Perancangan Robot Penelusur Menggunakan Algoritma Dijkstra dan Metode Maze Solver

Devising Search Robot Using Dijkstra Algorithm and Maze Solver Method

Yusuf Anshori[1], A. Y. Erwin Dodu[2], Fadli Kurniawan[3]
[1,2] Jurusan Teknologi Informasi, Fakultas Teknik, Universitas Tadulako,
Jl. Soekarno Hatta Km. 9 Palu Telp (0451) 422611 – 422355 Fax. (0451) 454014
[3] Jurusan Teknologi Informasi, Fakultas Teknik, Universitas Tadulako, Palu
e-mail: *[1]iyus.jr@gmail.com, [2]ayerwin.dodu@gmail.com, [3]vadlykurniawan@gmail.com

Abstrak

Dalam sebuah eksplorasi terdapat berbagai macam hal yang ingin di telusuri, baik dalam area terbuka maupun sebuah area tertutup. Pada area tertutup seperti reruntuhan, banyak hal yang bisa terjadi dan dapat membingungkan serta membahayakan nyawa manusia. Berdasarkan studi literatur yang dilakukan didapatkan bahwa dengan menggunakan robot yang diimplementasikan algoritma dijkstra dan metode maze solver dapat mengurangi biaya eksplorasi serta menghindari hal-hal yang dapat membahayakan nyawa manusia. Maka untuk algoritma dijkstra dan metode maze solver diambil dikarenakan algoritma dijkstra dapat menelusuri jalur terpendek dari titik manapun dan metode maze solver dapat menelusuri labirin yang tidak diketahui peta labirinnya sehingga dapat melakukan penelesuran labirin yang memiliki peta yang diketahui jalurnya dan peta yang tidak diketahui jalurnya. Capaian dari hasil penelitian ini adalah, mengimplementasikan algoritma dijkstra dan metode maze solver pada robot penelusur labirin. Ilmu penelitian ini dapat menjadi referensi dari studi pembelajaran dan pembuatan produk teknologi yang berguna bagi perusahaan maupun masyarakat.

Kata kunci— Robot, Eksplorasi, Algoritma Dijkstra, Maze Solver, Labirin.

Abstract

In doing an exploration, there are various kinds of things that you're expecting to find, both in an open area or in a closed area. In closed area such as ruins, anything can happen, which can confuse or endanger human life. Based on the literature study conducted, it was found that using robots implemented by the dijkstra algorithm and the maze solver method could reduce the cost of exploration and avoid things that could endanger human life. it is also because The Dijkstra Algorithm can trace the shortest path from any point, and the maze solver method can trace the maze of an unknown labyrinth so that it can trace both labyrinths that have maps the one that doesn't have maps, that made the dijkstra algorithm and the maze solver method chosen to be implemented to the robot, This research will be used as a reference of learning and making technology products that is surely useful for companies and society

Keywords— Robots, Exploration, Dijkstra Algorithm, Maze Solver, Labyrinths

1. PENDAHULUAN

Dalam sebuah eksplorasi terdapat berbagai macam hal yang ingin di telusuri, baik dalam area terbuka maupun sebuah area tertutup atau ruangan tertutup. Pada area tertutup seperti reruntuhan, banyak hal yang bisa terjadi dan dapat membingungkan serta membahayakan nyawa manusia. Dengan menggunakan sebuah robot penelusur yang di implementasikan dengan algoritma *dijkstra* dan metode *maze solver* hal tersebut dapat diatasi

dan dapat dikurangi biaya eksplorasi serta dapat mengetahui performa algoritma *dijkstra* dan metode *maze solver* dalam robot penelusur.

Saat ini telah ada beberapa penelitian terkait masalah tersebut dengan penelitian terdahulu [1,2]. Bagaimana menemukan jalur terpendek dengan menggunakan robot sebagai alat dalam menelusuri suatu labirin. Untuk mengatasi hal tersebut, solusi pemecahan masalah yang diusulkan adalah menggunakan algoritma *Short Path Finder*. Capaian yang dilakukan dalam penelitian ini Robot berhasil menelusuri labirin namun algoritma *short path finder* hanya mampu diterapkan dalam lintasan lurus dan tidak dapat diterapkan dalam lintasan melingkar [1].

