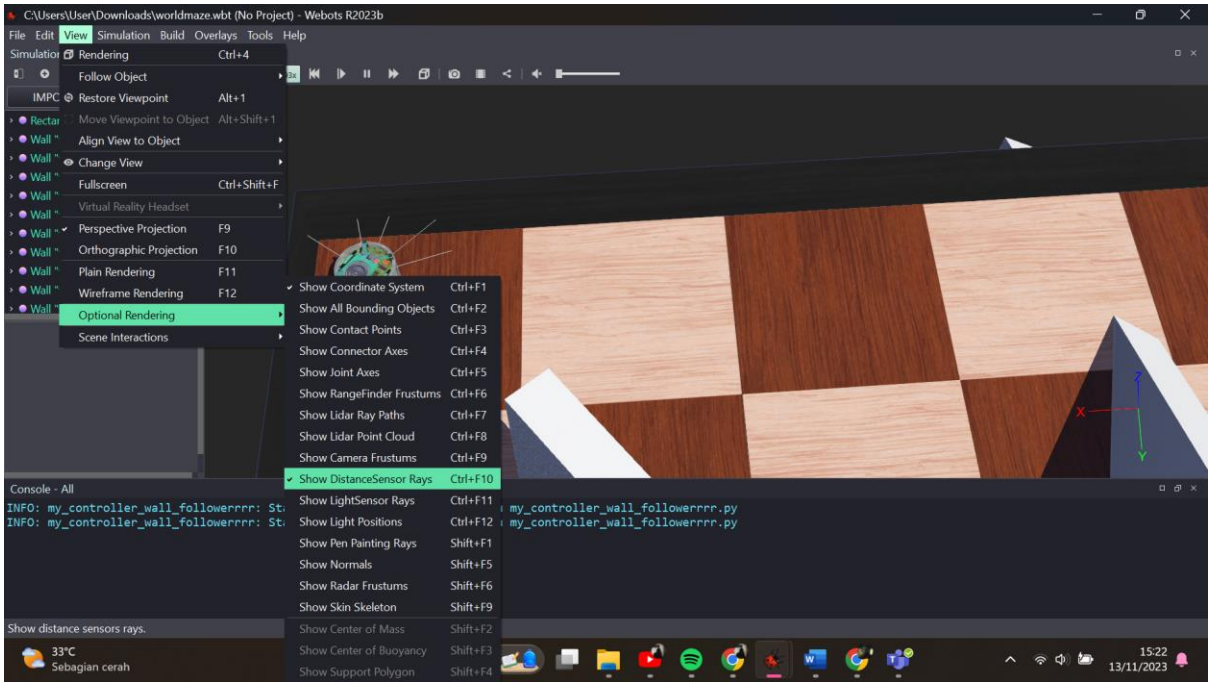
   - Juga pastikan untuk memberikan kecepatan pada kedua motor tersebut sehingga robot dapat bergerak ke depan

```
# Mengatur kecepatan motor kiri dan kanan
      left_motor.setVelocity(left_speed)
      right_motor.setVelocity(right_speed)
```

