ROBOT MICROMOUSE DENGAN MENGGUNAKAN ALGORITMA FLOOD-FILL

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat

Untuk Memperoleh Gelar Sarjana Teknik Elektro

Disusun oleh :

HARDI RIFKI AL'AMIN
NIM : 07524030

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
Y O G Y A K A R T A
2011

LEMBAR PENGESAHAN PEMBIMBING

ROBOT MICROMOUSE DENGAN MENGGUNAKAN ALGORITMA FLOOD-FILL

TUGAS AKHIR

Disusun oleh :

HARDI RIFKI AL'AMIN
NIM : 07524030

Yogyakarta, Oktober 2011
Telah disetujui dan disahkan oleh :

Pembimbing Tugas Akhir I	Pembimbing Tugas Akhir II
Dwi Ana Ratna Wati, ST., M.Eng.	**Medilla Kusriyanto, ST., M.Eng**

LEMBAR PENGESAHAN PENGUJI

ROBOT MICROMOUSE DENGAN MENGGUNAKAN ALGORITMA FLOOD-FILL

TUGAS AKHIR

Disusun oleh :

HARDI RIFKI AL'AMIN
NIM : 07524030

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Oktober 2011

Tim Penguji

Dwi Ana Ratna Wati, ST., M.Eng
Ketua

Tito Yuwono, ST., M.Sc.
Anggota I

Wahyudi Budi Pramono, ST., M.Eng.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Elektro
Universitas Islam Indonesia

Tito Yuwono, ST., M.Sc.

ABSTRAKSI

Robot *Micromouse* adalah sebuah robot cerdas yang dapat bergerak dengan bebas di dalam sebuah area labirin tanpa menyentuh objek sekitarnya, yang pada akhirnya robot mengetahui ke arah mana harus bergerak, berapa derajat harus berputar jika menemui jalan buntu pada area labirin tersebut, bekerja secara autonomous atau tanpa interfensi dari manusia. Robot *Micromouse* pada penelitian ini dirancang dengan sensor *proximity* yang dapat mendeteksi dinding dan rintangan lainnya pada labirin dengan cara mengembalikan nilai yang sebanding antara jarak robot dengan dinding kiri, dinding kanan, atau dinding di depan robot. Sensor encoder pada roda digunakan untuk mengetahui jarak yang dapat ditempuh robot *micromouse* dalam 1 kali putaran roda sehingga robot dapat mengetahui di sel labirin mana ia berada. Selanjutnya, data dari sensor ini akan disimpan oleh robot untuk pemetaan bentuk labirin. Algoritma *Flood-Fill* merupakan metode yang digunakan untuk menyelesaikan masalah pencarian rute terpendek pada labirin 5x5 dalam penelitian ini. Algoritma *Flood-Fill* melibatkan proses penomoran pada setiap sel dalam labirin dimana nomor-nomor ini merepresentasikan jarak setiap sel dengan sel tujuan. Sel tujuan yang ingin dicapai diberi nomor 0 dan sel–sel pada labirin yang memungkinkan untuk mencapai posisi tujuan, ditandai dengan cara n+1. Dengan algoritma ini, awalnya robot melakukan eksplorasi pada setiap sel pada labirin sehingga dihasilkan peta dari labirin tersebut, selanjutnya robot menelusuri sel-sel dengan nilai terkecil sesuai dengan peta yang telah dibuat dengan waktu yang lebih cepat dari waktu eksplorasi. Dari hasil pengujian eksplorasi pada beberapa lapangan, robot selalu berhasil menemukan *GOAL* dengan metode algoritma *flood-fill* ini, tetapi tidak semua lapangan yang peta jalur terpendeknya berhasil dibuat oleh robot yang mungkin disebabkan adanya informasi yang belum dimiliki robot ketika robot menemui beberapa kondisi baru di beberapa lapangan yang memerlukan tindakan khusus.

Kata kunci : Robot *Micromouse,* Algoritma *Flood-Fill*, Labirin, Jalur terpendek, Sensor *proximity,* Pemetaan, Encoder.

DAFTAR ISI

Halaman

BAB III PERANCANGAN SISTEM

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

HALAMAN MOTTO

"Takkan Pulang Sebelum Menang!"

(Tim Robot KRI Elektro UII 2007)

d(^_^)v

HALAMAN PERSEMBAHAN

Tugas akhir ini dipersembahkan untuk semua orang yang telah mendukung penulis dalam meraih kesuksesan, baik keluarga, teman, teman seperjuangan, dan teman yang jauh dimata tetapi dekat dihati. Terutama untuk Papa, Mama, Bang Andri dan kedua adik penulis yaitu Ayi dan Lia.

Semoga nantinya penulis bisa membanggakan mereka melalui segala hal-hal yang bersifat positif. Aamiin.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir di Universitas Islam Indonesia ini dengan baik.