Bagaimana cara memecahkan suatu labirin dengan suatu jalur yang memiliki sudut belokan yang mempunyai sudut $225^{\mathbf{o}}$ dan sudut $45^{\mathbf{o}}$, solusi pemecahan masalah yang diusulkan adalah dengan meng gunakan beberapa algoritma antara lain algoritma *line maze solver*, algoritma *flood fill* dan algoritma *pledge*. Capaian yang didapatkan dalam penelitian tersebut adalah robot dapat melintasi jalur yang memiliki sudut belokan yang mempunyai sudut $225^{\mathbf{o}}$ dan sudut $45^{\mathbf{o}}$[2].

Algoritma *dijkstra* merupakan salah satu bentuk algoritma *greedy*. Algoritma ini termasuk algoritma pencarian *graf* yang digunakan untuk menyelesaikan masalah lintasan terpendek dengan satu sumber pada sebuah *graf* yang tidak memiliki *cost* sisi negatif, dan menghasilkan sebuah pohon lintasan terpendek. Algoritma ini sering digunakan pada *routing*. Algoritma *dijkstra* menggunakan *adjacent list* untuk merepresentasikan sebuah jaringan. Secara garis besar algortima *dijkstra* membagi semua node menjadi dua, kemudian dimasukkan ke dalam tabel yang berbeda, yaitu tabel permanen dan tabel temporal. Tabel permanen berisi node awal dan node-node yang telah melalui proses pemeriksaan dan labelnya telah diubah dari temporal menjadi permanen. Tabel temporal berisi node-node yang berhubungan dengan node pada tabel permanen [3].

maze solving adalah algoritma yang digunakan robot untuk memecahkan lintasan yang dibuat untuk mendapatkan jalur yang tepat. *Maze* bisa berupa dinding atau garis. Algoritma ini memberikan opsi untuk berjalan mengikuti dinding kiri atau dinding kanan pada proses penelusuran labirin. Penelusuran yang dijalani tidak dapat menelusuri semua jalur yang ada pada labirin, namun dapat memberikan jalan keluar. Jalan keluar tersebut bisa berupa titik akhir labirin maupun titik awal labirin [4].

maze solving adalah salah satu algoritma yang digunakan pada robot untuk mencari jalur terpendek dari sebuah line maze. Terdapat 2 aturan dalam algoritma ini, yaitu [5]:

1. Left Hand Rule

   Dalam *left hand rule*, robot akan lebih memilih untuk belok kiri dari pada lurus atau belok kanan dan jika tidak ada belokan ke kiri akan lebih memilih lurus dari pada belok kanan.
2. Right Hand Rule.

   Dalam *right hand rule*, robot akan lebih memilih belok kanan dari pada lurus dan lebih memilih lurus dari pada belok kiri.

Sistem Navigasi *Wall follower* adalah suatu aksi robot untuk mengikuti dinding dan berada tidak jauh dari dinding, *wall follower* bekerja berdasarkan prinsip mengikuti suatu objek, dalam hal ini objek tersebut adalah dinding. *Wall follower* dapat di implementasikan pada beberapa kasus dalam kehidupan kita sehari – hari dengan menggunakan beberapa algoritma di dalamnya [6].

Robot adalah sebuah perangkat mekanik yang dapat melakukan pekerjaan fisik yang dikendalikan secara otomatis atau dikontrol oleh manusia. Namun demikian, terdapat empat karakteristik dasar yang harus dimiliki oleh setiap robot moderen. Karakteristik dasar tersebut adalah sensor, sistem kecerdasan (kontrol), peralatan mekanik (aktuator) dan sumber daya (power) [7].

Sensor Ultrasonik adalah sensor pembaca jarak pada suatu objek yang dipantulkan. Sensor ultrasonik memiliki gelombang dengan besar frekuensi diatas frekuensi gelombang suara yaitu lebih dari 20 KHz [8].

Microcontroller adalah suatu terobosan dalam teknologi mikroprosessor dan mikrokomputer, perbedaannya mikrokontroler hanya digunaka untuk menangani suatu aplikasi tertentu. Perbedaan mikrocontroller lain terletak pada perbandingan RAM dan ROM. Komputer memiliki RAM dan ROM yang besar, tetapi pada mikrokontroler sangat terbatas. ROM pada mikrokontroler digunakan untuk menyimpan program, sedangkan RAM untuk menyimpan data sementara. Mikrokontroler terdiri dari ALU (*Aritmatic Logical Unit*), CU (*Control Unit*), PC (*Program Counter*), SP (*Stack Pointer*), *Register, Timer, Interupt*. Mikrocontroller juga dilengkapi dengan beberapa piranti pendukung lain seperti ROM (*Read Only Memory*), RAM (*Random Accses Memory*), *decoder, communication interface, input/output* (I/O) serial atau parallel [9].

