Simulation of Braitenberg Method of Differential Wheeled Robot Using Gazebo ROS Features

Pradana Bagus Mauludin[1*], Noorman Rinanto[2], Lilik Subiyanto[3],
Afif Zuhri Arfianto[4], Agus Khumaidi[5], Zindhu Maulana Ahmad Putra[6]
[1,2,3,4,5] Automation Engineering Study Program
[6] Marine Electrical Engineering Study program
[1,2,3,4,5,6] Shipbuilding Institute of Polytechnic Surabaya, Indonesia
[1]pradanabagus@student.ppns.ac.id, [2]noorman.rinanto@ppns.ac.id

Abstract - Differential wheeled robots are one type of mobile robot that is easy to design and operate. However, cost limitations and the risk of hardware damage due to collisions when the robot fails to avoid obstacles often become obstacles in direct research on physical devices. To overcome this, this research was conducted in the form of simulations using the Robot Operating System (ROS) and Gazebo. The simulation utilizes the Burger model TurtleBot robot and applies the Braitenberg method as an obstacle avoidance strategy to detect and avoid obstacles in the surrounding environment. The Braitenberg method is an algorithm designed for wheeled vehicles to move automatically by utilizing data from left and right-side sensors to control motor rotation. Therefore, it is necessary to test the implementation of this method to ensure that the robot is able to move independently while avoiding obstacles in the vicinity.

Keywords: Turtlebot9, ROS, Gazebo, Turtlebot, Braitenberg

I. INTRODUCTION

The development of robotics technology continues to progress rapidly, especially in terms of mobility and control systems (navigation systems) [1]. One type of robot that is widely developed and applied in various fields is a wheeled robot. Wheeled robots offer advantages in terms of mechanical simplicity and ease of control, and are widely applied to service robots, indoor exploration robots, and autonomous vehicles. Based on the configuration of the number of wheels and the control system used, wheeled robots can be divided into several types, one of which is a two-wheeled robot. A two-wheeled robot is a type of mobile robot equipped with one wheel on the right side and one wheel on the left side. This robot requires a control system (controller) to maintain balance, because without proper control, the robot cannot stand or move stably [2].

Mobile robots with a differential two-wheel configuration are one type of robot that is widely used in research and industry because of its simplicity in design and its ability to maneuver in various environments. In maneuvering a two-wheel robot can also avoid obstacles in its place. One popular example of this robot is the TurtleBot, which is often used as a development platform in ROS (Robot Operating System) based robotics systems. In simulations using Gazebo, TurtleBot3 can be run to mimic the real behavior of robots in the physical world. With two main drive wheels located on the left and right sides. ROS 1 provides libraries that enable integration and testing of robotics systems in a modular and structured manner [3].

Simulation is an essential component in the testing phase of mobile robots, as it allows the evaluation of system performance in a safe, efficient, and controlled manner prior to implementation in a real environment. By using simulation, potential damage to both the robot and the test environment can be significantly reduced [4]. In the Gazebo simulation environment, LiDAR sensors can be represented virtually but still provide data that resembles real conditions. Simulation using Gazebo makes it easy for users to implement and test the Braitenberg algorithm in avoiding obstacles, without worrying about damage to the hardware. In addition, integration with ROS supports communication between nodes, data processing from LiDAR sensors, and motor control in a structured and flexible manner.

In the development of robotics technology, especially in mobile robots, various challenges arise that need to be overcome, one of which is the ability of robots to detect and avoid obstacles effectively to ensure safe and efficient navigation in various environmental conditions. One method used to process distance sensor data against motor rotation on mobile robots is the Braitenberg method. This method is inspired by the working principle of the biological nervous system, where motor responses are generated directly from sensory stimuli through simple yet effective connections. In the context of obstacle avoidance, the Braitenberg method allows robots to respond to the environment in real-time by connecting

the proximity sensor output directly to the motor actuators, resulting in intelligent-looking behavior even without complex algorithms.