   - Simpan kode dan klik tombol putar untuk memulai simulasi webots



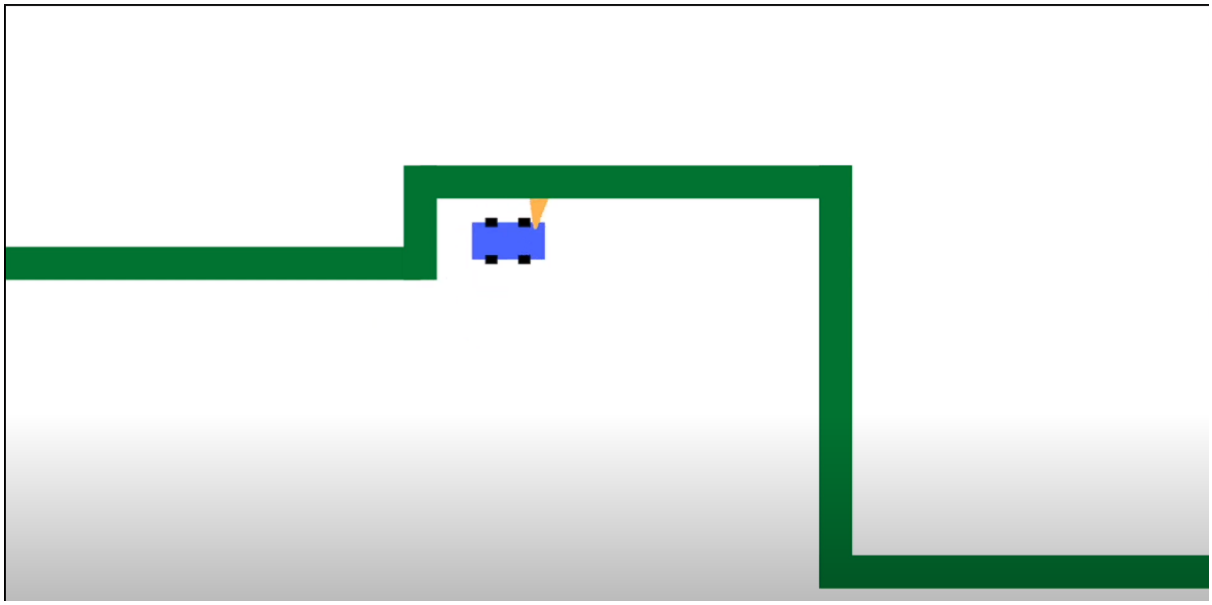
   - Untuk melihat sinar sensor klik menu bar > click view > optional rendering > show distance sensor rays



- Perbarui kode untuk mendeteksi dinding kiri

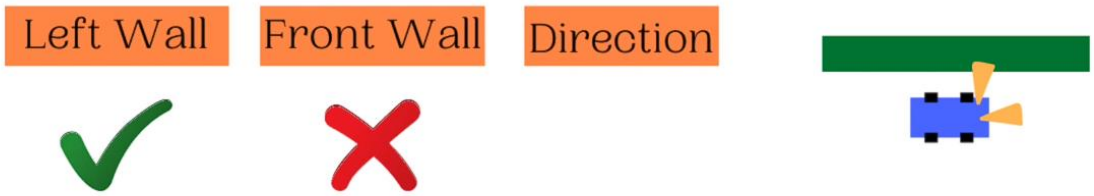
```
# Memproses data sensor
        left_wall = prox_sensors[5].getValue() > 80
        front_wall = prox_sensors[7].getValue() > 80
```

6. Part Six: Wall following logic
   Detailed explanation of how the following logic works. We will go over different scenarios including corner cases and how to solve it.



Seperti namanya, robot akan mengikuti dinding. Kalian dapat pilih sisi yang ingin dipilh, saya memilih sisi kiri. Robot harus bergerak sedemikian rupa untuk mengikuti dinding kiri.

- Ketika dinding berada di sebelah kiri dan tidak ada apa-apa di depan maka robot dapat dengan mudah melaju ke depan



- Jika mendeteksi dinding di depan robot, pada dasarnya di sudut, robot harus mengemudi ke kanan untuk melanjutkan

- Jika ada jalan kiri yang akan datang di depan robot, dinding di sebelah kiri akan menghilang. Jadi jika ada dinding di sebelah kiri robot, robot harus belok ke kiri dan konsep yang sama juga akan bekerja pada putar balik.

- Kasus terakhir tidak ada dinding di sisi kiri dan ada dinding di depan robot, karena kita mengikuti dinding kiri, robot perlu berbelok sedemikian rupa sehingga dinding berada di sisi kiri. Hal ini dapat dilakukan hanya dengan berbelok ke kanan

7. Part Seven: Wall following code in Python 3
   We will code the Webots controller in Python 3 for e-puck to display wall following behavior.