Selama mengerjakan tugas akhir ini, baik pelaksanaan maupun dalam pengerjaan laporan tugas akhir, penulis banyak memperoleh bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan terima kasih kepada :

1. ALLAH S.W.T. atas segala rahmat dan karunia-Nya, sehingga penulis bisa melaksanakan tugas akhir ini dengan baik.
2. Papa dan Mama yang selalu mendukung dan mendo'akan penulis.
3. Bang Andri, Kak Melly, Ayi dan Lia yang selalu mendukung dan mendo'akan penulis.
4. Buat seseorang yang selalu ada dipikiran penulis selama pengerjaan tugas akhir ini, yang jauh dimata dekat dihati dan pikiran ☺.
5. Bapak Tito Yuwono, ST., M.Sc. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Industri Universitas Islam Indonesia.
6. Ibu Dwi Ana Ratna Wati, ST., M.Eng. selaku dosen pembimbing tugas akhir I yang telah mengarahkan dan memberi masukan dengan penuh semangat kepada penulis guna mendapatkan sistem pengontrolan robot yang handal dan terbaik dalam penggunaan Algoritma *Flood-Fill* pada penelitian ini. Terimakasih Bu Ana ☺.

7. Bapak Medilla Kusriyanto, ST., M.Eng. selaku dosen pembimbing tugas akhir II yang telah mengarahkan dan memberi masukan dengan penuh semangat kepada penulis guna mendapatkan sistem robot yang handal dan rangkaian elektronika yang berkompetensi pada penelitian ini. Terimakasih Pak Medilla ☺.
8. Seluruh Dosen Jurusan Teknik Elektro, terima kasih atas bimbingannya selama ini.
9. Mbak Umi selaku Administrasi Jurusan Teknik Elektro, terimakasih atas kebaikannya selama ini dalam mengurusi segala administrasi yang saya butuhkan serta saran-sarannya yang sangat bermanfaat ☺.
10. Mas Heri selaku Laboran Lab. Kendali dan Mas Anwar selaku Laboran Lab. Elektronika yang telah banyak membantu dan membimbing penulis selama ini.
11. Kepada Teman-teman seperjuangan konsentrasi kendali yang telah banyak mendukung dan memberi semangat kepada penulis selama ini. Semoga kita semua sukses, Aamiin.
12. Kepada Keluarga Besar Elektro angkatan 2007, terimakasih atas kerjasamanya selama ini, terima kasih atas dukungan dan do'a-nya, semoga kita semua sukses dan di permudah jalannya oleh Allah S.W.T. Aamiin... Hidup Elektro!!!! ☺
13. Khusus buat Sapta Nugraha, ST., yang sudah banyak membantu penulis selama ini, terutama sudah membantu membawa lapangan robot ketika penulis akan melakukan demo robot ke kampus. Thank's Boss ;)

14. Kepada Keluarga Besar Teknik Elektro Universitas Islam Indonesia, terimakasih atas segala dukungan, semangat, dan do'a-nya selama ini. Jangan pernah berhenti untuk terus mengharumkan nama Elektro UII tercinta. Hidup Elektro!!!! ☺
15. Kepada Keluarga Besar Tim Robot Elektro UII, terimakasih atas kerjasama dan ilmunya selama ini, terus berjuang dalam mengharumkan nama Universitas Islam Indonesia. "*Takkan pulang sebelum menang!!!*" ☺
16. Kepada teman-teman seperjuangan SMA Cendana Duri, terimakasih atas dukungan, semangat dan do'a selama ini. Sukses buat kita semua ☺. Aamiin.
17. Kepada semua pihak yang tidak dapat penulis sebutkan satu persatu, semoga Allah S.W.T. memberikan balasan yang terbaik. Aamiin.

Penulis menyadari bahwa mungkin masih banyak terdapat kekurangan dalam penulisan laporan ini. Oleh karena itu, penulis terbuka untuk menerima saran dan kritik yang bersifat membangun untuk perbaikan ke depan. Akhir kata penulis mengucapkan mohon maaf yang sebesar-besarnya apabila ada kesalahan yang penulis perbuat dalam pelaksanaan tugas akhir ini. Semoga laporan ini bermanfaat bagi kita semua. Aamiin. ☺

Yogyakarta, 19 September 2011

Penulis

BAB I

PENDAHULUAN

1.1 Latar Belakang

Robot adalah sebuah alat mekanik yang dapat melakukan tugas fisik, baik menggunakan pengawasan dan kontrol manusia, ataupun menggunakan program yang telah didefinisikan terlebih dulu (kecerdasan buatan). Robot berasal dari bahasa *Czech*, *Robota*, yang berarti pekerja. Dalam perkembangannya, robot mempunyai beberapa klasifikasi menurut fungsi dari robot itu sendiri, diantaranya terdapat robot manipulator (*arm*) yang biasanya digunakan dalam berbagai industri, robot mobil dan robot berkaki, robot bawah air (*underwater*), dan robot terbang (*flying robot*). Pada saat ini, penelitian di bidang robotika semakin banyak dilakukan di berbagai belahan dunia, termasuk Indonesia, karena perkembangan teknologi robotika telah membuat kualitas kehidupan manusia semakin tinggi.

