PEMANFAATAN COPPELIA SIMULATOR UNTUK SIMULASI ROBOT PIONEER P3DX – STUDI KASUS ARENA KRC POLNES 2024

Teknologi Rekayasa Komputer B 2022

Teknologi Informasi, Politeknik Negeri Samarinda
Jalan Cipto Mangunkusumo, Kota Samarinda, Kalimantan Timur, 75131

Abstrak

Penggunaan perangkat lunak simulasi semakin penting dalam bidang robotika, memungkinkan pengujian dan pengembangan sistem robotik secara efisien dalam lingkungan virtual. Penelitian ini berfokus pada pemanfaatan simulator *CoppeliaSim* untuk memodelkan dan mensimulasikan kinerja robot *Pioneer P3DX*. Studi ini mengadopsi studi kasus spesifik pada arena KRC POLNES 2024 untuk mengevaluasi kemampuan navigasi dan pelaksanaan tugas robot tersebut. Melalui berbagai skenario simulasi, penelitian ini menganalisis respons robot terhadap berbagai kondisi lingkungan, konfigurasi rintangan, dan tugas yang telah ditentukan. Hasilnya menunjukkan efektivitas *CoppeliaSim* dalam menyediakan platform simulasi yang realistis dan fleksibel, serta memberikan wawasan berharga tentang kinerja dan potensi perbaikan *Pioneer P3DX*. Penelitian ini menyoroti peran penting simulasi dalam mengoptimalkan desain dan operasi robot, khususnya di lingkungan kompetitif dan dinamis seperti KRC POLNES 2024.

Kata Kunci: *CoppeliaSim, Pioneer P3DX,* simulasi robotik, KRC POLNES 2024, navigasi, penghindaran rintangan

Abstract

The use of simulation software has become increasingly essential in the field of robotics, enabling efficient testing and development of robotic systems in a virtual environment. This study focuses on utilizing the CoppeliaSim simulator to model and simulate the performance of the Pioneer P3DX robot. The research adopts the specific case study of the KRC POLNES 2024 arena to evaluate the robot's navigation and task execution capabilities. Through various simulation scenarios, the study analyzes the robot’s response to different environmental conditions, obstacle configurations, and predefined tasks. The results demonstrate the effectiveness of CoppeliaSim in providing a realistic and flexible platform for robotic simulations, offering valuable insights into the Pioneer P3DX's performance and potential improvements. This research highlights the significant role of simulation in optimizing robotic design and operations, particularly in competitive and dynamic environments such as KRC POLNES 2024.

Keywords*: CoppeliaSim, Pioneer P3DX, robotic simulation, KRC POLNES 2024, navigation, obstacle avoidance*

A. Pendahuluan

Latar Belakang

Perkembangan teknologi robotika di Indonesia terus mengalami kemajuan pesat, khususnya dalam konteks kompetisi robotika tingkat nasional. Kontes Robot Cerdas (KRC) yang diselenggarakan oleh Politeknik Negeri Samarinda (POLNES) pada tahun 2024 menjadi salah satu ajang prestisius yang membutuhkan persiapan matang dari para peserta. Dalam mempersiapkan robot untuk kompetisi, tim pengembang menghadapi tantangan berupa keterbatasan waktu dan sumber daya dalam melakukan pengujian robot secara fisik.

Penggunaan simulator robotika, khususnya CoppeliaSim (sebelumnya dikenal sebagai V-REP), menawarkan solusi efektif untuk mengatasi kendala tersebut. CoppeliaSim merupakan simulator robot yang memungkinkan pengembang untuk merancang, mensimulasikan, dan menguji algoritma navigasi robot dalam lingkungan virtual yang dapat disesuaikan dengan kondisi arena sesungguhnya. Pioneer P3DX, sebagai platform robot yang umum digunakan dalam kompetisi robotika, dapat disimulasikan secara akurat dalam CoppeliaSim, memberikan kesempatan bagi tim pengembang untuk melakukan iterasi pengembangan algoritma tanpa risiko kerusakan hardware dan dengan biaya yang minimal.

Rumusan Masalah

1. Bagaimana mengimplementasikan simulasi robot Pioneer P3DX dalam lingkungan CoppeliaSim yang merepresentasikan arena KRC POLNES 2024?
2. Seberapa akurat performa algoritma navigasi robot dalam simulasi dibandingkan dengan implementasi pada robot fisik?
3. Apa saja parameter-parameter kritis yang perlu dipertimbangkan dalam mentransfer hasil simulasi ke robot nyata?

Tujuan Penelitian

1. Mengembangkan model simulasi arena KRC POLNES 2024 dalam lingkungan CoppeliaSim yang akurat dan dapat digunakan untuk pengujian algoritma navigasi robot Pioneer P3DX.
2. Menganalisis tingkat keakuratan simulasi dibandingkan dengan pengujian pada robot fisik.
3. Mengidentifikasi dan mendokumentasikan parameter-parameter penting yang mempengaruhi keberhasilan transfer algoritma dari simulasi ke implementasi nyata.

Manfaat Penelitian

1. Membantu tim pengembang robot dalam mempersiapkan kompetisi dengan lebih efisien
2. Mempercepat proses iterasi pengembangan algoritma navigasi robot
3. Mengurangi biaya dan risiko kerusakan hardware selama fase pengembangan
4. Mendukung pengujian dan evaluasi algoritma dalam berbagai skenario tanpa memerlukan pengaturan fisik yang kompleks.
5. Meningkatkan kualitas algoritma navigasi melalui simulasi yang lebih aman dan terkendali.
6. Mempermudah analisis performa robot secara terukur.