- Membuat dua kecepatan masing-masing untuk motor epuck kiri dan kanan.

```
# Mengatur kecepatan motor kiri dan kanan berdasarkan logika kontrol
        left_speed = max_speed
        right_speed = max_speed
```

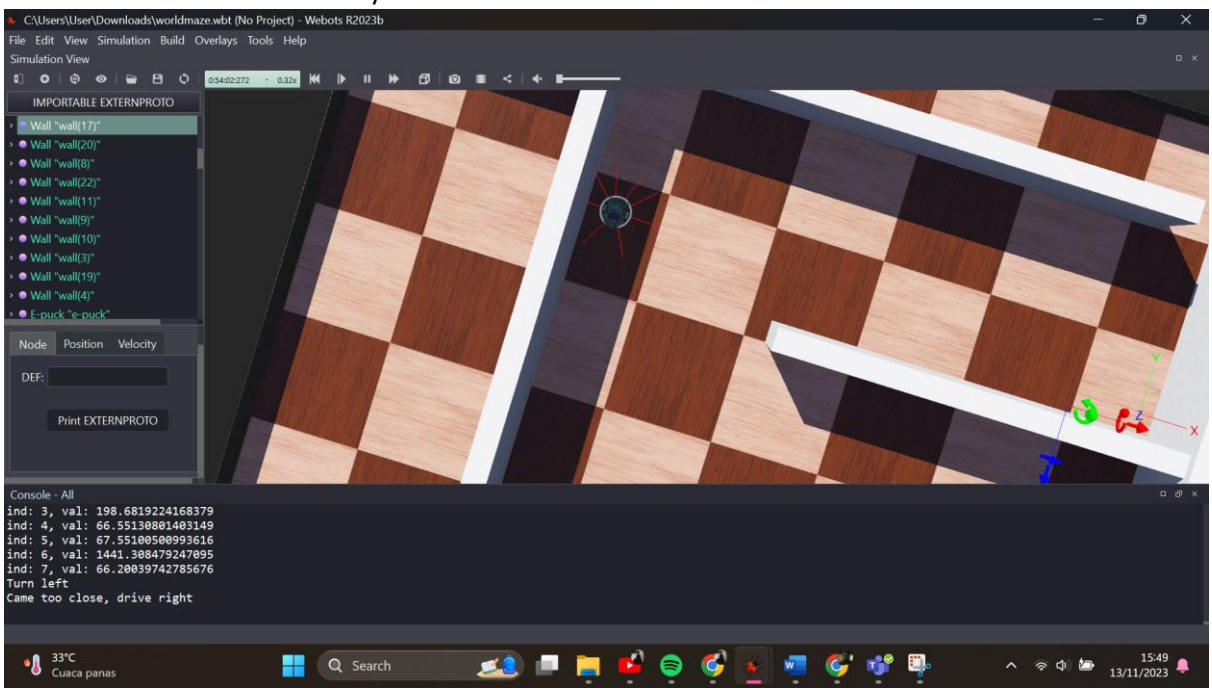
- Kita bisa mulai tindakan umum berbelok kanan di tempat ketika ada dinding di depan robot terlepas dari apakah ada dinding di sisi kiri atau tidak

```
- if front_wall:
-             print("Turn right in place")
-             left_speed = max_speed
-             right_speed = -max_speed
```

- Selanjutnya jika dinding terdeteksi di sebelah kiri kita melaju ke depan dan jika tidak ada dinding yang terdeteksi kita belok ke kiri dan atur max kecepatan 8

```
- else:
-             if left_wall:
-                 print("Drive forward")
-                 left_speed = max_speed
-                 right_speed = max_speed
-             else:
-                 print("Turn left")
-                 left_speed = max_speed / 8
-                 right_speed = max_speed
-
```