Belakangan ini, berbagai bencana atau kecelakaan seperti kebakaran, bocornya gas beracun, kecelakaan di pertambangan atau gempa bumi seringkali terjadi. Para petugas penyelamat atau pemadam kebakaran, seringkali harus menempuh bahaya dan risiko untuk menyelamatkan dan mengevakuasi korban kebakaran atau bencana alam. Terkadang para petugas juga harus membayar mahal tugas mulia itu dengan nyawanya. Untuk dapat membantu dan menggantikan tugas manusia yang berat ini, dibutuhkan robot yang memiliki kemampuan cerdas. Robot – robot pembantu dan penyelamat dapat mengambil

alih tugas ini. Salah satu cara untuk dapat membuat robot yang cerdas, adalah dengan cara mengimplementasikan metode – metode kecerdasan buatan pada robot tersebut.

Robot *Micromouse* merupakan salah satu tipe robot yang memiliki kecerdasan buatan didalamnya. Akhir-akhir ini, banyak digelar pertandingan robot *micromouse* di berbagai belahan dunia, dan mulai hangat dibicarakan di Indonesia. Robot *Micromouse* adalah robot cerdas yang dapat bergerak bebas di dalam sebuah labirin (maze) tanpa menyentuh objek sekitarnya, dan robot akan mengetahui ke arah mana harus bergerak, berapa derajat harus berputar jika menemui jalan buntu pada area labirin (maze) tersebut.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka didapat rumusan masalah sebagai berikut:

a. Bagaimana cara merancang robot mobil yang dapat bergerak bebas di dalam sebuah labirin (maze) tanpa menyentuh objek sekitarnya.
b. Bagaimana cara merancang robot mobil yang dapat mengetahui ke arah mana harus bergerak dan berapa derajat harus berputar jika menemui jalan buntu pada area labirin (maze).
c. Bagaimana merancang Metode Algoritma *Flood-Fill* yang handal pada robot.
d. Bagaimana merancang robot yang ukurannya tidak lebih dari 16.8cm x 16.8cm.

1.3 Batasan Masalah

Adapun batasan-batasan masalah dalam penelitian ini adalah:

a. Bahasa pemograman yang digunakan adalah bahasa BASIC pada *compiler* BASCOM AVR versi 1.11.9.0.
b. Mikrokontroler yang digunakan adalah ATMEGA32 dari AVR.
c. Mengimplementasikan Algoritma *Flood-Fill* pada robot *micromouse* dalam pencarian rute dari sel awal menuju sel yang dituju (*GOAL*) dan melakukan pemetaan pada labirin (maze) 5x5.
d. Bentuk lintasan dari labirin sudah ditentukan sebelumnya.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk mengetahui bagaimana metode Algoritma *Flood-Fill* jika diimplementasikan pada masalah pencarian rute dan pemetaaan pada sebuah labirin berukuran 5x5. Jika terus dikembangkan, sistem pada robot *micromouse* ini dapat digunakan untuk menciptakan robot yang dapat menggantikan pekerjaan manusia yang dapat membahayakan keselamatan, misalnya menyelamatkan korban yang terjebak didalam gedung yang terbakar, dan tempat-tempat yang mempunyai radiasi tinggi.

1.5 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian dan sistematika dari penulisan laporan tugas akhir.

BAB II TINJAUAN PUSTAKA

Berisi tentang teori-teori yang berkaitan dengan penelitian ini yang diambil dari berbagai sumber disertai sedikit ulasan mengenai penelitian yang serupa dengan penelitian yang penulis lakukan disertai perbedaan.

BAB III PERANCANGAN SISTEM

Bagian ini menjelaskan perancangan *hardware* yang meliputi perancangan mekanik robot, perancangan rangkaian elektronik robot dan perancangan algoritma *flood-fill* pada *software*.

BAB IV ANALISA DAN PEMBAHASAN

Bagian ini berisi penjelasan bagaimana hasil pengujian dari implementasi algoritma *flood-fill* dalam mencari rute terpendek dari posisi *HOME* menuju posisi *GOAL* pada labirin berukuran 5x5 dengan susunan lapangan yang berbeda-beda.

BAB V PENUTUP

Berisi kesimpulan dari penelitian yang penulis lakukan pada tugas akhir ini beserta saran-saran apa saja yang dapat dilakukan untuk pengembangan penelitian ini lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Nur Syafidtri Anita, 2010, dari Fakultas Ilmu Komputer Jurusan Sistem Komputer Universitas Gunadarma di Jakarta, telah melakukan penelitian tentang robot *micromouse* dalam paper miliknya yang berjudul "Robot Micromouse Dengan Menggunakan Algoritma Depth-First Search". Pada paper ini, Nur Syafidtri Anita sebagai peneliti menggunakan algoritma *Depth-First Search* untuk algoritma pemogramannya. Pada algoritma ini dipakai sistem seperti pohon akar, dimana robot memulai mencari dari node akar terus ke level ke-1 atau ke level yang lebih tinggi dari kiri ke kanan, kemudian berpindah ke level selanjutnya, demikian pula dari kiri ke kanan hingga ditemukannya solusi. Peneliti menjabarkan bahwa keuntungan dari algoritma ini adalah robot tidak akan menemukan jalur buntu, tetapi memiliki kelemahan membutuhkan memori yang banyak,karena harus menyimpan semua node dalam satu pohon, juga membutuhkan waktu yang cukup lama untuk dapat menemukan solusi karena akan menguji n level untuk mendapatkan solusi pada level yang ke-(n+1).