B. Metodologi Penelitan

Tinjauan Pustaka Sebelumnya

- Rohmer, E., Singh, S. P., & Freese, M. (2014) dalam penelitian berjudul "V-REP: A versatile and scalable robot simulation framework" yang dipublikasikan di IEEE/RSJ International Conference on Intelligent Robots and Systems, menjelaskan kemampuan V-REP (sekarang CoppeliaSim) dalam mensimulasikan berbagai jenis robot termasuk robot mobile seperti Pioneer P3DX. Penelitian ini menjadi dasar penggunaan CoppeliaSim sebagai platform simulasi yang reliabel.

- Araújo, A., Portugal, D., Couceiro, M. S., & Rocha, R. P. (2015) melalui penelitian "Integrating Arduino-based educational mobile robots in ROS" yang dipublikasikan di Journal of Intelligent & Robotic Systems, mendemonstrasikan penggunaan simulasi robot mobile untuk tujuan edukasi dan kompetisi, yang menunjukkan efektivitas pendekatan simulasi sebelum implementasi pada robot fisik.

- Baillie, P., & Park, C. H. (2019) dalam "Teaching Robotics With Robot Simulation Programs" yang dipublikasikan di International Conference on Robotics in Education (RiE), mengulas pentingnya penggunaan simulator robot dalam proses pembelajaran dan persiapan kompetisi robotika. Penelitian ini menunjukkan bahwa penggunaan simulator dapat meningkatkan efektivitas pembelajaran dan pengembangan algoritma robot.

Dasar Teori

1. Coppelia Simulator

   CoppeliaSim merupakan simulator robot yang dikembangkan oleh Coppelia Robotics sebagai platform pengembangan dan pengujian sistem robotika. Menurut Rohmer et al. (2014), CoppeliaSim menyediakan lingkungan pengembangan terintegrasi yang memungkinkan pengguna untuk membuat, mensimulasikan, dan menguji sistem robot kompleks dalam lingkungan virtual yang aman dan terkontrol. Simulator ini menggunakan physics engine yang memungkinkan simulasi dinamika robot secara akurat, serta mendukung berbagai bahasa pemrograman untuk pengembangan algoritma kontrol robot.

Gambar 1. 1 Logo Coppelia Simulator

Sumber :
https://www.coppeliarobotics.com/CoppeliaSim.png

Freese et al. (2010) dalam penelitiannya mendemonstrasikan bagaimana CoppeliaSim dapat digunakan untuk mensimulasikan berbagai aspek robotika, mulai dari kinematika dasar hingga sistem kontrol kompleks. Kemampuan simulator ini dalam merepresentasikan kondisi dunia nyata menjadikannya alat yang sangat berharga dalam pengembangan dan pengujian algoritma robotika sebelum implementasi pada platform hardware sesungguhnya.

2. Robot Pioneer P3DX

   Pioneer P3DX adalah robot mobile yang telah banyak digunakan dalam penelitian dan pengembangan robotika. Seperti yang dijelaskan oleh Siegwart et al. (2011) dalam bukunya "Introduction to Autonomous Mobile Robots", Pioneer P3DX merupakan platform robot *differential drive* yang dirancang khusus untuk aplikasi penelitian dan

pendidikan. Robot ini dilengkapi dengan array sensor sonar dan sistem odometri yang memungkinkan navigasi autonomous dalam berbagai lingkungan.

Gambar 1. 2 Robot Pioneer P3DX

Sumber : https://static.generation-robots.com/robot-mobile-pioneer-3-dx.jpg

Dalam konteks penelitian robotika, Yue et al. (2014) mendemonstrasikan penggunaan Pioneer P3DX untuk implementasi algoritma navigasi autonomous, dimana platform ini menunjukkan kemampuan dan reliabilitas yang tinggi dalam eksekusi tugas-tugas navigasi kompleks.

3. Wall Follower Robot

Algoritma wall following merupakan salah satu metode navigasi fundamental dalam robotika mobile. Borenstein dan Koren (2011) mendefinisikan wall following sebagai teknik navigasi dimana robot menggunakan dinding atau obstacle sebagai referensi untuk memandu pergerakannya. Metode ini menggunakan pembacaan sensor jarak untuk mempertahankan jarak yang konsisten dengan dinding referensi.

Gambar 1. 3 Arena Wall Follower

Sumber : https://i.ytimg.com/vi/g2gXwHSLLzA/maxresdefault.jpg

Penelitian yang dilakukan oleh Martinez et al. (2015) menunjukkan bahwa implementasi wall following dengan kontrol PID dapat menghasilkan performa navigasi yang stabil dan efisien pada robot mobile. Metode ini particularly efektif untuk navigasi dalam lingkungan terstruktur seperti koridor dan maze.

4. Bahasa Pemrograman Python

Python telah menjadi bahasa pemrograman pilihan dalam pengembangan aplikasi robotika modern. Menurut Cook (2015) dalam bukunya "Robotics with Python", Python menawarkan kombinasi ideal antara kemudahan penggunaan dan kekuatan pemrograman, dengan dukungan library yang ekstensif untuk aplikasi robotika.

