JURNAL INFOTEL
Informatics - Telecommunication - Electronics
Website: http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN: 2085-3688; e-ISSN: 2460-0997

Room cleaning robot movement using A* algorithm and imperfect maze

Vera Suryani[1,*], Kinkin Agustriana[2], Andrian Rakhmatsyah[3], Riska Reza Pahlevi[4]
[1,2,3,4]School of Computing, Telkom University
[1,2,3,4]Jl. Telekomunikasi, No. 1, Bandung 40257, Indonesia
[*]Corresponding email: verasuryani@telkomuniversity.ac.id

Received 17 January 2023, Revised 28 February 2023, Accepted 1 March 2023

Abstract — The room-circling motion of the cleaning robot must be meticulously calibrated to adopt the most efficient and effective path achievable. During room tracking, the robot's movement should reach all locations without obstruction and return to its starting point. It is essential to take into consideration the shortest path when conducting an exploration of the entire room to achieve maximum efficiency. The shortest path for room exploration might not always be considered by robots. This research simulated the movement of a room explorer robot using the imperfect maze method and searched a room that had not been explored using the A* algorithm. Since the robot must return to its initial location after finishing the cleaning process, an imperfect maze is chosen. Loops are possible in imperfect mazes because dead ends can be "removed" by connecting them to a nearby path. The A* algorithm was also utilized to discover the shortest route for the robot to return to its original position after the room exploration. The performance of the robot's mobility is determined by the parameters of optimal room exploration distance and response time. This research simulates the development of a robot prototype and its mobile control application using Android. The simulation results revealed that the imperfect maze could be applied to explore the room effectively, and A* algorithm is relatively optimal for exploring both the undiscovered room and the path to return to its original location.

Keywords – A* algorithm, imperfect maze, movement, room exploration

I. Introduction

As seen by the rising number of persons exposed to the virus, the transmission of today's hazardous diseases is incredibly quick. There are many measures to prevent the transmission of these viruses, including maintaining a clean environment. Room sterilization, which can be accomplished by spraying disinfectant or emitting Ultraviolet (UV) rays, is one method for maintaining cleanliness. The infection might still be transmitted to the cleaners if the sterilizing process is not performed correctly. To decrease the risk of the virus spreading to people, an automated sterilizing procedure with robots may be an alternate method.

Room cleaning robots are practical and cost-efficient. They are considerably less expensive than costly housekeepers. The machine can be programmed to clean precisely to specifications while requiring minimal maintenance. The robot employed to do the sterilization must be able to comb all rooms for optimal results. In doing the task, the robot can move automatically, controlled by the system control through the remote control or self-manage [1]–[4]. This process aims to make human tasks easier and to be highly useful in the cleaning process in the house [5]–[8]. To make the robot able to move and walk automatically, it needs a method to control the speed and direction of the cleaning. In this case, the imperfect maze method is used to determine the speed and direction determined as needed [9]–[11]. The imperfect maze method is one of the maze methods in which the path traversed is not unique but something highly possible for an intersection. In its journey, the robot prototype must go through all the paths and calculate the shortest distance to return to its initial place using the A* algorithm.

A study by Ullah *et al.* [10] proposed the Nearest-to-Final Goal and Unvisited Gap (FNUG), a novel gap-based strategy for traversal imperfect-unknown mazes. The technique is computationally efficient and can handle loops and dead ends thanks to the revisit

check. This technique primarily focuses on finding gaps around the robot by analyzing the depth scans and building a topological map in the shape of a tree using the coordinate system. The robot's unexplored neighbor gap is then chosen for route planning, relying on its Euclidean distance from the final destination. Lastly, Dijkstra's method determines a path from the robot's location to the sub-goal.

Numerous studies examine the usage of mazes, but few employ imperfect mazes. Some researchers [17]–[21] have utilized mazes for robot mobility. The research utilizing the maze will be described below.

In research conducted by Ecsedi *et at.* [12] entitled "The Development of an Autonomous Maze Robot" in 2019, a robot that could pass through a simple labyrinth by keeping on the path, turning, and finding the way out from the labyrinth was made. The results showed that the robot could pass through each classical labyrinth without touching the wall.