Untuk Algoritma *flood-fill*, telah dilakukan penelitian oleh Abdullah M N Rahman, Akhmad Hendriawan dan Reesa Akbar , 2010, dari Jurusan Teknik Elektronika Politeknik Elektronika Negeri Surabaya (PENS-ITS) dalam paper mereka yang berjudul "Penetapan Algoritma Flood Fill Untuk Menyelesaikan

Maze Pada Line Follower Robot". Mereka menggunakan Algoritma Flood Fill dalam algoritma pemograman penelitiannya. Perbedaannya dengan penelitian yang penulis lakukan adalah dalam jenis robot yang digunakan dan medan yang dipakai. Robot yang digunakaan adalah robot line follower dengan garis sebagai pengganti labirinnya, tentunya terdapat perbedaan pada robot dan sistem yang akan digunakan karena penulis menggunakan Robot *micromouse* yang berjalan menyusuri labirin yang berupa sekat-sekat dinding.

2.2 Robot Micromouse

2.2.1 Sejarah Robot Micromouse

Robot Micromouse pertama kali dibuat oleh **Claude Elwood Shannon** pada tahun 1950 yang diberi nama ***Theseus***. *Theseus*, robot tikus yang bisa bermain maze (lorong simpang siur) mengambil suatu hal positif yang lebih mendekati aslinya. Robot dikendalikan oleh suatu rangkaian relay, tikus magnetis seukuran aslinya mengembara pada lorong simpang siur dengan 25 penyiku. Lorong simpang siurnya bisa diubah sesuka hati dan tikus kemudian memeriksa secara menyeluruh jalan yang harus ditempuh untuk menemukan titik tujuan. Setelah melalui lorong simpang siur, tikus bisa ditempatkan di manapun dan akan bergerak secara langsung sampai ke tujuan ditempatkan di arena yang tidak biasanya, tikus itu akan mencari sampai mencapai suatu posisi yang dikenal dan kemudian mulai menuju titik yang dikendaki, dan menambahkan pengetahuan yang baru atas lintasan yang dilaluinya ke memorinya. Hal ini nampak sebagai alat yang bisa belajar untuk pertama kalinya, pada tingkatan ini.

Gambar 2.1 Shannon dan Theseus

Pada kasus Theseus, “otak” dan “otot” adalah dua bagian yang terpisah dari tikus itu sendiri dan kenyataannya ada di bawah lorong simpang siur itu. Otak adalah suatu rangkaian sekitar 100 relay, dan ototnya berupa sepasang motor yang dikemudikan secara elektromagnet yaitu oleh tindakan magnetis menggerakkan tikus melalui lorong simpang siur itu. Dengan pengembangan untaian kondisi padat, dimungkinkan membuat tikus yang menyatu (otak dan ototnya ada dalam diri tikus itu sendiri). Sehingga apabila dibandingkan dengan Theseus, otaknya lebih kecil tetapi ukuran tikusnya lebih besar. Pada tahun 1978 sejumlah insinyur membangun tikus maze-solving untuk IEEE Spectrum yang menyelenggarakan acara ‘Amazing Micro Mouse Maze Contest,’ yang mana Theseus sebagai penampilan tamu.

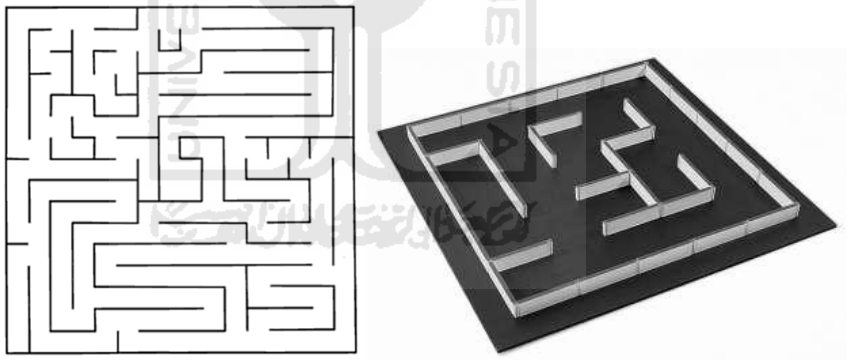
Jadi Robot Micromouse adalah sebuah robot cerdas yang dapat bergerak dengan bebas di dalam sebuah area labirin (maze) tanpa menyentuh objek sekitarnya, yang pada akhirnya robot mengetahui ke arah mana harus bergerak,

berapa derajat harus berputar jika menemui jalan buntu pada area labirin (maze) tersebut. Robot micromouse ini juga bekerja secara autonomous atau tanpa interfensi dari manusia.

