Penyelesaian Jalur Terpendek dengan menggunakan Algoritma Flood Fill pada Line Maze

Akhmad Hendriawan, Reesa Akbar
Jurusan Teknik Elektronika, Politeknik Elektronika Negeri Surabaya
Kampus PENS-ITS Sukolilo, Surabaya
Email: hendri@eepis-its.edu, reesa@eepis-its.edu

Abstrak - *Flood fill adalah metode yang umum digunakan untuk menyelesaikan maze dalam bentuk dinding . akan tetapi algoritma ini sangat jarang digunakan untuk menyelesaikan maze dalam bentuk garis. Oleh karena itu Pada paper ini diimplementasikan algoritma flood fill pada line follower robot untuk melakukan pencarian jalur dari tempat awal menuju tempat tujuan dalam suatu lingkungan terkontrol berupa maze dalam bentuk garis. Algoritma ini bekerja dengan mengisi sebuah area dengan penanda tertentu. Pada tahap awal algoritma ini akan membagi suatu area, menjadi sub-sub area yang lebih kecil, yang dapat didefinisikan sebagai suatu matrik. Kemudian mengisi sub-sub area tersebut dengan sebuah nilai awal, dimana nilai ini merupakan perhitungan awal untuk jarak dari masing-masing area tersebut dari titik tujuan. Nilai ini akan di-update sesuai dengan kondisi line maze yang dihadapi, sehingga nilai dari tiap area ini akan sesuai dengan kondisi lapangan (maze) yang dihadapi. Pada percobaan yang dilakukan, algoritma ini telah dapat berjalan sesuai yang diharapkan*

Kata kunci*—line follower robot, maze maping, flood fill, update, path finding*

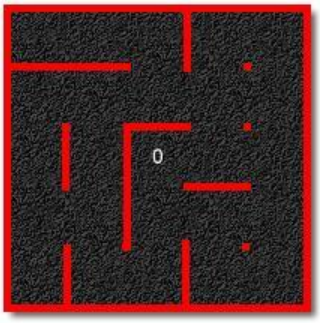
1. Pendahuluan

Line follower robot adalah sebuah robot yang dirancang untuk berjalan mengikuti garis. Namun dalam paper ini, robot tidak hanya bertugas untuk berjalan mengikuti garis saja melainkan juga harus bisa mencari jalan keluar (jalan menuju ke finish) dari suatu *maze*. Hal ini tentunya membutuhkan sistem kendali yang bisa membuat robot mampu melewati *maze* dengan baik dan dengan tingkat *error* seminimal mungkin. Dengan demikian, waktu yang ditempuh untuk mencapai tujuan menjadi lebih efektif.

2. Teori Penunjang

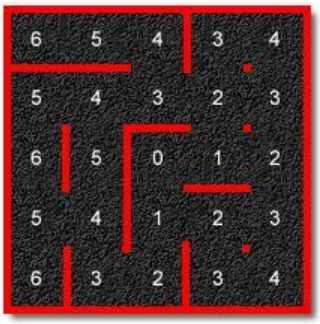
2.1 Algoritma *Flood fill*

Langkah yang paling tepat untuk dapat mengerti algoritma *flood fill* adalah dengan menggunakan analogi air yang ditumpahkan pada sebuah maze. Berikut penjelasannya,



Gambar 1 Kondisi awal maze

- Proses penumpahan air terpusat hanya pada satu titik (gambar 1) (*center*, selanjutnya titik ini akan dikenal sebagai *destination* atau tujuan)
- Air akan membanjiri titik center ini, kemudian mulai mengalir ke area disekitarnya, yang tidak terhalang oleh dinding (dapat diakses secara langsung)
- Secara virtual, maze dibagi menjadi beberapa kotak kecil (array)
- Kotak dimana titik *center* berada, diberi nilai ‘ 0 ’



Gambar.2 Kondisi maze setelah terpenuhi oleh air

- Kotak yang terisi air setelah *center*, akan diberi nilai (gambar 2)
- Kotak yang terisi air setelah golongan 1, akan diberi nilai 2
- Kotak yang terisi air setelah golongan 2, akan diberi nilai 3
- Dan begitu pula untuk kotak yang terisi air selanjutnya

Arti dari nilai di dalam masing-masing kotak adalah jumlah kotak yang harus ditempuh dari kotak tersebut untuk mencapai *center* (tujuan). Asumsikan kotak yang berada pada bagian bawah sebelah kiri merupakan *start*, kemudian ikutilah kotak yang memiliki nilai lebih kecil dari nilai kotak yang sedang ditempati. Rute yang akan

ISBN: 978-979-8689-13-0

terbentuk adalah rute terpendek yang dapat ditempuh dari *start* menuju ke *center*.