- Kemudian coba simulasi nya di webots

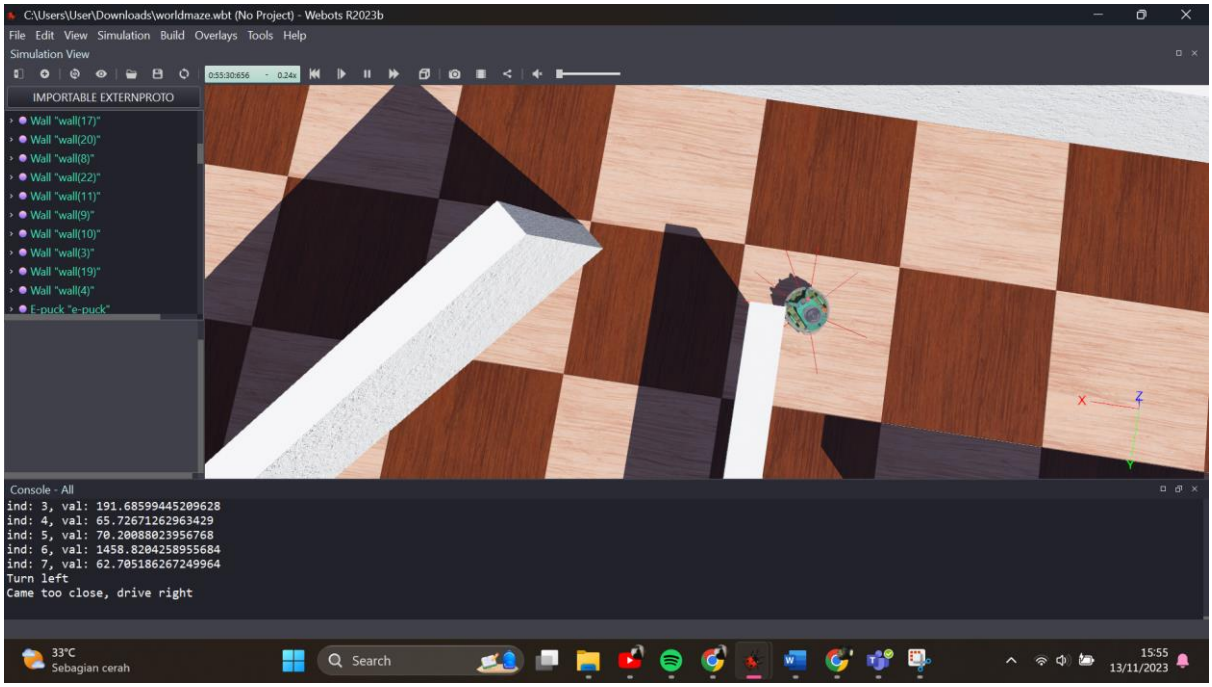


- Tambahkan code nya lagi, jika sudut kiri terdeteksi robot terlalu dekat ke dinding dan harus menjauh darinya dengan berbelok ke kanan

```
        # Memproses data sensor
            left_wall = prox_sensors[5].getValue() > 80
            left_corner = prox_sensors[6].getValue() > 80
            front_wall = prox_sensors[7].getValue() > 80
```