2.2.2 Spesifikasi Maze dan Robot Micromouse

A. Spesifikasi Maze

- labirin harus terdiri dari 16 x 16 maze yang terdiri dari maze yang berukuran l8cm x l8cm unit persegi. Dinding labirin harus mempunyai tinggi 5cm dan tebal 1,2cm. Lorong-lorong antara dinding harusmempunyai lebar l6,8cm. Dinding luar harus menghubungkan seluruh labirin.



Gambar 2.2 Labirin

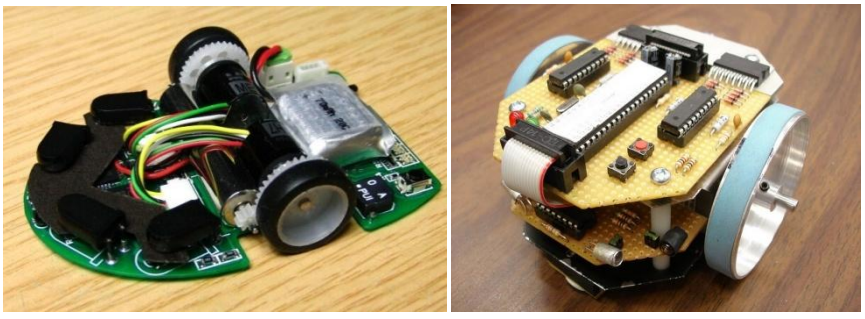
- Sisi dinding labirin harus putih, dan bagian atas dinding harus merah. Lantai labirin harus dibuat dari kayu dan di beri warna Hitam dof. Lapisan di atas dan sisi dinding harus dipilih untuk mencerminkan sinar infra-merah dan lapisan di lantai akan menyerapnya.

- Awal labirin harus terletak di salah satu dari empat sudut. Posisi mulai harus memiliki dinding pada tiga sisi. Orientasi posisi mulai harus sedemikian sehingga ketika dinding terbuka adalah ke 'utara', di luar dinding labirin berada di 'dalam' barat, dan 'selatan'. Di tengah-tengah labirin harus tempat terbuka yang terdiri dari 4 kotak unit. Tempat terbuka ini merupakan tujuan. Sebuah tanda berwarna merah dengan tinggi 20 cm dan 2,5 cm di setiap sisi, dapat ditempatkan di tengah tempat terbuka yang menjadi tujuan besar jika diminta oleh peserta.
- Labirin harus dibentuk sedemikian rupa sehingga ada setidaknya satu dinding yang menyentuh setiap titik kisi, kecuali untuk labirin tujuan.
- Dimensi labirin harus akurat ke dalam 5% atau 2 cm. Adanya spasi penyambungan labirin pada lantai tidak boleh lebih besar dari 1 mm. Perubahan lereng pada saat penyambungan tidak boleh lebih dari 4 derajat. Kesenjangan antara dinding kotak yang berdekatan tidak boleh lebih besar dari 1 mm.
- Sebuah sensor unutk posisi mulai akan ditempatkan pada perbatasan antara unit persegi pada posisi mulai dan unit labirin berikutnya. Sebuah sensor tujuan akan ditempatkan di pintu masuk ke maze tujuan. Sinar infra merah dari setiap sensor pada posisi horizontal dan posisi 1 cm di atas lantai.

- Harus ada beberapa jalur yang dapat dilalui untuk memperoleh posisi tujuan.

B. Spesifikasi Robot Micromouse

- Berukuran maksimum 25 cm x 25 cm. Tidak ada batasan untuk tinggi. Robot Micromouse harus benar-benar mandiri *(Autonomous)* dan tidak boleh menerima bantuan dari luar.
- Peserta dibebaskan memilih metode apa yang dipakai untuk penginderaan dinding labirin, yang penting, robot micromouse mouse tidak melakukan tindakan-tindakan yang dapat menyebabkan kerusakan pada dinding.
- Sumber robot harus berupa baterai.
- Jika robot micromouse dinyatakan dapat menyebabkan terjadinya kerusakan pada dinding labirin, maka juri tidak akan memperbolehkan robot micromouse itu untuk bertanding.
- Robot micromouse harus tidak menggunakan bahan-bahan yang dapat meledak.



