ANALISIS DAN REALISASI ROBOT MAZE SOLVER MENGGUNAKAN ALGORITMA DEPTH FIRST SEARCH

ANALYSIS AND REALIZATION OF MAZE SOLVER ROBOT USING DEPTH FIRST SEARCH ALGORITHM

Ahmad Syarif Hidayatullah[1], Agung Nugroho Jati[2], Casi Setianingsih[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik, Universitas Telkom

[1] Asyarifhidayatullah@telkomuniversity.ac.id, [2] Agungnj@telkomuniversity.ac.id, [3]Setiacasie@telkomuniversity.ac.id

Abstrak

Robot *maze solver* adalah salah satu robot autonomous yang membutuhkan algoritma. Algoritma adalah bagian penting dari robot autonomous. Robot *maze solver* berjalan dan mencari jalan menuju *finish* pada sebuah labirin. *Depth first search* adalah sebuah algoritma yang memiliki opsi backtracking yang sangat membantu robot *maze solver* untuk kembali ke simpul sebelumnya apabila robot menemui jalan buntu. Pada tugas akhir ini, dilakukan realisasi dan analisis terhadap robot *maze solver* dengan algoritma *depth first search* dalam menyelesaikan beberapa jenis labirin yang berbeda. Dengan menggunakan *right hand rule*, maka prioritas belokan robot ini adalah sebelah kanan. Fitur backtracking juga digunakan untuk mengatasi jalan buntu. Hasil dari penelitian ini, robot *maze solver* dengan algoritma *depth first search* menggunakan *right hand rule* memiliki kemampuan yang baik dalam menyelesaikan labirin. Hal ini ditunjukan dengan hasil pengujian dimana persentase keberhasilan sistem dalam menyelesaikan beberapa labirin adalah 83,3 persen dan rata-rata waktu penyelesaian 88,6 detik

Kata kunci : *Maze solver* Robot, Depth First Search

Abstract

Maze solver robot is an example of autonomous robot that using algorithm. Algorithm is an important part of autonomous robot. *Maze* solver robot used to trace and find way to *finish* from a *maze*. Depth first search is an algorithm which has a backtracking option that totally help the *maze* solver robot to going back to previously vertex of the *maze* if the robot finds *dead end*. In this final task, *maze* solver robot using depth first search algorithm are analyzed when tracing several kinds of *maze*. Using right hand rule, the priority turn of this robot is right. Backtracking feature will be used when robot finds *dead end*. Result of this final task is *maze* solver robot using depth first search algorithm and right hand rule has good ability to *finish* the *maze*. It can be shown by result when system tested to *finish* several *maze*s with 83,3 percentage of success and 88,6 second average *finish* time.

Keywords: *Maze solver* Robot, Depth First Search

1. Pendahuluan

Sejak robot dibuat dapat melakukan tugas-tugas secara mandiri, bidang Autonomous Mobile Robot semakin marak dikembangkan dalam banyak penelitian. Meskipun robot dengan sensor berbasis citra menjadi tren pada saat ini, namun robot line following masih banyak digunakan dalam berbagai penelitian. Robot line following digunakan karena memiliki desain rangkaian yang sederhana dan biaya pembuatan yang murah [1]. Robot line following ini juga dapat dikembangkan menjadi robot *maze solver* dengan ditambahkan algoritma kedalam robot tersebut.

Robot *maze solver* adalah robot yang dapat menemukan jalan keluar dari sebuah labirin dengan cepat dan tepat. Agar robot *maze solver* dapat menemukan jalan keluar dengan cepat dan tepat robot ini harus ditunjang dengan sensor dan algoritma yang tepat sesuai labirin yang akan digunakan. Salah satu algoritma yang dapat digunakan adalah algoritma depth first search. Algoritma ini mendukung fungsi *backtracking* sehingga robot dapat menjelajahi labirin tanpa takut tersesat pada jalan buntu [2].

Pada akhirnya robot *maze solver* ini dapat digunakan untuk membantu pekerjaan manusia terutama pekerjaan berulang. Robot ini dapat diimpelementasikan sebagai pembawa barang pada pergudangan ataupun sebagai robot untuk mencari longsor pada pertambangan bawah tanah.