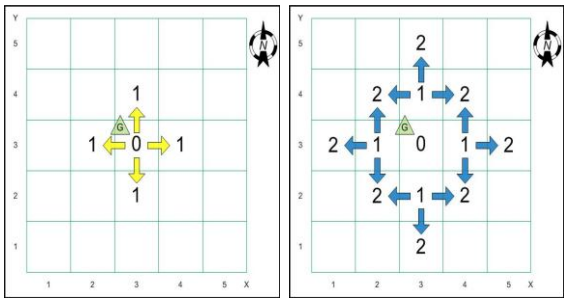
Penjelasan di atas adalah kondisi lapangan berupa *wall-mazed* sedangkan pada paper ini akan digunakan lapangan berupa *line-mazed*. Tujuaannya adalah agar memudahkan pemahaman tentang algoritma *flood fill.*

Tahap - tahap Algoritma Flood Fill

- *Generate* nilai awal untuk masing – masing *cell*

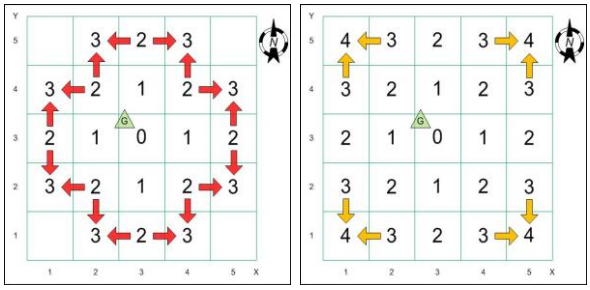
  Nilai pembobot awal diasumsikan pada lapangan terdapat jalur yang menghubungkan seluruh *cell* dengan seluruh *cell* tetangganya. *cell* yang berjarak 1 *cell* dari goal akan bernilai 1, yang berjarak 2 *cell* akan bernilai 2, dan seterusnya. Asumsi jarak disini dapat dicapai dengan arah empat mata angin, sehingga tidak ada gerak serong atau diagonal. Ilustrasi dari algrotirma diatas ditunjukkan pada gambar 3 dan gambar 4

  Berikut ini diberikan ilustrasi penjelasan diatas,



(a) (b)

Gambar 3 (a) *cell* yang berjarak 1 *cell* dari *goal* , (b) *cell* yang berjarak 2 *cell* dari *goal*.



(a) (b)

Gambar 4 (a) *cell* yang berjarak 3 *cell* dari *goal*, (b) *cell* yang berjarak 4 *cell* dari *goal*.

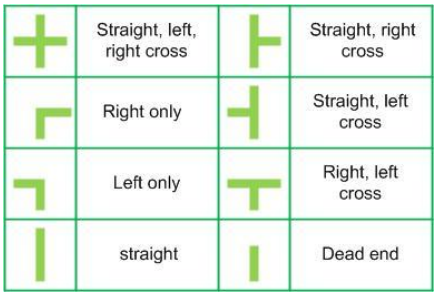
- *update* lapangan

  menyimpsn kondisi lapangan pada tiap *cell*, informasi bentuk lapangan (jalur) akan mempengaruhi besar nilai dari masing-masing *cell*. Karena bentuk lapangan akan mempengaruhi jarak antar *cell*. Gambar 5 adalah variasi kemungkinan bentuk jalur yang dapat terjadi.
- *update* nilai *cell*

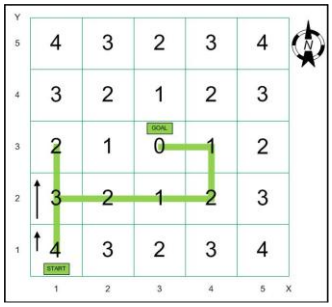
  yaitu merubah nilai dari *cell*, dengan tujuan menyesuaikan nilai *cell* dengan kondisi nyata (lapangan / jalur) yang ada,. proses ini dilakukan apabila,

1. Robot menemui jalan buntu.
2. Robot menemui cell tujuan yang nilainya lebih besar dari nilai cell tempat robot sekarang.