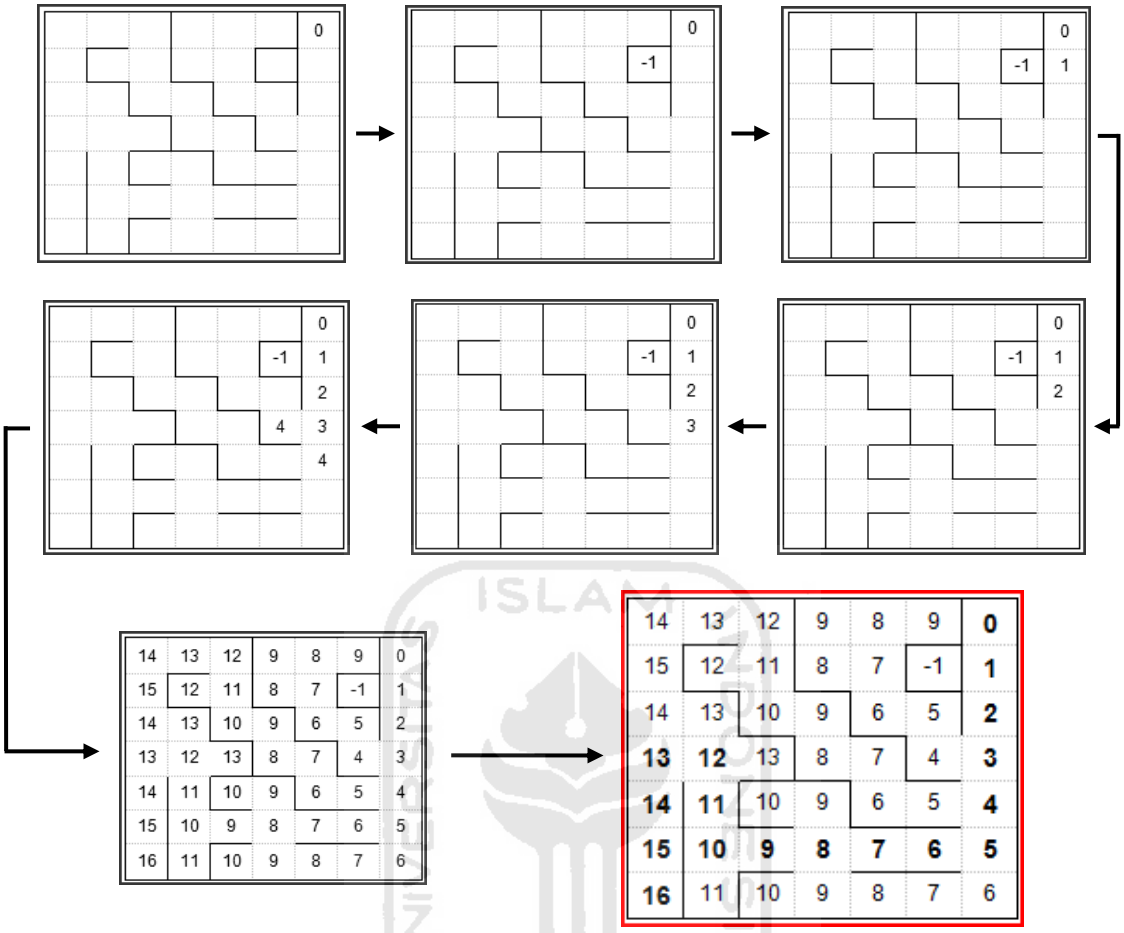
Gambar 2.3 Robot Micromouse

2.3 Algoritma *Flood-Fill*

Secara umum algoritma *Flood-Fill* akan menentukan daerah-daerah yang terhubung dengan suatu simpul dalam array multi-dimensi. Algoritma ini sering digunakan dalam program editor gambar bitmap untuk mewarnai suatu daerah terbatas dengan warna tertentu (*boundary fill*). Algoritma ini dapat diadaptasi untuk menyelesaikan permasalahan labirin secara dinamis. Algoritma Flood-Fill dapat dianalogikan seperti membanjiri labirin dengan air yang banyak. Air akan terus mengalir hingga mencapai lantai tempat tujuan. Jalur yang dilalui oleh tetesan air pertama di tempat tujuan merupakan jalur terpendek untuk mencapai tempat tujuan tersebut.

Algoritma *Flood-Fill* ini melibatkan proses penomoran setiap sel dalam labirin dimana nomor-nomor ini merepresentasikan jarak setiap sel dengan sel tujuan. Sel tujuan yang ingin dicapai diberi nomor 0. Setiap maze labirin yang mempunyai dinding di 4 sisinya, diberi nomor -1. Sel – sel pada labirin yang memungkinkan untuk mencapai posisi tujuan, kita tandai dengan cara n+1 sampai semua sel telah ditandai. Jika robot berada pada pada sel dengan nomor 1, maka robot tersebut hanya berjarak 1 sel dari sel tujuan. Jika robot berada pada sel dengan nomor 3, maka robot tersebut berjarak 3 sel dari posisi sel tujuan.