Labirin merupakan tempat yg penuh dengan jalan dan lorong yg berliku-liku dan simpang siur dan dipisahkan oleh tembok. Labirin sering kali dijadikan tantangan dalam permainan seperti *puzzle*, dimana terdapat objek dalam posisi awal harus menemukan jalan keluar pada posisi yang ditentukan [10].

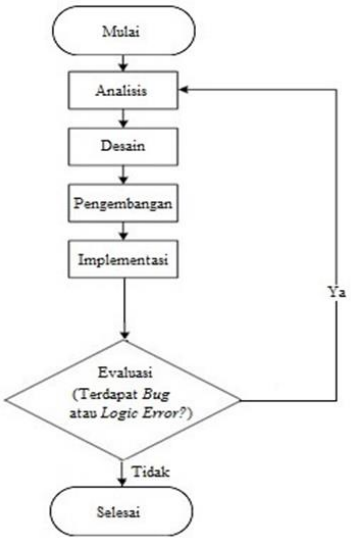
Dengan ini penulis memilih algoritma *dijkstra* dan metode *maze solver* dikarenakan algoritma dijkstra dapat menelusuri jalur terpendek dari titik manapun dan metode *maze solver* dapat menelusuri labirin yang tidak diketahui peta labirinnya sehingga dapat melakukan penelesuran labirin yang memiliki peta yang diketahui jalurnya dan peta yang tidak diketahui jalurnya.

Berdasarkan latar belakang di atas maka dapat dirumuskan permasalahannya adalah bagaimana mengimplementasi dan mengetahui performa algoritma *dijkstra* dan metode *maze solver* pada sebuah robot penulusur labirin?

Tujuan yang ingin dicapai pada penelitian ini adalah mengimplementasikan dan menguji performa algoritma *dijkstra* dan metode *maze solver* pada robot penelusur labirin.

2. METODE PENELITIAN

Adapun tahapan perancangan secara detil dan lengkap dapat dilihat pada *flowchart* tahapan perancangan penelitian. Adapun *flowchart* dapat dilihat pada gambar 1.



Gambar 1 *Flowchart* Tahapan Perancangan Penelitian

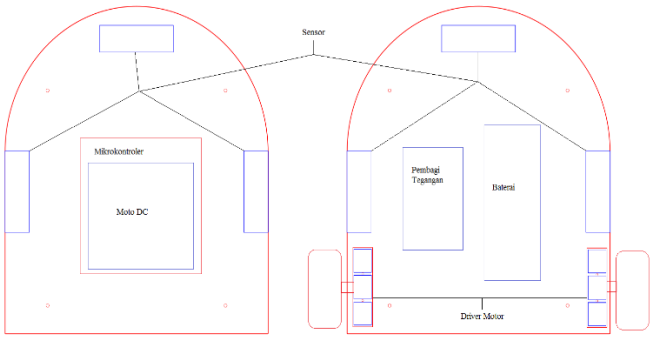
2.1. Analisis

Melakukan pengumpulan bahan, informasi, dan referensi yang relevan berkaitan dengan topik penelitian dan melakukan identifikasi masalah untuk menentukan solusi dan pembahasan apa yang diperlukan, berkaitan dengan topik penelitian serta mempelajari literatur

yang akan digunakan dengan masalah yang dihadapi untuk digunakan sebagai kajian pendukung teori dalam penelitian ini.