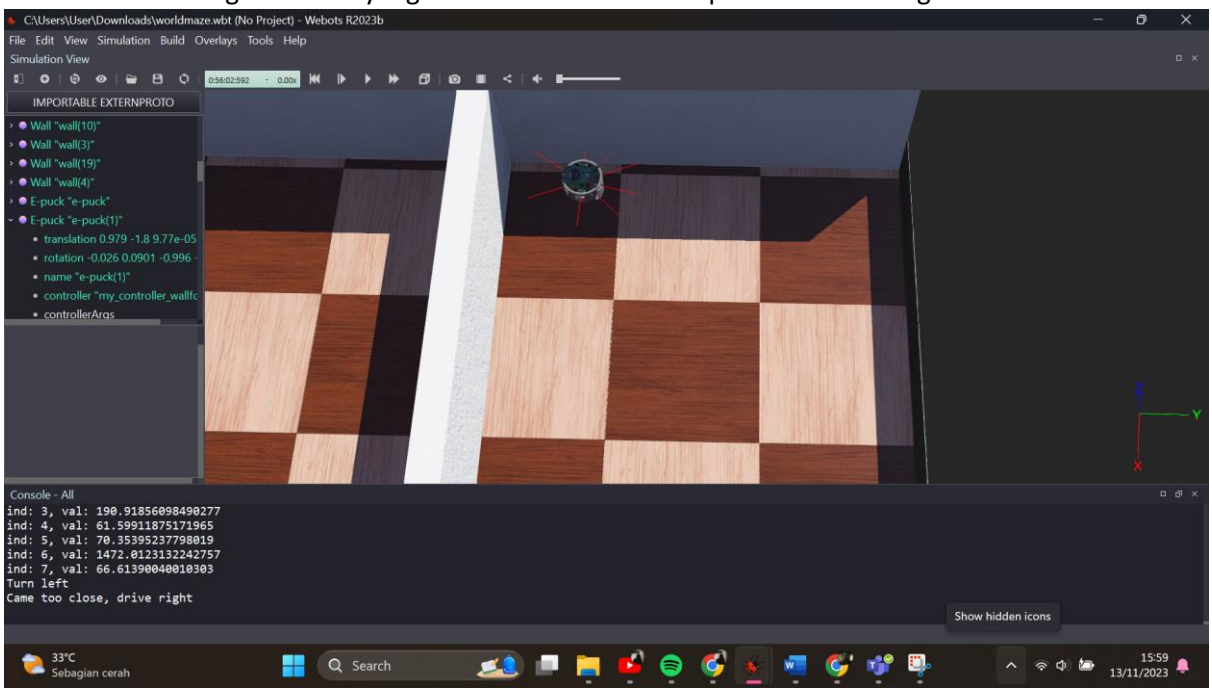
```
    if left_corner:
                print("Came too close, drive right")
                left_speed = max_speed
                right_speed = max_speed / 8
```