Gambar 1. 4 Logo Python

Sumber : https://logodownload.org/python-logo-2.png

C. Metodologi Penelitan

Alat dan Bahan

1. Perangkat Lunak
   a. CoppeliaSim Edu V4.5.1
   b. Python V3.12.1
   c. Library Math, RemoteAPI CoppeliaSim
   d. Visual Studio Code

2. Perangkat Keras
   Laptop : Acer Nitro 5 (AN515-58-55E6)

Konfigurasi

1. Menghubungkan ke CoppeliaSim

```
from coppeliasim_zmqremoteapi_client import RemoteAPIClient

# Menghubungkan ke CoppeliaSim
client = RemoteAPIClient()
sim = client.require('sim')
```

2. Inisialisasi Komponen Robot

```
# Mendapatkan handle untuk motor kiri, motor kanan, dan sensor ultrasonik

left_motor = sim.getObject('/PioneerP3DX/leftMotor')

right_motor = sim.getObject('/PioneerP3DX/rightMotor')

sensors = [sim.getObject(f'/PioneerP3DX/ultrasonicSensor[{i}]') for i in range(8)]
```

3. Fungsi mengatur kecepatan motor

```
def set_motor_speed(left_speed, right_speed):
    sim.setJointTargetVelocity(left_motor, left_speed)
    sim.setJointTargetVelocity(right_motor, right_speed)
```

4. Fungsi membaca jarak dari sensor

```
# Fungsi untuk membaca jarak dari sensor ultrasonik
def get_sensor_distance(sensor_handle):
    return sim.readProximitySensor(sensor_handle)[1] * 100  # Mengonversi dari meter ke cm
```

5. Parameter Robot

```
left_speed = 2
right_speed = 2
wheel_diameter = 47
wheel_radius = wheel_diameter / 100 / 2
wheel_circumference = 2 * math.pi * wheel_radius
move_duration = 8  steps = int(move_duration * 10)
```

6. Kontrol Simulasi

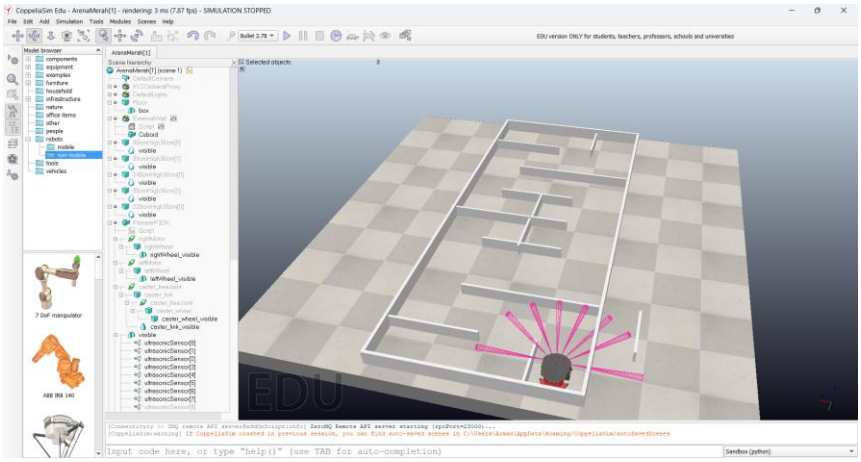
```
#Memulai Simulasi
sim.setStepping(True)
sim.startSimulation()

# Penghentian simulasi
sim.stopSimulation()
```

D. Timeline Projek

Minggu ke 1 (8 - 14 November 2024)

- Tim Merah minggu pertama

Tim coding membuat konfigurasi tentang pergerakan robot. Pada kali ini robot yang digunakan adalah robot Pioneer P3DX yang di coding menggunakan vscode.
Tim arena membuat struktur yang Dimana terdapat halangan untuk robot agar tidak hanya jalan. Arena ini yang terinspirasi dari lomba krc it fest berikut adalah gambar arena yang terbuat. Pada minggu berikutnya kami akan menambahkan tempat point dan checkpoin



Gambar 1 Progress Arena Tim Merah

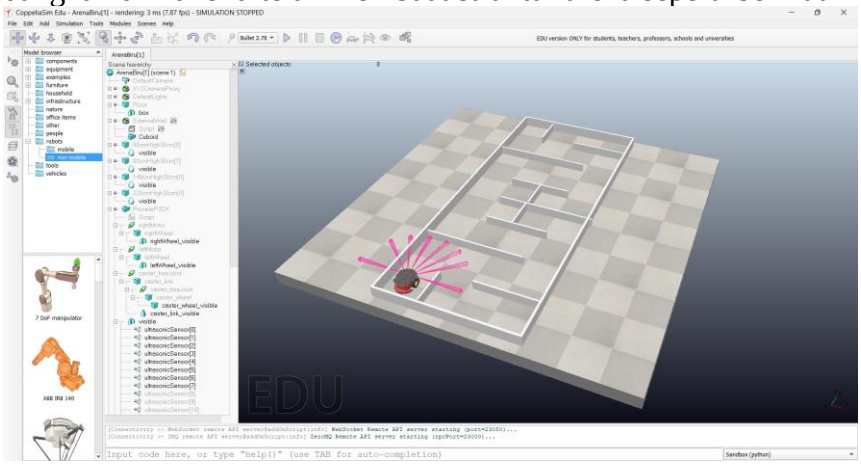
Source Code tim merah minggu pertama ada di Lampiran **Gambar 7**
Pergerakan robot tim merah di minggu pertama hanya beberapa tahp yaitu maju, berputar balik kiri, maju, berputar balik kiri kemudian maju sampai sensor mendeteksi objek kurang dari 10cm , tetapi setelah objek terdeteksi beberapa kali, koneksi vs code dan coppeliasimulator nya terputus karena terlalu banyak output yang terjadi karena ukuran dari arena belum fix yang membuat pengkuruan objek selalu kurang dari 10cm yang mengakibatkan overload dan memutuskan koneksi antara coppeliasimulator dan vscode.

```
Tahap 1: Robot bergerak maju...
Tahap 2: Robot berputar balik ke kiri...
Tahap 3: Robot bergerak maju...
Tahap 4: Robot berputar balik ke kiri...
Tahap 5: Robot bergerak maju sampai sensor 3 dan 4 mendeteksi objek kurang dari 10 cm...
Objek terdeteksi di depan, robot akan belok kiri...
Objek terdeteksi di depan, robot akan belok kiri...
Objek terdeteksi di depan, robot akan belok kiri...
Objek terdeteksi di depan, robot akan belok kiri...
```