2. Dasar Teori

2.1. Graph

Graph adalah kumpulan dari simpul(*vertex*) dan busur(*edge*). Biasanya graph digambarkan dengan kumpulan titik-titik yang dihubungkan oleh garis. Secara matematis graph dinyatakan sebagai:

G=(V,E)

Dimana:
G= Graph E= Busur
V= Simpul

Graph traversal adalah sebuah proses dimana setiap simpul dikunjungi secara sistematik. Pada *graph* traversal terdapat dua metode penelusuran yaitu *Depth first search* dan *Breadth First Search*. Hasil dari penelusuran dua metode tersebut adalah *path*. *Path* adalah serangkaian simpul-simpul berbeda yang *adjacent* secara berturut-turut dari satu simpul ke simpul lainnya [3].

2.2. Algoritma Depth First Search

Pada umumnya suatu persoalan memiliki lebih dari satu kemungkinan solusi. Labirin merupakan salah satu contoh persoalan yang dapat kita temui sehari-hari. Terdapat banyak cara penyelesaian suatu persoalan. Salah satu teknik penyelesaian persoalan adalah pencarian menggunakan pohon berakar *graph* ,pencarian solusi dilakukan dengan menelusuri simpul-simpul dalam pohon. Setiap simpul diperiksa hingga solusi ditemukan. Salah satu cara pembentukan *graph* adalah *depth first search* [3].

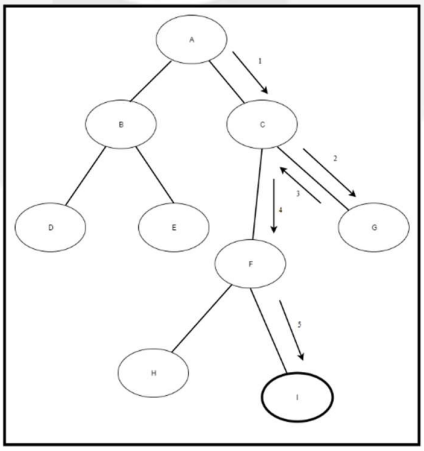
Seperti namanya algoritma *depth first search* dilakukan dengan mencari masuk ke simpul yang paling dalam dahulu. Apabila disimpul paling dalam tidak ditemukan tujuannya *finish*, maka pencarian akan kembali ke simpul sebelumnya dan mencari ke busur sebelahnya (*backtrack*). Pencarian ini berlanjut sampai *finish* ditemukan [4].

Pada algoritma *depth first search* terdapat dua opsi untuk pemilihan simpul yang akan diambil. Dua opsi tersebut adalah *right hand rule* dan left hand rule. Pada *right hand rule* simpul prioritas adalah sebelah kanan, simpul sebelah kanan akan dipilih dibanding lurus dan kiri. Apabila kanan tidak ada maka lurus akan dipilih dibanding kiri. Simpul kiri hanya akan dipilih jika simpul tersebut satu-satunya simpul yang tersedia. Left hand rule berlaku sebaliknya dari *right hand rule*. Hanya satu dari kedua opsi ini yang bisa dipilih dan dilakukan secara konsisten [2].

Pada algoritma *depth first search* terdapat beberapa istilah yang akan dipakai untuk menyelesaikan suatu skema pencarian. Istilah-istilah tersebut adalah sebagai berikut:

1. Start, start adalah titik awal dimana pencarian akan dimulai.
2. *Goal*/*Finish*, *goal* adalah simpul tujuan yang akan dicari solusinya.
3. Current state, current state adalah simpul yang akan diperiksa apakah simpul tersebut merupakan *goal*.
4. *Dead end*, *dead end* adalah sebuah simpul yang tidak memiliki *edge* lanjutan (buntu).

Berikut ini adalah contoh skema pencarian pada algoritma *depth first search* menggunakan *right hand rule*, dimana A adalah start dan I adalah *goal*.