A study by Hermanto *et al.* [13] depicted that the prototype showed that the A* algorithm could be used in a hexapod robot to get the fastest path to the destination. The average speed of the hexapod robot to walk from the initial point at 10,27 to the last point (20,5). The results of the first average were taken from the data without any interruption (30,92s); using the interruption, the average speed obtained was 34,59s.

In [14], the A* algorithm was used for the Non-Playable Character (NPC) Ghost, a gaming character who relentlessly pursues Pacman around the maze until Pacman loses his life and the game ends. The A* algorithm is implemented so that the ghost can locate the quickest route to Pac-Man, increasing the game's intensity. From the research findings, it can be stated that the implementation of the A* algorithm was effective 91.5% of the time but still requires refinement to achieve greater accuracy. This conclusion showed the ability of the A* algorithm to find the solution in minimum time compared to any existing graphic-seeking algorithm [15].

In this research, the A* algorithm is selected and combined with the imperfect maze mapping method for the room mapping consisting of rows and columns. The shortest route selection by the robot detected four directions (front, back, left, and right), and comparison and selection of the shortest values were prioritized in the horizontal axis (node x). In selecting the shortest path of the A* Algorithm, it applied the heuristic function in which this algorithm removed any unnecessary steps by considering that the removed steps were the ones that would never reach the expected solution. In other words, a heuristic is the optimization function that makes the A* algorithm better than any other. Moreover, heuristics may be enhanced to produce better outcomes [16].

This study aimed to make a simulation of the robot prototype for the floor cleaning process that can infiltrate human drudgery. The imperfect maze was used to help the robot prototype seek the path of room exploration, while the A* algorithm was used to find the path to return to the initial place.

The presentation of this paper starts with the related studies and the design of the system built, evaluation, and the last part consists of a conclusion and recommendation related to the research results.

II. Research Method

The research method section of our study includes several subsections, including the imperfect maze, A* algorithm, system design, and hardware block diagram.

A. Imperfect Maze

Maze is a route through which one has to find a way. The robot could use the maze to find the route it needs to go. The path passed from the initial point to the last point in the perfect maze is always unique or cannot be similar.

An imperfect maze uses paths that are not always unique; it can intersect by itself, making it possible for the occurrence of the loop, as seen in Fig. 1.

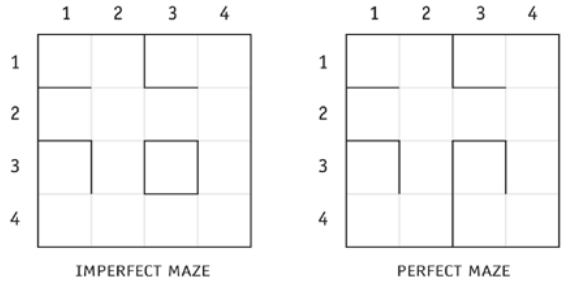

Fig. 1. Perfect and imperfect maze.

B. A Algorithm*

A* algorithm is one of the algorithms of path finding as the combination of Uniform Cost Search and Greedy Best First Search. Uniform Cost Search selects the smallest distance from the initial knot to the next until reaching the destination; meanwhile, Greedy Best First Search uses the heuristic function, where this function plays an essential role in controlling the search of the path in A* algorithm [13]. Several research utilize the A* algorithm extensively to discover the shortest pat [22]–[25]. Mathematically, the function used in A* algorithm can be written as the following equation:

$$f(n) = g(n) + f(n) \quad (1)$$

$$g(n) = \sqrt{Xn{+}Yn^2} \quad (2)$$

$$h(n) = |X(\text{target}) - X(n)| + |Y(\text{target}) - Y(n)| \quad (3)$$

where, $f(n)$ is the lowest estimated value, $g(n)$ is the value from starting node to n^{th} node, and $h(n)$

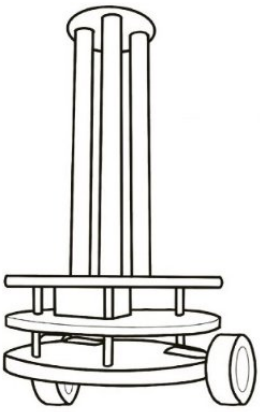

Fig. 2. Robot prototype.

is the heuristic function used to estimate the distance from node n to the destination location.