This research aims to design a simulated mobile robot using Gazebo software, and implement the Braitenberg method as an obstacle avoidance strategy. Through this approach, the robot is expected to be able to respond reactively to its surroundings based on proximity sensor data, so that it can avoid obstacles automatically in a virtual environment that resembles real conditions.

II. METHODS

In this research method, we will explain how the data from the LiDAR sensor is processed and used to implement the Braitenberg method. The flowchart in Figure 1 shows the stages of the system in avoiding obstacles by utilizing the LiDAR sensor integrated with the Braitenberg approach.

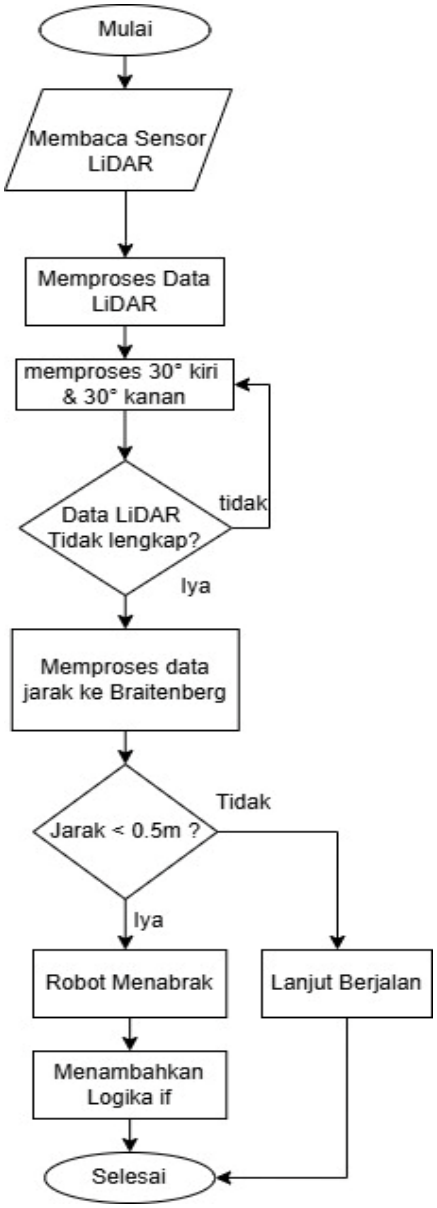

Figure 1 System Flowchart

In applying this method, the first step is to integrate the LiDAR sensor attached to the TurtleBot. This sensor adopts the principle of laser triangulation by using a camera that captures infrared rays reflected onto the object in front of the sensor, and the reflection time of the rays is used to calculate the distance between the sensor and the object or obstacle [5]. After the data from the LiDAR sensor is successfully obtained, the system will process the information to calculate the distance on the left and right sides with a tilt angle of 30 degrees each. If the data reading from the sensor is incomplete or fails, the system will automatically re-read to ensure data accuracy. The validated distance data is then used in the Braitenberg method approach for obstacle avoidance. When an object is detected at a distance of less than 0.5 meters in front of the robot, the robot has the potential to hit an obstacle, so it is necessary to add conditional logic (if statement) to the Python program to overcome this situation. Conversely, if the distance on the left and right sides exceeds 0.5 meters, it can be assumed that there are no obstacles around the robot and the robot will continue to move forward through an obstacle-free area.

ROS (Robot Operating System)

ROS (Robot Operating System) is an operating system for robots that provides various software in the form of tools, libraries, and packages used to control robots. ROS is developed by a community of developers and contributors in the field of robotics software [6]. Since its appearance, many institutions and researchers have utilized the RP2 robot from Willow Garage as an object of research and development in the field of robotics. In addition, the ROS community has also developed Turtlebot as a standard platform for ROS-based wheeled robots.