Gambar 2.1 Diagram Alir Algoritma Depth First Search

Pada skema pencarian diatas,dari start simpul A algoritma akan memilih *edge* sebelah kanan terlebih dahulu menuju simpul C. Karena C bukanlah *goal*, maka *edge* sebelah kanan akan dipilih lagi menuju simpul G. Pada simpul G tidak terdapat *edge* sehingga simpul G merupakan *dead end* atau buntu. Saat algoritma menemukan *dead end*, maka *current state* akan dikembalikan ke simpul sebelumnya dan memilih *edge* sebelahnya. Maka *current state* untuk sekarang adalah simpul F. Karena simpul F juga bukan merupakan *goal* maka akan dipilih *edge* sebelah kanan lagi. Ketika *current state* sama dengan *goal*, maka solusi dari skema pencarian sudah ditemukan. Solusi dari skema pencarian diatas adalah A-C-F-I.

Kelemahan dari algoritma *depth first search* adalah ketika *finish* berada pada simpul non-prioritas dan jauh didalam simpul lain maka waktu pencarian akan berlangsung lama. Algoritma *depth first search* tidak dapat mengenali lingkungan sekitarnya karena pada dasarnya algoritma *depth first search* merupakan algoritma pencarian [5].

2.3. Labirin

Labirin adalah jaringan jalan yang rumit dan berliku-liku. Pada robotika terdapat berbagai jenis labirin, yaitu labirin garis, labirin dinding dan gabungan keduanya. Pada tugas akhir ini akan digunakan labirin garis. Labirin garis biasanya adalah garis hitam diatas latar putih dapat juga sebaliknya. Labirin garis digunakan karena biaya pembuatan dan sensor yang digunakan untuk jenis labirin garis lebih murah dibanding labirin dinding [6] [7]

2.4. Arduino UNO

Arduino UNO adalah board mikrokontroller yang menggunakan ATmega328. Mikrokontroller ini terdiri dari 14 pin digital input/output, 6 input analog, USB connection, power jack, header ICSP, dan tombol reset. Arduino UNO diberi daya melalui koneksi USB atau external power supply yang berasal dari AC-to-DC adapter atau dapat menggunakan baterai. ATmega328 memiliki 32 KB flash memory untuk menyimpan code dengan kondisi 0,5 KB digunakan sebagai bootloader, 2 KB SRAM (Static Random Access Memory) dan 1 KB EEPROM (Electrically Erasable Programmable Read-Only Memory).

Gambar 2.2 Arduino UNO
(Sumber arduino.org Diakses tanggal 28 Mei 2017 20:52)

2.5. Driver Motor Shield L293D

Driver motor adalah suatu rangkaian yang memiliki fungsi mengatur arah dan kecepatan pada motor DC. Driver motor di desain untuk level TTL logic, mendorong beban induktif seperti relay, DC, dan stepping motors. Perangkat in memerlukan daya mulai dari 5V hingga 46V dengan arus 2A.

Gambar 2.3 Motor Driver Shield L293D
(Sumber playground.arduino.cc Diakses tanggal 28 Mei 2017 20:54)

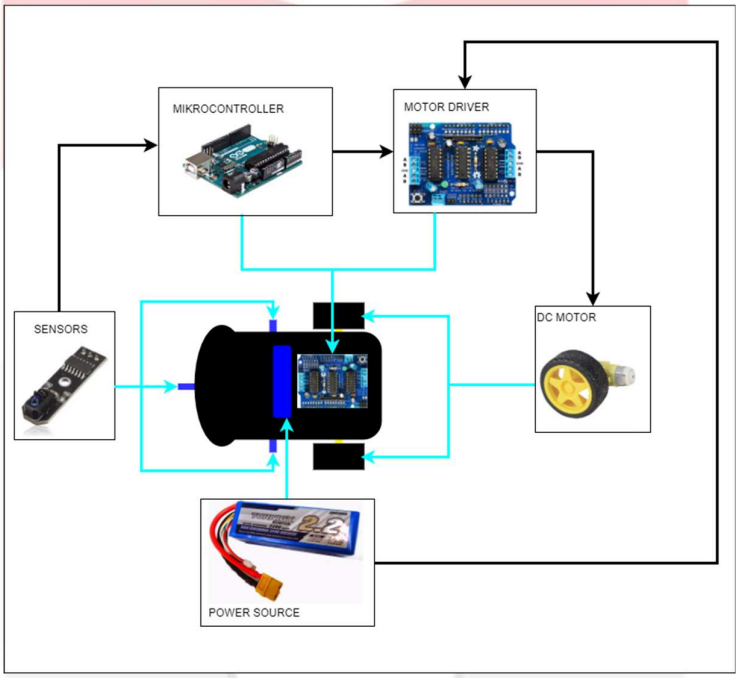
2.6. TCRT 5000 IR Sensor

TCRT 5000 IR Sensor adalah IR Emitter dan IR Phototransistor yang bekerja bersamaan. Sensor ini menerima pantulan cahaya inframerah untuk mendeteksi warna hitam dan putih berdasarkan kontras objek dan sifat reflektifnya.