Gambar 2 Output Code Tim Merah Minggu Pertama

- Tim Biru minggu pertama

Pada minggu pertama **tim coding** berhasil membuat robot menuju ke cekpoint 2 tetapi mengabaikan cekpoint 1 dan berhenti bergerak di cekpoint 2 source code pada lampiran **Gambar 8** sedangkan **tim arena** telah membuat struktur arena seperti berikut



Gambar 3 Progress Arena Tim Biru

Minggu ke 2 (15 - 21 November 2024)

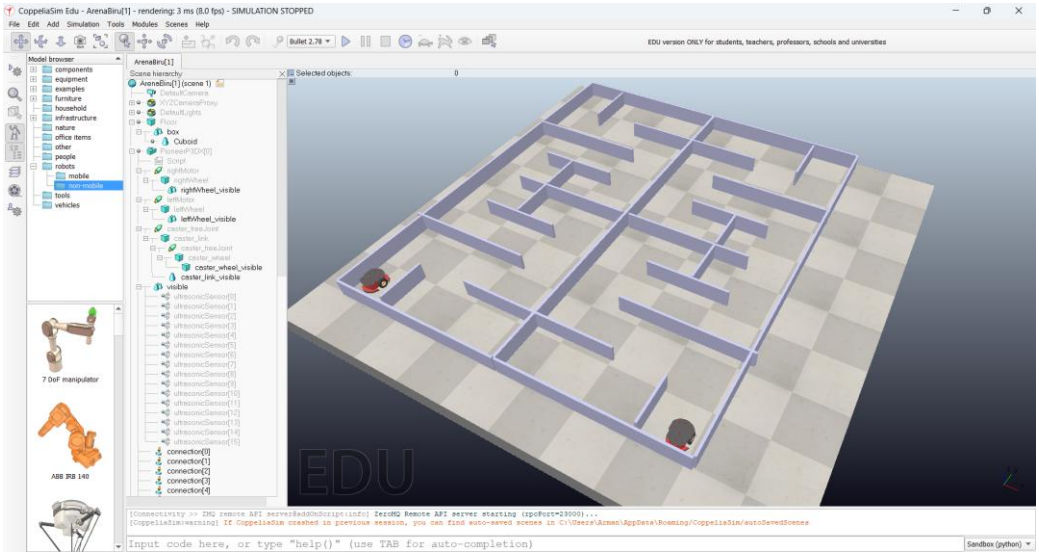
- Tim Merah Minggu Kedua

Pada Minggu kedua **Tim coding** mencoba menggunakan PID untuk robotnya dan mendapatkan hasil yang diharapkan karena robot bisa bergerak menuju cekpoin 2 walau dengan gerakan tidak stabil seperti ular berkelok – kelok untuk source codenya terdapat di Lampiran **Gambar 9**

- Tim Biru Minggu Kedua

Pada minggu kedua **Tim Coding** menambah kan parameter PID untuk gerakan robot nya dan mendapatkan hasil yang cukup yaitu bisa bergerak ke cekpoin tetapi tidak bisa berhenti untuk source codenya terdapat di Lampiran **Gambar 10**

Untuk **kedua tim arena merah dan biru** berkolaborasi untuk menyatukan antara arena merah dan arena biru seperti berikut:



Gambar 4 Arena Merah dan Arena Biru

Minggu ke 3 (22 - 28 November 2024)

- Tim Merah Minggu Ketiga

Pada minggu ketiga **tim coding** mengganti PID dengan parameter awal dan menambahkan program untuk membuat rute-rute yang di inginkan seperti rute maju, belok kiri, berhenti, putar balik dan seterusnya sampai ke finish total ada 1000 lebih baris code untuk pembuatan rute manual ini, untuk source code nya di link berikut :

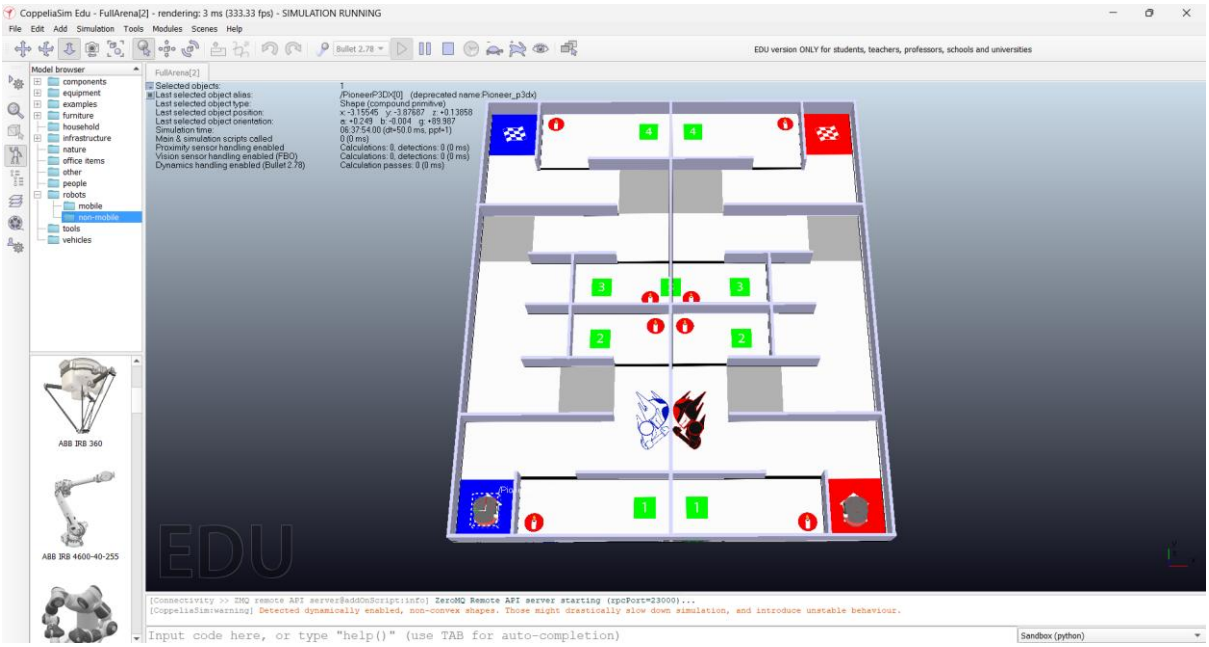
https://github.com/Arunayza/TRK_5B_2024_Kendali-Robotika/blob/main/main_merah.py