Dengan menganggap robot tidak dapat bergerak secara diagonal, maka simulasi pengisian nilai untuk maze dengan ukuran 7 x 7 tanpa dinding dapat dilihat pada gambar 2.4 berikut :

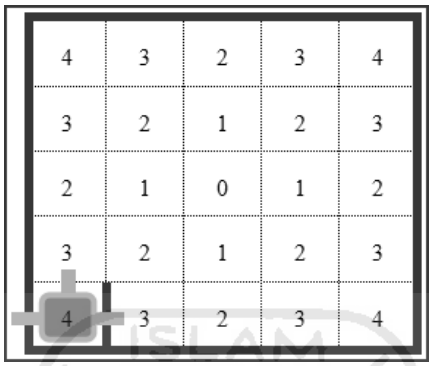


Gambar 2.4 Visualisasi proses penomoran labirin dengan Algoritma *Flood-fill*

Dari gambar 2.4 dapat dilihat sel dengan nomor (-1) merupakan sel yang tidak dapat dilalui oleh robot. Setiap nomor pada sel labirin menandakan jarak sel tersebut terhadap posisi tujuan. Nomor pada posisi mulai merupakan nomor tertinggi pada sel labirin, maka untuk dapat mencapai sel labirin tujuan robot hanya perlu mengikuti urutan nomor sesuai penomoran yang ada pada sel labirin, tentunya dengan memilih jalur yang terpendek. Oleh karena itulah Algoritma ini disebut Algoritma banjir.

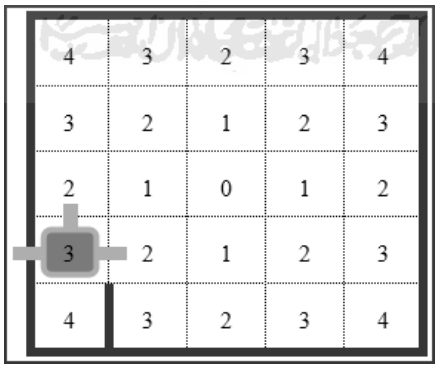
Berikut penjelasan lebih lanjut tentang bagaimana proses dari algoritma *Flood-Fill* ini bekerja pada robot micromouse pada maze dengan ukuran 5x5.

Perhatikan bahwa pada awalnya robot tidak tahu dimana lokasi dinding yang ada. Penomoran *maze* dilakukan berdasarkan jarak semua sel yang ada terhadap sel tujuan yang terletak di tengah.



Gambar 2.5 Robot pada posisi awal dengan peta buta

Dalam setiap pergerakan, robot perlu memeriksa sel-sel tetangga yang berdekatan untuk mengetahui apakah sel-sel di sekitar robot terhalang oleh dinding atau tidak, lalu berjalan menuju sel dengan jarak terendah.



Gambar 2.6 Robot mulai berjalan ke utara dan belum menemukan dinding

Pada gambar diatas, robot harus mengabaikan semua sel di sebelah barat karena terhalang oleh dinding dan harus melihat nomor sel yang berada di utara, timur, dan selatan karena tidak terhalang oleh dinding. Sel di utara bernomor 2,

sel di timur bernomor 2 dan sel di selatan bernomor 4. Sekarang robot berada pada posisi 3, robot perlu memilih sel mana yang memiliki nomor terkecil. Dalam hal ini ada dua sel yang mepunyai nilai “2”, yaitu sel di utara dan sel di timur, karena jika memilih sel timur maka robot harus berputar 90 derajat ke kiri, untuk mempersingkat waktu maka robot akan memilih sel utara yang hanya akan terus bergerak maju ke sel selanjutnya. Berikut gambaran proses penentuan keputusan penentuan dinding dapat dituliskan dalam program:

```
If      sel di utara terhalang oleh dinding Then
        Abaikan sel utara
Else
        Push sel utara ke stack untuk diperiksa
End If

If      sel di timur terhalang oleh dinding Then
        Abaikan sel timur
Else
        Push sel timur ke stack untuk diperiksa
End If
If      sel di selatan terhalang oleh dinding Then
        Abaikan sel selatan
Else
        Push sel selatan ke stack untuk diperiksa
End If
```

If sel di barat terhalang oleh dinding **Then**

Abaikan sel barat

Else

Push sel barat ke stack untuk diperiksa

End If

Tarik semua sel yang ada di stack

Sortir sel untuk mengetahui sel mana yang memiliki jarak terendah

Maju ke arah sel dengan jarak terendah yang didapatkan

Perlu diingat bahwa setiap sisi dinding dipakai bersama oleh dua buah sel (yang saling bertetangga tetapi terpisah oleh dinding , jadi ketika robot melakukan *update* nilai dari sebuah sel, maka robot juga harus melakukan *update* nilai sel-sel yang berdekatan dengan sel tersebut. Berikut gambaran proses *update* penentuan dinding dapat dituliskan dalam program:

If sel di utara terhalang oleh dinding **Then**

Pasang bit “utara” pada sel tempat kita berdiri

Pasang bit “selatan” pada sel di utara kita

Else

Do nothing

End If

If sel di timur terhalang oleh dinding **Then**

Pasang bit “timur” pada sel tempat kita berdiri

Pasang bit “barat” pada sel di timur kita

Else

Do nothing

End If

If sel di selatan terhalang oleh dinding **Then**

Pasang bit “selatan” pada sel tempat kita berdiri

Pasang bit “utara” pada sel di selatan kita

Else

Do nothing

End If

If sel di barat terhalang oleh dinding **Then**

Pasang bit “barat” pada sel tempat kita berdiri

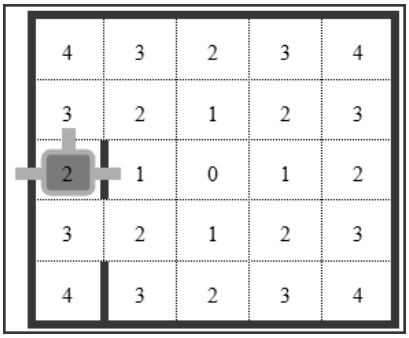
Pasang bit “timur” pada sel di barat kita

Else

Do nothing

End If

Seiring dengan perjalanan, robot akan menemukan dinding baru, dan diperlukan cara untuk meng-*update* penomoran sel tersebut.