Gambar 2.4 TCRT 5000 IR Sensor [10]
(Sumber mgsuperlabs.co.in Diakses tanggal 28 Mei 2017 21:09)

3. Pembahasan

3.1. Deskripsi Sistem



Gambar 3.1 Diagram Umum Sistem

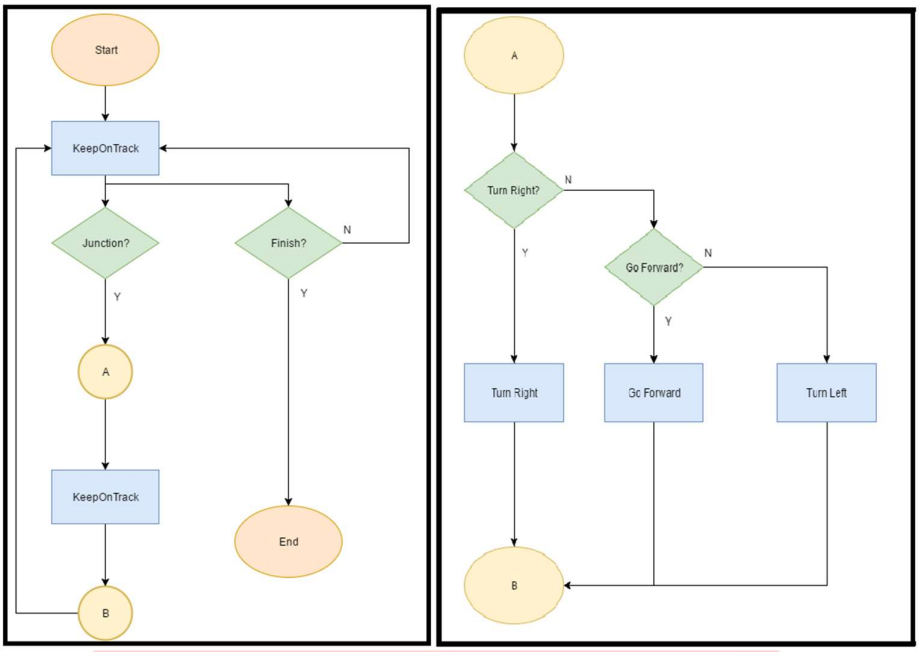
Robot *maze solver* ini akan mencari jalur dari start hingga *finish* menggunakan algoritma depth first search. Robot ini menggunakan sensor garis (IR sensor) untuk mendeteksi garis pada labirin. Sensor akan memberikan input kepada mikrokontroller. Pada mikrokontroller input dari sensor akan diproses sesuai perancangan algoritma yang akan dibuat. Hasil proses dari mikrokontroller akan memberikan input terhadap motor driver. Output dari mikrokontroller berupa berbagai perintah sesuai kondisi yang didapat dari sensor. Perintah-perintah yang didapatkan oleh motor driver akan menggerakkan DC motor sesuai perintah. Seluruh robot mendapatkan daya dari baterai LiPo.

3.2. Perancangan Sistem

3.2.1. Perancangan Perangkat Lunak dan Algoritma

Robot *maze solver* ini memiliki 3 bagian dalam perancangan perangkat lunak dan algoritmanya.Bagian pertama dari perangkat lunak robot ini adalah *Keep On the Line*. Bagian ini merupakan fitur dari robot line follower dimana fitur ini mempertahankan robot tetap berada diatas garis hitam.

Bagian kedua dari perangkat lunak robot ini adalah Putar balik. Bagian ini akan memutar robot 180º dan mengembalikan robot ke simpul sebelumnya. Bagian ini akan bekerja ketika robot menemui jalan buntu.Bagian terakhir adalah algoritma *depth first search* sendiri. Pada robot ini menggunakan algoritma *depth first search* dengan *right hand rule*.