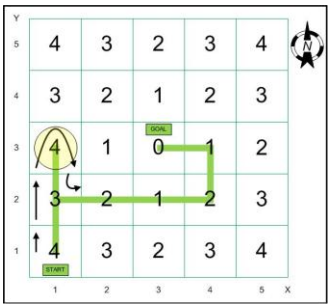
Ilustrasi ditunjukkan pada gambar 6,7,8 dan 9 Berikut akan diberikan gambar sebagai ilustrasi penjelasan di atas,



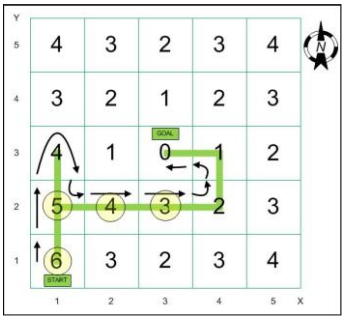
Gambar 5 kemungkinan bentuk jalur



Gambar .6 kondisi awal nilai *cell*



Gambar 7 contoh proses *update* pada kondisi *dead end*



Gambar 9 contoh proses *update* pada kondisi nilai *cell* tujuan lebih besar daripada *cell* sekarang (*current cell*)

- menentukan *cell* tujuan

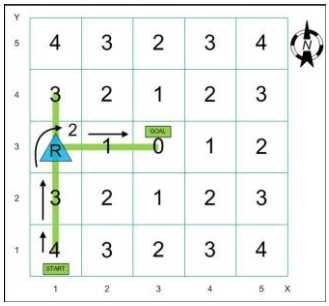
  robot akan membaca kondisi lapangan, dengan begitu robot akan mengetahui cell mana saja yang memungkinkan untuk dituju. Dari *cell* - *cell* ini

akan dicek, *cell* yang memiliki nilai terkecil akan menjadi tujuan robot.

- bergerak ke arah *cell* tujuan

  setelah *cell* tujuan ditentukan, robot akan bergerak menuju ke *cell* tersebut.

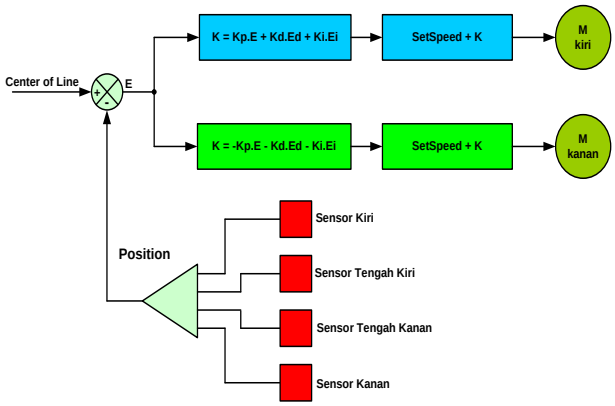
  Untuk kedua langkah terakhir, (menentukan *cell* tujuan, dan bergerak ke a rah *cell* tujuan). Ilustrasi ditunjukkan oleh gambar 10.



Gambar 10 contoh penentuan cell tujuan

2.2 Kontrol PID

Kontrol untuk mengendalikan kecepatan motor DC pada robot digunakan kontroler PID. Kontroler ini merupakan kombinasi antara kontrol P, I dan D. Dengan menggabungkan ketiga kontroler tersebut, maka akan diperoleh output yang cukup ideal dari yang diharapkan. Gambar 11 menunjukkan skema kombinasi PID dalam sebuah kontroler untuk motor DC.



Gambar 11 Blok diagram kontrol

Metode Ziegler-Nichols

Merupakan metode yang digunakan untuk tuning nilai dari konstanta – konstanta pada kontroler PID.

Penalaan parameter PID(gambar 11) didasarkan terhadap kedua konstanta hasil eksperimen, Ku dan Pu. Ziegler dan Nichols menyarankan penyetelan nilai parameter Kp, Ti dan Td berdasarkan rumus yang diperlihatkan pada tabel 1.