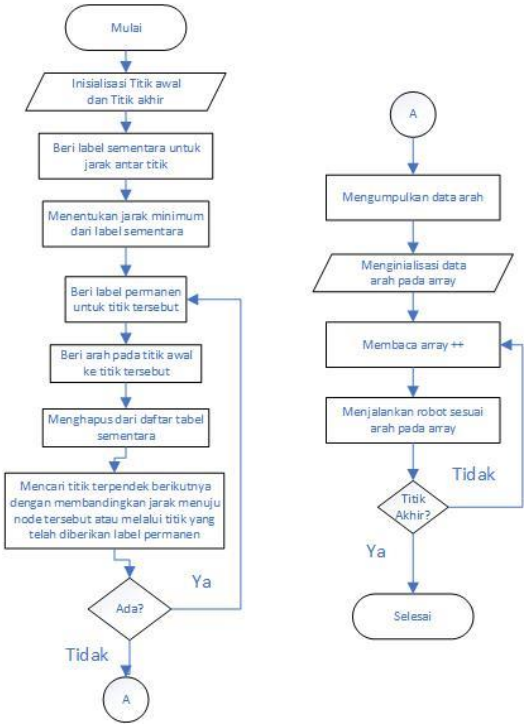
2.2. Desain

Melakukan perancangan kebutuhan robot dan desain kerangka robot serta tahapan-tahapan proses operasi mencakup proses perancangan pembuatan program menggunakan algoritma *dijkstra* dan metode *maze solver*. Adapun desain robot dapat dilihat pada gambar 2.



Gambar 2 Desain Robot

Proses perancangan software algoritma *dijktra* dan metode *maze solver* yang mencakup proses operasi dan dan proses penerapan dapat dilihat pada *flowchart* gambar 3 dan *flowchart* gambar 4.

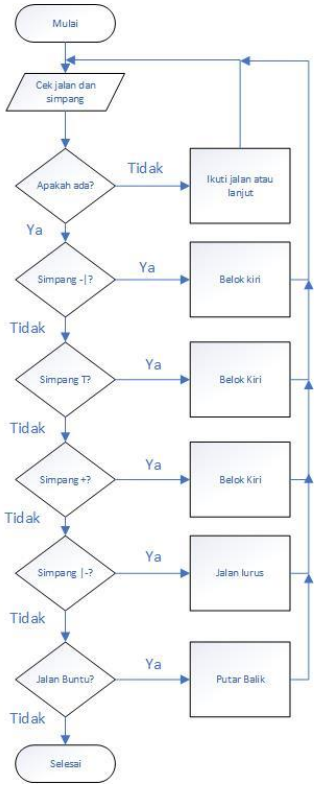


Gambar 3 *Flowchart* Proses Operasi dan Penerapan Algoritma *Dijkstra*

Dalam flowchart pada Gambar 3 dapat dijelaskan sebagai berikut:

a. Pertama dengan membuat peta yang diketahui dengan menginialisasi titik awal dan titik akhir dari jalur yang akan di tempuh.
b. Lalu memberikan label sementara untuk jarak dan titik.
c. Setelah itu melakukan pencarian jarak terpendek dari label sementara.
d. Titik yang mendapatkan jarak terpendek diberikan label permanen.
e. Memberi arah kepada titik yang baru saja diberikan label permanen.

f. Menghapus jarak terpendek dari label sementara.
g. Mencari lagi jarak terpendek berikutnya dengan membandingkan jarak dari titik tersebut atau titik yang diberi label permanen.
h. Jika ada titik yang masih bisa dicari maka kembali ke proses d, jika tidak lanjut ke proses selanjutnya.
i. Mengumpulkan data arah dari semua label permanen.
j. Membuat semua data arah menjadi sebuah array.
k. Membaca data array.
l. Robot berjalan sesuai arah yang terdapat pada array.
m. Jika sampai titik akhir maka robot akan selesai, jika belum maka akan kembali lagi membaca data array.



Gambar 4 *Flowchart* Proses Operasi dan Penerapan Metode *Maze Solver*

Dalam flowchart pada gambar 4 dapat dijelaskan sebagai berikut:

a. Mengecek jalan dan simpang.
b. Jika menemukan jalan dan simpang maka akan lanjut ke proses selanjutnya, jika tidak jalan terus dan kembali lagi mengecek jalan dan simpang.
c. Jika menemukan simpang -| maka akan memerintahkan robot untuk belok kiri dan mengecek lagi jalan dan simpang, jika tidak lanjut lagi ke proses selanjutnya.
d. Jika menemukan simpang T maka akan memerintahkan robot untuk belok kiri dan mengecek lagi jalan dan simpang, jika tidak lanjut lagi ke proses selanjutnya.
e. Jika menemukan simpang + maka akan memerintahkan robot untuk belok kiri dan mengecek lagi jalan dan simpang, jika tidak lanjut lagi ke proses selanjutnya.
f. Jika menemukan simpang |- maka akan memerintahkan robot untuk jalan lurus dan mengecek lagi jalan dan simpang, jika tidak lanjut lagi ke proses selanjutnya.
g. Jika menemukan jalan buntu maka akan memerintahkan robot putar balik dan mengecek lagi jalan dan simpang, jika tidak maka robot akan berhenti.

