Analisa dan Perbaikan Algoritma *Line Maze Solving* Untuk Jalur *Loop*, Lancip, dan Lengkung pada Robot *Line Follower (LFR)*

Febi Yanto[1], Irma Welly[2]
[1,2]Teknik Informatika UIN Sultan Syarif Kasim Riau
Jl. H.R. Soebrantas no. 155 KM. 18 Simpang Baru, Pekanbaru 28293
febiyanto@uin-suska.ac.id[1], irmawelly12@gmail.com[2]

***Abstrak* – Pengujian robot *line follower* pada penelitian ini menggunakan arduino uno. Pada penelitian sebelumnya algoritma *line maze solving* yang digunakan sebagai metode pada robot *line follower* masih menyisakan kendala, di mana robot belum bisa melewati *maze* dengan jalur *loop*, jalur lengkung lebih dari 225º, dan jalur lancip kurang dari sudut 45º. Oleh karena itu dilakukan perbaikan, dan didapatkan algoritma yang baru. Algoritma ini kemudian diuji kembali pada robot dan berhasil menyelesaikan *maze* dengan jalur *loop*, jalur lengkung lebih dari 225º, dan jalur lancip kurang dari 45º.**

Kata kunci – algoritma line maze solving, arduino uno, jalur lancip, jalur lengkung, jalur loop, line maze, robot

PENDAHULUAN

Robot *line tracer* atau *line follower* adalah robot yang bergerak dengan mengikuti jalur berupa garis berwarna hitam di atas permukaan berwarna putih atau sebaliknya [1]. Robot ini juga dapat digunakan dalam menyelesaikan *line maze*. Penyelesaian labirin (*maze*) dapat dilakukan dengan berbagai algoritma, di antaranya adalah algoritma *line maze solving*, algoritma *flood fill*, dan algoritma *pledge*. Algoritma *flood fill* adalah salah satu metode pemecahan labirin yang digunakan pada robot *micromouse*. Dalam algoritma *flood fill*, posisi start pada robot tidak bisa dipindah posisikan. Algoritma *pledge* merupakan algoritma yang memposisikan garis lurus sebagai garis *finish* dalam memecahkan labirin. Dalam algoritma *pledge*, apabila robot telah melewati garis *finish*, maka robot akan terjebak dalam *maze.* Algoritma *line maze solving* merupakan algoritma yang digunakan pada robot untuk mencari jalur terpendek dari sebuah *line maze*.

Terdapat 2 aturan dalam algoritma *line maze solving*, yaitu *left hand rule* dan *right 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. Sebaliknya dalam *right hand rule*, robot akan lebih memilih belok kanan dari pada lurus atau kiri dan lebih memilih lurus dari pada belok kiri [2].

Penelitian yang telah dilakukan sebelumnya adalah mengenai penggunaan algoritma *line maze solving* pada robot *line tracer* dalam menyelesaikan sebuah *maze* oleh Richard T. Vannoy II. Penelitian tersebut hanya dilakukan pada *maze* dengan jalur lurus dan belokan serta persimpangan dengan sudut 90˚. Kemudian robot *line follower* diteliti lanjut oleh Raja Joko Musridho dan robot berhasil melewati jalur *zig-zag* dan lancip [3].

Penelitian dari sebelumnya masih menyisakan beberapa kendala di mana robot masih belum bisa bergerak di jalur lengkung dengan sudut lebih dari 225º, jalur lancip kurag dari 45º, dan jalur *loop*. Maka dalam penelitian yang penulis lakukan adalah "Analisa dan perbaikan algoritma *line maze solving* untuk jalur *loop*, jalur lancip, dan lengkung pada robot *line follower* (LFR)".

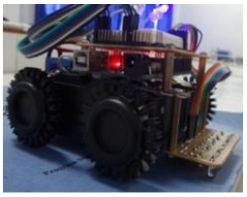
LANDASAN TEORI

A. Robot

Robotika adalah multidisiplin ilmu yaitu komputer, elektronika, dan mekanika [1]. Robotika merupakan satu cabang teknologi yang berhubungan dengan desain, konstruksi, operasi, disposisi struktural, pembuatan, dan aplikasi dari robot.

B. *Line Follower Line Follower*

Robot merupakan robot yang bisa bergerak mengikuti garis panduan yaitu garis putih yang diletakkan di atas permukaan berwarna hitam, atau sebaliknya garis hitam yang diletakkan di atas permukaan berwarna putih[4]. Cara kerja robot *line tracer* atau *line follower* robot yaitu dengan menggunakan sensor garis seperti *infrared* yang mendeteksi adanya garis pada permukaan lintasan robot, dan informasi yang telah diterima oleh sensor garis akan diteruskan ke penggerak atau motor tersebut agar motor dapat menyesuaikan dengan gerak tubuh robot sesuai dengan garis yang dideteksi.



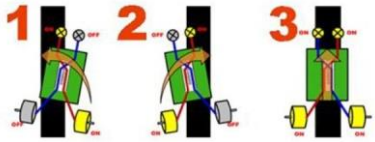
Gambar 1. *Line Follower*

Robot *line tracer* memiliki dua buah sensor (A-kiri dan B-kanan) yang terhubung dengan kedua motor (kiri dan kanan) secara bersilangan melalui sebuah *processor/driver*. Berikut penjabarannya:

a. Ketika sensor A mendeteksi garis sedangkan sensor B tidak, ini berarti posisi robot berada di sebelah kanan garis. Sehingga motor kanan akan aktif sementara motor kiri tidak, sehingga robot akan bergerak ke kiri.

b. Jika sensor B mendeteksi garis dan sensor A tidak, ini berarti robot berada di sebelah kiri garis. Maka motor kiri akan aktif dan motor kanan mati, sehingga robot akan bergerak ke kanan.

c. Jika sensor A dan sensor B sama-sama mendeteksi garis, artinya robot berada di atas garis. Maka kedua motor akan aktif dan robot bergerak lurus.



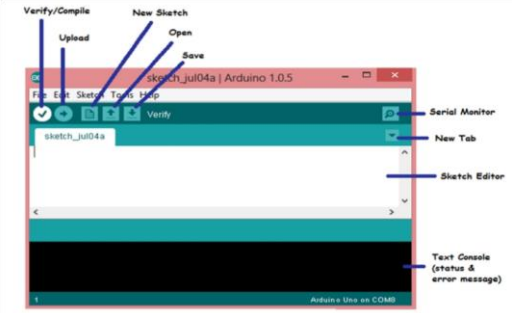
Gambar 2. Penjabaran Sederhana Cara Kerja *Line Tracer*

C. Arduino

Arduino terbagi menjadi dua, yaitu *hardware* dan *software*[5]. *Hardware* merupakan papan input/output (I/O) dan *software* meliputi penulis program berupa *Integrated Development Environment* (IDE), penghubung (bukan secara fisik) dengan computer berupa *driver*, serta pengembang program berupa contoh program dan *library* [6]. 1. *Hardware*: Arduino Uno (ATMega 328) Robot *line tracer* yang dijadikan sebagai media penelitian ini menggunakan papan *microcontroller* Arduino Uno yang berbasis ATMega328.

Gambar 3. Arduino Uno

Software: IDE Arduino 1.0.5 IDE Arduino terdiri dari *editor* program, *compiler*, dan *uploader*. *Editor* program berfungsi sebagai tempat bagi pengguna untuk menulis dan mengedit program. Karena *microcontroller* hanya memahami kode biner, maka diperlukan *compiler* untuk mengubah kode program menjadi kode biner. Dan agar kode biner bisa dimuat ke dalam *microcontroller*, maka dibutuhkan sebuah modul yang bernama *uploader*. Kode program sering juga disebut dengan istilah *sketch* [6].



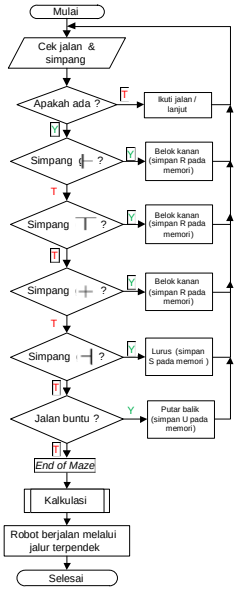
Gambar 4. Gambar Penjelasan Singkat *Tools* Arduino 1.0.5

D. Algoritma *Line Maze Solving*

Algoritma *line 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 *left hand rule* dan *right 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. Sebaliknya dalam *right hand rule*, robot akan lebih memilih belok kanan dari pada lurus dan lebih memilih lurus dari pada belok kiri. Dengan adanya line maze solving, robot bisa memecahkan labirin (*maze*) dengan berbagai cara seperti *left hand rule* dan *right hand rule* [2]. Robot yang telah diprogram dengan algoritma ini diletakkan di titik start sebuah *line maze*. Kemudian saat menjumpai persimpangan, robot akan memilih untuk berbelok sesuai dengan prioritas dari rule yang digunakan. Hal ini terus berlanjut sambil robot menyimpan/merekam jalur yang telah dilewati ke dalam memori hingga robot menemui akhir maze (*end of maze*).

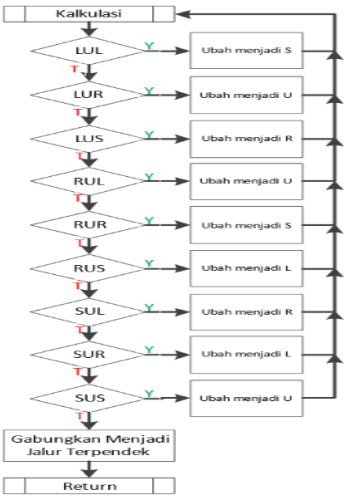
Proses perekaman jalur ini disebut sebagai *mapping*. Setelah selesai, robot langsung melakukan kalkulasi/perhitungan atas jalur yang telah tersimpan dalam memori robot untuk mendapatkan jalur terpendek dalam menyelesaikan *maze* tersebut pada penelusuran ke dua kalinya [7].

flowchart dari algoritma *line maze solving* dapat dilihat pada Gambar 5.



Gambar 5. *Flowchart* Algoritma *Line Maze Solving* dengan *Right Hand Rule*

Setelah selesai, robot langsung melakukan kalkulasi atas jalur yang telah tersimpan dalam memori robot untuk mendapatkan jalur terpendek dalam menyelesaikan *maze* tersebut pada penelusuran ke dua kalinya. Berikut adalah *flowchart* proses kalkulasi tersebut.

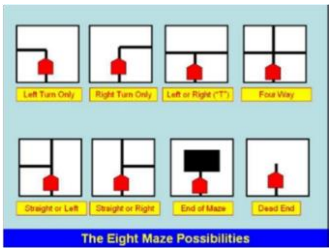


Maze Solving

Gambar 6. *Flowchart* Kalkulasi Algoritma *Line*

Terdapat 8 kemungkinan dalam *maze* yang ia sebut dengan "*The Eight Maze Possibilities*" [2]. Kedelapan kemungkinan tersebut adalah sebagai berikut:

1. tidak ada pilihan lain selain belok kiri (*left turn only*),
2. tidak ada pilihan lain selain belok kanan (*right turn only*),
3. kiri atau kanan (*left or right*),
4. kiri, lurus, atau kanan (*four way*),
5. lurus atau kiri (*straight or left*) ,
6. lurus atau kanan (*straight or right*),
7. jalan buntu (*dead end*) ,
8. *End of maze.*



Gambar 7. *The Eight Maze Possibilities*

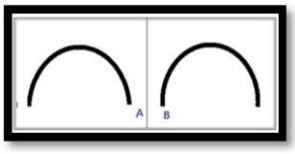
Left turn only dan *right turn only* bukanlah merupakan sebuah persimpangan, sebab hanya ada satu belokan. Sehingga robot tidak menyimpan apapun ke dalam memori. *Dead end* juga bukan sebuah persimpangan, namun karena *dead end* menandakan bahwa jalur yang dipilih adalah salah maka robot melakukan *U-turn* atau putar balik dan menyimpannya ke dalam memori.

ANALISA

A. *Maze Possibilities*

Jalur Lengkung Jalur lengkung yang akan digunakan pada robot *line follower* kali ini adalah jalur lengkung yang sudut pergeserannya melebihi sudut 225º. Pada jalur lengkung ini terdapat 2 *maze possibilities* seperti berikut:

A = kiri lingkar
B = kanan lingkar

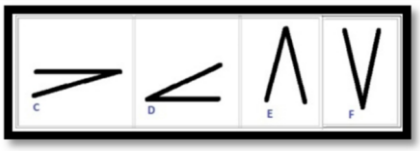


Gambar 8. Dua Maze Possibilities Jalur Lengkung

B. *Maze Possibilities* Jalur Lancip

Pada analisa *maze possibilities* jalur lancip, penulis membatasi pembahasan gambar analisa agar tidak menimbulkan banyak hasil perbedaan pada sensor yang ada dalam robot *line tracer* atau *line follower*. Adapun gambaran analisa dari *maze possibilities* jalur lancip dapat dilihat pada gambar berikut.

C = kanan lancip sudut 45º
D = kiri lancip sudut 45º
E = atas lancip 35º
F = bawah lancip 35º

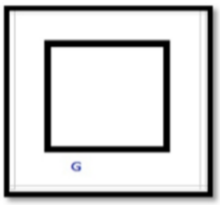


Gambar 9. Empat Maze Possibilities Jalur lancip

C. *Maze Possibilities* Jalur *Loop*

Terdapat 1 macam gambaran analisa untuk maze possibilities untuk jalur loop. Hal ini dilakukan agar tidak terjadi banyak perbedaan terhadap sensor robot line tracer atau line follower. Gambaran analisa maze possibilities untuk jalur loop dapat dilihat pada gambar seperti berikut.

G = persegi

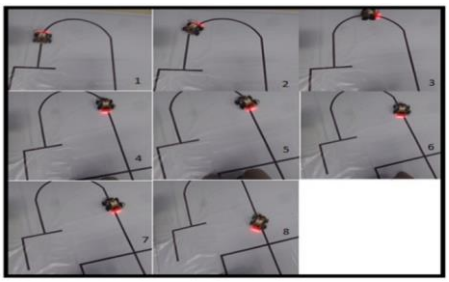


Gambar 10. Empat Maze Possibilities Jalur lancip

PENGUJIAN DAN PERBAIKAN

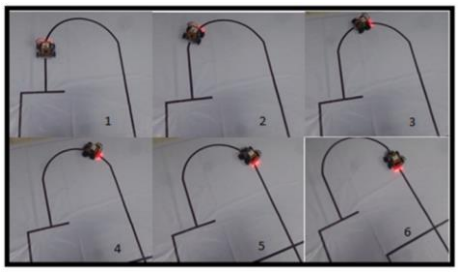
A. Pada Jalur Lengkung

Pada saat melakukan pengujian algoritma *line maze solving* untuk jalur lengkung, robot *line tracer* yang dijalankan masih belum bisa bergerak dengan sempurna. Pada penelitian sebelumnya jalur lengkung bisa dilewati oleh robot karena jalur sebelumnya tidak ditetapkan sudutnya. Sementara pada penelitian ini robot harus bergerak melewati jalur lengkung dengan sudut di atas 225^o. Oleh sebab itu robot masih mengalami beberapa gangguan seperti pergerakan yang belum sempurna yaitu gerakan yang patah-patah.



Gambar 11. Sebelum Perbaikan

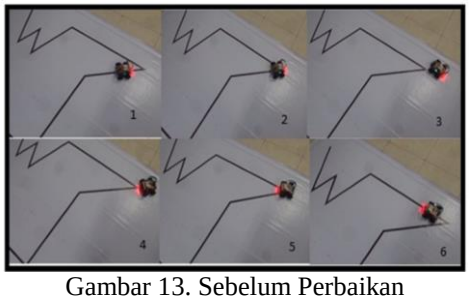
Oleh sebab itu dilakukan penganalisaan dan perbaikan pada algoritma. Setelah dianalisa, ditambahkan PID pada motor DC saat melewati jalur lurus sebelum melewati jalur lengkung agar robot bisa melewati jalur dengan baik.



Gambar 12. Setelah Perbaikan

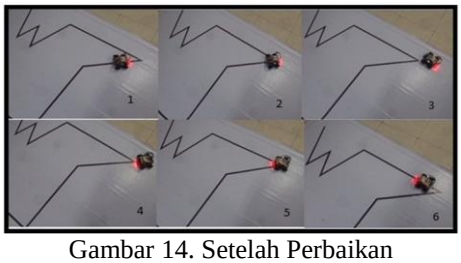
B. Pada Jalur Lancip

Untuk jalur lancip dengan sudut di bawah 45^o seperti pada *maze possibilities* C dan D dengan sudut 40^o, robot kesulitan membaca jalur lancip yang ada. Begitu juga untuk *maze possibilities* E dan F dengan sudut 35^o. Hal ini terjadi karena semakin mengecilnya sudut pada jalur lancip yang ada membuat robot semakin kesulitan membaca sensor dan robot menganggap bahwa itu jalur lurus. Akibat dari sudut yang kecil membuat robot terkadang bergerak lurus dan sensor robot yang paling tepi sering membaca garis semula sehingga robot kembali ke jalur awal bahkan robot juga pernah melewati jalur garis hitam di luar dari *maze* yang telah ditentukan.



Gambar 13. Sebelum Perbaikan

Kemudian dilakukan analisa dan setelah itu diperbaiki sehingga didapat algoritma yang baru dan hasilnya robot berhasil begreak seperti pada gambar beirkut.

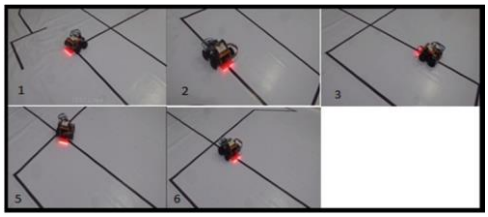


Gambar 14. Setelah Perbaikan

C. Pada Jalur *Loop*

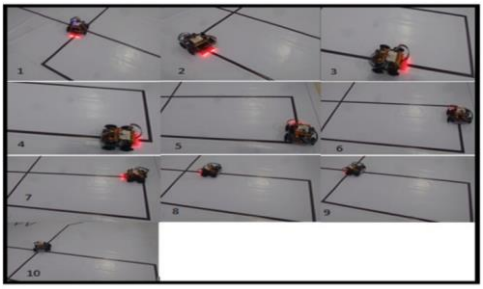
Robot *line tracer* yang digerakkan dengan menggunakan algoritma *line maze solving* pada jalur *loop* ini masih mengalami beberapa kendala. Diantaranya yaitu robot masih belum begitu bisa membaca sensor untuk “11111” apabila belum melewati garis *finish*. Maksudnya di sini ialah

robot yang apabila berada di jalur lurus “00100” kemudian menemukan jalur loop yang jika dibaca sensornya menjadi “11111”, robot kembali ke jalur semula atau dikatakan berputar di area yang ada. Robot juga sering mengalami bergerak kembali ke arah awal atau bergerak lurus.



Gambar 15. Sebelum Perbaikan

Setelah dilakukan perbaikan algoritma yang baru robot akhirnya bisa bergerak melewati jalur *loop* dengan baik dan benar seperti pada gambar berikut.



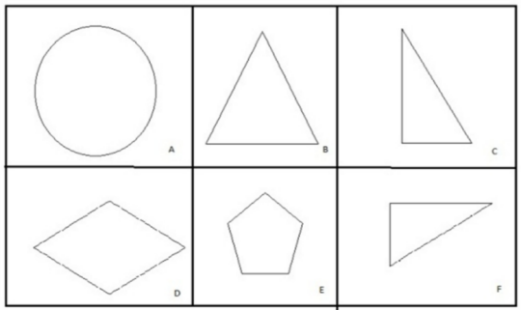
Gambar 16. SetelahPerbaikan

KESIMPULAN DAN SARAN

Berdasarkan penelitian ini adalah algoritma *line maze solving* tidak dapat digunakan untuk menyelesaikan *maze* dengan jalur lancip, jalur lengkuing, dan jalur *loop*. Sehingga perlu dilakukan perbaikan dan tambahan dari program yang baru agar dapat menyelesaikan jalur-jalur pada *maze* tersebut dengan baik. Berikut adalah penjabaran hasil akhir dari robot *line tracer* atau *line follower* yang terbaru pada jalur maze yang diteliti pada penelitian ini:

1. Robot berhasil melewati jalur lengkung dengan sudut lebih dari 225^o dengan baik.
2. Robot berhasil melewati jalur lancip dengan sudut kurang dari 45^o dengan baik. Namun pada saat pengulangan di jalur terpendeknya, robot terkadang *error* dan pernah melewati di luar jalur berwarna hitam.
3. Robot berhasil melewati jalur *loop* dengan baik.
4. Dengan tambahan PID, robot berhasil menstabilkan gerakan jalan saat melewati jalur-jalur yang ada pada *line maze*.
5. Robot berhasil bergerak mengikuti *line maze* hingga menemukan garis *finish* atau *end of maze*.
6. Setelah melewati *maze* dengan jalur lengkung, jalur lancip, dan jalur *loop*, robot kembali melewati jalur *maze* dan berhasil menentukan jalur terpendek dengan menghilangkan jalur *loop* dari *maze* dengan benar.
7. Untuk mencegah terjadinya *error* pada gerakan robot diusahakan untuk menghindari robot dari sinar matahari dan landasan untuk *line maze* harus datar dan bersih.

Maze possibilities pada penelitian ini hanya fokus pada jalur lengkung dengan sudut lebih dari 225^o, jalur lancip kurang dari 45^o, dan jalur *loop*. Maka untuk penelitian selanjutnya dapat dilakukan dengan *maze possibilities* yang terdapat pada jalur *loo*p dengan model dengan bentuk lain seperti lingkaran, segitiga atau bentuk lainnya. Dan penelitian selanjutnya robot bisa dikembangkan dengan menggunakan metode yang lain seperti metode *flood fill* atau *pledge.* Untuk contoh jalur *loop* bisa diilhat seperti pada gambar berikut ini:



Gambar 17. Saran *Maze Possibilities* Jalur *Loop*Untuk Penelitian Selanjutnya

REFERENSI

[1] Sholahuddin, Asep, dan Setiawan Hadi. “Penerapan Jaringan Syaraf Tiruan Pada Pengenalan Pola Robot Line Follower”. *Prosiding Seminar Nasional Sains dan Teknologi Nuklir*. PTNBR – BATAN Bandung, 4 Juli 2013.

[2] Vanoy, Richard T., II. *Design a Line Maze* Solving Robot: *Teaching a Robot to Solve a Line Maze*. April 2009.

[3] Musridho, R. Joko. Yanto, Febi. 2014. *Analisa Performa Algoritma Line Maze Solving Pada Jalur Lengkung Dan Zigzag*. Prosiding SNTIK5 UIN SUSKA RIAU.

[4] Rudiyanto, Hariz Bafdal. "Rancang Bangun Robot Pengantar Surat Menggunakan Mikrokontroler AT89S51". Jurnal Skripsi.

Jurusan Teknik Elektro, Universitas Gunadarma, 2010.

[5] Margolis, Michael. *Make an ArduinoControlled Robot*. Sebastopol: O'Reilly Media, Inc., 2013.

[6] Djuandi, Feri. *Pengenalan Arduino*. Juli 2011. [Online] Tersedia: http://www.tobuku.com/docs/ArduinoPengenalan.pdf, diakses 4 juli 2014.

[7] Hendriawan, Akhmad, dan Reesa Akbar. *Penyelesaian Jalur Terpendek dengan menggunakan Algoritma Maze Mapping Pada Line Maze*. Jurusan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya, Institut Teknologi Sepuluh November, 2010.