Tabel 1 Penalaan parameter PID dengan metode osilasi

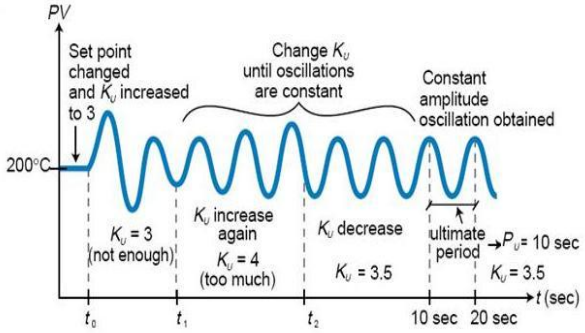
Tipe Kontroler	Kp	Ti	Td
P	0,5.Ku		
PI	0,45.Ku	½ Pu	
PID	0,6.Ku	0,5 Pu	0,125 Pu

3. Perancangan Sistem

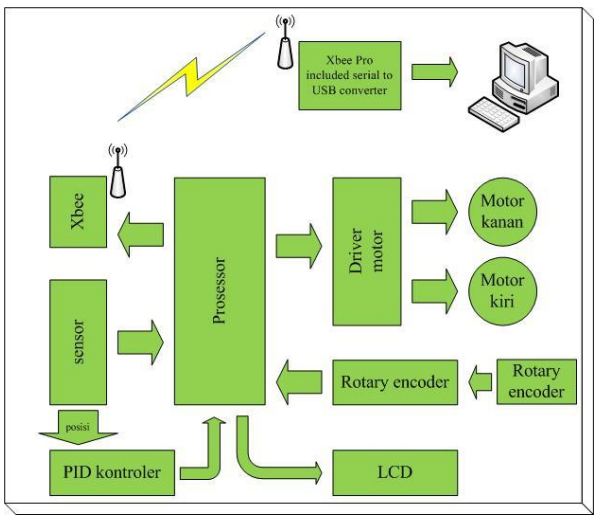
Secara umum, rancangan system yang dibuat adalah tampak pada gambar 12

Sistem ini seperti halnya kerja *line follower robot* pada umumnya, namun diberikan beberapa fitur – fitur tambahan, secara umum penambahan fitur – fitur yang dilakukan adalah,

1. PID kontroler, berguna untuk kontrol gerakan *line follower robot*, agar dapat berjalan dengan baik, dalam artian tidak banyak berosilasi.
2. Rotary encoder, sebagai sensor untuk mengetahui jarak yang telah ditempuh.
3. Komunikasi serial, digunakan untuk mengirim data error pada kontroler, sehingga error dapat diamati dan dianalisa.



Gambar 11 ilustrasi respon terhadap perubahan nilai konstanta pada saat tuning

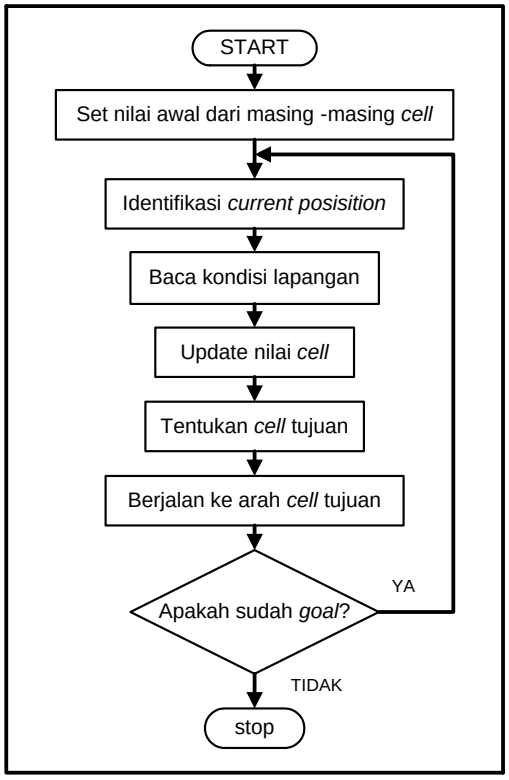


Gambar 12 Konfigurasi system

3.1 Flood Fill

Penjelasan dari flow chart pada gambar 13 adalah sebagai berikut, robot berjalan dari posisi start, setiap 25 cm robot akan berhenti, mengecek apakah terdapat persimpangan atau tidak, ada atau tidaknya persimpangan akan dibaca oleh robot sebagai data, data

dari lapangan ini akan masuk sebagai data pada virtual array. Data ini selanjutnya akan menentukan nilai-nilai untuk *neighbour cell* atau dengan kata lain robot melakukan proses update. Nilai-nilai neighbour cell ini kemudian dibandingkan, dan robot akan bergerak ke arah *neighbour cell* yang memiliki nilai paling kecil. Begitu seterusnya hingga robot menemukan *finish.*

