Implementasi Perencanaan Jalur Menggunakan Algoritma Dijkstra pada Robot Roda Mecanum

Senanjung Prayoga, S.Pd.,M.T[1], Elmaria Ompu Sunggu[2]

[12]Politeknik Negeri Batam,
Jurusan Teknik Elektro, Program Studi Teknik Robotika,
Jl Ahmad Yani Batam Kota, Kota Batam, Kepulauan Riau 29461

Email: senanjung@polibatam.ac.id, lmatiana5711@gmail.com

***Abstrak*— Dengan kemajuan teknologi, banyak pekerjaan manusia yang berisiko digantikan oleh robot. Robot harus dapat bermanuver secara mandiri tanpa kendali manusia. Penelitian ini mengembangkan robot yang mampu bergerak dari titik awal ke tujuan serta menghindari tabrakan menggunakan algoritma Dijkstra, yang efektif dalam mencari jalur terpendek. Algoritma ini diimplementasikan pada robot mecanum yang dilengkapi dengan mikrokontroler Arduino, mini PC, baterai, motor DC dengan encoder, driver motor, dan perangkat lunak terkait. Penelitian ini menguji lima peta dengan titik awal dan tujuan yang sama, namun dengan letak halangan yang berbeda. Hasil pengujian menunjukkan bahwa robot mecanum berhasil menemukan jalur dan menghindari halangan dengan baik. Dengan kecepatan maksimal rata-rata 2,6 m/s, robot dapat menempuh jarak 3,2 meter tanpa halangan dalam waktu 14,96 detik, serta jarak 4,2 meter dengan halangan dalam waktu 22,82 detik. Penelitian ini menunjukkan potensi algoritma Dijkstra dalam perencanaan jalur dan navigasi robot, serta menggarisbawahi pentingnya penggunaan perangkat keras dan perangkat lunak yang tepat untuk mencapai kinerja optimal. Hasil ini dapat menjadi dasar bagi pengembangan lebih lanjut dalam bidang robotika, khususnya dalam aplikasi robot otonom di lingkungan yang dinamis.**

Kata Kunci : Robot Mecanum, Implementasi Algoritma Dijkstra, Perencanaan Jalur

I. PENDAHULUAN

Dunia bergerak menuju teknologi otomatis di mana robot menggantikan manusia di pekerjaan berisiko tinggi. Kecerdasan buatan dan teknik komputasi lunak memainkan peran penting dalam meniru aktivitas manusia, memungkinkan robot digunakan di berbagai sektor seperti industri, militer, medis, logistik, pertanian, dan transportasi. Pada tahun 2022, terdapat 422.000 unit robot di sektor industri, dengan pertumbuhan 12% per tahun. Robot harus mampu bergerak mandiri, menjadikan perencanaan jalur dan penghindaran rintangan sebagai aspek penting dalam robotika. Perencanaan jalur memungkinkan robot mencapai tujuan tanpa bertabrakan dengan rintangan, bahkan dalam lingkungan kompleks, dan berusaha untuk membuat jalur seoptimal mungkin [1].

Untuk memungkinkan robot menggantikan peran manusia secara efektif, navigasi yang baik merupakan hal yang krusial. Perencanaan jalur menjadi aspek kunci dalam penelitian teknologi robot, yang menjamin robot dapat menyelesaikan tugas yang diberikan dengan spesifik [2].

Navigasi robot seluler melibatkan empat langkah utama: persepsi, lokalisasi, kognisi dan perencanaan jalur, serta kontrol gerak. Pertama, robot mengumpulkan informasi lingkungan melalui sensor. Kedua, robot menentukan posisi dan orientasinya relatif terhadap lingkungan. Ketiga, robot merencanakan jalur optimal menuju tujuan. Terakhir, robot mengendalikan pergerakannya untuk mengikuti jalur yang telah direncanakan [3].

Perencanaan jalur hanya dapat dilakukan dengan baik jika robot memiliki informasi lengkap tentang peta lingkungan sekitarnya. Oleh karena itu, robot harus memiliki kemampuan untuk melakukan lokalisasi (mengetahui posisinya sendiri dalam peta) dan pemetaan (membuat peta lingkungan sekitarnya) secara simultan. Dengan demikian, robot dapat merencanakan jalur yang tepat dan aman untuk bergerak di lingkungan yang kompleks [4].

Tujuan utamanya adalah memastikan jalur yang pendek, biaya rendah, dan waktu perjalanan singkat dari posisi awal ke target yang telah ditentukan. Selain itu, juga bertujuan untuk menemukan jalur yang aman, optimal, dan bebas tabrakan [5].

Salah satu metode yang sering digunakan untuk menyelesaikan masalah perencanaan jalur adalah menggunakan algoritma graf. Beberapa algoritma graf yang populer adalah algoritma Dijkstra dan algoritma Floyd-Warshall. Algoritma Dijkstra adalah salah satu algoritma perencanaan jalur yang pertama kali diusulkan. Algoritma ini cocok untuk menyelesaikan masalah jalur terpendek dari satu titik ke titik lainnya. Algoritma ini bekerja dengan memperluas pencarian dari titik awal ke lapisan-lapisan luar hingga mencapai titik akhir untuk menemukan jalur terpendek [6].

Algoritma Dijkstra adalah salah satu algoritma yang

digunakan dalam teori graf untuk mencari jalur terpendek antara dua simpul dalam sebuah graf yang memiliki bobot positif. Algoritma ini bekerja dengan menghitung bobot terkecil dari simpul awal ke simpul tujuan, sehingga menemukan jalur yang paling efisien berdasarkan bobot yang diberikan [7].

Masalah jalur terpendek mengacu pada proses mencari jalur terpendek dalam sebuah graf dengan cara menghitung jumlah bobot antar simpul. Dalam teori graf, setiap simpul dihubungkan oleh tepi yang memiliki bobot tertentu, dan tujuan dari masalah ini adalah menemukan jalur yang memiliki jumlah bobot terkecil dari titik awal ke titik tujuan [8].

Fitur utama dari algoritma ini adalah memulai pencarian dari titik awal, kemudian memperluas pencarian secara bertahap ke lapisan luar. Algoritma ini menemukan titik berikutnya yang paling dekat dengan target dan terus melakukan iterasi lapis demi lapis hingga mencapai titik target. Hal ini bertujuan untuk meningkatkan efisiensi dan akurasi dalam perencanaan jalur [9].

Robot bergerak segala arah menggunakan roda mecanum dan algoritma Dijkstra untuk perencanaan jalur. Robot ini dapat bergerak ke semua arah, memungkinkan penggunaan di industri, kedokteran, dan kompetisi robot [10].

Berdasarkan hasil penelitian dan pengujian di beberapa lingkungan, algoritma Dijkstra terbukti lebih efisien dan praktis. Algoritma ini dapat memberikan rute terpendek dengan cepat pada peta yang sudah diketahui letak rintangan dan rutenya, yang datanya diperoleh dari SLAM. Penelitian ini bertujuan untuk membuktikan bahwa algoritma ini dapat memecahkan masalah perencanaan jalur, sehingga robot mecanum dapat berjalan dengan baik dan menghindari rintangan saat bergerak.

Jurnal ini disusun sebagai berikut: Bagian II menyajikan landasan teori komponen utama robot pseudocode algoritma dijkstra beserta pengertian dari algoritma tersebut, Bagian III menyajikan data analisis dan hasil dari implementasi algoritma dijkstra pada robot dan Bagian IV menyajikan kesimpulan dari isi seluruh hasil penelitian pada jurnal ini beserta referensi yang diterapkan.

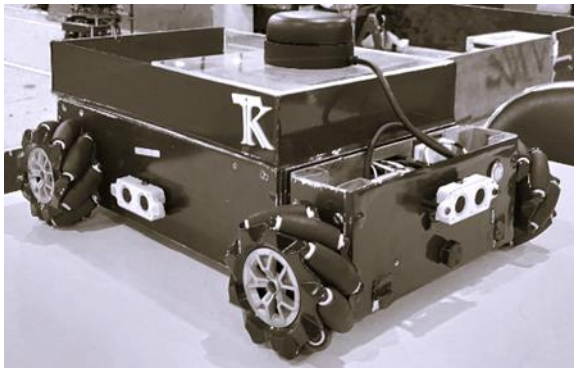
II. METODE PENELITIAN

Penelitian ini menggunakan metode yaitu dengan studi literatur dan pengumpulan data. Studi literatur dilakukan untuk memperoleh data yang diperlukan melalui referensi seperti buku, jurnal, artikel maupun pencarian disitur internet. Pengumpulan data dilakukan untuk mengumpulkan dan menganalisa data pencarian jalur [11].

Penelitian ini bertujuan membuat perencanaan jalur yang efisien, cepat, terpendek, dan mampu menghindari halangan, sehingga robot dapat bernavigasi dengan menggunakan ROS dan mikrokontroler Arduino Atmega 2560 serta komunikasi rosserial. Hasil penelitian ini adalah robot yang bergerak mengikuti jalur dari titik awal ke tujuan berdasarkan data input.

A. Design Robot

Robot mecanum dalam penelitian ini menggunakan 4 roda mecanum dan dikendalikan oleh mikrokontroller Arduino Atmega 2560. Motor penggeraknya adalah motor DC 12V dengan encoder, dan driver motor L298N digunakan untuk menggerakkan robot. Robot ini memiliki kecepatan 2,6 meter per detik. Ukuran robotnya adalah 379 mm x 210 mm x 90 mm, dengan diameter roda 97 mm dan berat 3,2 kg. Untuk design robot roda mecanum ada pada gambar 1.



Gambar 1. Design Robot mecanum

B. Perangkat Keras Robot

Perangkat keras yang digunakan untuk penelitian perencanaan jalur terhadap robot terdapat pada Tabel 1.

Tabel 1
Daftar Perangkat Keras

Jenis	*Jumlah*
Arduino Atmega 2560	*1*
Motor driver L298N	*2*
Motor Dc dengan encoder	*4*
RPLidar A2M12	*1*
Mini PC intel NUC	*1*
Roda Mecanum	*4*
Baterai Li Po 12V	*3*
Step down DC	*1*

Arduino Atmega 2560 digunakan sebagai kontrol level rendah pada robot, dengan 54 digital input/output, termasuk 14 pin PWM output dan 16 pin analog input. Kontrol level rendah mengacu pada teknik kontrol robot melalui sensor langsung, sementara mini PC digunakan sebagai kontrol level tinggi untuk memproses data sensor lingkungan. Mikrokontroler adalah sistem komputer dalam satu chip yang mengendalikan dan memproses data dari input ke output. Driver motor L298N mengontrol kecepatan dan arah motor DC, populer di dunia elektronika [12].

Motor DC dengan encoder sebagai aktuator pada robot dan encoder sebagai umpan balik untuk mendeteksi kecepatan motor. Rotary encoder tersusun dari suatu piringan tipis yang memiliki lubang-lubang pada bagian lingkaran piringan. LED ditempatkan disalah satu sisi piringan sehingga cahaya akan menuju piringan, disisi lainnya terdapat photo-transistor sehingga photo-transistor ini dapat mendeteksi cahaya dipancarkan oleh LED [13].

RPLidar A2M12 adalah sensor lidar 360 derajat yang dikembangkan oleh SLAMTEC, dengan radius pindai maksimum 12 meter. Sensor ini memancarkan cahaya laser ke objek, lalu menangkap dan menganalisis pantulannya untuk mendeteksi objek. LIDAR menggunakan dua metode pengukuran jarak: triangulasi, yang menggunakan prinsip trigonometri, dan Time of Flight (ToF), yang mengukur waktu tempuh cahaya. RPLIDAR digunakan pada robot otonom untuk

menghindari tabrakan dan mendeteksi rintangan dalam ruangan [14].

Roda mecanum sebagai alat gerak robot, roda mecanum adalah suatu sistem yang dirancang untuk robot *autonomous*, sistem ini diharapkan dapat berfungsi untuk menggerakkan robot dengan fleksibilitas, kelebihan dari roda mecanum yaitu dapat bergerak secara serentak dan mandiri arah rotasi dan translasi artinya keempatnya dapat bergerak setiap saat ke segala arah tanpa orientasi. Baterai digunakan sebagai daya robot dan yang terakhir yaitu Step down atau penurun tegangan yang digunakan untuk menurunkan tegangan dari 24v ke 19v yang akan disalurkan ke mini pc sebagai power mini pc [15].

C. Perangkat Lunak Robot

Perangkat Lunak Robot yang digunakan pada penelitian ini dapat dilihat pada Tabel 2.

Tabel 2
Perangkat Lunak Robot

Nama software	*Versi*
Arduino IDE	*versi 2.2.1*
ROS Noetic	*versi ubuntu 20.04*
Visual Studio Code	*versi 1.84.2*
Rviz	*versi 1.14.20*

Berikut adalah penjelasan dari Tabel 2, Arduino IDE (*Integrated Development Environment)* adalah yang digunakan untuk membuat sketch pemrograman atau bisa dibilang sebagai media untuk pemrograman pada *board* yang di program. Arduino IDE dibuat dari bahasa pemrograman Java yang dilengkapi dengan *library* C/C++ yang membuat operasi *input/*output lebih mudah. IDE berperan untuk menulis program, meng-*compile* menjadi kode biner dan meng-*upload* ke dalam memory mickrokontroler [5].

Arduino IDE dipenelitian ini digunakan sebagai media program motor mulai dari arah putaran, kecepatan dan odometry robot. ROS (Robot Operating System) adalah kumpulan perpustakaan perangkat lunak dan alat yang menyediakan kerangka kerja untuk mengembangkan dan menjalankan aplikasi robotika. ROS bersifat open source dan modular, memungkinkan penggunaan, modifikasi, dan kontribusi dari siapapun. Pengguna dapat memilih komponen dan paket sesuai kebutuhan proyek. ROS menggunakan konsep node, yaitu proses yang melakukan tugas tertentu, seperti mengendalikan sensor atau kamera. Node berkomunikasi melalui topik, yang membawa pesan jenis tertentu, seperti data sensor, perintah, atau gambar [16].

Dipenelitian ini menggunakan ROS jenis Noetic sebagai *middleware* komunikasi antar program. RViz adalah antarmuka grafis ROS untuk memvisualisasikan informasi dari berbagai topik. Dalam penelitian ini, RViz digunakan untuk visualisasi perencanaan jalur. Visual Studio Code, editor buatan Microsoft yang kompatibel dengan Windows, macOS, dan Linux, digunakan untuk mengedit dan menyimpan kode C++ robot [17].

D. Perencanaan Jalur

Perencanaan jalur adalah aspek penting dalam penelitian pengendalian gerak robot. tujuan utamanya adalah memastikan bahwa robot dapat bergerak melalui area yang kompleks tanpa mengalami tabrakan, meskipun terdapat berbagai rintangan atau kondisi jalan yang tidak stabil [18].

Metode perencanaan jalur terbagi menjadi dua kategori utama: metode klasik dan metode heuristik. Metode klasik meliputi Cell Decomposition (CD), Probabilistic Road Map (PRM), Potential Field (PF), dan Rapidly Random Tree (RRT). Metode heuristik mencakup Algoritma Dijkstra, A*, D*, Artificial Neural Network (ANN), Algorithm Genetic (GA), dan Particle Swarm Optimization (PSO). Pendekatan ini digunakan dalam perencanaan jalur robot mobile karena kelengkapan, efisiensi, dan optimalitasnya. Penelitian ini menggunakan Algoritma Dijkstra, yang dikenal sebagai salah satu algoritma terbaik untuk mencari jalur terpendek antara dua titik sejak diperkenalkan pada tahun 1968 [3].

Algoritma ini andal dan tahan terhadap gangguan, namun memiliki kompleksitas tinggi dan rentan terhadap optimum lokal [19].

Perencanaan jalur terbagi menjadi dua: global untuk lingkungan yang dikenal dan lokal untuk lingkungan yang tidak diketahui. Penelitian ini menggunakan perencanaan jalur global, di mana robot memetakan jalur terlebih dahulu dan mencari jalur terdekat untuk mencapai tujuan yang ditentukan, menggunakan metode grid dan visualisasi melalui Rviz [20].

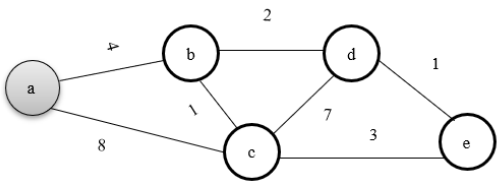
Dalam pencarian jalur terpendek, algoritma Dijkstra mencari bobot minimal dari graf berbobot untuk mendapatkan jarak terpendek antara dua titik. Algoritma dimulai dari node sumber dan mengiterasi untuk menemukan node dengan bobot terkecil. Titik yang dipilih dipisahkan dan tidak diperhatikan lagi. Langkah-langkah Dijkstra adalah:

1) Inisialisasi peta dengan node sumber bernilai 0 dan node tujuan ∞.
2) Telusuri node tetangga dari node sumber dan perbarui jarak jika lebih kecil.
3) Labeli node yang sudah diperiksa dan abaikan di iterasi berikutnya.
4) Pilih node dengan jarak terkecil yang belum dilewati sebagai node sumber baru.
5) Ulangi langkah sampai mencapai node tujuan.
6) Algoritma selesai dan menghasilkan jalur terpendek.

Pengujian dilakukan untuk mendapatkan kesesuaian jarak dari titik awal sampai titik akhir pada peta yang sudah ditenntukan jaraknya oleh algoritma dijkstra.

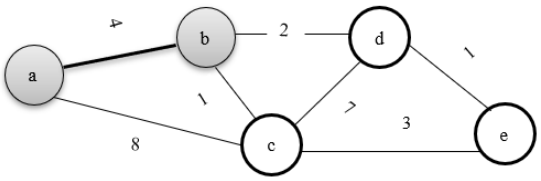
Untuk mengetahui lebih spesifiknya dari cara proses perhitungan djikstra berikut dibawah ini langkah-langkahnya yaitu :

1) Dalam penerapan algoritma dijkstra membutuhkan parameter asal dan akhir seperti pada gambar 2.



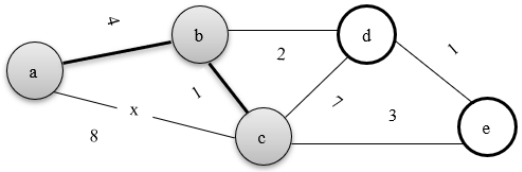
Gambar 2. Menentukan titik awal

2) Gambar 2 menunjukkan titik a adalah titik awal dan titik tujuan e. Algoritma Dijkstra akan mencari titik terdekat yang terhubung dengan titik a dan memiliki bobot kecil.



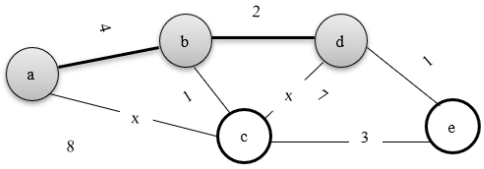
Gambar 3. Bobot terendah terhubung dengan titik a

3) Gambar 3 menunjukkan bahwa titik yang memiliki bobot minimum adalah b. selanjutnya algoritma Dijkstra akan membandingkan bobot dari titik yang terhubung dengan b dengan cara sebagai berikut :
   - Titik a – b – d = 6
   - Titik a – b – c = 5
4) Dari perbandingan diatas maka didapatkan titik a-b-c memiliki bobot yang terkecil, sehingga algoritma dijkstra akan melanjutkan perhitungan bobot yang terhubung dengan c. Pada tahap ini titik d tidak menjadi pilihan karena bobot yang dimiliki lebih besar dari bobot a-b-d dan a-b-c seperti pada gambar 4.



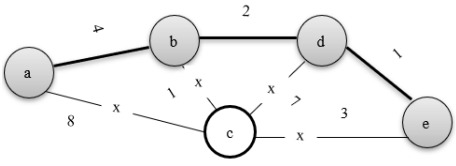
Gambar 4. Bobot terendah terhubung dengan c

5) Gambar 4 menunjukkan bahwa titik a-b-c-d = 12, sedangkan a-b-c-e = 8 dan sudah mencapai titik tujuan yaitu titik e, akan tetapi a-b-c-e belum bisa menjadi pilihan karena masih ada jalur lain yang memungkinkan memiliki bobot lebih rendah yaitu a-b-d dan belum dibandingkan oleh algoritma.



Gambar 5. Bobot terendah terhubung dengan d

6) Gambar 5 menunjukkan bahwa titik d terhubung dengan c dan e, akan tetapi titik c tidak menjadi pilihan karena bobot yang dimiliki a-b-c-d melebihi bobot minimum a-b-c-e, sehingga algoritma dijkstra akan lansung menghitung bobot a-b-d-e. Pada tahap ini, didapat nilai minimum a-b-d-e = 7 sedangkan a-b-c-e = 8, maka hasil jalur yang memiliki bobot rendah adalah a-b-d-e dapat dilihat pada gambar 6.



Gambar 6. Hasil akhir jalur terpendek

Berikut dibawah ini adalah pseudocode dari algoritma Dijkstra.

```
1: Function dijkstra (S, source)
2:  dist = map ()
3:  Q = set ()
4:  for i in S:
5:      if ( i = source) :
6:          dist[i] := 0
7:      else:
8:          dist[i] :=infinity
9:          add i to Q
10: while Q is not empty:
11:    i :=i in Q with minimum dist [i]
12:    remove i from Q
13:    for each neighbor u for i and u in Q:
14:        newDist := dist[i] + weight(i,u)
15:        if  (newDist < dist[u]):
16:            dist[u] := newDist
17:  return dist[]
```

Gambar 7. Pseudocode algoritma dijkstra

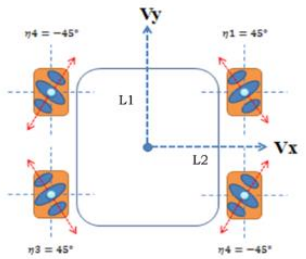
Berikut ini penjelasan pada gambar 7 diatas pilih node source pada graf S yang akan dijadikan node sumber dan tetapkan $dist_{source} = 0$. Untuk node selain source , tetapkan nilai dist = ∞. Masukkan seluruh node pada sebuah himpunan Q yang menunjukkan node – node yang masih belum ditentukan nilai shortest pathnya. Kemudian setelah itu pilih sebuah node i pada Q dengan nilai dist terkecil. Hapus i dari Q kemudian periksa setiap node u yang merupakan tetangga dari i yang belum ada pada Q. Nilai $dist_u$ diperbaharui apabila $dist_u < dist_i + w_{iu}$, dengan w_{iu} adalah besar bobot dari sisi yang menghubungkan i dan u dan ulangi langkah sampai ketemu jalur nya dan algoritma selesai.

E. Kinematika Mecanum

Penurunan kinematika robot mecanum adalah langkah awal dalam merancang trajectory, yaitu lintasan optimal yang harus diikuti robot untuk mencapai tujuannya dari posisi awal. Penelitian ini menggunakan persamaan kinematika terbalik (inverse) dan kinematika maju (forward) untuk mendukung perencanaan jalur dengan algoritma Dijkstra pada robot mecanum.

Persamaan inverse kinematik digunakan untuk menentukan kecepatan motor roda yang diperlukan untuk mencapai posisi atau orientasi tertentu (θ FL, θ FR, θ BL, θ BR) berdasarkan kecepatan linear (Vx, Vy) dan kecepatan angular (ω) yang diinginkan. Persamaan kinematika maju digunakan untuk menentukan posisi dan orientasi robot berdasarkan kecepatan motor dan konfigurasi roda. Dengan mengetahui kecepatan roda, dapat dihitung kecepatan robot ke arah depan (Vx), ke samping (Vy), dan rotasional (ω).

Untuk bergerak maju, keempat roda harus bergerak ke depan dengan kecepatan yang sama. Untuk bergerak ke kiri, roda 1 dan 4 bergerak ke belakang, sedangkan roda 2 dan 3 berhenti. Untuk berputar searah jarum jam, roda 1 dan 3 maju, sementara roda 2 dan 4 mundur. Jika roda 1 dan 3 bergerak lebih cepat daripada roda 2 dan 4, robot akan bergerak membentuk lingkaran searah jarum jam [21].



Gambar 8. Konfigurasi dan Resultan gaya robot mecanum.

$$\begin{bmatrix}\theta_{FL}\\ \theta_{FR}\\ \theta_{BL}\\ \theta_{BR}\end{bmatrix} = \frac{1}{2\pi r}\begin{bmatrix}1 & -1 & -(d+e)/2\\ 1 & 1 & (d+e)/2\\ 1 & 1 & -(d+e)/2\\ 1 & -1 & (d+e)/2\end{bmatrix}\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix} \quad (1)$$

$$\begin{bmatrix}V_x\\ V_y\\ \omega\end{bmatrix} 2\pi r\begin{bmatrix}\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}\\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4}\\ -\frac{1}{2(d+e)} & \frac{1}{2(d+e)} & -\frac{1}{2(d+e)} & \frac{1}{2(d+e)}\end{bmatrix} \quad (2)$$

Keterangan :
Θ_{FL} : Kecepatan roda depan-kiri atau front-left (rps)
Θ_{FR} : Kecepatan roda depan-kanan atau front-right (rps)
Θ_{FL} : Kecepatan roda belakang-kiri atau Back-Left (rps)
Θ_{FL} : kecepatan roda belakang-kanan atau Back-Right (rps)
r : Radius roda (m)
d : Jarak roda kiri ke roda kanan (m)
e : Jarak roda depan ke roda belakang (m)
Vx : Kecepatan robot dalam arah maju (m/s)
Vy : kecepatan robot dalam arah samping (m/s)
ω : Kecepatan rotasional robot (rad/s)

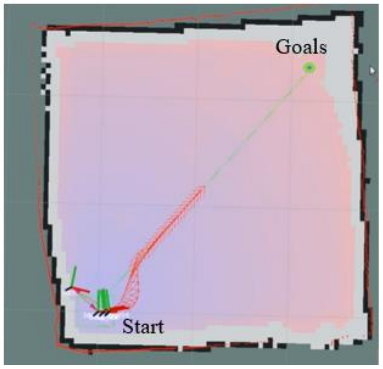
III. HASIL DAN ANALISA

Dalam penelitian ini, algoritma Dijkstra diterapkan untuk merencanakan jalur terpendek pada peta grid 2D berukuran 3.3 m x 3.3 m yang mewakili lingkungan nyata robot. Peta tersebut berisi beberapa rintangan yang ditempatkan secara acak untuk mensimulasikan kondisi nyata. Pengujian dilakukan pada 5 peta dengan posisi halangan yang berbeda, namun dengan titik awal yang sama (x = 0, y = 0) dan tujuan yang sama (x = 2.4, y = 2.3) setiap kali. Langkah pengujian meliputi :

1) Inisialisasi lingkungan dengan menyiapkan peta grid dan rintangan.
2) Inisialisasi algoritma dengan menentukan titik awal dan tujuan, lalu menjalankan algoritma untuk menentukan jalur terpendek.
3) Mengukur panjang jalur, waktu tempuh, dan mencatat keberhasilan dalam menghindari rintangan.
4) Menganalisis data yang diperoleh.
5) Memvisualisasikan hasil jalur.

Gambar 9. Peta lingkungan nyata.

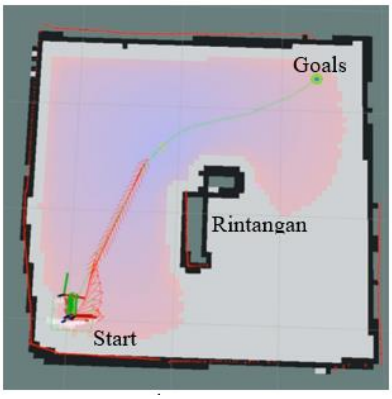
A. Pengujian Pertama Peta 1



Gambar 10. Peta 1.

Pada gambar 10, hasil pengujian robot mecanum di Peta 1 tanpa rintangan menunjukkan bahwa robot menempuh jarak 3.21 m dalam 14.96 detik dengan kecepatan rata-rata 0.2 m/s. Kecepatan ini dihitung dari pembagian jarak total dengan waktu total. Hasil ini mencerminkan kemampuan robot untuk menavigasi lingkungan sederhana dengan efisiensi dan stabilitas yang baik, serta dapat digunakan sebagai baseline untuk membandingkan performa robot di kondisi yang lebih menantang.

B. Pengujian Kedua Peta 2



Gambar 11. Peta 2.

Pada pengujian kedua di Peta 2 (gambar 11), yang memiliki rintangan tambahan, robot mecanum menempuh jarak 3.38 meter dalam 16.08 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun jarak dan waktu tempuh lebih panjang dibandingkan Peta 1, robot tetap mencapai tujuan dengan kecepatan konstan.

Hasil ini menunjukkan kemampuan robot untuk beradaptasi dengan rintangan di lingkungan yang lebih kompleks.

C. Pengujian ketiga Peta 3

Gambar 12. Peta 3.

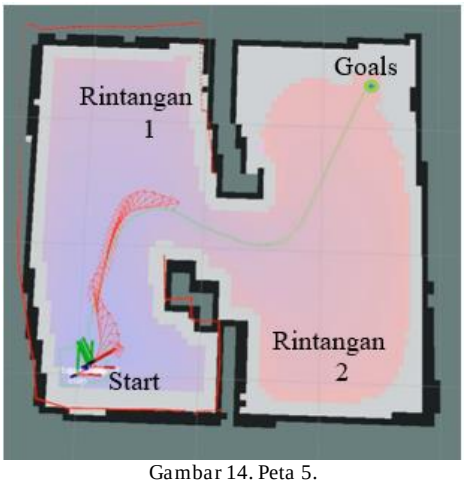
Pada pengujian ketiga (Gambar 12) di Peta 3, yang memiliki lebih banyak rintangan, robot mecanum menempuh jarak 4.14 meter dalam 22.13 detik dengan kecepatan rata-rata 0.2 m/s. Jarak dan waktu tempuh lebih panjang dibandingkan Peta 1 dan 2, menunjukkan kompleksitas rintangan mempengaruhi jalur yang diambil robot. Hasil ini menunjukkan bahwa algoritma path planning berhasil mengarahkan robot melalui jalur yang aman meskipun lebih panjang, dan robot tetap dapat beradaptasi dengan lingkungan yang lebih kompleks sambil mempertahankan kecepatan yang konsisten.

D. Pengujian Keempat Peta 4

Gambar 13. Peta 4.

Pada pengujian keempat (Gambar 13) di Peta 4, robot mecanum menempuh jarak 4.07 meter dalam 19.12 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun jarak hampir sama dengan Peta 3, waktu tempuh lebih singkat, menunjukkan rintangan di Peta 4 lebih mudah dihindari. Hasil ini menunjukkan kemampuan robot untuk beradaptasi dengan konfigurasi rintangan yang berbeda sambil mempertahankan kecepatan yang konsisten.

E. Pengujian kelima Peta 5



Gambar 14. Peta 5.

Pada pengujian kelima (Gambar 14) di Peta 5, robot mecanum menempuh jarak 4.29 meter dalam 22.85 detik dengan kecepatan rata-rata 0.2 m/s. Meskipun posisi awal dan tujuan sama dengan pengujian sebelumnya, rintangan diatur berbeda, menambah tantangan. Jarak dan waktu tempuh lebih panjang dibandingkan Peta 4, menunjukkan kemampuan robot untuk beradaptasi dengan rintangan baru dan mempertahankan kecepatan yang konsisten, serta efektivitas algoritma path planning dalam menemukan jalur optimal. Berikut ini hasil ringkasan dari pengujian-pengujian yang dilakukan dengan menggunakan lima data peta dapat dilihat pada Tabel 3.

Tabel 3
Hasil Pengujian terhadap Peta 1 -5

Keterangan	Peta 1	Peta 2	Peta 3	Peta 4	Peta 5
Waktu Eksekusi (Detik)	14.96	16.08	22.13	19.12	22.85
Jarak ()	3.21	3.38	4.14	4.07	4.2
Posisi X (m)	2.33	2.32	2.33	2.32	2.35
Posisi Y (m)	2.35	2.35	2.36	2.36	2.34
Error terhadap posisi X	0.07	0.08	0.07	0.08	0.05
Error terhadap posisi Y (m)	0.05	0.05	0.06	0.06	0.04
Rata- rata Velocity (m/s)	0.21	0.21	0.21	0.21	0.18
				Rata – Rata Velocity Robot	0.19

Dari tabel 3, pada peta 1 tanpa rintangan, robot menempuh jarak 3.21m dalam waktu 14.96 detik. Posisi akhir robot sedikit berbeda dari target dengan error 0.07m pada sumbu x dan 0.05m pada sumbu y. Kecepatan rata-rata adalah 0.21 m/s. Pada peta 2 dengan rintangan tambahan, robot menempuh jarak 3.38m dalam 16.08 detik. Error terhadap posisi akhir adalah 0.08m pada sumbu x dan 0.05m pada sumbu y. Kecepatan rata-rata tetap 0.21 m/s. Pada peta 3 dengan lebih banyak rintangan, robot menempuh jarak 4.14m dalam 22.13 detik. Error terhadap

posisi akhir adalah 0.07m pada sumbu x dan 0.06m pada sumbu y. Kecepatan rata-rata menurun menjadi 0.18 m/s. Pada peta 4, robot menempuh jarak 4.07 m dalam 19.12 detik. Error terhadap posisi akhir adalah 0.08m pada sumbu x dan 0.06m pada sumbu y. Kecepatan rata-rata kembali menjadi 0.21 m/s dan yang terakhir pada peta 5 dengan rintangan baru, robot menempuh jarak 4.20 m dalam 22.85 detik. Error terhadap posisi akhir adalah 0.05m pada sumbu x dan 0.04 m pada sumbu y. Kecepatan rata-rata adalah 0.18 m/s.

IV. KESIMPULAN

Penelitian ini meneliti efektivitas algoritma dijkstra dalam perencanaan jalur untuk robot mecanum. Hasil penelitian menunjukkan bahwa algoritma dijkstra sangat efektif dalam menentukan jalur terpendek dari titik awal ke tujuan, bahkan dalam lingkungan yang kompleks. Robot yang menggunakan algoritma ini mampu mencapai kecepatan rata-rata 0.19 m/s, dengan jarak tempuh terpendek sebesar 3.2 meter dalam waktu 14.96 detik, dan jarak terjauh sebesar 4.2 meter dalam waktu 22.85 detik. Kecepatan rata-rata robot dalam semua uji adalah 0.18 m/s. Selain itu, error posisi rata-rata robot adalah 0.6% pada sumbu X dan 0.5% pada sumbu Y yang menunjukkan tingkat akurasi yang tinggi.

Algoritma dijkstra juga terbukti mampu mempertahankan kinerja yang stabil meskipun menghadapi rintangan tambahan di sepanjang jalur. Hal ini menunjukkan kemampuan algoritma dalam beradaptasi dalam berbagai kondisi lingkungan yang berbeda-beda. Saran pengembangan untuk meningkatkan kinerja algoritma dijkstra dalam perencanaan jalur pada robot mecanum meliputi integrasi algoritma path planning lainnya, optimasi kinerja, pengujian di lingkungan dinamis, peningkatan sensor, penyesuaian algoritma untuk jenis rintangan yang bervariasi, evaluasi aplikasi nyata, uji coba di platform lain, dan peningkatan model untuk aplikasi skala besar.

REFERENCES

[1] C. Wang and J. Mao, "Summary of AGV Path Planning," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, IEEE, Oct. 2019, pp. 332–335. doi: 10.1109/EITCE47263.2019.9094825.

[2] Y. Li, Z. Huang, and Y. Xie, "Path planning of mobile robot based on improved genetic algorithm," *Proc. - 2020 3rd Int. Conf. Electron Device Mech. Eng. ICEDME 2020*, pp. 691–695, 2020, doi: 10.1109/ICEDME50972.2020.00163.

[3] M. N. A. Wahab, C. M. Lee, M. F. Akbar, and F. H. Hassan, "Path Planning for Mobile Robot Navigation in Unknown Indoor Environments Using Hybrid PSOFS Algorithm," *IEEE Access*, vol. 8, pp. 161805–161815, 2020, doi: 10.1109/ACCESS.2020.3021605.

[4] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid Path Planning Algorithm Based on Membrane Pseudo-Bacterial Potential Field for Autonomous Mobile Robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019, doi: 10.1109/ACCESS.2019.2949835.

[5] H. Zhao, Z. Nie, F. Zhou, and S. Lu, "A Compound Path Planning Algorithm for Mobile Robots," *Proc. 2021 IEEE Int. Conf. Power Electron. Comput. Appl. ICPECA 2021*, pp. 541–545, 2021, doi: 10.1109/ICPECA51329.2021.9362724.

[6] Y. Jiang *et al.*, "Path Planning for Mobile Robots Based on Improved RRT Algorithm," *ICARM 2022 - 2022 7th IEEE Int. Conf. Adv. Robot. Mechatronics*, pp. 793–798, 2022, doi: 10.1109/ICARM54641.2022.9959538.

[7] W. Andriani, "Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall Penentuan Jalur Lintasan Terpendek Stasiun Tegal Menuju Hotel," *BATIRSI-Bahari Tek. Inform. dan …*, vol. 4, no. 2, pp. 1–8, 2021, [Online]. Available: https://e-journal.stmik-tegal.ac.id/index.php/batirsi/article/view/42

[8] A. Alyasin, E. I. Abbas, and S. D. Hasan, "An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method," *4th Sci. Int. Conf. Najaf, SICN 2019*, pp. 11–14, 2019, doi: 10.1109/SICN47020.2019.9019345.

[9] Q. Liu, H. Xu, L. Wang, J. Chen, Y. Li, and L. Xu, "Application of dijkstra algorithm in path planning for geomagnetic navigation," *Proc. IEEE Sens. Array Multichannel Signal Process. Work.*, vol. 2020-June, pp. 19–22, 2020, doi: 10.1109/SAM48682.2020.9104382.

[10] Y. Xu, Y. Zhao, and S. L. Zhao, "Grid Line Tracking Omnidirectional Robot Based on Visible Light Sensor and Mecanum Wheel PID Control," *Proc. - 2021 7th Int. Symp. Mechatronics Ind. Informatics, ISMII 2021*, pp. 57–60, 2021, doi: 10.1109/ISMII52409.2021.00019.

[11] F. R. Khoir, A. A. Subandri, F. N. Alanshori, Z. M. H. Solihah, M. Munawir, and A. S. Perdana, "Perencanaan Rute Optimal Kunjungan Destinasi Wisata Bandung dengan Algoritma Dijkstra Pada C++," *J. Teknol. Dan Sist. Inf. Bisnis*, vol. 6, no. 2, pp. 275–281, 2024, doi: 10.47233/jteksis.v6i2.1167.

[12] H. T. Saputra and A. Muhaimin, "Robot Pemindah Benda Dengan Kendali Joystick Ps2 Wireless Berbasis Wemos," *J. Ilmu Komput.*, vol. 11, no. 02, pp. 80–85, 2022, [Online]. Available: http://jik.htp.ac.id

[13] G. Rahmatillah and B. Suprianto, "Sistem Pengendalian Kecepatan Motor DC Pada Prototipe Lift Menggunakan Kontroler Pi Berbasis Arduino," *J. Tek. Elektro*, vol. 9, no. 2, pp. 269–276, 2020.

[14] M. Fikri and M. Rivai, "Sistem Penghindar Halangan dengan Metode Lidar pada Unmanned Surface Vehicle," *J. Tek. ITS*, vol. 8, no. 2, pp. 127–132, 2020, doi: 10.12962/j23373539.v8i2.43153.

[15] E. C. Orozco-Magdaleno, F. Gomez-Bravo, E. Castillo-Castaneda, and G. Carbone, "Evaluation of Locomotion Performances for a Mecanum-Wheeled Hybrid Hexapod Robot," *IEEE/ASME Trans. Mechatronics*, vol. 26, no. 3, pp. 1657–1667, 2021, doi: 10.1109/TMECH.2020.3027259.

[16] L. Qiong, C. Xudong, H. Jizhuang, and M. Ruihao, "Research on robot path planning method based on tangent intersection method," *Proc. - 2020 Int. Conf. Inf. Sci. Parallel Distrib. Syst. ISPDS 2020*, pp. 272–276, 2020, doi: 10.1109/ISPDS51347.2020.00063.

[17] H. Huang *et al.*, "Dynamic path planning based on improved D∗ algorithms of Gaode map," *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 1121–1124, 2019, doi: 10.1109/ITNEC.2019.8729438.

[18] Y. Ding, H. Ma, and S. Li, "Path Planning of Omnidirectional Mobile Vehicle Based on Road Condition," *Proc. 2019 IEEE Int. Conf. Mechatronics Autom. ICMA 2019*, pp. 1425–1429, 2019, doi: 10.1109/ICMA.2019.8816402.

[19] Z. Nie and H. Zhao, "Research on Robot Path Planning Based on Dijkstra and Ant Colony Optimization," in *2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, IEEE, Nov. 2019, pp. 222–226. doi: 10.1109/ICIIBMS46890.2019.8991502.

[20] H. Liu, L. Huang, and H. Ye, "Autonomous path planning strategy for water-Air amphibious vehicle based on improved A* algorithm," *IEEE Int. Conf. Ind. Informatics*, vol. 2020-July, pp. 812–816, 2020, doi: 10.1109/INDIN45582.2020.9442090.

[21] F. Fahmizal, A. Priyatmoko, and A. Mayub, "Implementasi Kinematika Trajectory Lingkaran pada Robot Roda Mecanum," *J. List. Instrumentasi dan Elektron. Terap.*, vol. 3, no. 1, pp. 25–30, 2022, doi: 10.22146/juliet.v3i1.74760.