The development of robotics technology has made significant progress along with the increasing needs in the academic and industrial fields. This progress encourages many companies in the robotics industry to develop and produce various important components, such as Integrated Circuit (IC), sensors, and microcontrollers, which act as part of the hardware system on the robot. Innovation in these components is the main foundation in creating robotics systems that are increasingly sophisticated and adaptive to the needs of the times.

Braitenberg Method

Braitenberg is a two-wheeled vehicle model equipped with sensors on each motor. This model uses the principle of sensorimotor coupling, where each motor directly drives one wheel based on the response of the connected sensors, thus allowing the vehicle to react to environmental stimuli automatically [7]. In this research, the Braitenberg method is used to implement an obstacle avoidance mechanism in the robot simulation arena. LiDAR sensors are utilized to detect the presence of obstacles on the right and left sides of the robot. The information from the sensor is then used to stimulate the

movement of the right and left motors, so that the robot can react adaptively to the surrounding environmental conditions.

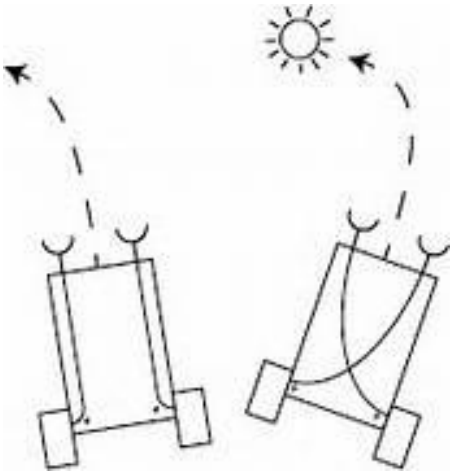

Figure 2. Braitenberg Vehicle

Figure 6 shows an illustration of a wheeled robot model based on the Braitenberg principle. If the left sensor is connected to the left motor and the right sensor to the right motor, the robot will move away from the obstacle, because the sensor response will increase the speed of the motor on the same side as it detects the object. Conversely, if the left sensor is connected to the right motor and the right sensor to the left motor, the robot tends to approach the obstacle, because cross sensorimotor coupling causes an increase in speed on the opposite side to the detection position. From the above understanding, it can be explained about the program algorithm that has been made below.

```
Listening Code Braitenberg

kanan_weight = max(0.0, min(1.0, 1.0 - kanan_avg))
kiri_weight = max(0.0, min(1.0, 1.0 - kiri_avg))
kanan_motor = base_speed * (1.0 - kiri_weight)
kiri_motor = base_speed * (1.0 - kanan_weight)
twist.linear.x = (kanan_motor + kiri_motor) / 2.0
twist.angular.z = (kiri_motor - kanan_motor) / 0.2
```

The code above is an application of the Braitenberg algorithm with a negative cross-coupling (cross-inhibition) approach, where the motor speed is determined based on the distance data from the LiDAR sensor. The calculation process of the Braitenberg algorithm for motor speed follows the following equation.

$$n_l = 1 - \frac{R_l}{R_{max}} \quad (1)$$

$$n_r = 1 - \frac{R_r}{R_{max}} \quad (2)$$

$$M_l = V \times n_r \quad (3)$$

$$M_r = V \times n_l \quad (4)$$

$$v_{lin} = \frac{M_r + M_l}{2} \quad (5)$$

$$\omega = \frac{M_l + M_r}{L} \quad (6)$$

The calculation process in the Braitenberg method starts by determining the left weight n_l based on the left sensor distance R_l, and the right weight n_r based on the right sensor distance R_r, with both normalized to the maximum sensor distance R_{max}. These left and right weight values are then used to calculate the velocities of the left motor M_l and the right motor M_r through cross-correlation to the base velocity V, where the right sensor affects the left motor, and vice versa. Next, the robot's linear velocity v_{lin} is calculated as the average of the velocities of the two motors, i.e. $[\![M_r \; sum \; up \; M_l]\!]$ _1 and then divided by two. The angular velocity ω is obtained from the difference in the speed of the two motors divided by the distance between the wheels (L).