- Tim Biru Minggu Ketiga

Sama seperti tim merah , **tim coding** membuat rute manual dengan mengulang-ulang program seperti belok, maju, berhenti dan seterusnya sampai finish. Untuk source codenya di link berikut :

https://github.com/Arunayza/TRK_5B_2024_Kendali-Robotika/blob/main/main_biru.py

untuk tim Arena melanjutkan menambahkan checkpoint , finish dan gambar-gambar seperti yang ada pada referensi. Berikut adalah screenshotnya :



Gambar 5 Arena

Minggu ke 4 (29 November – 6 Desember 2024)

- Tim Merah Minggu Keempat

Pada minggu keempat, tim coding berhasil menambahkan fungsi sensor ultrasonik di sisi kiri, kanan, dan depan perangkat. Penambahan ini bertujuan untuk memungkinkan sistem mendeteksi rintangan atau tembok di tiga arah tersebut, sehingga perangkat dapat melakukan gerakan belok atau berhenti untuk menghindari hambatan dan mencapai garis finish. Namun, meskipun sensor sudah berfungsi dengan baik, hasil percobaan yang diulang masih tidak konsisten. Hal ini kemungkinan disebabkan oleh penggunaan algoritma PID (Proportional-Integral-Derivative), yang meskipun efektif dalam mengatur gerakan, ternyata tidak memberikan kestabilan yang konsisten. Variabilitas dalam pengaturan parameter PID atau gangguan dari sensor dapat menyebabkan fluktuasi pada respons perangkat, sehingga hasil gerakan menjadi tidak stabil.

- Tim Biru Minggu Keempat

Pada minggu keempat, meskipun tim coding telah berhasil menambahkan fungsi sensor ultrasonik di sisi kiri, kanan, dan depan perangkat, tim masih menghadapi kendala yang sama seperti minggu sebelumnya. Robot tetap mengalami masalah seperti menabrak rintangan dan

overuse (penggunaan berlebihan), yang menyebabkan perangkat tidak dapat bergerak dengan lancar dan mencapai tujuan secara efisien.