- Pastikan untuk simpan file codenya dan jalankan lagi simulasinya

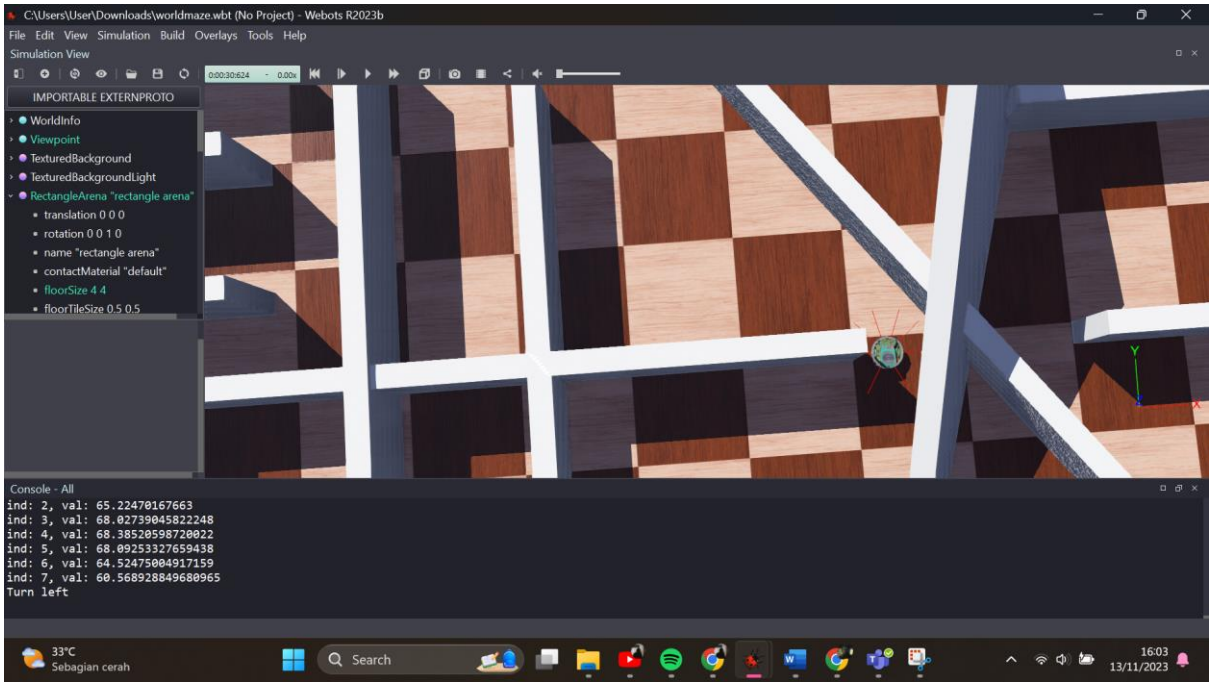


Kalian dapat melihat robot dengan mudah mengikuti dinding dan setelah memutarnya tidak terlalu dekat dengan robot

- Kita dapat mengamati apa yang terjadi jika robot berada di sudut dan robot akan berbelok sesuai dengan arahan yang dilakukan tadi di dalam pembuatan codingan.



- Skenario lain, berikutnya akan ada belokan kiri dan belokan yang sempit, sekali lagi robot terus mengikuti dinding kiri dan berkeliling labirin



- Simulasi webots wall follower robot menggunakan robot e-puck di webots telah selesai tutorialnya.

- Codingan lengkap dari wall follower robot menggunakan Python total 74 baris:

```
"""Daffa Asyqar Ahmad Khalisheka"""
"""1103200034"""
"""UTS Robotika Telkom University"""
"""2023"""

# Mengimpor modul 'Robot' dari pustaka 'controller' dalam Webots
from controller import Robot

# Fungsi untuk menjalankan logika robot mengikuti dinding
def run_robot(robot):
    """Wall following robot"""

    # Mendapatkan nilai timestep dari lingkungan simulasi
    timestep = int(robot.getBasicTimeStep())
    max_speed = 6.28  # Kecepatan maksimum robot

    # Mengaktifkan motor kiri dan kanan
    left_motor = robot.getMotor('left wheel motor')
    right_motor = robot.getMotor('right wheel motor')

    # Mengatur motor agar berputar tanpa batas dan memiliki kecepatan awal 0
    left_motor.setPosition(float('inf'))
    left_motor.setVelocity(0.0)
    right_motor.setPosition(float('inf'))
    right_motor.setVelocity(0.0)

    # Mengaktifkan sensor jarak sebanyak 8 buah
    prox_sensors = []
    for ind in range(8):
        sensor_name = 'ps' + str(ind)
        prox_sensors.append(robot.getDistanceSensor(sensor_name))
        prox_sensors[ind].enable(timestep)

    # Loop utama:
    # - melakukan langkah simulasi hingga Webots menghentikan kontroler
    while robot.step(timestep) != -1:
        # Membaca nilai sensor jarak dan mencetaknya ke konsol
        for ind in range(8):
            print("ind: {}, val: {}".format(ind, prox_sensors[ind].getValue()))

        # Memproses data sensor
        left_wall = prox_sensors[5].getValue() > 80
        left_corner = prox_sensors[6].getValue() > 80
        front_wall = prox_sensors[7].getValue() > 80
```

```
•
•           # Mengatur kecepatan motor kiri dan kanan berdasarkan
    logika kontrol
•           left_speed = max_speed
•           right_speed = max_speed
•
•           if front_wall:
•               print("Turn right in place")
•               left_speed = max_speed
•               right_speed = -max_speed
•           else:
•               if left_wall:
•                   print("Drive forward")
•                   left_speed = max_speed
•                   right_speed = max_speed
•               else:
•                   print("Turn left")
•                   left_speed = max_speed / 8
•                   right_speed = max_speed
•
•               if left_corner:
•                   print("Came too close, drive right")
•                   left_speed = max_speed
•                   right_speed = max_speed / 8
•
•           # Mengatur kecepatan motor kiri dan kanan
•           left_motor.setVelocity(left_speed)
•           right_motor.setVelocity(right_speed)
•
•   # Kode untuk membersihkan setelah kontrol selesai (tidak
    diimplementasikan)
•   # Enter here exit cleanup code.
•
•   # Membuat instance dari robot dan menjalankan fungsi run_robot
•   if __name__ == "__main__":
•       my_robot = Robot()
•       run_robot(my_robot)
•
```