2.3. Pengembangan

Melakukan pembuatan robot antara lain melakukan persiapan *hardware* yang akan digunakan, membuat robot, arena dan melakukan pengkodingan terhadap program algoritma *dijkstra* dan metode *maze solver* yang akan di implementasikan ke dalam robot.

2.4. Implementasi

Menyempurnakan robot dan mengimplementasikan algoritma *dijkstra* serta penginputan program metode *maze solver* ke *mikroprosesor*, selanjutnya dilakukan pengujian robot secara menyeluruh untuk memastikan fungsi-fungsi dari robot telah berjalan dengan baik sesuai yang diharapkan dengan memastikan tidak ada *bug* ataupun *logic error* pada perangkat lunak dan kerusakan pada perangkat keras.

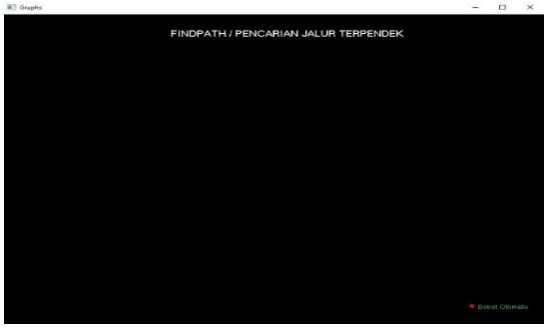
2.5. Evaluasi

Menuliskan kesimpulan melalui analisa yang diperoleh serta mencatat *bug* maupun *logic error* yang ditemukan untuk di analisa kembali untuk pengembangan robot kedepan.

3. HASIL DAN PEMBAHASAN

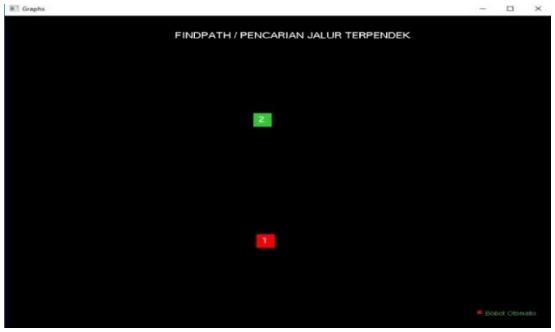
3.1. Implementasi Algoritma *Dijkstra*

Pada implementasi algoritma dijkstra menggunakan aplikasi berbasis *desktop* yang hanya berfungsi untuk mencari jalur terpendek pada peta yang sudah diketahui jalur nya, adapun bentuk dan rupa aplikasi implementasi algoritma *dijkstra* dapat dilihat pada gambar 5, gambar 6, gambar 7, gambar 8 dan gambar 9.



Gambar 5 Form Awal

Pada gambar 5 adalah sebuah *form* awal dimana dalam semua area *form* bisa dibuatkan area peta yang berupa node-node dan simpul yang akan direncanakan.

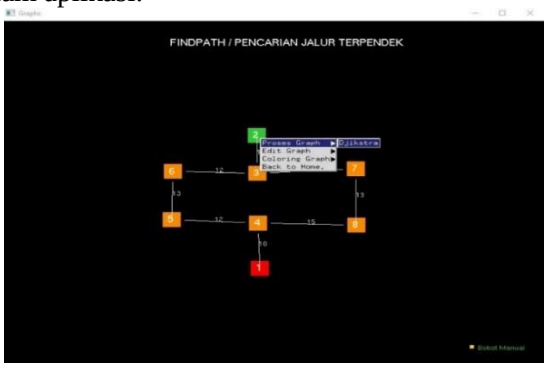


Gambar 6 Pemberian Node Awal dan Node Akhir

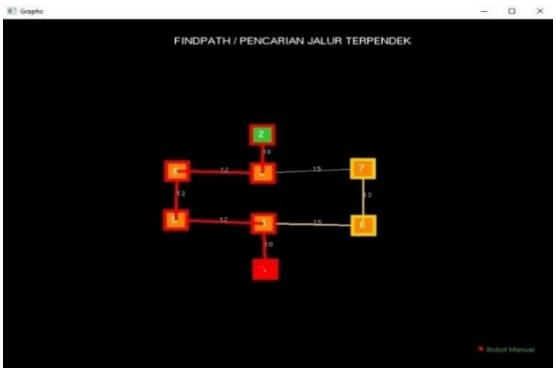
Pada Gambar 6 dilakukan pemberian node awal yang berupa titik atau kotak berwarna merah dan node akhir yang berupa titik yang berwana hijau.