Gazebo Application

Developers and communities in the field of robotics have presented Gazebo and ROS as simulation software that allows users to learn and develop robots without the need to physically build them. Gazebo is an open source simulation application that can be used to simulate hardware or dynamic systems. This platform supports robot design and artificial intelligence (AI) system training with fairly accurate simulations, both in complex indoor and outdoor environments. Gazebo runs on the Linux operating system and requires adequate graphics support. Gazebo has also been integrated in the ROS installation package. ROS itself is designed for robot software development, while Gazebo is used to simulate its hardware aspects.

Turtlebot Burger

R Turtlebot is a small mobile burger developed by Robotis and the ROS (Robot Operating System) community. The robot uses a two-wheel differential configuration and is designed as an open-source platform intended for development and research activities in the field of robotics, particularly in terms of mapping and obstacle avoidance.

The TurtleBot3 Burger is compact, lightweight, and modular, making it easy to change and expand as needed. The robot is equipped with essential components, such as an OpenCR microcontroller, Raspberry Pi or Jetson Nano, and various sensors, including motor encoders, LiDAR sensors, and IMUs. Burger's TurtleBot3 supports ROS and supports Gazebo simulation. Figure 2 shows the shape of the burger turtlebot.

In this research, we will simulate the movement of wheeled robots without using physical hardware, so as to minimize the risk of direct collision or damage. Simulation of obstacle avoidance on this robot will be carried out using one of the Gazebo platforms provided by ROS.

Figure 3 Turtlebot Burger

Arena Planning

The design of the simulation environment for TurtleBot was carried out using the Gazebo application. In this study, two types of arenas with different design variations and dimensions were developed, each designed to evaluate the robot's ability to avoid obstacles in various environmental scenarios. Figure 4 shows Arena 1, which is the default simulation environment built into TurtleBot known as Stage 4. This arena was used in the first simulation trial. Arena 1 has a relatively simple structure, with walls as barriers that serve as obstacles. The design resembles a maze with several narrow passages that challenge the robot's navigation and obstacle avoidance capabilities effectively.

Next, Figure 5 shows Arena 2, which is also the built-in simulation environment of TurtleBot and is known as House. Compared to Arena 1, Arena 2 has a higher level of complexity. The design of this arena resembles the interior of a house, complete with room dividers and narrow pathways, thus adding challenges for the robot in detecting and avoiding obstacles. Both types of arenas are used to evaluate the performance of the obstacle avoidance algorithm under different environmental conditions, in order to test the reliability and adaptability of the developed system.