- Berikut penjelasan setiap baris baris codenya:

1. Import Library:

```
# Mengimpor modul 'Robot' dari pustaka 'controller' dalam Webots
from controller import Robot
```

Mendeklarasikan penggunaan kelas Robot dari modul controller yang disediakan oleh Webots.

2. Fungsi run_robot:

```
# Fungsi untuk menjalankan logika robot mengikuti dinding
def run_robot(robot):
    """Wall following robot"""
```

Fungsi utama yang menerima objek Robot sebagai parameter.

3. Pengaturan Awal:

```
# Mendapatkan nilai timestep dari lingkungan simulasi
    timestep = int(robot.getBasicTimeStep())
    max_speed = 6.28  # Kecepatan maksimum robot
```

Mengambil waktu langkah dan mengatur kecepatan maksimum robot.

4. Inisialisasi Motor:

```
# Mengaktifkan motor kiri dan kanan
    left_motor = robot.getMotor('left wheel motor')
    right_motor = robot.getMotor('right wheel motor')
```

Mendapatkan referensi ke motor kiri dan kanan dari robot.

5. Konfigurasi Motor:

```
# Mengatur motor agar berputar tanpa batas dan memiliki kecepatan awal 0
    left_motor.setPosition(float('inf'))
    left_motor.setVelocity(0.0)
    right_motor.setPosition(float('inf'))
      right_motor.setVelocity(0.0)
```

Mengaktifkan motor untuk kontrol kecepatan dan posisi tak terbatas.

6. Inisialisasi Sensor Proximity:

```
# Mengaktifkan sensor jarak sebanyak 8 buah
    prox_sensors = []
    for ind in range(8):
        sensor_name = 'ps' + str(ind)
        prox_sensors.append(robot.getDistanceSensor(sensor_name))
        prox_sensors[ind].enable(timestep)
```

Menginisialisasi sensor proximity sebanyak delapan, mengatur nama dan mengaktifkannya dengan waktu langkah yang telah diambil sebelumnya.

7. Loop Utama:

```
8.  # Loop utama:
9.      # - melakukan langkah simulasi hingga Webots menghentikan
   kontroler
10.       while robot.step(timestep) != -1:
```

Loop utama yang akan dijalankan selama simulasi berlangsung.

8. Baca Sensor:

```
9. # Membaca nilai sensor jarak dan mencetaknya ke konsol
10.           for ind in range(8):
11.               print("ind: {}, val: {}".format(ind,
   prox_sensors[ind].getValue()))
```

Membaca nilai sensor proximity dan mencetaknya. Nilai-nilai ini nantinya digunakan untuk pengambilan keputusan.

9. Proses Data Sensor:

```
10.          # Memproses data sensor
11.           left_wall = prox_sensors[5].getValue() > 80
12.           left_corner = prox_sensors[6].getValue() > 80
13.           front_wall = prox_sensors[7].getValue() > 80
```

Menganalisis data sensor untuk menentukan apakah ada dinding di sebelah kiri, sudut kiri, atau di depan robot.