Gambar 3.2 Diagram Alir

3.2.2. Perancangan Perangkat Keras dan Tata Letak Sensor

Pada perancangan perangkat keras dan tata letak sensor robot *maze solver* ini terbagi atas dua bagian. Bagian-bagian tersebut adalah sebagai berikut:

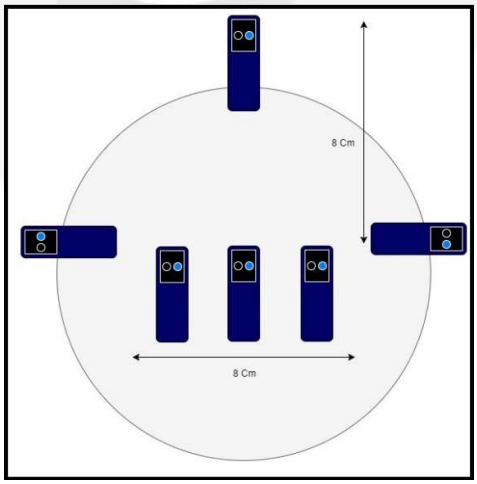
1. Perangkat Keras

Body dari robot *maze solver* ini dibuat menggunakan *body* kit akrilik. *Body* robot sendiri terbagi atas dua bagian, yaitu bagian *body* dan bagian sensor. Pada bagian *body* dipasang 2 dua buah motor DC, 1 buah mikrokontroller, motor driver, baterai LiPo dan juga *breadboard*.

Mikrokontroller yang digunakan pada tugas akhir ini adalah Arduino UNO. Pada Arduino UNO port digital 2,5,6,9,10 dan 13 digunakan sebagai port untuk inputan dari sensor. Motor driver yang digunakan adalah L293D Shield. Pada motor driver port yang digunakan adalah M1 dan M2 sebagai port output untuk motor DC. Daya didapatkan dari baterai LiPo yang dihubungkan pada input daya eksternal yang terdapat pada motor driver.

2. Tata Letak Sensor

Robot *maze solver* ini menggunakan 6 sensor garis. 3 sensor untuk algoritma *depth first search*, 2 sensor untuk bagian *Keep On the Line* dan 1 sensor untuk bagian putar balik. Karena lebar garis pada labirin yang akan digunakan adalah 8 cm, maka tata letak sensor-sensor tersebut adalah sebagai berikut:



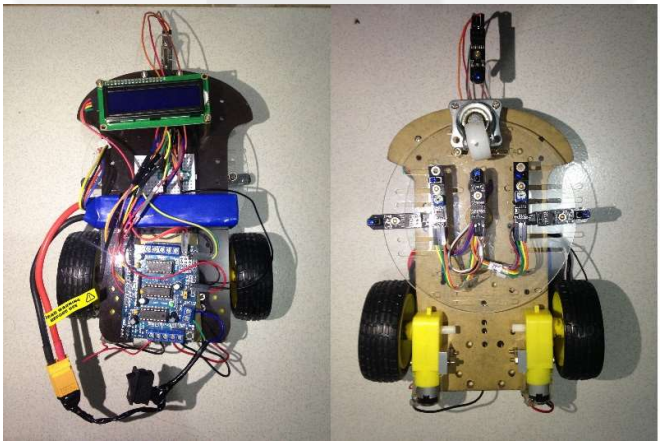
Gambar 3.3 Tata Letak Sensor

3.3. Implementasi Sistem

Perangkat lunak dalam hal ini sistem dan algoritma diolah dengan Arduino IDE. Bahasa pemprograman pada Arduino IDE sendiri mirip seperti bahasa C. Perancangan perangkat lunak dan algoritma diimplementasikan sebagai berikut:

```
1: KeepOnTheLine(active)
2: if (S.Kanan ≡ hitam)and(S.Tengah ≡ hitam)and(S.Kiri ≡ hitam) then
3:     KeepOnTheLine(stop)
4:     Robot belok kanan
5: else
6:     if (S.Kanan ≡ hitam)and(S.Tengah ≡ hitam)and(S.Kiri ≡ putih) then
7:         KeepOnTheLine(stop)
8:         Robot belok kanan
9:     else
10:        if (S.Kanan ≡ hitam)and(S.Tengah ≡ putih)and(S.Kiri ≡ putih) then
11:            KeepOnTheLine(stop)
12:            Robot Belok kanan
13:        else
14:            if (S.Kanan ≡ putih)and(S.Tengah ≡ hitam)and(S.Kiri ≡ hitam)
   then
15:                KeepOnTheLine(stop)
16:                Robot lurus
17:            else
18:                if (S.Kanan ≡ putih)and(S.Tengah ≡ putih)and(S.Kiri ≡
   hitam) then
19:                    KeepOnTheLine(stop)
20:                    Robot Belok kiri
21:                else
22:                    if S.putarbalik ≡ putih then
23:                        KeepOnTheLine(stop)
24:                        TurnAround
25:                    end if
26:                end if
27:            end if
28:        end if
29:    end if
30: end if
```

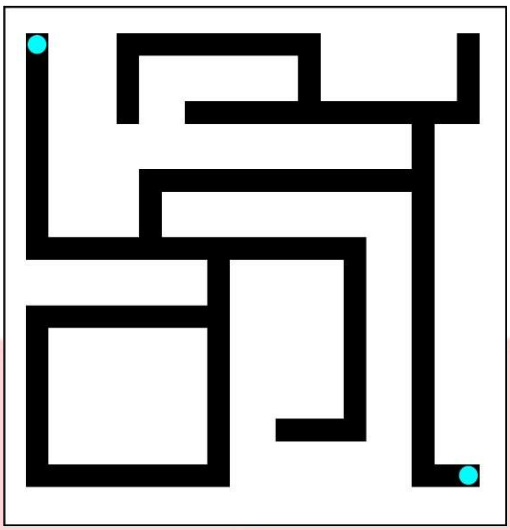
Perancangan *body* robot diimplementasikan sebagai berikut:



Gambar 3.4 Implementasi Robot

4. Pengujian dan Analisis

Untuk melakukan pengujian robot dan algoritma, dibutuhkan sebuah *maze*. *Maze* tersebut harus dapat menguji semua fitur yang ada pada sistem robot. *Maze* tersebut dibuat seperti yang terlihat pada gambar berikut:



Gambar 4.1 Labirin Pengujian

Robot diuji sebanyak 10 kali dengan voltase input antara 10,7 V sampai 11,2 V. Dari pengujian tersebut dicatat waktu dan keberhasilan robot untuk menyelesaikan labirin. Hasil dari pengujian robot terlihat seperti yang ada pada tabel 4.1.

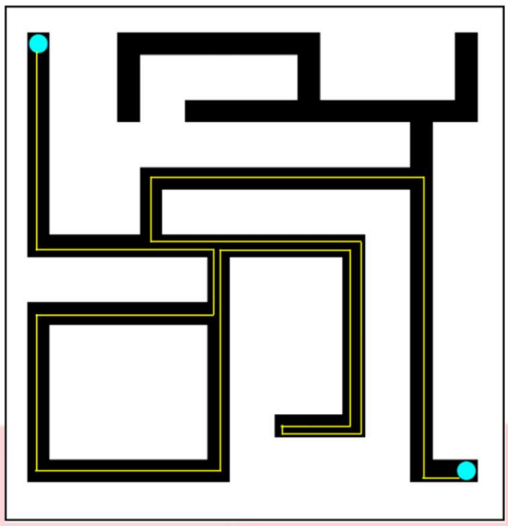
Tabel 4.1 Pengujian Robot pada labirin

Pengujian Ke-	Waktu(detik)	Result
1	89	Berhasil
2	-	Gagal
3	99	Berhasil
4	108	Berhasil
5	133	Berhasil
6	130	Berhasil
7	150	Berhasil
8	121	Berhasil
9	-	Gagal
10	126	Berhasil

Dari 10 kali pengujian, robot dapat menyelesaikan 8 kali dengan berhasil dengan rata-rata waktu 119.5 detik. Pada 3 pengujian lainnya, robot gagal mencapai *finish* dikarenakan terjadinya kerancuan informasi yang didapat dari sensor. Contohnya ketika robot melewati persimpangan yang hanya memiliki opsi belok kanan atau kiri, ketika robot mencapai persimpangan tersebut dengan posisi yang tidak lurus maka sensor hanya mendapatkan informasi untuk belok kiri sedangkan garis pada sebelah kanan terlambat diinformasikan.