Gambar 7 Pemberian Semua Node yang Ada

Pada gambar 7 melakukan pemberian semua node yang ada pada peta dan akan diimplementasikan kedalam aplikasi.



Gambar 8 Pemberian Simpul



Gambar 9 Hasil Proses Implentasi Algoritma Dijkstra

Pada gambar 8 dilakukan pemberian bobot atau jarak pada simpul dan akan dilakukan proses algoritma *dijkstra* dan menghasilkn jarak terpendek yang berada pada gambar 9

3.2. Implementasi *Maze Solver*

Maze solver mempunyai skema kerja berdasarkan data sensor yang diterima lalu diproses secara logika sehingga dapat menentukan arah pada robot. Adapun proses logika tersebut dapat dilihat pada tabel 1

Tabel 1 Logika Maze solver

No.	Sensor Depan	Sensor Samping	Sensor Sudut	Kondisi
1	Dekat	Dekat	Dekat	Kanan
2	Dekat	Dekat	Dekat	Maju
3	Dekat	Dekat	Jauh	Kiri
4	Dekat	Sedang	Dekat	Kanan
5	Dekat	Sedang	Sedang	Kanan
6	Dekat	Sedang	Jauh	Kiri
7	Dekat	Jauh	Dekat	Kanan
8	Dekat	Jauh	Sedang	Maju
9	Dekat	Jauh	Jauh	Kiri
10	Jauh	Dekat	Dekat	Kanan
11	Jauh	Dekat	Sedang	Maju
12	Jauh	Dekat	Jauh	Maju
13	Jauh	Sedang	Dekat	Maju
14	Jauh	Sedang	Sedadng	Maju
15	Jauh	Sedang	Jauh	Kiri
16	Jauh	Jauh	Dekat	Maju
17	Jauh	Jauh	Sedang	Maju
18	Jauh	Jauh	Jauh	Kiri

Adapun penjelasan lebih lanjut adalah sebagai berikut:
a. Sensor Depan Dekat = 0 cm – 9 cm
b. Sensor Depan Jauh = > 9 cm
c. Sensor Samping Dekat = 0 cm – 10 cm
d. Sensor Samping Sedang = >10 cm - <15 cm
e. Sensor Samping Jauh = >15 cm
f. Sensor Sudut Dekat = 0 cm – 10 cm
g. Sensor Sudut Sedang = >10 cm - <15 cm
h. Sensor Sudut Jauh = > 15 cm

3.3. Pengujian Sistem Robot

Pengujian robot ini dilakukan dengan menguji cepat waktu tempuh robot dalam menempuh jalur yang dilalui dan untuk menguji seberapa keberhasilan implementasi algoritma *dijkstra* dan *maze solver* pada robot dalam penemuan jalur labirin. Pengujian algortima *dijkstra*

dilakukan sebanyak 5 kali pada 2 peta yang berbeda dan titik yang berbeda serta pengujian *maze solver* ini dilakukan sebanyak 10 kali dengan 3 peta yang berbeda-beda. Adapun nilai pengujian yang akan diambil adalah sebagai berikut:

a. Keberhasilan algoritma *dijkstra* dalam menemukan jalur terpendek
b. Waktu tempuh algoritma *dijkstra*
c. Keberhasilan *maze Solver* dalam menelusuri jalur
d. Waktu tempuh *maze Solver*

Hasil pengujian robot dapat dilihat pada tabel 2, tabel 3, tabel 4, tabel 5 dan tabel 6.

Tabel 2 Hasil Pengujian *Dijkstra* Pada Peta Implentasi Algoritma *Dijkstra* A

No	Berhasil Menelesuri	Waktu Tempuh	Waktu Komputasi Algoritma
1	Ya	14s	22.8s
2	Ya	15s	23.4s
3	Ya	14s	23.0s
4	Ya	16s	22.7s
5	Ya	16s	22.5s

Tabel 3 Hasil Pengujian *Dijkstra* Pada Peta Implementasi Algoritma *Dijkstra* B

No	Berhasil Menelesuri	Waktu Tempuh	Waktu Komputasi Algoritma
1	Ya	20s	32.1s
2	Ya	22s	32.1s
3	Ya	21s	31.8s
4	Ya	29s	31.9s
5	Ya	27s	28.8s

Dari hasil pengujian diatas terdapat beberapa perbedaan dari waktu komputasi dan waktu tempuh robot, hal ini dikarenakan penulis mengatur jarak dan node awal yang berbeda-beda.

Tabel 4 Hasil pengujian *maze solver* pada peta A

No	Berhasil menelusuri fokus kiri	Berhasil menelusuri fokus kanan	Waktu tempuh fokus kiri	Waktu tempuh fokus kanan
1	Ya	Ya	16s	15s
2	Ya	Ya	15s	15s
3	Ya	Ya	15s	16s
4	Ya	Ya	16s	15s
5	Ya	Ya	16s	16s
6	Ya	Ya	17s	17s
7	Ya	Ya	17s	17s
8	Ya	Ya	16s	16s
9	Ya	Ya	17s	16s
10	Ya	Ya	17s	16s
Rata-Rata Waktu yang Ditempuh			16.2s	15.9s

Tabel 5 Hasil pengujian *maze solver* pada peta B

No	Berhasil menelusuri fokus kiri	Berhasil menelusuri fokus kanan	Waktu tempuh fokus kiri	Waktu tempuh fokus kanan

1	Ya	Ya	26s	17s
2	Ya	Ya	25s	16s
3	Ya	Ya	26s	16s
4	Ya	Ya	27s	16s
5	Ya	Ya	26s	17s
6	Ya	Ya	25s	15s
7	Ya	Ya	27s	15s
8	Ya	Ya	25s	16s
9	Ya	Ya	25s	17s
10	Ya	Ya	25s	15s
Rata-Rata Waktu yang Ditempuh			25.7s	16s

Tabel 6 Hasil pengujian *maze solver* pada peta C

No	Berhasil menelusuri fokus kiri	Berhasil menelusuri fokus kanan	Waktu tempuh fokus kiri	Waktu tempuh fokus kanan
1	Ya	Ya	13s	29s
2	Ya	Ya	13s	29s
3	Ya	Ya	12s	28s
4	Ya	Ya	13s	29s
5	Ya	Ya	13s	30s
6	Ya	Ya	14s	30s
7	Ya	Ya	12s	28s
8	Ya	Ya	14s	29s
9	Ya	Ya	13s	28s
10	Ya	Ya	14s	28s
Rata-Rata Waktu yang Ditempuh			13.1s	28.8s

Berdasarkan hasil pengujian dari 4 tipe pengujian dari 2 algoritma yang berbeda serta 5 peta yang telah dibuat, menunjukkan bahwa robot mampu mengimplementasikan algoritma *dijkstra* dan dapat menulusuri labirin dengan baik dalam mengimplementasikan *maze solver*. Berdasarkan nilai yang didapatkan masih terdapat nilai yang beragam atau perbedaan waktu tempuh dan waktu komputasi dalam menelusuri labirin, hal ini disebabkan dari beberapa faktor antara lain sebagai berikut :

1. Perbedaan jarak antara node pada percobaan pertama dan percobaan lainnya mempengaruhi waktu komputasi algoritma dijkstra, seperti pada percobaan pertama peta A algoritma dijkstra node a ke node b mempunyai jarak 10cm sedangkan pada percobaan kedua peta A algoritma dijkstra mempunyai jarak 100cm.
2. Perbedaan lokasi node awal juga berpengaruh pada waktu tempuh dan waktu komputasi algoritma.
3. Penggunaan terus menerus sehingga daya baterai menjadi semakin berkurang sehingga membuat ketidakstabilan penyerapan daya sensor dan mengganggu kinerja sensor dalam membaca jarak dinding.
4. Perbedaan kecepatan motor driver kiri dan kanan, sehingga membuat arah pergerakan tidak lurus kedepan melainkan lurus menyamping dan menimbulkan pergerakan baru.

5. Perbedaan Arena atau Peta yang membuat lama tempuh penelusuran beragam berdasarkan metode penelusuran yang digunakan.

4. KESIMPULAN

Berdasarkan pengujian dan analisis yang dilakukan pada implementasi algoritma dijkstra dan metode maze solver pada robot penelusur labirin, maka dapat diperoleh kesimpulan sebagai berikut:

1. Robot dapat mengimplementasikan algoritma dijkstra pada robot dan dapat menelusuri peta yang sudah teridentifikasi jalurnya.
2. Robot dapat menelusuri labirin dan mampu mengimplementasikan metode maze solver pada peta yang teridentifikasi dan peta yang tidak teridentifikasi.
3. Kecepatan waktu penelusuran labirin disebabkan oleh beberapa faktor, antara lain medan arena dan metode yang digunakan sehingga tiap penelusuran dapat menghasilkan waktu yang cepat maupun waktu yang lama.