10. Inisialisasi Kecepatan Awal:

```
11.      # Mengatur kecepatan motor kiri dan kanan berdasarkan
   logika kontrol
12.           left_speed = max_speed
13.           right_speed = max_speed
14.
```

Mengatur kecepatan awal untuk motor kiri dan kanan dengan nilai maksimum (max_speed).

11. Pengecekan Dinding Depan:

```
12.
13.           if front_wall:
14.               print("Turn right in place")
15.               left_speed = max_speed
16.               right_speed = -max_speed
```

Jika sensor di depan mendeteksi adanya dinding (front_wall), robot akan mencetak pesan dan memutar dirinya di tempat dengan mengatur kecepatan motor kiri ke nilai maksimum dan kecepatan motor kanan ke nilai maksimum yang dibalik (negatif).

12. Logika Ketika Tidak Ada Dinding Depan:

```
13.      else:
```

Jika tidak ada dinding di depan, maka akan dilakukan pengecekan lebih lanjut.

13. Logika Ketika Terdeteksi Dinding Kiri (left_wall):

```
    if left_wall:
                    print("Drive forward")
                    left_speed = max_speed
                    right_speed = max_speed
```

Jika terdeteksi dinding di sebelah kiri, maka robot akan maju lurus dengan mengatur kecepatan motor kiri dan kanan ke maksimum.

14. Logika Ketika Tidak Ada Dinding Kiri:

```
    else:
                    print("Turn left")
                    left_speed = max_speed / 8
                    right_speed = max_speed
```

Pertama, pesan "Turn left" dicetak sebagai indikasi bahwa robot akan berbelok ke kiri. Selanjutnya, kecepatan motor diatur untuk menciptakan gerakan berbelok ke kiri:

left_speed = max_speed / 8: Kecepatan motor kiri diatur menjadi sekitar satu per delapan (1/8) dari kecepatan maksimum. Hal ini menghasilkan gerakan berbelok yang lebih lambat ke kiri.

right_speed = max_speed: Kecepatan motor kanan diatur ke kecepatan maksimum, sehingga roda kanan tetap bergerak maju dengan kecepatan penuh.

15. Logika Ketika Terdeteksi Sudut Kiri (left_corner):

```
    if left_corner:
                    print("Came too close, drive right")
                    left_speed = max_speed
                    right_speed = max_speed / 8
```

Jika terdeteksi sudut di sebelah kiri (mungkin terlalu dekat), maka robot akan bergerak ke kanan dengan mengurangi kecepatan roda kanan.

18. Set Kecepatan Motor Akhir:

```
    # Mengatur kecepatan motor kiri dan kanan
            left_motor.setVelocity(left_speed)
            right_motor.setVelocity(right_speed)
```

Mengatur kecepatan motor kiri dan kanan berdasarkan logika kontrol yang telah diatur sebelumnya. Setelah logika dijalankan, kecepatan motor yang dihasilkan akan menggerakkan robot sesuai dengan kondisi yang ditemui.

19. Main Program:

```
  # Membuat instance dari robot dan menjalankan fungsi
  run_robot
  if __name__ == "__main__":
      my_robot = Robot()
      run_robot(my_robot)
```

Keseluruhan, kode ini mengimplementasikan kontroler untuk robot yang dapat mengikuti dinding dan mengatasi sudut-sudut di lingkungan simulasi Webots. Logika gerak didasarkan pada

pembacaan sensor proximity dan menentukan kecepatan motor kiri dan kanan sesuai dengan kondisi sensor.

Link Webots e-puck: https://cyberbotics.com/doc/guide/epuck?version=R2021b#e-puck-model
Credit to Youtube Kajal Gada https://www.youtube.com/@KajalGada
Link Youtube tutorial https://youtu.be/tHENC-HEIW8?si=gnH-iyjb50ZBHdf-