Salah satu kegagalan pembacaan sensor lain adalah ketika robot menemui belokan ke kanan, sedangkan kapasitas dari baterai lebih tinggi dari pada saat dilakukan kalibrasi terhadap labirin. Maka robot akan terlewat membaca garis pada sebelah kanan dan menganggap belokan tersebut adalah jalan buntu lalu melakukan putar balik.

Berikut ini adalah *path* yang dihasilkan oleh sistem robot saat dilakukan pengujian:



Gambar 4.2 *Path* hasil pengujian

Dari *path* diatas dapat kita lihat bahwa algoritma *depth first search* berjalan dengan baik dalam menentukan jalur yang dipilih robot. Dengan *right hand rule* algoritma memprioritaskan belok kanan dibanding berjalan lurus dan belok kiri. Sehingga pada *path* diatas robot berputar terlebih dahulu dan menemukan *dead end*, lalu kembali ke simpul sebelumnya dan menemukan jalur menuju *finish*.

5. Kesimpulan dan Saran

5.1. Kesimpulan

Dari hasil yang didapatkan dari penelitian ini, didapatkan kesimpulan sebagai berikut:

1. Sistem robot *maze solver* dengan algoritma *depth first search* mampu bekerja dengan baik. Dari 10 kali pengujian persentase keberhasilannya adalah 80 persen dengan rata-rata waktu 119.5 detik.
2. Algoritma *depth first search* dapat menyelesaikan *maze* dengan *loop* selama *finish* tidak berada pada *edge* non-priotitas dan terhubung pada *loop*.
3. Berdasarkan hasil penelitian ini, algoritma *depth first search* mampu untuk melakukan pencarian jalan keluar pada suatu lingkungan (*maze*) yang tidak diketahui.

5.2. Saran

Saran untuk penelitian selanjutnya adalah:

1. Pada penelitian ini digunakan sistem robot tidak dapat mendeteksi adanya *loop* pada labirin, yang mengakibatkan robot tidak dapat menyelesaikan labirin. Oleh sebab itu, penelitian selanjutnya dapat menambahkan method baru untuk dapat mendeteksi *loop*.
2. Pada penelitian ini sistem robot tidak dapat mengenali atau memetakan lingkungan sekitarnya. Pada penelitian selanjutnya dapat ditambahkan fitur pemetaan dengan menggunakan mikrokontroller yang lebih baik lagi.

Daftar Pustaka:

[1] Yichao Li, Xiaoling Wu and Dongik Shin.2012. *Improved Line Following Optimization,*Algorithm for Mobile Robot. 7th International Conference of Computing and Convergence Tecnology (ICCCT), 2012.

[2] Md. Sazzad Mahmud, Ujjal Sarker, Md. Monirul Islam, Prof. Dr. Hasan Sarwar. 2012. *A Greedy Approach in Path Selection for DFS Based Maze-map Discovery Algorithm for an Autonomous Robot*. Department of CSE, United International University, Bangladesh.

[3] Yogie Noviyanto, 2008. *Analisis Dan Implementasi Engine Constraint Satisfaction Problem (CSP) Dengan Menggunakan Algoritma Depth- First Search (DFS) With Backtrack dan Heuristic Most Constraining Variable*, Fakultas Informatika, Telkom Univ., Bandung, Indonesia

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. 2009. *Introduction to Algorithms*, 3rd Edition, MIT Press, Cambridge, Massachusetts London, England, pp.603-623.

[5] Aulian M. Fathan, Agung Nugroho Jati and Rendy Efra Saputra.2016. *Mapping algorithm using ultrasonic and compass sensor on autonomous robot.* IEEE International Conference on Control, Electronic,Renewable Energy and Communications

[6] Mehran Pakdaman and M. Mehdi Sanaatiyan. 2009. *Design and Implementation of line follower robot.* IEEE International conference on computer and Electrical Engineering

[7] M. Iqbal Nugraha, Akhmad Herdiawan and Reesa Akbar. *Penerapan Algoritma Maze Mapping untuk menyelesaikan maze pada line tracer*. Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia.