Proses Analisis dalam Perbaikan Algoritma *Line Maze Solving* untuk Jalur Lengkung dan Zig-zag

Raja Joko Musridho[1], Febi Yanto[2], Deddy Gusman[3]
[1,3] Teknik Informatika, Fakultas Teknik, Universitas Pahlawan Tuanku Tambusai
Jln. Tuanku Tambusai No.23 Bangkinang 28412 INDONESIA
[1]rajajoko@universitaspahlawan.ac.id
[3]deddy@universitaspahlawan.ac.id

[2] Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sultan Syarif Kasim Riau
Jln H.R. Soebrantas No. 155 Km. 15 Pekanbaru 28293 INDONESIA
[2] febiyanto@uin-suska.ac.id

Intisari— **Algoritma *Line Maze Solving* (LMS) merupakan algoritma untuk robot pengikut garis yang digunakan pada labirin dengan persimpangan sudut siku-siku. Penelitian sebelumnya telah melakukan perbaikan pada algoritma LMS sehingga dapat digunakan pada labirin yang memiliki jalur lengkung dan *zig-zag*. Perbaikan tersebut dilakukan karena algoritma LMS hanya mengenali persimpangan dengan sudut siku-siku. Penelitian ini menunjukkan proses analisa dalam perbaikan algoritma LMS dengan lebih rinci sehingga perbedaan antara algoritma LMS sebelum dan sesudah perbaikan dapat dilihat dengan jelas.**

Kata kunci— Robot Pengikut Garis, Labirin, Algoritma Line Maze Solving.

Abstract— **Line Maze Solving algorithm is an algorithm for line follower robot that is used on maze with right angle intersections. Previous work has improved the algorithm so that it can be used for the maze with curved and zig-zag intersections. The improvement was done because the algorithm only recognizes right angle intersections. This research presented the analysis process of the improvement of the algorithm in details, hence the difference between Line Maze Solving algorithm before and after the improvement can be seen clearly.**

Keywords— Line Follower Robot, Maze, Line Maze Solving Algorithm.

I. PENDAHULUAN

Robot yang bergerak dengan mengikuti jalur berupa garis disebut juga dengan robot *line follower* atau *line tracer*. Jalur ini dapat berupa garis terang di atas permukaan berwarna gelap atau garis berwarna gelap di atas permukaan berwarna terang (Rudiyanto, 2010). Selain untuk mengikut garis saja, robot ini juga dapat digunakan untuk mencari jalur terpendek dalam sebuah labirin yang terbuat dari kumpulan garis dan persimpangan (Vannoy II, 2009). Pemecahan masalah labirin ini dapat diselesaikan dengan berbagai algoritma, seperti *flood fill*, *pledge*, dan *line maze solving* (Musridho et al., 2018). Kemampuan robot dalam mengambil keputusan ketika akan menentukan jalan mana yang akan dipilih dalam labirin ditentukan oleh algoritma-algoritma tersebut.

Dalam *line maze solving*, terdapat dua aturan utama yang akan menentukan pergerakan robot ketika menemui persimpangan, yaitu aturan tangan kanan dan aturan tangan kiri. Aturan tangan kanan memprioritaskan untuk berbelok ke kanan dari pada lurus, lebih mengutamakan lurus dari pada berbelok ke kiri dan lebih mengutamakan belok kanan dari pada berbelok ke kiri. Sebaliknya dalam aturan tangan kiri, robot menjadi lebih mengutamakan belok kiri dari pada lurus atau belok kanan dan lebih mengutamakan lurus dari pada berbelok ke kanan (Vannoy II, 2009).

Penelitian sebelumnya telah memperbaiki algoritma *line maze solving* yang sebelumnya hanya diperuntukkan bagi labirin dengan persimpangan sudut siku-siku agar dapat digunakan pada labirin yang memiliki jalur zig-zag dan lengkung (Musridho et al., 2018). Pada penelitian tersebut, robot telah berhasil berjalan dengan baik di labirin dengan jalur lengkung dan zig-zag. Namun penelitian tersebut belum menunjukkan hasil uji coba yang telah dilakukan pada labirin yang sudah dirancang memiliki jalur dengan persimpangan siku-siku serta jalur lengkung dan zig-zag.

II. LANDASAN TEORI

A. *Robot*

Kata "robot" pertama kali muncul pada tahun 1921 dalam drama oleh Karel Capek, seorang penulis yang berasal dari Ceko. Judul dari drama tersebut adalah "Rossum's Universal Robot (RUR)". Asal katanya adalah "robota" yang bermakna tenaga kerja paksa (Herrera et al., 2015; Siswaja, 2008). Selain itu, kata "robotics" muncul pada Tahun 1942 dalam sebuah karya fiksi ilmiah yang ditulis oleh Isaac Asimov yang berjudul "Runaround". Ia kemudian melanjutkan cerita tersebut ke karangannya yang sangat terkenal yang berjudul "I, Robot" (Siswaja, 2008).

B. *Line Follower*

Seperti yang sudah dijelaskan pada bagian pendahuluan, robot *line follower* adalah robot yang bergerak mengikuti garis yang berwarna terang di atas permukaan berwarna gelap atau sebaliknya (Rudiyanto, 2010). Sensor yang digunakan adalah sensor cahaya (Jatmiko et al., 2010). Garis dideteksi oleh robot melalui hasil bacaan dari sensor cahaya tersebut, yang kemudian dikirim ke mikrokontroler yang kemudian akan menentukan motor mana saja yang harus berputar dan ke arah mana putaran dari motor tersebut agar robot tetap berada di atas garis sesuai dengan algoritma yang digunakan (Sholahuddin & Hadi, 2013). Robot yang digunakan pada penelitian ini dapat dilihat pada gambar 1 berikut.

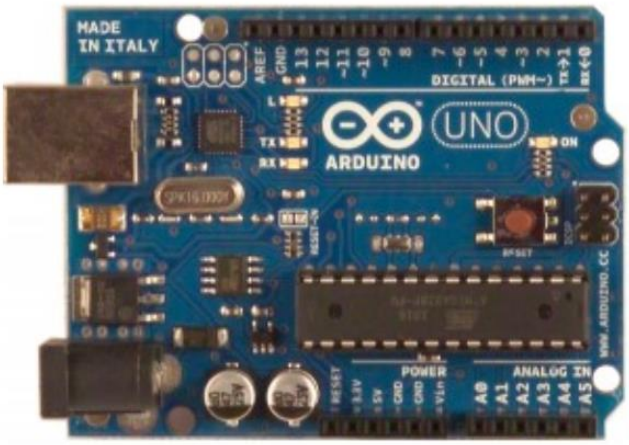
Gambar 1 Robot *Line Follower* yang digunakan dalam Penelitian (Musridho, 2014)

Paling sedikit dibutuhkan 2 buah sensor, 2 buah motor dan sebuah mikrokontroler untuk dapat membuat sebuah robot *line follower*. Kedua sensor tersebut terhubung secara bersilangan dengan dua buah motor yang seluruhnya terhubung melalui mikrokontroler. Pergerakan motor kiri ditentukan oleh sensor kanan dan begitu juga sebaliknya (Sholahuddin & Hadi, 2013). Berikut penjelasan lebih rinci:

1. Ketika sensor kiri mendeteksi garis dan sensor kanan tidak, artinya robot berada di kanan garis. Maka motor kanan akan berputar maju sementara motor kiri tetap diam, sehingga robot bergerak ke arah kiri.
2. Ketika sensor kanan mendeteksi garis dan sensor kiri tidak, artinya robot berada di kiri garis. Maka motor kiri akan berputar maju sementara motor kanan tetap diam, sehingga robot bergerak ke arah kanan.
3. Ketika kedua sensor mendeteksi garis, artinya robot berada di atas garis. Maka kedua motor berputar maju sehingga robot bergerak lurus di atas garis.

C. *Arduino (Hardware)*

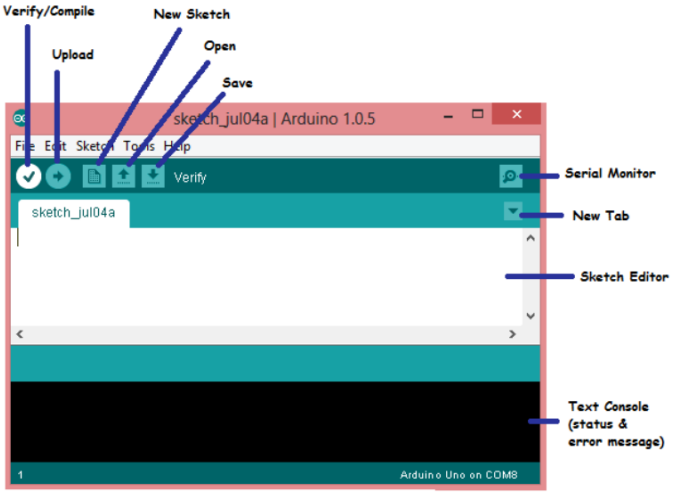
Seperti yanArduino terbagi menjadi dua macam, yaitu *software* dan *hardware* (Margolis, 2012). *Hardware* Arduino adalah papan mikrokontroler yang memilik pin masukan dan keluaran atau *input/output* (I/O). Pada penelitian ini, robot menggunakan Arduino Uno sebagai mikrokontroler yang mengendalikan dan menentukan sifat dari robot. Bentuk dari Arduino Uno dapat dilihat pada gambar 2 berikut.



Gambar 2 Arduino Uno (Djuandi, 2011)

D. *Arduino Integrated Development Environment (IDE) (Software)*

Arduino *Integrated Development Environment* (IDE) merupakan *software* yang digunakan untuk membuat dan mengupload program ke mikrokontroler Arduino dan lainnya. Di dalam Arduino IDE terdapat *program editor*, *compiler* dan *uploader* (Djuandi, 2011). *Program Editor* adalah ruang bagi *programmer* untuk menulis dan mengubat kode atau program yang kemudian diunggah ke mikrokontroler. *Compiler* bertugas mengubah program menjadi kode biner, karena mikrokontroler hanya memahami kode biner. *Uploader* merupakan sebuah modul yang bertugas untuk mengunggah kode biner yang sudah dibuat oleh *compiler* ke mikrokontroler. Program yang sudah ditulis disebut *sketch*. Arduino IDE 1.0.5 dapat dilihat pada gambar 3 berikut.



Gambar 3 Arduino IDE 1.0.5 (Musridho, 2014)

E. Algoritma Line Maze Solving (LMS)

Algoritma *Line Maze Solving* (LMS) merupakan suatu algoritma yang diperuntukkan bagi robot *line follower* untuk menemukan jalur terpendek dari titik A ke titik B dalam suatu labirin garis. Berdasarkan penjelasan yang diberikan oleh (Vannoy II, 2009), robot yang sudah deprogram dan diisi dengan algoritma LMS diletakkan di garis start yang ditentukan pada labirin garis, robot kemudian bergerak mengelilingi labirin. Ketika menemui persimpangan, robot menentukan untuk berbelok ke arah mana sesuai dengan aturan yang digunakan pada algoritma LMS, aturan tersebut bisa berupa aturan tangan kanan atau tangan kiri. Robot merekam jalur yang sudah dipilihnya ke dalam memori hingga garis atau titik *finish* ditemukan. Proses ini disebut juga dengan *mapping* (Mishra & Bande, 2008).

Ketika robot sampai di titik *finish*, jalur terpendek dihitung berdasarkan laluan yang sudah direkam di memori (Vannoy II, 2009). Pada proses ini, robot perlu diletakkan kembali secara manual di titik awal. Robot kemudian bergerak mulai dari titik awal hingga titik *finish* melalui jalur terpendek yang sudah dihitung. Persimpangan yang direkam adalah sebagai berikut:

1. Belok kanan = R
2. Belok kiri = L
3. Jalan buntu = U
4. Lurus = S

Proses perubahan rekaman untuk penghitungan jalur terpendek adalah sebagai berikut [3]:

1. LUL = diubah menjadi S
2. LUR = diubah menjadi U
3. LUS = diubah menjadi R
4. RUL = diubah menjadi U
5. RUR = diubah menjadi S
6. RUS = diubah menjadi L
7. SUL = diubah menjadi R
8. SUR = diubah menjadi L
9. SUS = diubah menjadi U

III. METODOLOGI PENELITIAN

Robot yang digunakan pada penelitian ini adalah sebuah robot *line follower* yang menggunakan mikrokontroler Arduino Uno. Robot memiliki 4 roda yang dikendalikan oleh 2 motor, 2 roda kiri dikendalikan oleh satu motor dan 2 roda kanan dikendalikan oleh satu motor lainnya. Pada robot ini terdapat 5 buah sensor cahaya dan 5 buah led yang terletak dekat dengan masing-masing sensor cahaya. Baterai yang digunakan pada robot ini berjumlah 6 buah, 2 untuk motor kiri, 2 untuk motor kanan dan 2 untuk mikrokontroler serta sensor dan LED.

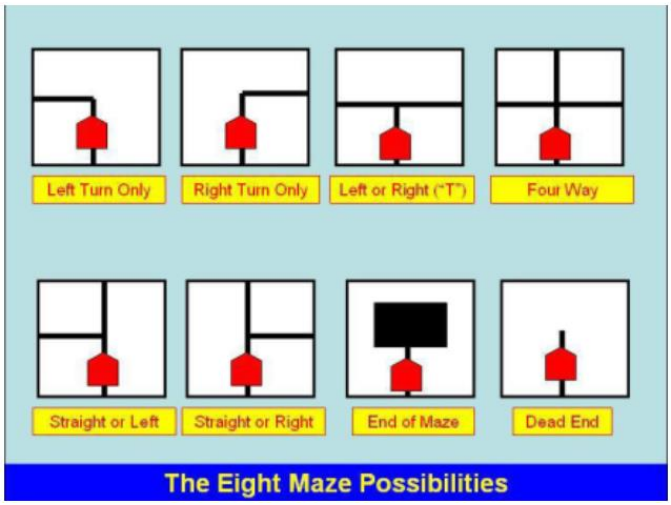
Pengujian dilakukan pada jalur yang dibuat dengan isolasi berwarna hitam dengan lebar 15mm. Isolasi tersebut disusun hingga berbentuk labirin garis di atas *backdrop* berwarna putih polos. Pengujian terhadap robot dilakukan di atas labirin tersebut.

IV. HASIL DAN PEMBAHASAN

A. Labirin Persimpangan Siku-siku

Terdapat 8 macam kemungkinan pada labirin garis dengan sudut persimpangan siku-siku yang dapat dilihat pada gambar 4. Berikut adalah 8 macam kemungkinan tersebut:

1. Kiri saja (bukan persimpangan, tidak perlu direkam),
2. Kanan saja (bukan persimpangan, tidak perlu direkam),
3. Kiri atau kanan (persimpangan T),
4. Kiri, lurus atau kanan (simpang 4),
5. Lurus atau kiri (persimpangan T),
6. Lurus atau kanan (persimpangan T),
7. Akhir labirin (garis/titik *finish*), dan
8. Jalan buntu (bukan persimpangan tapi perlu direkam).



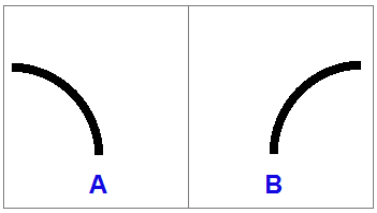
Gambar 4 Delapan Macam Kemungkinan Labirin (Vannoy II, 2009)

Kiri saja dan kanan saja bukan merupakan persimpangan sehingga robot tidak perlu merekam jalur yang dilaluinya jika robot melewati kiri saja atau kanan saja. Jalan buntu juga bukan merupakan persimpangan, namun tetap perlu direkam ke dalam memori karena jalan buntu merupakan jalan yang

salah karena robot berputar kembali dan kembali ke jalur sebelumnya.

B. Jalur Lengkung dan Zig-zag

Jalur lengkung hanya terdapat 2 kemungkinan, yaitu melengkung ke kiri (A) atau melengkung ke kanan (B) (Musridho et al., 2018). Kedua kemungkinan tersebut dapat dilihat pada gambar 5 berikut.

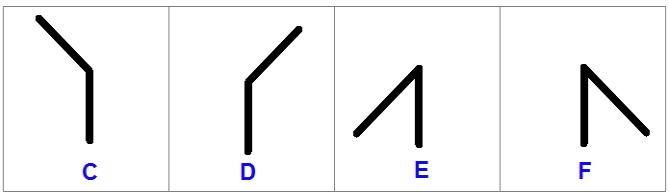


Gambar 5 Dua Macam Kemungkinan Jalur Lengkung (Musridho et al., 2018)

Untuk jalur zig-zag, sudut jalur dibatasi hanya 45° dan 135° agar bacaan sensor tidak terlalu bervariasi. Oleh karena itu, maka terdapat 4 kemungkinan pada jalur zig-zag:

C = kiri tumpul,
D = kanan tumpul,
E = kiri lancip,
F = kanan lancip.

Keempat kemungkinan tersebut dapat dilihat pada gambar 6 berikut.



Gambar 6 Empat Macam Kemungkinan Jalur Zig-zag (Musridho et al., 2018)

C. Pembacaan pada Labirin Persimpangan Siku-siku

Terdapat 5 digit yang masing-masing mewakili hasil bacaan dari seluruh sensor cahaya pada robot mulai dari paling kiri hingga paling kanan. Angka 1 bermakna sensor mendeteksi garis, angka 0 bermakna sensor tidak mendeteksi adanya garis. Berikut penjelasan untuk berbagai macam kemungkinan hasil bacaan dari 5 sensor yang ada pada robot ketika berada di labirin dengan persimpangan siku-siku.

1. Jalur Lurus

Berbagai kemungkinan hasil bacaan pada jalur lurus adalah “00001”, “00011”, “00010”, “00110”, “00100”, “01100”, “01000”, “11000” atau “10000”.

Jika hasil bacaan adalah “00001”, “00011” dan “00010”, artinya robot berada di kanan garis. Jika hasil bacaan adalah “01100”, “00100” dan “00110”, artinya robot berada di tengah garis. Jika hasil bacaan adalah “10000”, “11000” dan “01000”, artinya robot berada di kiri garis.

2. Kanan Saja

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang kanan saja adalah “01111” dan “00111”. Ini sama dengan ketika robot berada di simpang kanan atau lurus. Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan yang selanjutnya adalah “00000”, maka robot berada di simpang kanan saja, berbelok ke kanan dan tidak menyimpan jalur apapun di memori karena ini bukan merupakan persimpangan.

3. Kiri Saja

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang kiri saja adalah “11110” dan “11100”. Ini sama dengan ketika robot berada di simpang kiri atau lurus. Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan yang selanjutnya adalah “00000”, maka robot berada di simpang kiri saja, berbelok ke kiri dan tidak menyimpan jalur apapun di memori karena ini bukan merupakan persimpangan.

4. Simpang T (Kiri atau Kanan)

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang T (kiri atau kanan) adalah “01111”, “1111” dan “11110”. Ini sama dengan ketika robot berada di atas garis *finish* atau simpang empat (kiri, lurus atau kanan). Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan selanjutnya adalah “00000”, artinya robot berada di simpang kiri atau kanan yang bisa juga disebut simpang T.

Jika robot menggunakan aturan tangan kanan, maka robot berbelok ke kanan. Jika menggunakan aturan tangan kiri, maka robot berbelok ke kiri. Kemudian robot menyimpan bacaan jalur di memori untuk kemudian digunakan ketika mencari jalur terpendek.

5. Simpang Empat (Kiri, Lurus atau Kanan)

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang empat (kiri, lurus atau kanan) adalah “01111”, “1111” dan “11110”. Ini sama dengan ketika robot berada di atas garis *finish* atau simpang T (kiri, lurus atau kanan). Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan selanjutnya adalah “01100”, “00100” atau “00110”, artinya robot berada di simpang empat (kiri, lurus atau kanan).

Jika robot menggunakan aturan tangan kanan, maka robot berbelok ke kanan. Jika menggunakan aturan tangan kiri, maka robot berbelok ke kiri. Kemudian robot menyimpan bacaan jalur di memori untuk kemudian digunakan ketika mencari jalur terpendek.

6. Kiri atau Lurus

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang kiri atau lurus adalah “11110” dan “11100”. Ini sama dengan ketika robot berada di simpang kiri saja. Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan yang selanjutnya adalah “01100”, “00100” atau “00110”, maka robot berada di simpang kiri atau lurus, berbelok ke kiri jika menggunakan aturan tangan kiri atau lurus jika menggunakan aturan tangan kanan, dan menyimpan bacaan jalur di memori untuk kemudian digunakan ketika mencari jalur terpendek.

7. Kanan atau Lurus

Berbagai kemungkinan hasil bacaan jika robot berada pada simpang kanan atau lurus adalah “01111” dan “00111”. Ini sama dengan ketika robot berada di simpang kanan saja. Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan yang selanjutnya adalah “01100”, “00100” atau “00110”, maka robot berada di simpang kanan atau lurus, berbelok ke kanan jika menggunakan aturan tangan kanan atau lurus jika menggunakan aturan tangan kiri, dan menyimpan bacaan jalur di memori untuk kemudian digunakan ketika mencari jalur terpendek.

8. Garis *Finish*

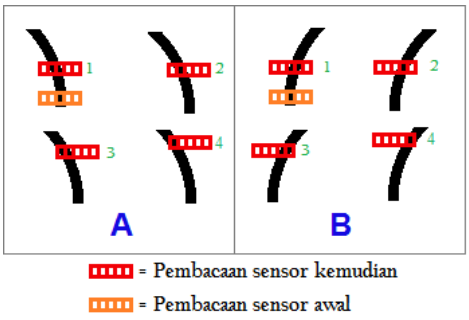
Berbagai kemungkinan hasil bacaan jika robot berada pada garis *finish* adalah “01111”, “11111” dan “11110”. Ini sama dengan ketika robot berada di simpang empat (kiri, lurus atau kanan). Untuk membedakannya, robot perlu mendapatkan bacaan selanjutnya. Jika bacaan yang selanjutnya adalah “11111”, maka robot berada di garis *finish*, berhenti selama waktu yang ditentukan dan menghitung jalur terpendek berdasarkan rekaman jalur yang sudah dilewati yang disimpan di memori.

9. Jalan Buntu

Kemungkinan hasil bacaan jika robot berada di jalan buntu adalah “0000”. Untuk mengetahui apakah jalan tersebut jalan buntu atau tidak, robot bergantung pada bacaan sebelumnya. Jika bacaan sebelumnya adalah “01000”, “01100”, “00100”, “00110” atau “00010”, maka robot berada di jalan buntu. Robot kemudian berputar arah dan menyimpan jalur yang sudah dilewatinya di memori untuk kemudian digunakan ketika mencari jalur terpendek.

D. Pembacaan pada Jalur Lengkung

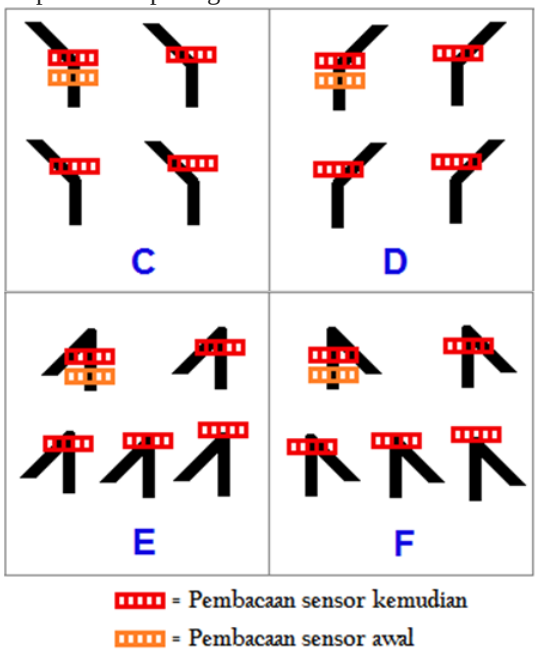
Saat menemui kemungkinan tipe A dan B (gambar 5) pembacaan sensor adalah sama seperti saat robot berjalan pada jalur lurus, yaitu “00001”, “00011”, “00010”, “00110”, “00100”, “01100”, “01000”, “11000” atau “10000”. Tugas robot pun sama seperti saat berjalan lurus di mana robot harus menjaga kestabilan posisi agar pembacaan sensor selalu "00100", baik itu saat menemukan kemungkinan tipe A dengan pembacaan "01100", “01000”, “11000”, dan “10000”, atau tipe B dengan pembacaan "00110", “00010”. “00011”, dan “00001”. Perbedaannya adalah bahwa robot mendapatkan tugas lebih dalam menjaga kestabilan. Apabila robot cukup cepat misalnya setelah berjalan di jalur lurus yang cukup panjang, lalu robot menemui jalur lengkung maka pembacaan sensor bisa hingga menjadi “00000” karena robot keluar jalur. Namun ini tidak menjadikan robot menganggap bahwa posisinya berada di jalur buntu, karena sebelum keluar jalur robot mendapatkan hasil bacaan “10000” atau “00001”. Hal ini juga berlaku pada jalur zig-zag dengan sudut tumpul, yaitu maze possibilities C dan D (gambar 6). Penggambaran untuk pembacaan sensor terhadap jalur lengkung dapat dilihat pada gambar 7 berikut.



Gambar 7 Perubahan Pembacaan Sensor Terhadap Jalur Lengkung

E. Pembacaan pada Jalur Zig-zag

Pada jalur zig-zag, hal yang perlu dilakukan robot ketika berada di kemungkinan tipe C dan D (gambar 6) adalah sama dengan ketika robot berada pada jalur lurus dan juga pada jalur lengkung. Lain halnya dengan tipe E dan F (gambar 6), berbagai kemungkinan hasil bacaan sensor jika robot berada pada tipe E adalah “10100”, “11100", “01100”, “00100”, dan “00000”. Dan untuk tipe F adalah “00101”, “00111”, “00110”, “00100”, dan “00000”. Pembacaan “10100” dan “00101” tidak ada pada algoritma dari Vannoy II, sehingga perlu ditambahkan agar robot dapat mengetahui bahwa ini adalah jalur lancip ke kiri. Penggambaran perubahan pembacaan sensor dapat dilihat pada gambar 8 berikut.



Gambar 8 Perubahan Pembacaan Sensor Terhadap Jalur Zig-zag

V. IMPLEMENTASI DAN PENGUJIAN

Setelah analisa berbagai kemungkinan hasil bacaan sensor pada berbagai macam kemungkinan jalur dan simpang yang ada pada labirin, maka algoritma yang baru didapatkan.

Algoritma ini kemudian diubah ke dalam bahasa pemrograman untuk Arduino.

A. *Hasil Pengujian pada Labirin Persimpangan Siku-siku*

Hasil implementasi dan pengujian sebelum dilakukan perbaikan pada algoritma dari (Vannoy II, 2009), robot gagal untuk berbelok di kemungkinan jalur kiri saja dan kanan saja. Hal ini disebabkan algoritma awal memerintahkan robot untuk berbelok ke kiri atau ke kanan selama 0,5 detik. Jangkauan putaran yang diraih oleh robot dalam waktu 0,5 detik sangat bergantung pada kondisi baterai. Jika baterai masih full, robot bahkan bisa berputar terus menerus karena setiap 0,5 detik tidak terbaca garis apapun. Begitu juga ketika baterai sudah lemah, robot berputar dan menyimpan rekaman jalur buntu (yang dilakukan lebih dari 1 kali) pada memori karena bacaan sensor setiap 0,5 detik adalah "00000".

Perbaikan yang sudah dilakukan adalah membuat robot berputar hanya hingga mendeteksi garis. Dan selama apapun waktu yang dibutuhkan, robot tidak akan menyimpannya sebagai jalan buntu walau bacaan "00000" karena perintah yang ada adalah berhenti berbelok ketika salah satu sensor membaca garis.

B. *Hasil Pengujian pada Jalur Lengkung*

Hasil implementasi dan pengujian sebelum dilakukan perbaikan pada algoritma dari (Vannoy II, 2009), robot menganggap bahwa posisinya ketika tiba-tiba bacaan menjadi "00000" adalah berada pada jalan buntu. Ini menyebabkan robot menyimpan rekaman jalan buntu di memori, padahal robot berada pada jalur lengkung yang bukan jalan buntu, bukan persimpangan dan bukan garis *finish*.