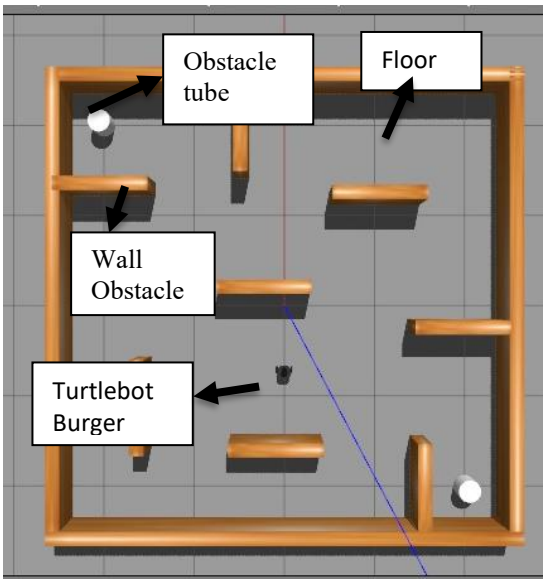

Figure 2.Arena 1

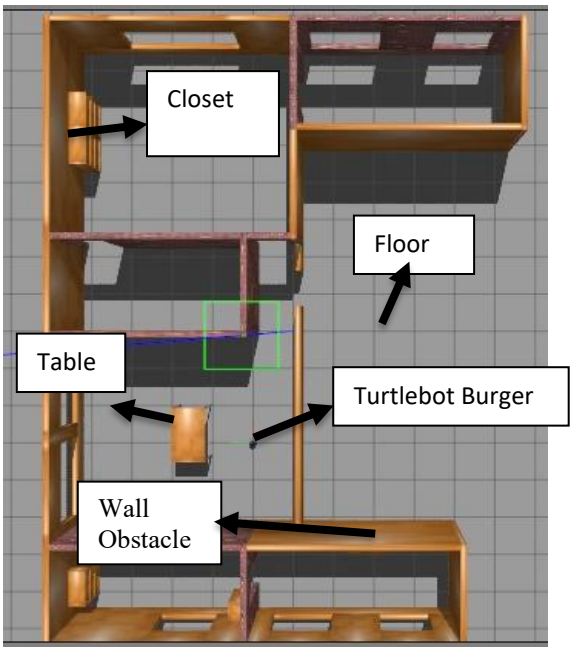

Figure 5. Arena 2

III. RESULT AND DISCUSSION

LiDAR Sensor Testing

The coin acceptor test was conducted partially to determine the LiDAR sensor testing is used to obtain data from scanning using a laser beam. The data generated includes information such as angle_min, angle_max, angle_increment, time_increment, scan_time, range_min, range_max, ranges, and intensities. LiDAR sensor testing was carried out using the Gazebo platform, because the sensor was already installed by default on TurtleBot Burger. To obtain and display the scan data, the rostopic echo /scan command was used through the terminal. The results of the scan are shown in Figure 6, which includes parameters such as angle_min, angle_max, angle_increment, time_increment, scan_time, range_min, range_max, ranges, and intensities. Through this test, it is expected that the sensor is able to function optimally in processing data into accurate distance information.

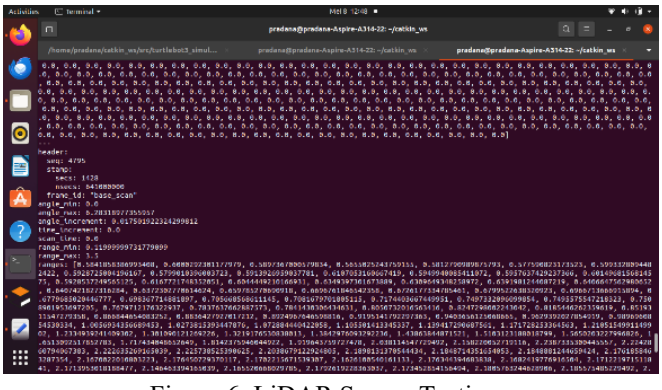

Figure 6. LiDAR Sensor Testing

Arena Testing 1

This test starts by entering the catkin_ws folder in the folder that has the turtlebot package that has been downloaded. Then run Gazebo on ROS and run the Braitenberg node with

the file name braitenberg.py contained in the ROS package that was previously created. Testing was carried out using TurtleBot Burger in the Stage 4 arena on Gazebo, with the aim of evaluating whether the Braitenberg algorithm can function properly in avoiding obstacles.

