Implementation of LiDAR Sensor for Mobile Robot Delivery Based on Robot Operating System

Fiqri Ahmad Agung [a,1], Herizon [a,2*], Era Madona [a,3], Muhammad Rohfadli [a,4], Ja'far [b,1]

[a] *Department of Electrical Engineering, Politeknik Negeri Padang, Indonesia*
Faculty of Power and Aeronautical Engineering, Warsaw University of Technology, Poland

[a,2] *herizon.pnp.ac.id*, [b,1]*jafar.stud@pw.edu.pl*
** corresponding author*

A B S T R A C T

Recent research has focused on the development of mobile delivery robots as part of the rapidly evolving autonomous technology. The main focus is on creating mobile robots capable of efficiently delivering goods without human intervention. This involves designing systems, core technologies, and overcoming various challenges. The navigation system of the delivery robot is enhanced by the use of LIDAR (Light Detection and Ranging) sensors. One of the sensors used is the RPLIDAR A1M8 which is capable of detecting nearby objects with a distance reading of 360°. Trials were conducted to evaluate the robot's navigation performance with a load variation of 0 kg to 20 kg. The test results show that RPLIDAR A1M8 can read distances up to 1200 cm with an error rate of about 3.15%, and a minimum distance of 15 cm with an error of 1.45%. The durability of this sensor reading is well maintained, with a presentation error of only 0.4%. Mapping trials in the room produced an average error of 18.57% without tracking, and 5.02% with tracking. The weight of the load carried by the robot affects the speed and travel time. The heavier the load, the robot speed decreases from 0.264 m/s at 0 kg load to 0.127 m/s at 20 kg load. Likewise, the travel time increases with the weight of the load, from 7.6 seconds to 15.6 seconds.

K E Y W O R D S

Article history

Received October 10, 2023
Revised October 19, 2023
Accepted December 31, 2023

Keywords

Navigation System, RPLIDAR A1M8, Mobile Robot Delivery, ROS, SLAM

1. INTRODUCTION

The development of technology and industrial automation has triggered the development of various types of robots. Robots are used to increase productivity, with ease of use and various types, such as humanoid, bioloid, and mobile robots. One obvious application is the widely used delivery robot. Robot technology uses feedback control systems to operate efficiently and can cooperate with humans in various contexts.

In the beginning, robots were created to replace the role of humans in dangerous situations. They can complete tasks quickly, accurately, and efficiently thanks to pre-programmed programs[1]. The main advantage of robots is their tireless endurance in carrying out tasks, so their use provides great advantages in tasks, so their use provides great benefits in various sectors, including industry and services. To determine the direction and motion as desired, the robot must be equipped with navigation sensors[2].

A simple method of navigation in robots involves the use of Ultrasonic sensors to detect the distance to obstacles around the robot. However, Ultrasonic sensors have limitations, such as their limited detection distance and low accuracy against angular[3]. To overcome the limitations of distance and angle readings, Light Detection and Ranging (LIDAR) proximity sensors are used. LIDAR is a remote sensing technology that has great potential to assist in mapping, monitoring, and estimating the location of spatial elements in various applications. Due to its high data density and high level of accuracy, LIDAR sensors are very suitable for use in mapping and robot navigation[4][5].

The Mobile Robot is built using the Hector SLAM method with the help of the RPLIDAR A1M8 sensor and the Robot Operating System (ROS). Hector SLAM is a Simultaneous Localization and Mapping (SLAM) method that allows a robot to create a map and simultaneously determine its position on the map[6].

In using the SLAM method, the Robot Operating System (ROS) is used as a sensor data processing system that controls the Autonomous Mobile Robot. ROS plays an important role in managing sensor data and controlling the robot to support the SLAM method.

The purpose of making Mobile Robots is to replace the role of humans in moving goods with the aim of improving work safety and avoiding risks such as being crushed by goods. This project utilizes Hector SLAM method with the help of RPLidar A1M8 sensor and Robot Operating System (ROS) to achieve the goal.

2. METHODS

2.1. Mobile Robot

Mobile Robots are a type of robot that uses wheels or legs to move and is equipped with sensors. They have a wide range of applications in sectors such as industry, transportation, and the military, and are often used in movement control and surveillance.

2.2. Metode SLAM

SLAM (Simultaneous Localization and Mapping) is a method used by a robot to create a map while being able to localize itself on the map it creates. This method allows the robot to perform two tasks at once, namely location mapping and positioning in the map it creates. Mapping is the integration of information from various sensors into a consistent model, while localization is the estimation of the robot's position and orientation in the map[7]. The SLAM method is commonly used in the development of Mobile Robots for reading flat areas. The robot is equipped with sensors such as LIDAR or SONAR to perform good readings. There are several ways to implement SLAM algorithms in a 2D context, such as Hector SLAM, G-Mapping, and Karto SLAM. SLAM algorithms can be grouped into three categories: first, based on the type of sensor used to measure the distance (such as laser, ultrasound, or odometry); second, based on the calculation method (with Kalman filter or particle filter algorithms)[8]; and third, based on the structure used to measure the distance. These SLAM algorithms are all contained in the Robot Operating System package[9].