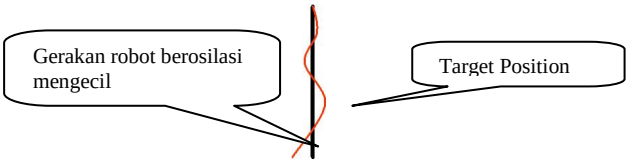


Gambar 13 *flow chart* algoritma *flood fill*

3.2 Kontrol PID

Kontrol PID dalam paper ini digunakan untuk mengontrol posisi robot saat berjalan agar bisa selalu berada di tengah-tengah garis seperta pada gambar 14. output yang diberikan adalah berupa nilai untuk pengaturan kecepatan motor.

Kontrol untuk mengendalikan kecepatan motor DC pada robot digunakan kontroler PD. Kontroler ini merupakan kombinasi antara kontrol P dan D. Dengan menggabungkan kedua kontroler tersebut, maka akan diperoleh output yang cukup ideal dari yang diharapkan.



Gambar 14 Pola gerakan robot berosilasi

Dalam penggunaannya, posisi sensor terhadap garis mengartikan error yang terjadi. Ilustrasi posisi sensor dan error ditunjukkan oleh gambar 15.

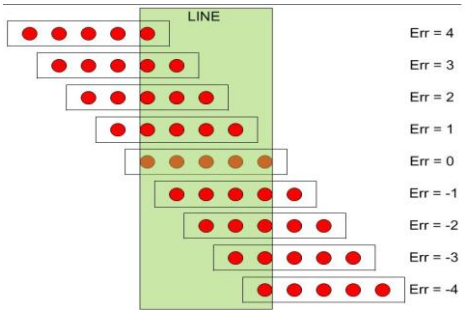
Dalam aplikasinya, maka peran dari kontroler ini dapat diterapkan dalam program dengan formulasi seperti berikut:

- pwmKiri = PwmRef + (Kp.error + Kd.(error-old_error) + Ki.(error + ak_error))
- pwmKanan = PwmRef - (Kp.error - Kd.(error-old_error) - Ki.(error + ak_error))

Ket:
PwmRef adalah nilai pwm yang diinginkan pada saat error = 0

3.3 Simulator Flood Fill

Pembuatan simulator yang bertujuan untuk mempermudah pengamatan pada kinerja algoritma flood fill. Karena algoritma ini menggunakan *virtual array* untuk mengidentifikasi tiap-tiap *cell*nya. Sehingga tidak dapat dilihat secara langsung ketika robot berjalan dilapangan. Pada simulator ini ditampilkan *array* untuk *cell value*, bentuk lapangan (sesuai input *user*), dan *array* untuk aksi dari robot.

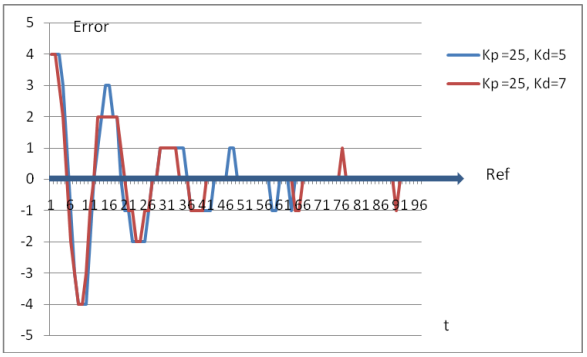


Gambar 15 nilai error berdasarkan posisi sensor

4. Pengujian

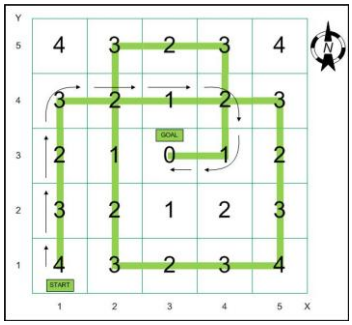
4.1 Pengujian Kontroler

Pengujian ini dilakukan dengan menjalankan linefollower robot pada suatu garis lurus, dan posisi awal robot berada pada kondisi yang memiliki error paling besar. Kemudian kontroler akan melakukan aksi yaitu mengatur kecepatan antara motor kanan dan kiri agar dapat mencapai posisi referensi. Hasil pengujian yang terlihat pada gambar 16 menunjukkan kontroller dapat bekerja dengan baik dan mempunyai respon yang cukup cepat