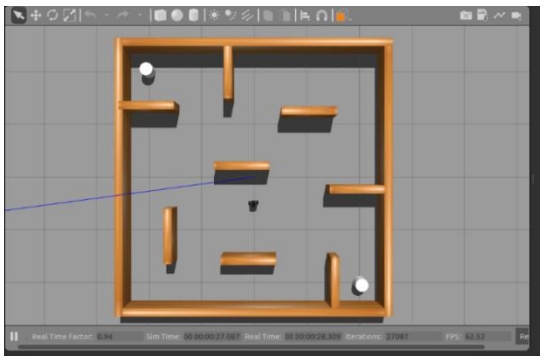

Figure 7. Stage 1 Testing

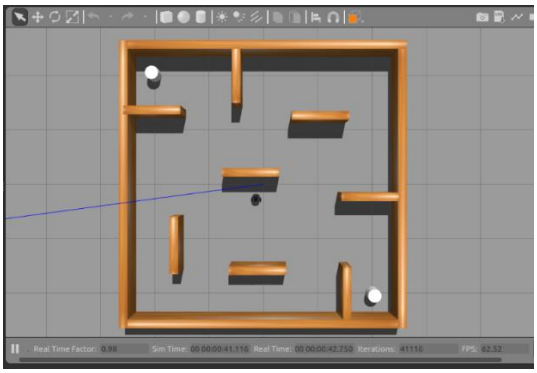

Figure 8. Stage 2 Testing

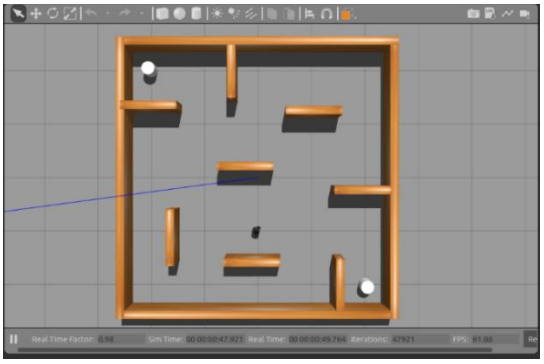

Figure 9. Stage 3 Testing

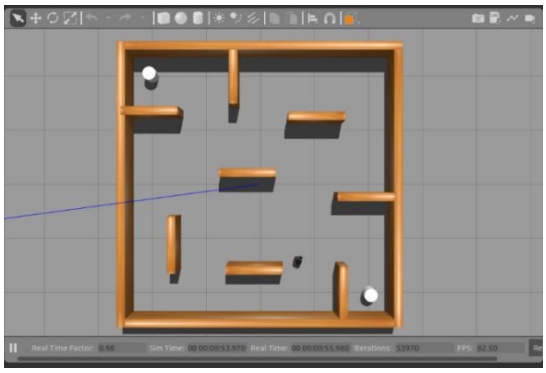

Figure 10. Stage 4 Testing

In Figure 7, it can be seen that the robot moves straight when it first runs the braitenberg.py node until the 28th second. After 42 seconds, as shown in Figure 8, the robot starts to detect an obstacle in front of it. Then, at the 49th second as shown in Figure 9, the robot turns left and returns to the starting point. Based on the experiments in Arena 1, the robot shows the ability to maneuver well in finding free space and avoiding obstacles in the form of maze walls.

Arena Testing 2

The second test was conducted in the 2-house arena by running the Gazebo simulation using the turtlebot3_house. launch file. This test procedure is similar to the first test, both in terms of the program and the way the robot operates, but is carried out in a different and more complex environment because there are narrow rooms resembling the interior of a house. Experiments in this arena can be observed through the images presented below.