C. *Hasil Pengujian pada Jalur Lengkung*

Hasil implementasi dan pengujian sebelum dilakukan perbaikan pada algoritma dari (Vannoy II, 2009) adalah ketika robot menemui kemungkinan tipe C dan D (gambar 6), robot melakukan hal yang sama dengan ketika menemui jalur lengkung. Sementara untuk tipe E dan F (gambar 6), robot dapat berbelok dengan baik setelah algoritma untuk berbelok hingga jumpa garis ditambahkan. Namun masih terdapat kekurangan, yaitu ketika robot tidak cukup lurus sehingga bacaan bukan "11100" atau "00111" dan hal ini menyebabkan robot tidak memberikan reaksi apapun. Karena "10100" atau "00101" tidak ada dalam algoritma awal (Vannoy II, 2009).

D. *Setelah dilakukan Perbaikan*

Hasil implementasi dan pengujian setelah dilakukan perbaikan (Musridho et al., 2018), robot berhasil berbelok dengan baik di persimpangan kiri saja, kanan saja, jalur lengkung, jalur zig-zag.

VI. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Proses analisa berbagai kemungkinan bacaan sensor pada berbagai kemungkinan jalur dan persimpangan di labirin siku-siku, lengkung dan zig-zag sudah dijabarkan dengan lebih baik dan rinci pada penelitian ini dibandingkan penelitian sebelumnya (Musridho et al., 2018). Sehingga seluruh kegagalan dan keberhasilan algoritma yang dijelaskan di penelitian ini hanya berupa sedikit penjelasan yang dibuat berdasarkan penelitian tersebut.

B. *Saran*

Untuk pengembangan dalam penelitian berikutnya disarankan agar robot ditambahkan algoritma yang dapat menelusuri balik jalur terpendek secara langsung dari garis *finish* dan kembali ke garis *start*.

REFERENSI

Djuandi, F. (2011). Pengenalan Arduino. *Tokobuku.Com. Jakarta*.

Herrera, R. R., Funes, F. G., & del Castillo, M. A. S. A. (2015). Design and Implementation of an Affective Computing for Recognition and Generation of Behaviors in a Robot. In *Multibody Mechatronic Systems* (pp. 567–578). Springer.

Jatmiko, W., Febrian, A., Jovan, F., Salsabila, S., Heriyandi, F., & Wibisono, A. (2010). *Robot Lego Mindstroms : Teori dan Praktek*. UI Press.

Margolis, M. (2012). *Make an Arduino-controlled robot*. " O'Reilly Media, Inc."

Mishra, S., & Bande, P. (2008). Maze solving Algorithms for micro mouse. *SITIS 2008 - Proceedings of the 4th International Conference on Signal Image Technology and Internet Based Systems*, 86–93. https://doi.org/10.1109/SITIS.2008.104

Musridho, R. J. (2014). *Analisa Performa Algoritma Line Maze Solving pada Jalur Lengkung dan Zig-Zag*. Universitas Islam Negeri Sultan Syarif Kasim.

Musridho, R. J., Yanto, F., Haron, H., & Hasan, H. (2018). Improved Line Maze Solving Algorithm for Curved and Zig-zag Track. *2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, 1–6.

Rudiyanto, H. B. (2010). Rancang Bangun Robot Pengantar Surat Menggunakan Mikrokontroler AT89S51. *Jurnal Skripsi. Jurusan Teknik Elektro, Universitas Gunadarma*.

Sholahuddin, A., & Hadi, S. (2013). Penerapan Jaringan Syaraf Tiruan Pada Pengenalan Pola Robot Line Follower. *Prosiding Seminar Nasional Sains Dan Teknologi Nuklir. PTNBR--BATAN Bandung*, *4*.

Siswaja, H. D. (2008). Prinsip Kerja dan Klasifikasi Robot. *Media Informatika*, *7*(3), 147–157.

Vannoy II, R. T. (2009). *Design a Line Maze Solving Robot - Teaching a Robot to Solve a Line Maze*. http://www.richardvannoy.info/line-maze-algorithm.pdf