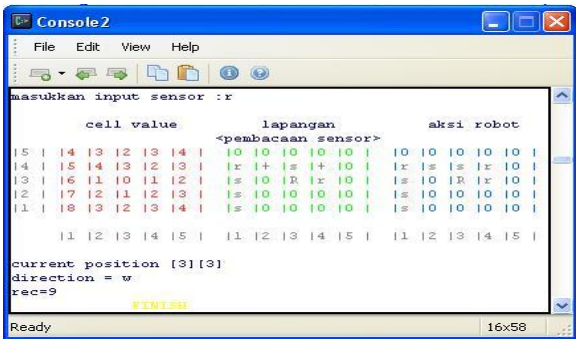


Gambar 16 Respon controller PID

4.2 Pengujian Simulator Flood Fill

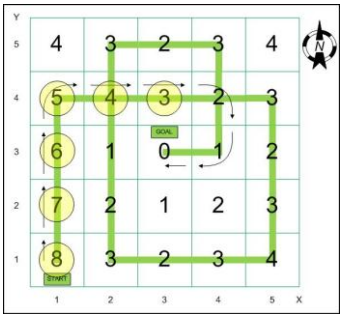


Gambar 17 contoh gambar dari lapangan yang akan di simulasikan



Gambar 18 visualisasi hasil simulasi algoritma *flood fill* ketika robot telah mencapai *goal* untuk lapangan pada gambar 4.2

Dapat dilihat pada gambar 18, pada array berwarna hijau terlihat pola atau jalur yang ditempuh robot. Pada *array* yang berwarna merah terlihat nilai-nilai dari *cell* yang telah di-*update*. Berikut akan diberikan gambar untuk memperjelas hasil peng-*update*-an *cell*.



Gambar 19 hasil *update cell value* setelah robot mencapai *goal*

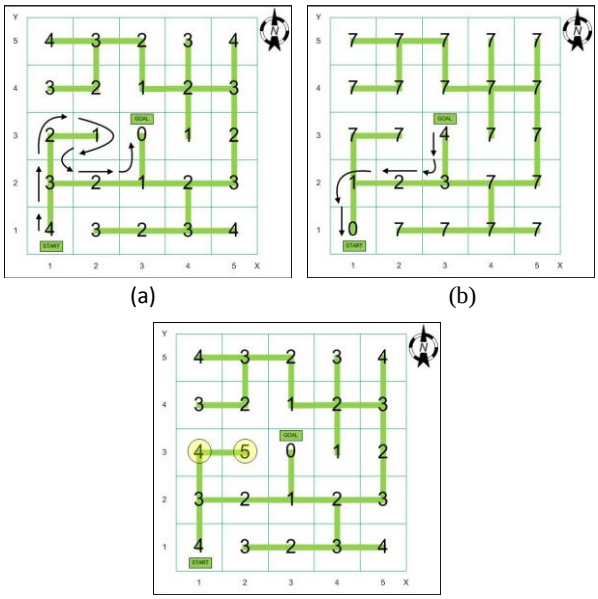
Bila diamati nilai –nilai awal *cell* yang terdapat pada gambar 17 dibandingkan dengannilai – nilai *cell* yang terdapat pada gambar 19, terdapat beberapa *cell* yang telah diubah nilainya, *cell-cell* ini adalah *cell* yang telah dilewati oleh robot, dan nilainya diubah menjadi sesuai dengan kondisi lapangan yang ada, yaitu nilai *cell* merupakan jarak *cell* tersebut dengan *cell* tujuan (*goal*).

4.3 Pengujian Robot di Lapangan

Pengujian dilapangan merupakan pengujian penting untuk mengetahui apakan metode floodfill yang dirancang dan bekerja dengan baik pada level simulasi dapat diterapkan pada robot.