5. SARAN

Adapun beberapa saran yang bisa diberikan untuk pengembangan lebih lanjut terhadap penelitian ini adalah sebagai berikut:

1. Robot yang dibuat masih bisa dikembangkan lebih lanjut dan dapat menggunakan algoritma yang lain seperti algoritma a* dan algoritma Two Queues.
2. Metode *maze solver* dapat digunakan pada peta yang belum teridentifikasi untuk mendapatkan pencarian jalan keluar dan dapat di implementasikan pada program yang mempunyai kasus yang sama.
3. Algoritma dijkstra hanya dapat digunakan dalam kondisi peta telah diketahui sebelumnya dan masih bisa dikembangkan kedepannya untuk kondisi peta yang belum teridentifikasi atau secara real time.

DAFTAR PUSTAKA

[1] Maarif, A., 2014, Perancangan Line Maze Solving Robot Dengan Algoritma Short Path Finder, *Skripsi*, Program Studi Teknik Elektro, Unversitas Islam Indonesia, Yogyakarta.

[2] Yanto, F., dan Welly, I., 2015, Analisa dan Perbaikan Algoritma Line Maze Solving Untuk Jalur Loop, Lancip, dan Lengkung pada Robot Line Follower (LFR), *Jurnal CoreIT*, Vol.1, No.2, Desember 2015, Universitas Islam Negeri Sultan Syarif Kasim Riau, Riau.

[3] Retnati, W. E. Y., Istiadi, D., dan Roqib, A., 2015, Pencarian SPBU Terdekat Dan Penetuan Jarak Terpendek Menggunakan Algoritma Dijkstra (Studi Kasus Kabupaten Jember), *Jurnal Nasional Teknik Elektro* Vol : 4, No : 1, Maret 2015, Universitas Jember, Jember.

[4]Nurmalasari, M., Triyanto, D. & Brianorman, Y. 2015. Implementasi Algoritma Maze Solving Pada Robot Line Follower, *Jurnal Coding Sistem Komputer Untan*, Volume 03 Nomor 2 (2015), Universitas Tanjungpura, Pontianak.

[5]Saralita, M. 2015. Robotika - Drop-BOT : Robot Pengirim Barang. *https://www.academia.edu/29590052/Robotika_-_Drop-BOT_Robot_Pengirim_Barang*, diakses tgl 15 Desember 2018.

[6]Hasyim, Y. & Putri, A.R. 2017. Implementasi Sistem Navigasi Wall Following Dengan Metode Fuzzy Logic Untuk Robot Pemadam Api Divisi Berkaki Onix II Pada KRPAI Tahun 2017, *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)* Volume 02, Nomor 01, Mei 2017 : 26 – 31, STKIP PGRI Tulungagung Jalan Major Sujadi Timur 7, Tulungagung.

[7] Rasyid, M,A., Firdaus & Derisma. 2016. Rancang Bangun Robot Pengering Lantai Otomatis Menggunakan Metode Fuzzy, *jsiskom Jurnal Sistem Komputer* Vol.6, No.2, 2016, Universitas Diponegoro, Semarang.

[8]Rohmanu, A. & David, W. 2018. Sistem Sensor Jarak Aman Pada Mobil Berbasis Mikrokontroller Arduino ATMEGA328, *Jurnal Informatika SIMANTIK* Vol. 3 No. 1. STMIK Cikarang, Cikarang.

[9]Muchtar, H. & Hidayat, A. 2017. Implementasi WAVECOM Dalam Monitoring Beban Listrik Berbasis Mikrokontroler, *Jurnal Teknologi Universitas Muhammadiyah Jakarta* Vol. 9 No. 1, Januari 2017, Universitas Muhammadiyah Jakarta, Jakarta.

[10] Pribadi, O. 2015. Maze Generator Dengan Menggunakan Algoritma Depth-First-Search, *Jurnal TIMES* Vol. IV No 1 : 1-5 , 2015, STIMIK TIME, Medan.