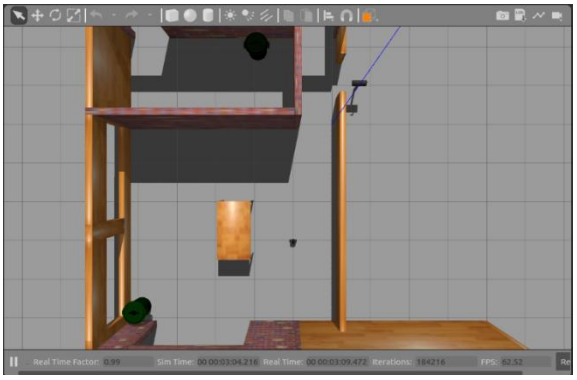

Figure 11. House 1 Testing

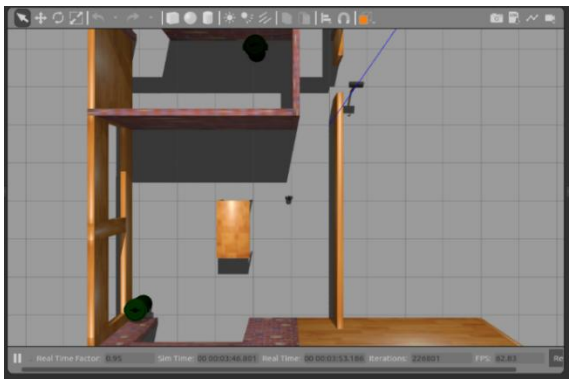

Figure 12. House 2 Testing

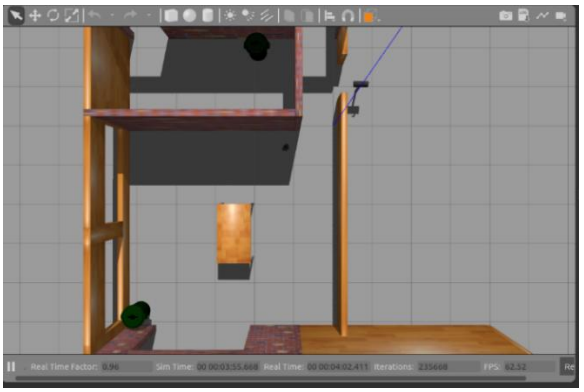

Figure 13. House 3 Testing

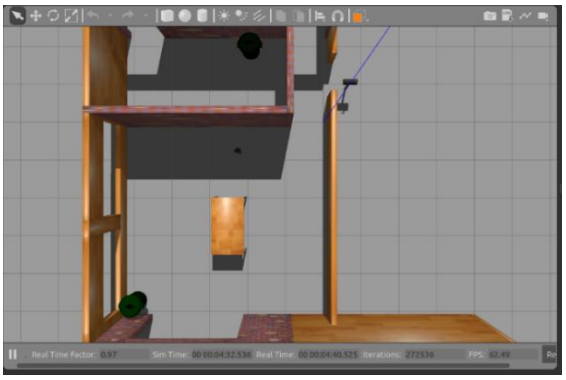

Figure 14. House 4 Testing

Based on Figure 14, it can be seen that the robot successfully avoided the obstacle and performed a turning maneuver to return to the starting point of its movement.

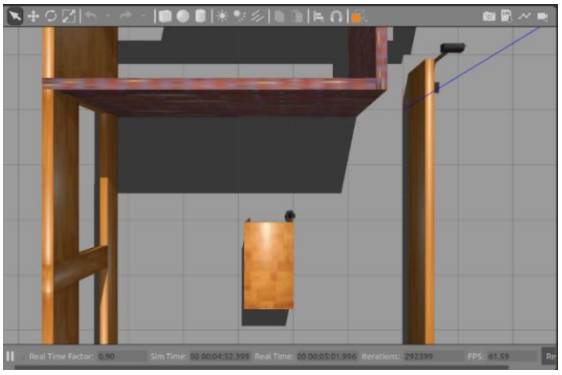

Figure 15. House 5 Testing

When running the braitenberg.py node, the robot moved straight ahead at 3 minutes 53 seconds, as shown in Figure 12. Furthermore, at 4 minutes 2 seconds, the robot detected an obstacle, then turned around and turned right to return to the starting position, as shown in Figure 13. However, on its way, the robot encountered an obstacle when it hit a table leg, as shown in Figure 15. At the time of the collision, the robot did not show an avoidance response and remained stationary. This is because the LiDAR sensor does not read any objects within less than 0.5 meters on the left or right side, so the motor speed remains constant and does not trigger any avoidance movement. Under these conditions, the detected distance on the left side is 2.48 meters, while the right side is 1.97 meters, as shown in Figure 16. To overcome this obstacle, it is necessary to add logic to the program so that the robot can continue its movement.