2.3. Robotic Operating System (ROS)

The Robot Operating System (ROS) is a platform that uses open source as its base. This platform includes various tools and libraries that are useful in the process of developing programs or applications for robotic systems. By using ROS, developers have access to various features and functions that can be used openly, facilitating the development of robotic programs more effectively and efficiently[10]. ROS has a modular structure, making it easy to integrate old programs with those being developed. Communication between processes is done through nodes that communicate via topics. Nodes, which are programs for specific functions, can publish data as publishers (for example, data from laser scanners) and function as subscribers that receive and display data[11]. This data streaming process occurs in a local TCP network known as the ROS network. A number of nodes that communicate with each other are organized into packages in the Robot Operating System (ROS). The package consists of nodes and various libraries. In the ROS communication system, the concept of publisher and subscriber nodes is used, which can be explained visually through illustrations or figure 1.

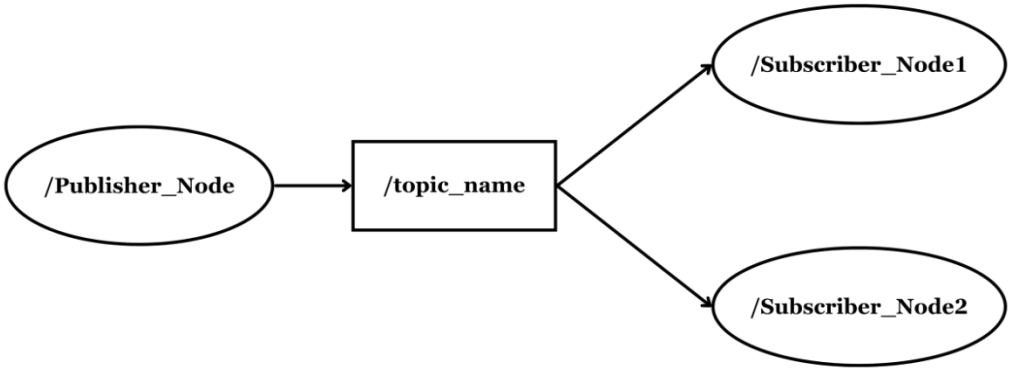


FIG 1. ROS system with publisher and subscriber nodes

In the Robot Operating System (ROS), there are two types of nodes, namely publishers and subscribers. In general, a node has no direct knowledge of which nodes are its partners in communication. Instead, a node can publish information to a topic, while other nodes that need the information will subscribe to the same topic. The node that publishes is called the publisher, while the one that subscribes is called the subscriber. Topics are identified by /topic_name, and are used as TCP or UDP-based data communication channels. Each topic has a message type with a specific format. A message is a type of data in the Robot Operating System (ROS). Within ROS, there is an RViz package that allows visualization of 2D map results. RViz, which stands for ROS Visualization, is a package in ROS that serves as a 2D or 3D visualization tool[8]. Its functions include visualizing the robot and sensor data obtained from the LIDAR sensor.

2.4. Light Detection and Ranging (LIDAR)

LIDAR (Light Detection and Ranging) is a remote sensor technology that uses the properties of scattered light to determine the distance and information about an object from its intended target. This sensor uses laser pulses to determine the distance of an object. The working principle involves sending a laser beam to the object which

is then reflected back to the sensor. The reflected light is received, then picked up and analyzed by the detector. Changes in the composition of the light received from the target are identified as objects. The output of the LIDAR process includes the angle and length of the distance affected by the reflection of the object. The output of the LIDAR sensor is a distance measured at a certain angle. To determine the time required by the sensor to measure the distance between the sensor and the plane, it can be calculated using a certain formula. The Lidar Sensor Working System is shown in Figure 2.

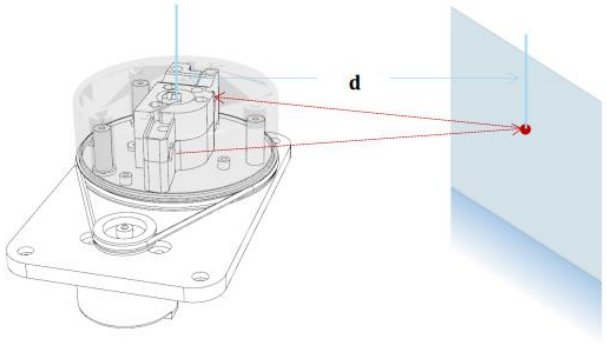


FIG 2. Lidar Sensor Working System

$$d = C\frac{t}{2}$$

Ket:
d = distance between the sensor and the measured object (m)
c = Speed of Light ($3x10^8 m/s$)
t = Signal Travel Time (s)

2.5. Microcontroller Arduino

One of the famous ATmega microcontroller modules is arduino. Arduino nano is one type of ATmega328P-based arduino board that is small in size making it suitable for use in projects that do not require many I/O pins.

2.6. System Design

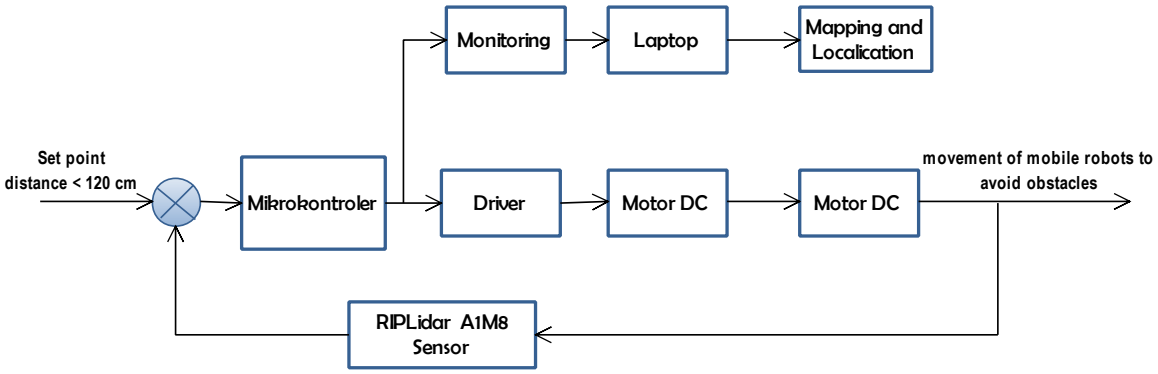


FIG 3. Block Control Diagram

The block diagram shown in figure 3 explains the processes contained in the system visually, where in the block diagram there are parts that represent the components contained in the system, the following is an explanation of the block diagram in figure 3:

a. In the lidar sensor block as an input medium that reads the distance of obstacles traveled by the mobile robot.
b. The arduino microcontroller block processes the data provided by the lidar sensor, where the arduino processes the error data received to synchronize the movement of the mobile robot.
c. The motor driver block will receive a pwm signal issued from the arduino which can adjust the speed of the dc motor on the mobile robot.
d. In this dc motor block which is used to move the mobile robot according to the commands given from the arduino microcontroller and is regulated by its speed from the results of the arduino microcontroller signal to the motor driver.

e. The laptop block processes the distance data obtained on the arduino microcontroller and converts the data by creating a mapping process that uses the ROS package in the form of an RViz display, so that it can display maps and localization through which the mobile robot passes.

The working principle of the mobile delivery robot uses the wall following method where when the robot is run to start delivering goods, the mobile robot will track according to a predetermined set point distance in order to avoid obstacles that are near the mobile robot. In addition, this mobile delivery robot also provides a mapping and localization feature that can create a map of the navigation results carried out on the mobile robot in wall following.

To do mapping, it can also be used with manual control, namely by ordering the mobile robot to move based on keyboard input from another laptop. This can be done by connecting the laptop on the mobile robot with the control laptop used by the user. For this manual work system, remote desktop from one laptop to another. And for the mapping results can also be displayed from the control laptop used by the user[4].

2.7. Software Design

a. Flowchart Manual Control Movement

To map the room manually, the robot is controlled using keyboard commands from a laptop to the Arduino Nano. The command is sent serially. This manual control aims to correct the room mapping results so that nothing is missed. Commands are sent in the form of characters. The list of characters sent is as listed in Table 1.

TABLE 1. *Mobile Robot Drive Character Delivery*

characters sent	command
w	forward robot
s	backward robot
a	left robot
d	right robot
q	robot stop

The flow diagram of sending commands to the robot is shown in Figure 4.

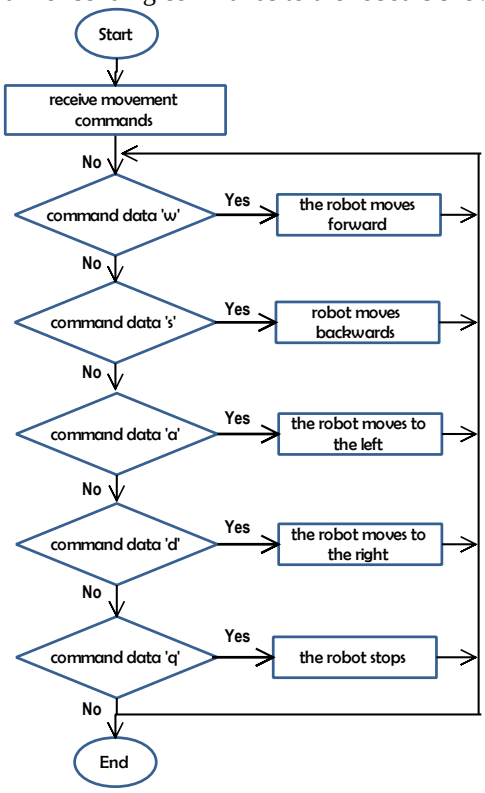


FIG 4. Manual Control Motion Flow Chart 4 Manual Control Motion Flow Chart

b. Flowchart Autonomous Movement

The autonomous movement created in this research is generated from the reading of the RPLidar A1M8 sensor. The reading data from this sensor is in the form of a distance value at an angle of 0 ° to

360 ° which will be published by a publisher node and then will be subscribed to by the subscriber node, which is the node that executes the autonomous movement program. The following is a block diagram of the Autonomous Movement process shown in Figure 5.

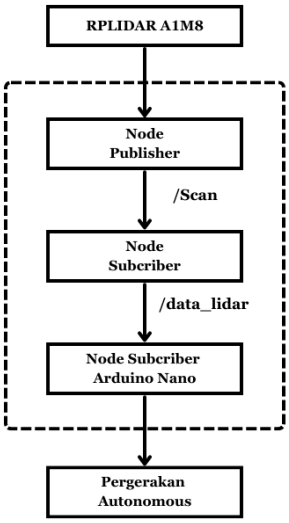


FIG 5. Flowchart Autonomous Movement

c. Mapping and Localization

The Mapping and Localization process in this research uses the Hector SLAM method. To get Hector SLAM, you must install the package first, and that has been done before. The flowchart of using Hector SLAM on this mobile robot shown in Figure 6.

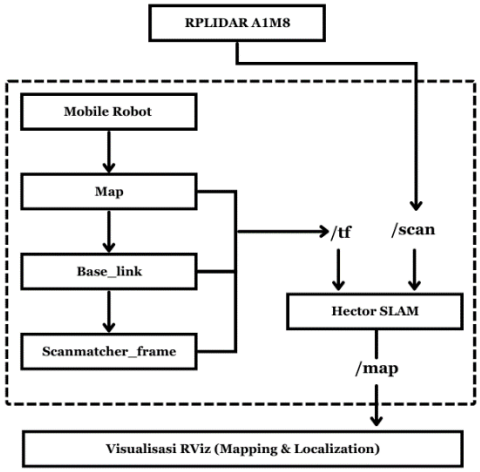


FIG 6. *Flowchart Mapping and Localization Hector SLAM*

The RPLIDAR A1 M8 sensor will publish a data into the /scan topic, where the data from this /scan topic can be seen in 'sensor_msg/LaserScan', the data sent by the RPLIDAR A1M8 sensor will be used to create a two-dimensional (2D) map visualization using the scan matching method found in Hector SLAM. Because the mobile robot made in this study does not use odometry data, so to get robot coordinates continuously ROS has a package, namely "/tf" (transform). Transform itself functions as a standard way to determine or track the frame coordinates of the mobile robot. In this study, the "/tf" used is map, base_link and scanmatcher frames. Frame map allows researchers to know the initial position of the robot. base_link is a frame coordinate that is on the robot body. base_link is used if the algorithm used does not use odometry data. While the scanmatcher frame is used to get the coordinates of the mobile robot continuously so that when the "/tf" type is merged, the frame and the coordinates of the mobile robot will appear on the map formed. Mapping and Localization results will be visualized in RViz through the /map feature. The /map feature here serves to implement the creation of a two-dimensional (2D) map based on data from "/tf" (package transform) and /scan (RPLidar A1M8 sensor).

2.8. Mechanical Design

Mobile Robot mechanics are made according to the needs of this research. The mechanical design of this mobile robot is made with Solidwork design software. The design of the mobile robot mechanical design is made to adjust the size of the laptop which is the main device that will become the system on the Mobile Robot. The design of the mobile robot frame design is shown in the figure 7.

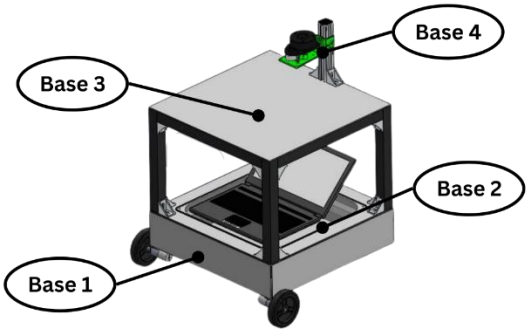


FIG 6. Mechanical Design of Mobile Robot

In this robot mechanic there are 4 bases. The first base, which is the bottom base, is where the mobile robot control system components such as dc motors, motor drivers, batteries and arduino microcontrollers are placed. The second base, is the base where the laptop is placed which is the main system. The third base, is the base used as the placement of goods to be carried by the mobile robot, while the top base, namely the fourth base, is occupied by the RPLIDAR A1 sensor. The placement of the LIDAR sensor must be free from obstacles that come from the robot itself. So that the placement of this lidar sensor is at the very top.

The framework of this mobile robot is made of iron, while the outer layer is covered with acrylic on the side of the lower base and 1.5 cm plywood on the upper side of the lower base and the middle base where the goods are placed. The robot consists of 2 12 V dc motors with a large torque of 12 kg equipped with wheels for each motor.

3. RESULTS AND DISCUSSION

3.1. RPLIDAR A1M8 Sensor Distance Reading Testing

a. Maximum Distance Reading

In this test to test the maximum distance that can be reached by the A1M8 RPLIDAR sensor. The following test results are obtained based on Table 2.

TABLE 2. Maximum Distance Readings

Maximum Distance Measurement			
No	**Distance (cm)**	**Distance Read (cm)**	**Error (%)**
0	0	0	0
1	50	51	1
2	100	103	3
3	150	152	2
4	200	205	5
5	250	255	5
6	300	304	4
7	350	353	3
8	400	404	4
9	450	456	6
10	500	505	5
11	550	556	6
12	600	603	3
13	650	655	5

14	700	704	4
15	750	754	4
16	800	805	5
17	850	856	6
18	900	906	6
19	950	957	7
20	1000	1006	6
21	1050	1055	5
22	1100	1107	7
23	1150	1157	7
24	1200	1208	8
25	1250	0	-
Average Error (%)			5,38

For the calculation of the sensor reading error formula against the object, you can use the following formula:

$$Error = \frac{Distance\ (cm) - Sensor\ Readings}{Actual\ Value\ (Measured\ Distance\ (cm))}\ x\ 100$$

b. Minimum Distance Reading

In this test to test the minimum distance that can be reached by the A1M8 RPLIDAR sensor. The following test results are obtained based on Table 3.

TABLE 3. Minimum distance reading

minimum distance reading			
No	Distance (cm)	**Distance Read (cm)**	**Error (%)**
1	0	0	-
2	15	16	1
3	25	27	2
4	35	36	1
5	45	46	1
6	55	56	1
7	65	67	2
8	75	77	2
9	85	87	2
10	95	97	2
11	105	107	2
Average Error (%)			1,45

For the calculation of the sensor reading error formula against the object, you can use the following formula:

$$Error = \frac{Distance\ (cm) - Sensor\ Readings}{Actual\ Value\ (Measured\ Distance\ (cm))}\ x\ 100$$

3.2. Lidar Measurement and Readout at Light Intensity

This test is carried out to ensure that the RPLIDAR A1M8 lidar sensor readings are not affected by room conditions with low light intensity. This test was carried out indoors with a distance of only 100 cm and only the front of the sensor was tested. In indoor testing with the lights on with a lux value of 100 lux, while the lights off the lux value of the lamp is 0 lux. This value is obtained from the conversion results in the lux meter application on the cellphone.

a. Indoor Lights On

TABLE 4. Lidar Sensor Readings with the Room Lights On (100 lux)

Indoor Lights On (100)			
No	**Jarak (cm)**	**Distance Read (cm)**	**Error (%)**
1	0	0	0,0
2	10	10,8	0,8
3	20	20,4	0,4
4	30	30,3	0,3
5	40	40,3	0,3
6	50	50,3	0,3
7	60	60,3	0,3
8	70	70,2	0,2
9	80	80,5	0,5
10	90	90,7	0,7
11	100	100,6	0,6
Average Error (%)			0,4

In Table 4, it can be seen the results of measuring the distance of the RPLIDAR A1M8 sensor to the object with the condition of the room lights on which is worth lux 100. From the tests carried out, obtained the highest error value of 0.8% and the lowest of 0.2%. Error calculations obtained from the results of each distance tested using the error formula, as follows:

$$Error = \frac{Distance\ (cm) - Sensor\ Readings}{Actual\ Value\ (Measured\ Distance\ (cm))}\ x\ 100$$

From the results of testing each distance carried out, the overall average error is 0.4%. By using the total error calculation, as follows:

$$Average = \frac{large\ number\ of\ errors}{lots\ of\ data}$$

$$Average = \frac{e1 + e2 + e3 + e4 + e5 + e6 + e7 + e8 + e9 + e10 + e11}{n}$$

$$Average = \frac{0{,}0 + 0{,}8 + 0{,}4 + 0{,}3 + 0{,}3 + 0{,}3 + 0{,}3 + 0{,}2 + 0{,}5 + 0{,}7 + 0{,}7}{11}$$

$$Average = 0{,}4\ \%$$

b. Indoor Lights On

TABLE 5. Lidar Sensor Readings with the Room Lights Out (0 lux)

Indoor Lights Out (0)			
No	**Distance (cm)**	**Distance Read (cm)**	**Error (%)**
1	0	0	0,0
2	10	10,8	0,8
3	20	20,4	0,4
4	30	30,3	0,3
5	40	40,3	0,3
6	50	50,3	0,3
7	60	60,3	0,3
8	70	70,2	0,2
9	80	80,5	0,5
10	90	90,7	0,7
11	100	100,6	0,6
Average Error (%)			0,4

In Table 5 can be seen the results of measuring the distance of the RPLIDAR sensor to the object with the condition of the paam lamp room which has a lux value of 0. From the tests carried out, the highest error value was obtained at 0.8% and the lowest was 0.2%. Error calculations obtained from the results of each distance tested using the error formula, as follows:

$$Error = \frac{Distance\ (cm) - Sensor\ Readings}{Actual\ Value\ (Measured\ Distance\ (cm))}\ x\ 100$$

From the results of testing each distance carried out, the overall average error is 0.4%. By using the total error calculation, as follows:

$$Average = \frac{e1 + e2 + e3 + e4 + e5 + e6 + e7 + e8 + e9 + e10 + e11}{n}$$

$$Average = \frac{0{,}0 + 0{,}8 + 0{,}4 + 0{,}3 + 0{,}3 + 0{,}3 + 0{,}3 + 0{,}2 + 0{,}5 + 0{,}7 + 0{,}7}{11}$$

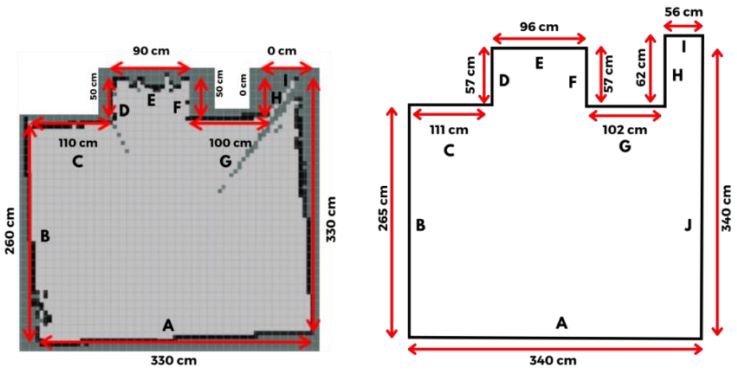
$$Average = 0{,}4\ \%$$

3.3. Testing Mapping Results

Testing the mapping results using hector SLAM compared to real world conditions, every 1 pixel of the mapping results is 100 cm. Testing was carried out in a room with a room size of 340 cm x 340 cm. but in the room there are still items in the form of cabinets.

a. Testing Mapping Results Without Tracking

This test is carried out by positioning the mobile robot in the middle of the room and mapping around the mobile robot without making the mobile robot move. The following mapping results are obtained:



(a) (b)

FIG 7. Testing Mapping Results without tracking (a) RViz Measurement Results (b) Actual Measurement Results

In Figure 7 is the measurement result obtained through the RViz display so that it can compare the measurement value from the one obtained with the original one can be seen from the following Table 6.

TABLE 6. testing results of mapping with tracking

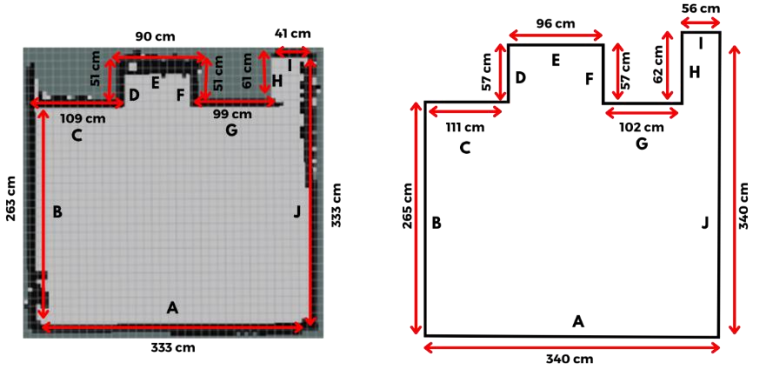
Testing Mapping Results Without Tracking			
wall position	**Actual wall length (cm)**	**Measured wall length (cm)**	**Error (%)**
A	340	330	2,94
B	265	260	1,89
C	111	110	0,90
D	57	50	12,28
E	96	90	6,25
F	57	50	12,28
G	102	100	1,96
H	62	0	100,00

I	56	0	100,00
J	340	330	2,94
	Average		**18,57**

In Table 6, it can be seen the results of the mapping carried out by the mobile robot in a state of silence or not moved (without tracking). From the tests carried out, the average error value is 18.57%. The error obtained is quite large, because the error value on the wall index H and wall index I gets a large error. This is because the sensor cannot reach the readings at the H wall position and the I wall position.

b. Testing Mapping Results with Tracking

This test is carried out by activating the mobile robot that tracks the left wall area. The following mapping results are obtained:



(a) (b)

FIG 8. Testing Mapping Results with tracking (a) RViz Measurement Results (b) Actual Measurement Results

TABLE 7. testing results of mapping with tracking

Testing Mapping Results Without Tracking			
wall position	**Actual wall length (cm)**	**Measured wall length (cm)**	**Error (%)**
A	340	333	2,06
B	265	263	0,75
C	111	109	1,80
D	57	51	10,53
E	96	90	6,25
F	57	51	10,53
G	102	99	2,94
H	62	61	1,61
I	56	41	26,79
J	340	333	2,06
	Average		**5,02**

In Table 7, we can see the results of the mapping done by the mobile robot when it is activated and tracking the room to be mapped. From the tests carried out, the average error value is 5.02%. During the mapping process with tracking the reading of the room can be done thoroughly, so as to get a smaller error value.

3.4. Straight Walk Mobile Robot Testing

Testing the speed of the mobile robot when walking straight to find out the speed of the robot with the load that can be carried by the mobile robot without hitting the wall. In this test, several load variations were carried out, namely no load, 5 kg load, 10 kg load, 15 kg load, and 20 kg load by experimenting 5 times with each load on the mobile robot. For the calculation formula for the speed of the mobile robot running straight, namely:

$$v = \frac{\Delta s}{\Delta t} = \frac{x_t - x_0}{t - t_0}$$

Description:

$v = Speed\ (m/s)$
$s = Track\ Length\ (m)$
$t = Time\ (t)$

From the results of the tests carried out, the average test of each load carried by the mobile robot can be seen from Table 8.

TABLE 8. average results of straight road test

No	Load Experiment	Distance (m)	Time (t)	Speed (m/s)
1	No burden	2	7,6	0,264
2	Load 5 Kg	2	8,46	0,237
3	Load 10 Kg	2	10,56	0,19
4	Load 15 Kg	2	12,14	0,165
5	Load 20 Kg	2	15,76	0,127
6	Load 21 Kg	2	∞	0

The following is a graphical display of the results of Table 8based on time and speed.

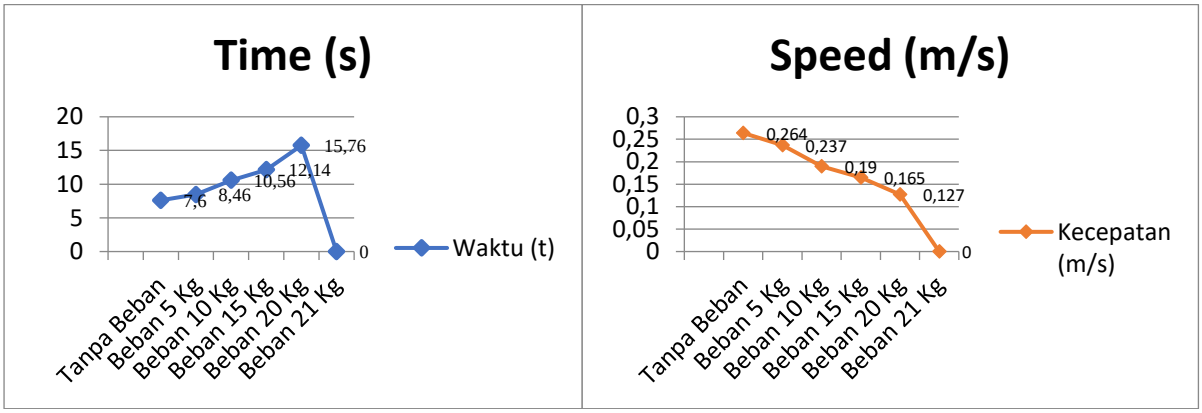


Figure 3. 7 line graph image of mobile robot travel time on a straight road

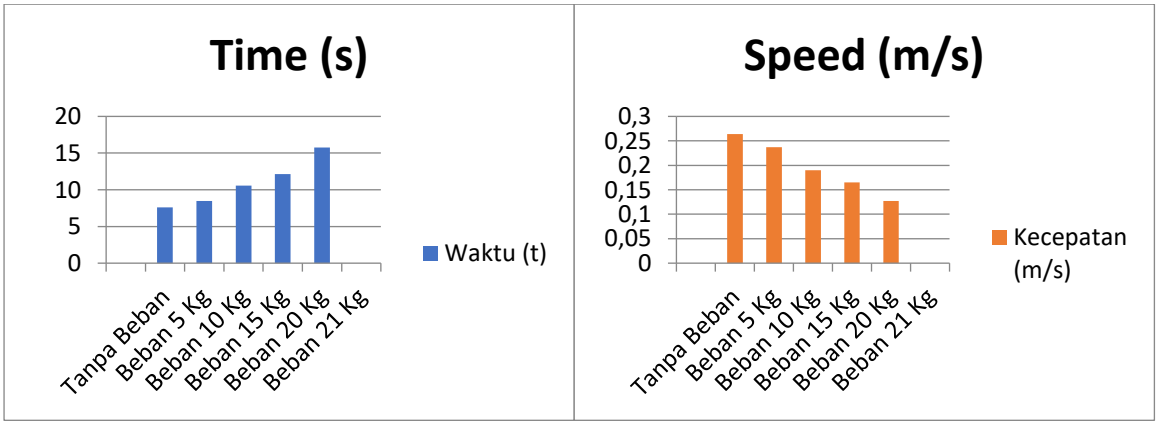


FIG 9. Bar graph of mobile robot travel time on a straight road

3.5. Testing of Mobile Robots Walking Turning

Testing the speed of the robot when turning 90° to ensure the navigation system is able to navigate properly and does not hit obstacles or objects in front of the mobile robot. This test was carried out to determine the value of the difference in mobile robot speed in a 90□ turning movement with different load variations. This test was carried out using several variations of load, namely no load, 5 kg load, 10 kg load, 15 kg load, and 20 kg load by

carrying out 5 trials for each load on the mobile robot. For the formula for calculating the speed of a mobile robot walking around, namely:

- look for frequency
  $f = \frac{n}{t}$
  information:
  $f = frekuensi\ (Hz)$
  $t = \ time\ (s)$
  $n = rotation\ angle\ (90°)$
- find the angular velocity
  $\omega = 2\pi f$
  Information:
  $\omega = the\ angular\ velocity (rad/s)$
  $\pi = phi\ (3{,}14)$
  $f = frekuensi\ (Hz)$
- find linear speed
  $v = \ \omega\ x\ r$
  Information:
  $v = linear\ speed (m/s)$
  $\omega = the\ angular\ velocity\ (rad/s)$
  $r = turning\ radius\ (m)$

From the results of the tests carried out, the average test of each load carried by the mobile robot can be seen from Table 9.

TABLE 9. average results of turning road test

No	load experiment	Radius (m)	Time (t)	Speed (m/s)
1	**0 Kg**	3	18,29	0,258
2	**5 Kg**	3	22,17	0,213
3	**10 Kg**	3	33,83	0,139
4	**15 Kg**	3	40,24	0,117
5	**20 Kg**	3	60,51	0,078
6	**21 Kg**	3	∞	0

The following is a graphical display of the results of Table 9 based on time and speed.

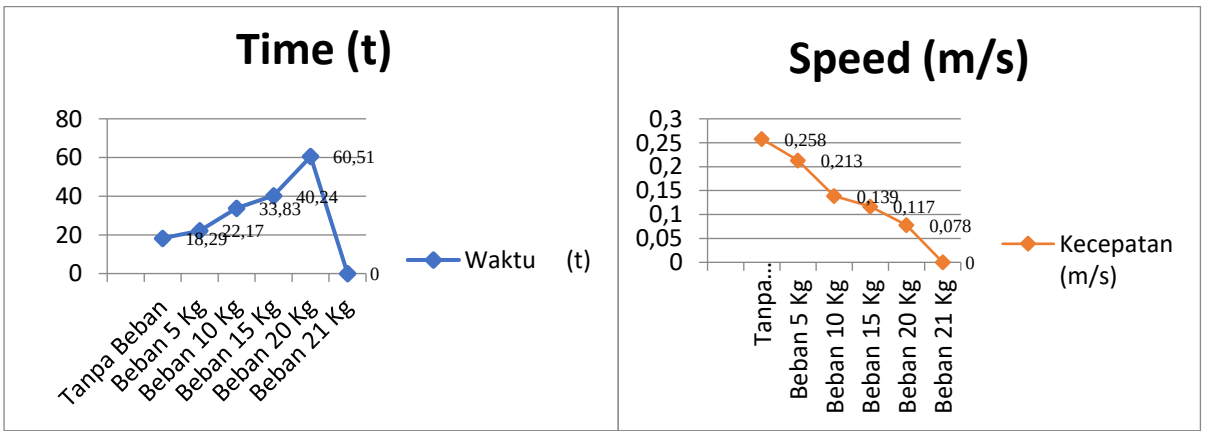


FIG 10. line graph image of the travel time of the mobile robot turning road

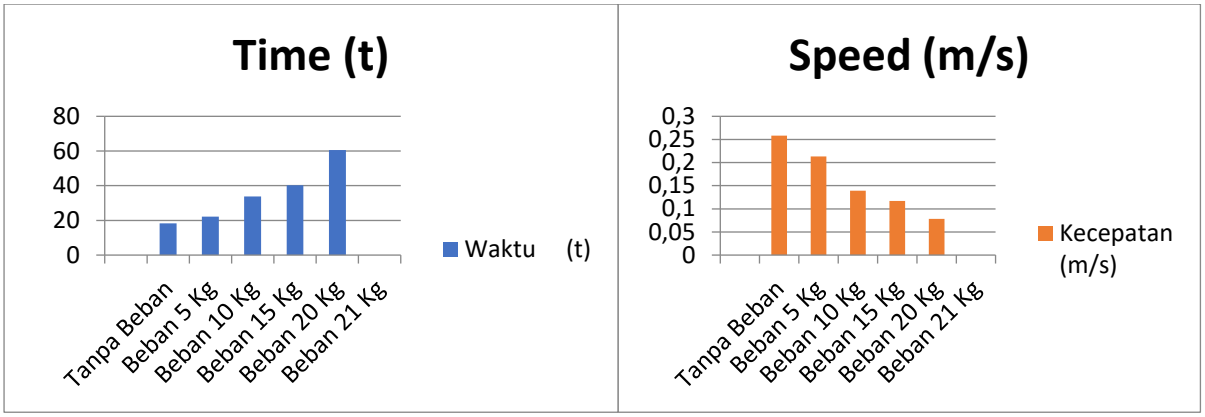


FIG 11. line graph image of mobile robot speed on turning road

4. CONCLUSION

Based on testing and data analysis, it can be concluded that: The RPLidar sensor is capable of reading a distance of 1200 cm with a reading error value of 3.15%, and a minimum reading distance of 15 cm with a reading error value of 1.45%. When reading light at 0 lux, the Lidar sensor remains stable, as proven in testing with a wall object, the error value is 0.4%. In mapping tests carried out in rooms without tracking, an average error of 18.57% was obtained, while testing with tracking obtained an average error of 5.02%. The heavier the load carried by the mobile robot, the smaller the speed of the mobile robot.The heavier the load carried by the mobile robot, the longer it will take for the mobile robot to move to the destination point.

REFERENCES

[1] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, "Lidar-based obstacle avoidance for the autonomous mobile robot," *Proc. 2019 Int. Conf. Inf. Commun. Technol. Syst. ICTS 2019*, pp. 197–202, 2019, doi: 10.1109/ICTS.2019.8850952.

[2] R. . P. Maulana I., A. Rusdinar, "Lidar Aplication for Mapping and Navigating on Clossed Environtment," *e-Proceeding Eng.*, vol. 5, pp. 1–8, 2018.

[3] & S. S. Z. L. , S. M. I., "Perancangan Dan Implementasi Mapping System Untuk Navigasi Roner (Robot Cleaner)," *E-Proceeding Appl. Sci.*, pp. 2092–2101, 2018.

[4] M. Rohfadli, "2021 5 th International Conference on Electrical , Telecommunication and Computer Engineering Gas Leak Inspection System Using Mobile Robot Equipped With LIDAR," 2021.

[5] D. Ghorpade, A. D. Thakare, and S. Doiphode, "Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR," *2017 Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2017*, pp. 1–6, 2017, doi: 10.1109/ICCUBEA.2017.8463846.

[6] Z. Liu, "Implementation o f SLAM and path planning for mobile robots under ROS framework," no. Icsp, 2021.

[7] M. S. and K. M. R. E., "THE IMPLEMENTATION OF HECTOR SLAM ON THE EARTHQUAKE VICTIMS FINDER ROBOT".

[8] K. Makita, D. Brscic, and T. Kanda, "Recognition of Human Characteristics Using Multiple Mobile Robots with 3D LiDARs," *2021 IEEE/SICE Int. Symp. Syst. Integr. SII 2021*, pp. 650–655, 2021, doi: 10.1109/IEEECONF49454.2021.9382640.

[9] S. Nagla, "2D Hector SLAM of Indoor Mobile Robot using 2D Lidar," *ICPECTS 2020 - IEEE 2nd Int. Conf. Power, Energy, Control Transm. Syst. Proc.*, pp. 18–21, 2020, doi: 10.1109/ICPECTS49113.2020.9336995.

[10] Q. M., "ROS: an open-source Robot Operating System. In Open-source software," *ICRA*, 2009.

[11] Z. Ma, L. Zhu, P. Wang, and Y. Zhao, "ROS-Based Multi-Robot System Simulator," pp. 4228–4232, 2019.