E. Daftar Tim

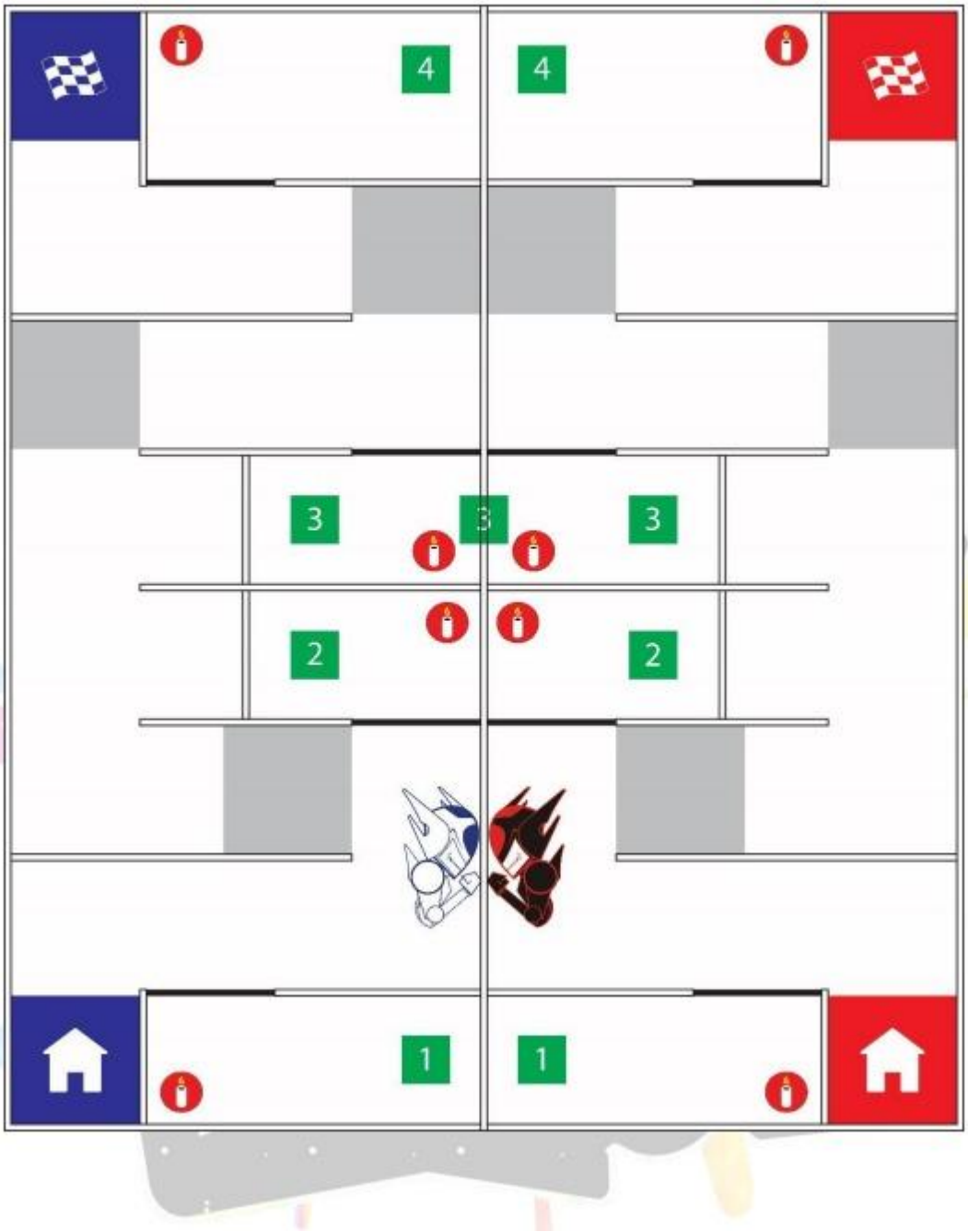
NO	NIM	NAMA	Tim		
			1	2	3
1	226661027	Zaidan Alfarizy Putra Fadilah	TIM 1		
2	226661028	Varian Rhesa	TIM 1		
3	226661029	Muhammad Ardika	TIM 1		
4	226661030	Haikal Syakir Mawarid		TIM 2	
5	226661031	Yhunike Widiya Pratama		TIM 2	
6	226661032	Sabat Jati Kristianto		TIM 2	
7	226661033	Restu Sanjaya Sihotang		TIM 2	
8	226661034	Armand Maulana	Project Leader		
9	226661035	Muhammad Fauzi Ariyo	TIM 1		
10	226661036	Muhammad Fiqri Haikal Syah		TIM 2	
11	226661037	Achmad Nur Azmi			/TIM 1/
12	226661038	Akhmad Difa Tri Saputra Hidayat			/TIM 1/
13	226661039	Muhamad Fauzan Ramdhani			/TIM 2/
14	226661040	Ahmad Reza Saputra			/TIM 2/
15	226661041	Odie Nugraha	TIM 1		
16	226661042	Adhe Ryza Dyandra Rahman			/TIM 2/
17	226661043	Muhammad Zainal Arifin			/TIM 2/
18	226661044	Soni Heri Saputra	TIM 1		
19	226661045	Ikhsan Daffa Assaddin			/TIM 1/
20	226661046	Zia Jauhari Juanda		TIM 2	
21	226661047	Andrian Pramana Putra	TIM 1		
22	226661048	Rizky Febrian			/TIM 1/
23	226661049	Muhammad Bintang Al Kausar		TIM 2	
25	226661051	Rizky Catur Risnanda	TIM 1		
26	226661052	Rakhmad Dhani		TIM 2	
	Note	/WAKIL/ dari tim 1 & 2 di tim 3			

Gambar 6 Daftar Tim

Tabel Kontribusi						
NO	NIM	NAMA	Minggu (tanggal)			
			(1) 8-14	(2) 15-21	(3) 22-28	4
1	226661027	Zaidan Alfarizy Putra Fadilah				
2	226661028	Varian Rhesa				
3	226661029	Muhammad Ardika				
4	226661030	Haikal Syakir Mawarid				
5	226661031	Yhunike Widiya Pratama				
6	226661032	Sabat Jati Kristianto				
7	226661033	Restu Sanjaya Sihotang				
8	226661034	Armand Maulana				
9	226661035	Muhammad Fauzi Ariyo				
10	226661036	Muhammad Fiqri Haikal Syah				
11	226661037	Achmad Nur Azmi				
12	226661038	Akhmad Difa Tri Saputra Hidayat				
13	226661039	Muhamad Fauzan Ramdhani				
14	226661040	Ahmad Reza Saputra				
15	226661041	Odie Nugraha				
16	226661042	Adhe Ryza Dyandra Rahman				
17	226661043	Muhammad Zainal Arifin				
18	226661044	Soni Heri Saputra				
19	226661045	Ikhsan Daffa Assaddin				
20	226661046	Zia Jauhari Juanda				
21	226661047	Andrian Pramana Putra				
22	226661048	Rizky Febrian				
23	226661049	Muhammad Bintang Al Kausar				
25	226661051	Rizky Catur Risnanda				
26	226661052	Rakhmad Dhani				
	Note	Hijau = Berkontribusi				

Gambar 7 Tabel Kontribusi per Minggu

F. Lampiran



Gambar 8 Referensi Arena

Dimensi Arena :

1. Lebar Lorong 40 cm
2. Tebal Dinding 2 cm
3. Tinggi dinding dari Lantai 15cm
4. Lebar alas 1,5 meter alas, dengan total 3 meter
5. Panjang arena 3,5 meter

Source Code Minggu Pertama :