C. System Design

The design of the system mechanism consists of hardware, software, dan brain ware.

1) Hardware

As the microcontroller in this system, Arduino is in NodeMCU to connect to the software on the user's smartphone. When the robot obtains the input value, it can walk according to the column and lines. For the robot's movement and its hindrance, they are detected by IR Sensor.

2) Software

The design of this software is to manage the values of the room size on the software determined. In addition, the software can also be used to control the robot's speed manually.

3) Brain ware

Users can control the robot's movement and insert the room size available in the robot application. The system of the robot prototype built has the design as seen in Fig. 2.

In Fig. 3, there are three infrared sensors (front, left, and right) to detect any interruptions during the search. NodeMCU is the connector of the software to the user's smartphone. Relay as the dc motor and the vacuum cleaner fan. The encoder controls the robot's speed distance sensor, while the battery controls the robot's power.

D. Hardware Block Diagram

The diagram block in Fig. 4 explains the relation of each hardware to control the robot. Microcontroller, such as the Arduino Mega 2560, is used to process input and produce output values. The system input comes from the infrared sensor to detect the interruption in the form of high and low digital values, sensor gyroscope mpu6050 as the determiner of the direction of the robot speed, and nodeMCU esp8266 as the Speed signaller when the robot is manually controlled. The process output results on the microcontroller in the form of pwm values, to move the left and right servo motor and high and low digital values to activate the vacuum cleaner fan.

III. Result

In the results section of our study, we present the findings of several subsections, including the testing results of the Infrared (IR) obstacle avoidance sensor, the result of the test on the LM393 speed sensor, the result of the test on software functionality, and the result of the test on the A* algorithm.

A. Testing Result of IR Obstacle Avoidance Sensor

Based on the data from the test results, as shown in Table 1, the infrared sensor used was to get accurate test values. This result was based upon the data of the test that later on were compared to real size using rulers, showing the good results from sensor 1, sensor 2, and sensor 3.

Table 1. Infrared Sensor Reading

No.	Score of Infrared Sensor Reading			Actual distance
	(1)	(2)	(3)	
1	1 cm	1 cm	1 cm	1 cm
2	2 cm	2 cm	2 cm	2 cm
3	3 cm	3 cm	3 cm	3 cm
4	4 cm	4 cm	4 cm	4 cm
5	5 cm	5 cm	5 cm	5 cm

B. The Result of the Test on the LM393 Speed Sensor

The accuracy of the distance with the input was discovered in eight experiments, according to the results of the test on the LM393 sensor in Table 2 from 10 times of testing. While, the remains obtained inappropriate results. These findings were because the influence of the number of pulse encoders required in experiments 4 and 6 was not appropriate with the number of pulses required. From the calculation results in the equation to seek the number of pulse encoders in Chapter 3, the number of pulses in experiment 4 exceeded the required number of pulses, *i.e.*, 11. In contrast, in experiment 6, the pulse number was fewer than the number of pulses required. This result caused the distance of the robot's mileage was not as appropriate as it should be.

Table 2. Infrared Sensor Reading

Number of Attempt	Distance measurement results (cm)	Expected Distance	Distance Accuracy	Encoder Reading Data
1	30 cm	30 cm	Ok	11 pulse
2	30 cm	30 cm	Ok	11 pulse
3	30 cm	30 cm	Ok	11 pulse
4	31 cm	30 cm	Not ok	12 pulse
5	30 cm	30 cm	Ok	11 pulse
6	29 cm	30 cm	Not ok	10 pulse
7	30 cm	30 cm	Ok	11 pulse
8	30 cm	30 cm	Ok	11 pulse
9	30 cm	30 cm	Ok	11 pulse
10	30 cm	30 m	Ok	11 pulse

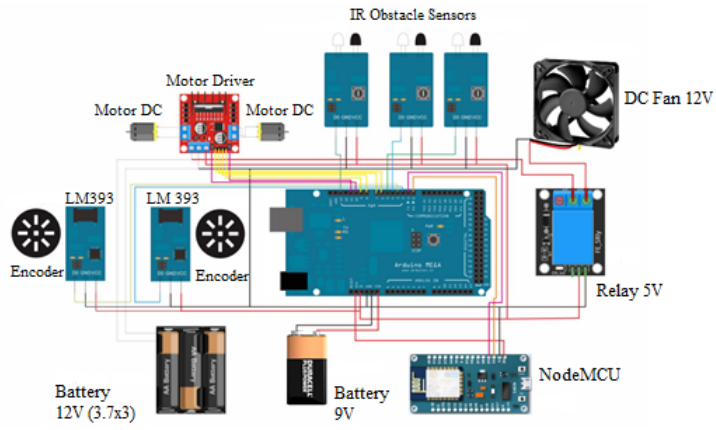

Fig. 3. Hardware components of the robot.

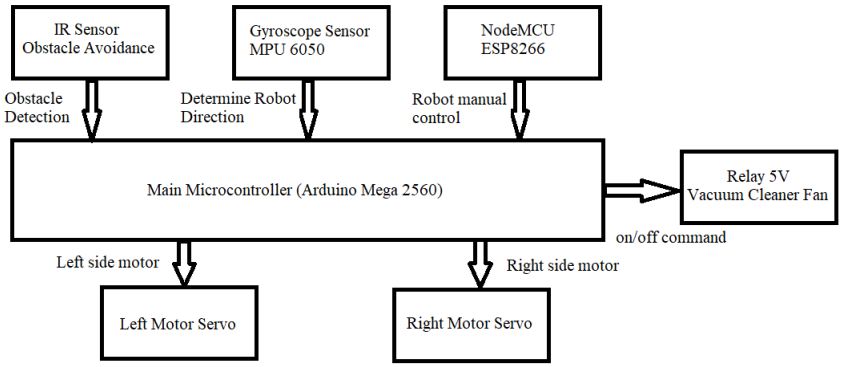

Fig. 4. Hardware block diagram.

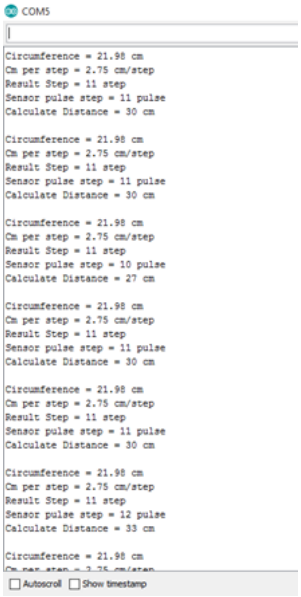

Fig. 5. Encoder result.

Table 3. The Results of the Test on the Functionality of System Software

No	Command input	Expected condition	Real Condition	Suitability of Conditions
1	Press the top arrow button	Robot did the action to move forward	Robot did the action to move forward in a stable and straight condition	Proper
2	Press the bottom arrow button	Robot did the action to move back in a straight line	Robot did the action to move back with a stable and straight condition	Proper
3	Press the right arrow button	Robot did the action to turn around clockwise	Robot did the action to turn around clockwise	Proper
4	Press the left arrow button	Robot did the action counter clockwise	Robot did the action counter action	Proper
5	Press the automatic button	Robot moved to the automatic mode	Robot moved to the automatic mode and did the room searching	Proper

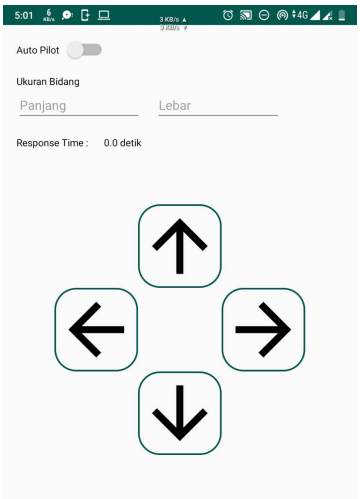

Fig. 6. Application interface.

C. *Result of the Test on Software Functionality*

From the test results on the functionality in Table 3, the robot could do the basic functionalities such as moving forward and back, turning right and left, and changing the mode of the robot into the automatic mode. Fig. 6 depicted the application interface for the simulation.

The test on the response time was done to ob-

serve how long the time took when the users gave the command for the robot functionality through the smartphone. Response time was calculated when the smartphone sent the request for the command message to the device of nodeMCU. The Arduino Mega was then asked to process the command message request into a robot action, and nodeMCU delivered a confirmation message to the smartphone informing the user that the operation had been completed.

The test was done by managing the distance between the robot and smartphone in the range of 1 to 7 meters. Then, the messages were input by 30 times to obtain the average values of response time in each distance. The commands were input randomly based upon the actions of the robot such as moving forwards, backwards, left and right. Based on the results of the test as seen in the graph in Fig. 7, the distance between the robot and smartphone affected the values of the response time obtained.

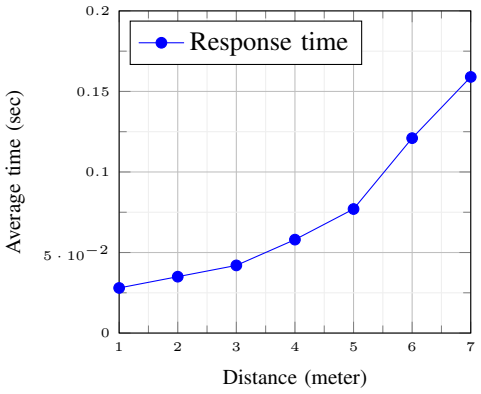

Fig. 7. Application response time.

This outcome arose because the response time values were precisely related to the test distance. Fig. 7 illustrates that as the distance between robot and smartphone increases, so does the response time. In contrast, the response time decreases as the distance between the robot and smartphone decreases.

This limitation is a result of the network technology used to transmit data from the from the smartphone to the robot.

D. Result of the Test on A Algorithm*

Nodes 1 and 2 were used to evaluate the robot's ability to explore varied locations, particularly 6x5 m and 6x6 m. The first node at coordinate (2,6) had the F-value of 12,3.

$$g(n) = \sqrt{Xn^2 + Yn^2} = \sqrt{2^2 + 6^2} = 6.3 \quad (4)$$
$$h(n) = |1-2| + |1-6| = 6 \quad (5)$$
$$f(n) = 6.3 + 6 = 12.3 \quad (6)$$

The second node at coordinate (1,5) had the F-value of 9,1.

$$g(n) = \sqrt{Xn^2 + Yn^2} = \sqrt{1^2 + 5^2} = 5.1 \quad (7)$$
$$h(n) = |1-1| + |1-5| = 4 \quad (8)$$
$$f(n) = 5.1 + 4 = 9.3 \quad (9)$$

From the results (6) and (9), it can be found that the second node had the lowest F-value; thus, the position shifted to the second node. After moving, all new nodes around Tnow were calculated to obtain the lowest F-value as the path passed through Tnow to the initial point. It can be found that the traveling route from the last point to the destination expected was through the following coordinates (1,5), (1,4), (1,3), (1,2), and coordinate (1,1) as the coordinate of the initial storage of the robot. The processes for finding pathways in an imperfect maze using the A* algorithm are explained in Tables 4 and 5.

Table 4. Calculation Results of Algorithm A* for Size 6x6

6	(1,6) LAST POINT	(2,6) G = 6,3 H = 6 F = 12,3	(3,6)	(4,6)	(5,6)	(6,6)
5	(1,5) G = 5,1 H = 4 F = 9,1 Tnow	(2,5) G = 5,4 H = 5 F = 10,4	(3,5)	(4,5)	(5,5)	(6,5)
4	(1,4) G = 4,1 H = 4 F = 4,1 Tnow2	(2,4) G = 4,5 H = 4 F = 8,5	(3,4)	(4,4)	(5,4)	(6,4)
3	(1,3) G = 3,2 H = 2 F = 5,2 Tnow3	(2,3) G = 3,6 H = 3 F = 6,6	(3,3)	(4,3)	(5,3)	(6,3)
2	(1,2) G = 2,2 H = 1 F = 3,2 Tnow4	(2,2) G = 2,8 H = 2 F = 4,8	(3,2)	(4,2)	(5,2)	(6,2)
1	(1,1) G = 1,4 H = 0 F = 1,2 INITIAL POINT	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)
X/Y	1	2	3	4	5	6

First node at coordinate (5,5) had the F-value of 15,1.

$$g(n) = \sqrt{Xn^2 + Yn^2} = \sqrt{5^2 + 5^2} = 7.1 \quad (10)$$
$$h(n) = |1-5| + |1-5| = 8 \quad (11)$$
$$f(n) = 7.1 + 8 = 15.1 \quad (12)$$

Second node at coordinate (6,4) had the F-value of 15,2.

Table 5. Calculation Results of Algorithm A* for Size 6x5

5	(1,5)	(2,5) G=5,4 H = 5 F=10,4	(3,5) G=5,8 H=5 F=10,8 Tnow2	(4,5) G=6,4 H = 7 F=13,4 Tnow1	(5,5) G=7,1 H = 8 F=15,1 Tnow	(6,5) LAST POINT
4	(1,4)	(2,4) G = 4,8 H = 4 F= 8,8	(3,4) G = 5 H = 5 F = 10 Tnow3	(4,4) G= 5,7 H = 6 F=11,7	(5,4) G=6,4 H = 7 F=13,4	(6,4) G= 7,2 H = 8 F=15,2
3	(1,3) G=3,2 H = 2 F= 5,2	(2,3) G= 3,6 H = 3 F= 6,5 Tnow5	(3,3) G=2,2 H = 4 F=6,2 Tnow4	(4,3)	(5,3)	(6,3)
2	(1,2) G=2,2 H = 1 F= 3,2 Tnow7	(2,2) G= 2,8 H = 2 F= 4,8 Tnow6	(3,2) G= 3,6 H = 3 F= 6,6	(4,2)	(5,2)	(6,2)
1	(1,1) INITIAL POINT	(2,1) G=2,2 H = 1 F = 3,2	(3,1)	(4,1)	(5,1)	(6,1)
X/Y	1	2	3	4	5	6

$$g(n) = \sqrt{Xn^2 + Yn^2} = \sqrt{6^2 + 4^2} = 7.2 \quad (13)$$

$$h(n) = |1 - 6| + |1 - 4| = 8 \quad (14)$$

$$f(n) = 7.2 + 8 = 15.2 \quad (15)$$

From the results (12) and (15), it can be inferred that the first node had the lowest F-value in the room with the size of 6x5 meters; thus, the position was shifted to the first node. After moving, all new nodes around Tnow were calculated to get the lowest path passed by the Tnow to the initial point. It can be found that the travel route from the last point to the destination expected was through the following coordinates (5,5), (4,5), (3,5), (3,4), (3,3), (2,3), (2,2), and coordinate (1,2) as the coordinate of the initial storing place of the robot.

IV. Conclusion

From the results of this research, it can be concluded that the robot prototype successfully works automatically. It followed the rules of the ordered imperfect maze and the direction was determined. The robot can also perform floor cleaning and pass through all floor blocks from the initial points to the last points. The robot can also explore the room while illuminating using a UV light for sterilization. The system's performance from the response time parameter showed that the optimal distance between the robot and the smartphone was around 1 to 7 meters. The farther the distance between the robot and the smartphone, the larger the response time values. This research can be improved using hardware with higher specifications for better results. The use of other search algorithms or improving the A* algorithm can also be used in determining the fastest path. The results can be compared to determine the most optimal algorithms.

References

[1] M. B. Alatise, G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, 2020.

[2] B. K. Patle, G. Babu L, A. Pandey, D. R. K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot," *Def. Technol.*, vol. 15, no. 4, pp. 582—606, 2019.

[3] S. Erfani, A. Jafari, and A. Hajiahmad, "Comparison of two data fusion methods for localization of wheeled mobile robot in farm conditions," *Artif. Intell. Agric.*, vol. 1, pp. 48—55, 2019.

[4] F. H. Ajeil, I. K. Ibraheem, A. T. Azar, and A. J. Humaidi, "Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments," *Sensor Journal*, vol. 20, no. 7, 2020.

[5] M. A. H. Ali and M. Mailah, "Path planning and control of mobile robot in road environments using sensor fusion and active force control," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2176—2195, 2019.

[6] M. A. Viraj J. Muthugala, M. Vega-Heredia, R. E. Mohan, S. R. Vishaal, "Design and control of a wall cleaning robot with adhesion-awareness," *Symmetry Journal*, vol. 12, no. 1, 2020.

[7] A. Z. Hasibuan, and M. S. Asih, "Rancang bangun robot vacum cleaner berbasis mikrokontroler dengan pengendali smartphone android," *InfoTekJar Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 1, 2019.

[8] V. Oza and P. Mehta, "Arduino robotic hand: Survey paper," in *2018 Int. Conf. Smart City Emerg. Technol. ICSCET 2018*, pp. 1—5, 2018.

[9] M. Ihsan, D. Suhaimi, M. Ramli, S. M. Yuni, and I. Maulidi, "Non-perfect maze generation using Kruskal algorithm," *J. Nat.*, vol. 21, no. 1, 2021.

[10] Z. Ullah , X. Chen , S. Gou, Y. Xu, and M. Salam, "FNUG: Imperfect mazes traversal based on detecting and following the nearest-to-final-goal and unvisited gaps," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, 2022.

[11] L. Peachey, "Parameterized maze generation algorithm for specific difficulty maze generation," *Association for Computing Machinery*, vol. 1, no. 1, 2022.

[12] E. Ecsedi, H. Silaghi, E. Mihok, and V. Spoiala, "The development of an autonomous maze robot," in *Int. Conf. Eng. Mod. Electr. Syst.*, pp. 169—172, 2019.

[13] D. Hermanto and S. Dermawan, "Penerapan algoritma A-Star sebagai pencari rute terpendek pada robot hexapod," *J. Nas. Tek. Elektro*, vol. 7, no. 2, pp. 122–129, 2018.

[14] F. Badri, M. F. A. Habib, "Implementasi algoritma A* (A Star) pada NPC (non-playable character) game pacman menggunakan game engine unity 5 berbasis android," *Teknika Journal*, vol. 4, no. 2, pp. 49–56, 2020.

[15] N. Kumar and S. Kaur, "A review of various maze solving algorithms based on graph theory," *IJSRD - Int. J. Sci. Res. Dev.*, vol. 6, no. 12, pp. 2-–6, 2019.

[16] Y. Li, D. Dong, X. Guo, "Mobile robot path planning based on improved genetic algorithm with A-star heuristic method," in *IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2020.

[17] H. Batti, C. B. Jabeur, H. Seddik, "Autonomous smart robot for path predicting and finding in maze based on fuzzy and neuro-Fuzzy approaches," *Asian Journal of Control*, pp. 1–10, 2020.

[18] S. Paul, C. B. C. Latha, "Shortest path traversal in a maze with wall following robot," *AIP Conference Proceedings*, 2022.

[19] X. Zhang, Y. Liu, D. Hu, L. Liu, "A maze robot autonomous navigation method based on curiosity and reinforcement learning," in *The 7th International Workshop on Advanced Computational Intelligence and Intelligent Informatics (IWACIII2021), Beijing, China*, 2021.

[20] M. Nadour, L. Cherroun, "Using flood-fill algorithms for an autonomous mobile robot maze navigation," *International Journal of System Assurance Engineering and Management*, vol. 13, pp. 546-–555, 2022.

[21] R. Covaci, G. Harja, I. Nascu, "Autonomous maze solving robot," in *IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR*, 2020.

[22] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of A* pathfinding," in *Procedia Comput. Sci.*, vol. 179, no. 2020, pp. 507-–514, 2021.

[23] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved A-Star based path planning algorithm for autonomous land vehicles," *Int. J. Adv. Robot. Syst.*, vol. 17, no. 5, pp. 1—13, 2020.

[24] O. O. Martins, A. A. Adekunle, O. M. Olaniyan, and B. O. Bolaji, "An Improved multi-objective a-star algorithm for path planning in a large workspace: Design, implementation, and evaluation," *Sci. African*, vol. 15, 2022.

[25] Y. Li, D. Dong, and X. Guo, "Mobile robot path planning based on improved genetic algorithm with A-star heuristic method," in *Information Technology and Artificial Intelligence Conference, 2020*, vol. 9, pp. 1306—1311, 2020.