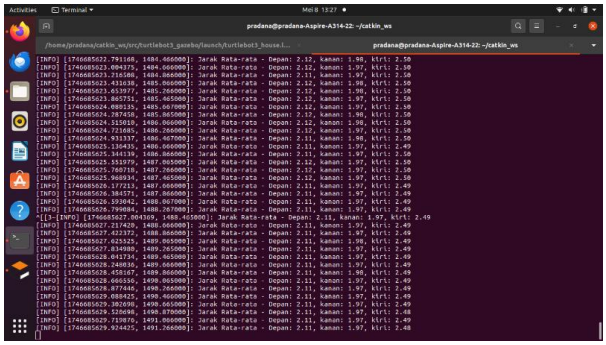

Figure 16. Distance The Robot Hit

IV. CONCLUSION

Based on the test results and implementation of the Braitenberg algorithm on the TurtleBot Burger robot in the Gazebo simulation environment, it can be concluded that this method is able to perform the obstacle avoidance function effectively as long as the robot is in a fairly open area. However, in narrow or closed environments, the robot tends to get stuck and has difficulty finding an exit path or free space. The advantages of applying the Braitenberg method include its ability to operate only by utilizing sensor data from the left and right sides without requiring a mapping process, as well as relatively simple and non-complex logic calculations.

REFERENCES

[1] M. S. Ummah, "Implementation of Braitenberg Method and Odometry in Behavior Based Architecture for Fire Extinguishing Robot Navigation System," Sustain., vol. 11, no. 1, pp. 1–14, 2019,

[2] M. M. Rokhmat, "Implementation of a two-wheeled robot balance system using differential integral proportional controller," J. Mhs. TEUB, 2013.

[3] Jalil, Full Guide : Robot Operating System (ROS). Yogyakarta: Andi, 2022.

[4] Nirmala, R. Hidayati, R. S. Komputer, and U. T. Pontianak, "Ros Based Car Robot Autonomous Navigation System On Robot," vol. 10, no. 2, pp. 288–296, 2024.

[5] Louise, Y. Susanthi, and Muliady, "Mapping and Navigation for Food Delivery Robots in ROS-Based Restaurants," Techné J. Ilm. Elektrotek., vol. 22, no. 1, pp. 111–128, 2023,

[6] D. A. N. W. J. Teahan, "Using Compression to Find Interesting Behaviors in Hybrid Braitenberg Vehicles," no. November 2020, 2021.

[7] S. Stoyanov, K. Gerov, "Mobile Robot Simulation and Navigation in ROS and Gazebo," Proceedings of the 2020 International Conference on Mathematical Methods and Computational Techniques in Science and Engineering (MCS), pp. 113–120, 2020.

[8] N. Pinrath et al., "Development of a Real-Time Simulator for a Semi-Autonomous Robot Utilizing Braitenberg Algorithm in CoppeliaSim and ROS," Journal of Robotics and Mechatronics, vol. 34, no. 3, pp. 631–642, 2022.

[9] R. Mengacci, G. Zambella, G. Grioli, D. Caporale, M.G. Catalano, A. Bicchi, "An Open-Source ROS-Gazebo Toolbox for Simulating Robots With Compliant Actuators," Frontiers in Robotics and AI, vol. 8, Art. no. 713083, Aug. 2021.

[10] A. Jalil, "Robot Operating System (ROS) dan Gazebo sebagai Media Pembelajaran Robot Interaktif," ILKOM Journal of Computer Science, vol. 10, no. 3, pp. 284–289, Dec. 2018.

[11] B. Udugama, "Mini bot 3D: A ROS Based Gazebo Simulation," arXiv preprint arXiv:2302.06368, Feb. 2023.

[12] Y. Hu, J. Zhang, "Manipulation Task Simulation using ROS and Gazebo," Proceedings of the 2014 IEEE International Conference on Robotics and Automation, pp. 599–604, 2014.

[13] L. Huang et al., "Research on Path Planning Based on Braitenberg Robot Collision Avoidance Method," Proceedings of the 2021 International Conference on Mechanical Engineering and Electrical Automation, Atlantis Press, pp. 211–216, 2021.