- Tim Merah Minggu Pertama

```
import time
from coppeliasim_zmqremoteapi_client import RemoteAPIClient
import math
# Menghubungkan ke CoppeliaSim
client = RemoteAPIClient()
sim = client.require('sim')
# Mendapatkan handle untuk motor kiri, motor kanan, dan sensor ultrasonik
left_motor = sim.getObject('/PioneerP3DX/leftMotor')
right_motor = sim.getObject('/PioneerP3DX/rightMotor')
sensors = [sim.getObject(f'/PioneerP3DX/ultrasonicSensor[{i}]') for i in
range(8)]
# Fungsi untuk mengatur kecepatan motor
def set_motor_speed(left_speed, right_speed):
    sim.setJointTargetVelocity(left_motor, left_speed)
    sim.setJointTargetVelocity(right_motor, right_speed)
# Fungsi untuk membaca jarak dari sensor ultrasonik
def get_sensor_distance(sensor_handle):
    return sim.readProximitySensor(sensor_handle)[1] * 100  # Mengonversi
dari meter ke cm
# Memulai simulasi
sim.setStepping(True)
sim.startSimulation()
# Parameter kecepatan dan durasi
left_speed = 2
right_speed = 2
wheel_diameter = 47  # diameter roda dalam cm
wheel_radius = wheel_diameter / 100 / 2  # konversi ke meter
wheel_circumference = 2 * math.pi * wheel_radius  # lingkaran roda dalam
meter
distance_traveled = 0  # Inisialisasi jarak tempuh
# Tahap 1: Maju selama beberapa detik
move_duration = 8  # waktu bergerak maju dalam detik
steps = int(move_duration * 10)  # Asumsi 10 langkah per detik
try:
    # Tahap 1: Robot bergerak maju selama beberapa detik
    print("Tahap 1: Robot bergerak maju...")
    for _ in range(steps):
        set_motor_speed(left_speed, right_speed)
        sim.step()

    # Tahap 2: Putar balik ke kiri
    print("Tahap 2: Robot berputar balik ke kiri...")
    turn_duration = 3  # durasi putar balik
    turn_steps = int(turn_duration * 10)
    for _ in range(turn_steps):
```

```
        set_motor_speed(-2, 2)  # Motor kiri mundur, motor kanan maju untuk
belok kiri
        sim.step()
    # Tahap 3: Robot bergerak maju selama beberapa detik
    print("Tahap 3: Robot bergerak maju...")
    for _ in range(steps):
        set_motor_speed(left_speed, right_speed)
        sim.step()

    # Tahap 4: Putar balik ke kiri
    print("Tahap 4: Robot berputar balik ke kiri...")
    turn_duration = 3  # durasi putar balik
    turn_steps = int(turn_duration * 10)
    for _ in range(turn_steps):
        set_motor_speed(-2, 2)  # Motor kiri mundur, motor kanan maju untuk
belok kiri
        sim.step()

    # Tahap 5: Robot bergerak maju hingga sensor 3 dan 4 mendeteksi objek
kurang dari 10 cm
    print("Tahap 5: Robot bergerak maju sampai sensor 3 dan 4 mendeteksi
objek kurang dari 10 cm...")
    while True:
        # Membaca jarak dari sensor 3 dan 4
        distance_3 = get_sensor_distance(sensors[3])
        distance_4 = get_sensor_distance(sensors[4])
        # Mengecek jika keduanya mendeteksi objek kurang dari 20 cm
        if distance_3 < 20 and distance_4 < 20:
            print("Objek terdeteksi di depan, robot akan belok kiri...")
            turn_duration = 3  # durasi putar balik
            turn_steps = int(turn_duration * 10)
            for _ in range(turn_steps):
                set_motor_speed(2, -1)  # Motor kiri maju, motor kanan
mundur untuk belok kiri
                sim.step()

finally:
    sim.stopSimulation()
```

Gambar 9 Source Code Tim Merah Minggu Pertama

- Tim Biru Minggu Pertama

```
import time
from coppeliasim_zmqremoteapi_client import RemoteAPIClient

# Connect to CoppeliaSim
client = RemoteAPIClient()
```

```
sim = client.require('sim')
# Get sensor handle
sensorKiri_handle = sim.getObject('/PioneerP3DX/ultrasonicSensor[1]')
# Start the simulation
sim.setStepping(True)
sim.startSimulation()
try:
    while True:
        # Read sensor values
        detectedKiri, distanceKiri =
sim.readProximitySensor(sensorKiri_handle)[:2]
        # Print detected distances if an object is detected
        if detectedKiri:
            print(f"sensorKiri: Detected distance = {distanceKiri} meters")
        else:
            print("sensorKiri: No object detected")
        # Step simulation and wait before next reading
        sim.step()

finally:
    # Stop simulation
    sim.stopSimulation()
```

Gambar 10 Source Code Tim Biru Minggu Pertama

Source Code Minggu Kedua :

- Tim Merah minggu kedua

```
import time
from coppeliasim_zmqremoteapi_client import RemoteAPIClient
import math

# Menghubungkan ke CoppeliaSim
client = RemoteAPIClient()
sim = client.require('sim')

# Mendapatkan handle untuk motor kiri, motor kanan, dan sensor ultrasonik
left_motor = sim.getObject('/PioneerP3DX/leftMotor')
right_motor = sim.getObject('/PioneerP3DX/rightMotor')

# Sensor ultrasonik (0 = kiri depan, 2 = kiri tengah, 6 = kanan, 4 = depan
tengah)
sensors = [sim.getObject(f'/PioneerP3DX/ultrasonicSensor[{i}]') for i in
range(8)]

# Fungsi untuk membaca jarak dari sensor ultrasonik
def get_sensor_distance(sensor_handle):
    result, state, detected_point, *_ =
sim.readProximitySensor(sensor_handle)
    if state:  # Jika sensor mendeteksi objek
        return math.sqrt(sum(coord**2 for coord in detected_point)) * 100  #
Konversi ke cm
    else:
        return float('inf')  # Jika tidak ada deteksi, jarak tak terhingga
# PID Controller
class PIDController:
    def __init__(self, kp, ki, kd):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.prev_error = 0
        self.integral = 0
    def compute(self, target, actual):
        error = target - actual
        self.integral += error
        derivative = error - self.prev_error
        self.prev_error = error
        return self.kp * error + self.ki * self.integral + self.kd *
derivative

# Memulai simulasi
```

```
sim.setStepping(True)
sim.startSimulation()


# PID Parameters
pid = PIDController(kp=0.8, ki=0.1, kd=0.2)  # Tuning parameter PID
target_distance = 5  # cm, jarak target dari dinding kiri
front_threshold = 10  # cm, ambang deteksi halangan di depan

try:
    while True:
        # Membaca jarak dari sensor kiri (sensor 2) dan depan (sensor 4)
        distance_left = get_sensor_distance(sensors[2])
        distance_front = get_sensor_distance(sensors[4])
        # Jika ada halangan di depan
        if distance_front < front_threshold:
            print("Obstacle detected in front. Turning...")
            # Hentikan sementara dan belok
            set_motor_speed(-2, 2)  # Putar ke kanan
            time.sleep(0.5)  # Durasi putaran
            continue
        # Menghitung koreksi kecepatan dengan PID
        correction = pid.compute(target_distance, distance_left)
        # Menyesuaikan kecepatan motor berdasarkan koreksi
        base_speed = 2  # Kecepatan dasar
        left_speed = base_speed - correction
        right_speed = base_speed + correction
        # Membatasi kecepatan motor untuk stabilitas
        left_speed = max(min(left_speed, base_speed + 1), base_speed - 1.5)
        right_speed = max(min(right_speed, base_speed + 1), base_speed -
1.5)

finally:
    sim.stopSimulation()
```

Gambar 11 Source Code Tim Merah Minggu Kedua

- Tim Biru Minggu Kedua

```
import time
from coppeliasim_zmqremoteapi_client import RemoteAPIClient
import math

# Menghubungkan ke CoppeliaSim
client = RemoteAPIClient()
sim = client.require('sim')

# Mendapatkan handle untuk motor kiri dan kanan
```

```
left_motor = sim.getObject('/PioneerP3DX/leftMotor')
right_motor = sim.getObject('/PioneerP3DX/rightMotor')

# Mendapatkan handle untuk sensor jarak
left_sensor = sim.getObject('/PioneerP3DX/ultrasonicSensor[0]')
front_sensor = sim.getObject('/PioneerP3DX/ultrasonicSensor[3]')
# Fungsi untuk mengatur kecepatan motor
def set_motor_speed(left_speed, right_speed):
    sim.setJointTargetVelocity(left_motor, left_speed)
    sim.setJointTargetVelocity(right_motor, right_speed)
# Fungsi untuk membaca sensor jarak
def read_sensor(sensor):
    detection_state, detected_distance, _, _, _ =
sim.readProximitySensor(sensor)
    if detection_state:
        return detected_distance  # Mengembalikan jarak jika ada deteksi
    else:
        return None  # Tidak ada deteksi
# Memulai simulasi
sim.setStepping(True)
sim.startSimulation()

# Parameter
wall_distance = 0.2  # Jarak yang diinginkan dari dinding (dalam meter)
frontwall_distance = 0.3
base_speed = 5  # Kecepatan dasar motor
turning_speed = 3  # Kecepatan untuk berbelok
distance_tolerance = 0.05  # Toleransi jarak dari dinding (5 cm)
# PID parameters
Kp = 20.0  # Proportional gain
Ki = 0.2   # Integral gain
Kd = 3.0   # Derivative gain
# Initialize error values
prev_error = 0
integral = 0
try:
    while True:
        left_distance = read_sensor(left_sensor)
        front_distance = read_sensor(front_sensor)
        # Front obstacle avoidance
        if front_distance and front_distance < frontwall_distance:  # Jika
ada halangan di depan
            print("Obstacle detected in front. Turning...")
            for _ in range(10):
                set_motor_speed(turning_speed, -turning_speed)

        elif left_distance is not None:  # Jika sensor kiri mendeteksi
dinding
```

```
            # PID control for maintaining distance from the wall
            error = wall_distance - left_distance  # Calculate the error
            integral += error  # Integral term
            derivative = error - prev_error  # Derivative term
            prev_error = error  # Update previous error for the next cycle
            # PID control equation for adjusting motor speed
            adjustment = Kp * error + Ki * integral + Kd * derivative
            left_motor_speed = base_speed + adjustment
            right_motor_speed = base_speed - adjustment
            # Limit motor speeds to avoid excessive turning
            left_motor_speed = max(min(left_motor_speed, base_speed +
turning_speed), -base_speed - turning_speed)
            right_motor_speed = max(min(right_motor_speed, base_speed +
turning_speed), -base_speed - turning_speed)
            print(f"Adjusting motor speeds: Left = {left_motor_speed:.2f},
Right = {right_motor_speed:.2f}")
            set_motor_speed(left_motor_speed, right_motor_speed)  # Set
adjusted speeds

            print("No wall detected on the left. Moving forward...")
            set_motor_speed(base_speed-3, base_speed)  # Maju lurus jika
tidak ada dinding

finally:
    # Stop simulation
    # Menghentikan simulasi
    sim.stopSimulation()
```

Gambar 12 Source Code Tim Biru Minggu Kedua