- **Percobaan 1**

Hasil dari pengujian pada robot di lapangan telah menunjukkan hasil yang sama ketika dilakukan dalam simulasi, namun terkadang terdapat kesalahan dikarenakan robot salah dalam membaca kondisi garis yang ada. Hal ini dapat diketahui karena setiap pembacaan ditampilkan langsung di LCD yang terdapat pada *hardware* robot. Gambar 20 dan tabel 2 adalah hasil dari beberapa percobaan yang telah dilakukan



(a) (b)

(c)

Gambar 20 variasi posisi start dan goal model 1, (a) jalur berangkat, (b) jalur pulang, (c) nilai cell yang di-*update*

Keterangan :

- Start point : 1,1Goal point : 3,3 Start direction : north

Tabel 2 hasil pewaktuan pada variasi model 1

model 1			
percobaan ke-	berangkat pertama (s)	kembali (s)	berangkat kedua (s)
1	9.8	5.1	4.8
2	9.9	4.9	4.9
3	9.9	4.9	4.8
4	10	4.8	4.7
5	gagal		
rata - rata	9.9	4.925	4.8

- **Percobaan 2**

Pada percobaan terdapat perbedaan antara waktu berangkat dan waktu pulang, karena pada proses

berangkat robot sempat menemui jalan buntu. Dan pada waktu pulang jalur buntu ini telah dihindari oleh robot. Sehingga waktu yang diperlukan pun lebih singkat. Dalam percobaan 2 yang ditunjukkan pada tabel 3 dijumpai kegagalan pada *running* dikarenakan faktor mekanik dari robot yang kurang baik.

Tabel 3 hasil pewaktuan pada variasi model 2

model 2			
percobaan ke-	berangkat pertama (s)	kembali (s)	berangkat kedua (s)
1	9.5	6	5.7
2	9.6	5.9	5.8
3	9.4	5.9	5.8
4	gagal		
5	9.6	5.8	5.9
rata - rata	9.525	5.9	5.8

- **Percobaan 3**

Pada percobaan 3 ini selisih antara waktu berangkat dan waktu pulang sangat kecil, karena pada proses berangkat robot tidak menemui jalan buntu. Namun proses berangkat memiliki tahapan yang lebih kompleks dibandingkan proses pulang. Sehingga memerlukan waktu yang sedikit lebih lama. Dalam percobaan ini dijumpai kegagalan pada *running* (tabel 4) dikarenakan faktor mekanik dari robot yang kurang baik.

Tabel 4 hasil pewaktuan pada variasi model 3

model 3			
percobaan ke-	berangkat pertama (s)	kembali (s)	berangkat kedua (s)
1	7	6.7	6.6
2	7.1	7.2	6.5
3	7	6.8	6.7
4	6.9	6.8	6.6
5	gagal		
rata - rata	7	6.875	6.6

5. Kesimpulan

- Nilai Kp=25 dan Kd=7, merupakan nilai setting yang didapat untuk *line follower robot* yang dibuat pada *final project* ini, Penggunaan kontroler akan menjaga kestabilan jalan dari robot, sehingga mampu menjaga keakuratan dalam pembacaan jarak oleh *rotary encoder*, dan juga pembacaan bentuk lapangan oleh sensor garis.
- Rotary encoder yang digunakan telah dapat berjalan dengan baik, error maksimal yang di dapat saat pengujian adalah sebesar 4.9 % dan error rata-ratanya adalah sebesar 0.57%.
- Pada saat dilakukan pengujian *running* algoritma *flood fill* langsung di robot, masih dijumpai error dengan persentase sebesar 20 %, hal ini disebabkan karena pengerjaan mekanik yang kurang baik.
- Setiap pengambilan keputusan yang dilakukan oleh algoritma *flood fill* selalu mempertimbangkan ke arah *cell* yang lebih dekat dengan *cell* tujuan dan apabila terdapat dua *cell* yang memiliki nilai yang sama, maka akan diputuskan untuk memilih waktu tempuh yang lebih singkat. Waktu tempuh untuk berjalan lurus adalah 0.8 s sedangkan berbelok memerlukan waktu tempuh 1.3 s

6. Pustaka

[1] Giessel David ; “Building a Mouse”, UAF MicroMouse Home Page ; 2007

[2] Mishra Swarti, Bande Pankaj ; ”Maze Solving Algorithm for Micro Mouse”, International Conference on Signal Image Technology and Internet Based Systems ; 2008

[3] W Eddy ; ”PID for Line Follower”, Chicago Area Robotics Group ; 2007

[4] Chaubey Pranjal ; ”The Modified Flood Fill Simulator”, The Robotics Institute ; 2008

[5] Maeda, Y. Kuswadi, Son. M, Nuh. Sulistyo MB. *Kontrol Automatik*. Politeknik Elektronika Surabaya; 1993.