Gambar 2.7 Robot menemukan dinding dan harus melakukan *re-flood*

Setelah maju 1 sel, robot menemukan dinding baru di sebelah timur sehingga robot tidak bisa bergerak ke sel disebelah timur yang memiliki nilai sel lebih kecil dari nilai sel tempat robot berada.

Dua sel di utara dan selatan memiliki nilai sel yang lebih tinggi dari nilai sel tempat robot berada sekarang, sehingga robot tidak bisa bergerak ke arah mana pun karena sel di utara dan selatan nilai sel nya lebih besar dari nilai sel tempat robot berada dan di barat dan timur robot terdapat dinding sel. Apabila kondisi seperti ini ditemukan maka robot perlu melakukan penomoran ulang *(re-flood)* dengan tetap menyimpan informasi pemetaan dinding sejauh ini. Robot akan menubah nilai sel tempat posisi robot berada sekarang dengan cara nilai sel tetangga di tambah 1 atau 3 + 1 = 4.

Berikut gambaran program untuk modifikasi prosedur proses algoritma *flood-fill* dalam melakukan *update* nilai sel tempat robot berada:

```
Memastikan bahwa stack telah kosong
Push sel tempat robot berada sekarang ke stack
Ulangi instruksi berikut sampai stack kosong :
Repeat hingga stack kosong
        Tarik sel yang terdapat di stack
{
If      nilai sel ini = 1 + nomor terkecil dari sel tetangga yang telah
        terbuka Then
        Do nothing
```

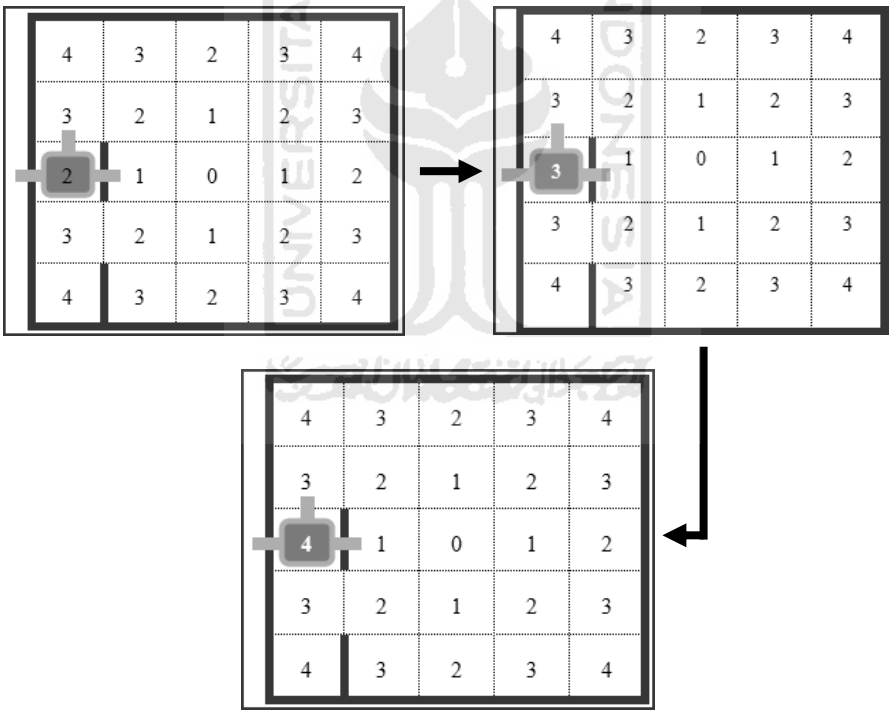
Else

Ubah sel ini menjadi 1 + nomor terkecil dari sel tetangga yang telah terbuka

Push semua sel tetangga yang telah terbuka ke dalam stack untuk di periksa

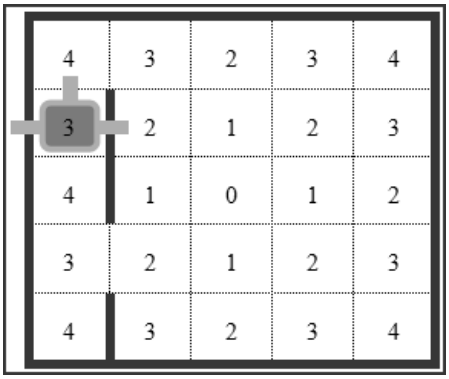
End If

End Repeat



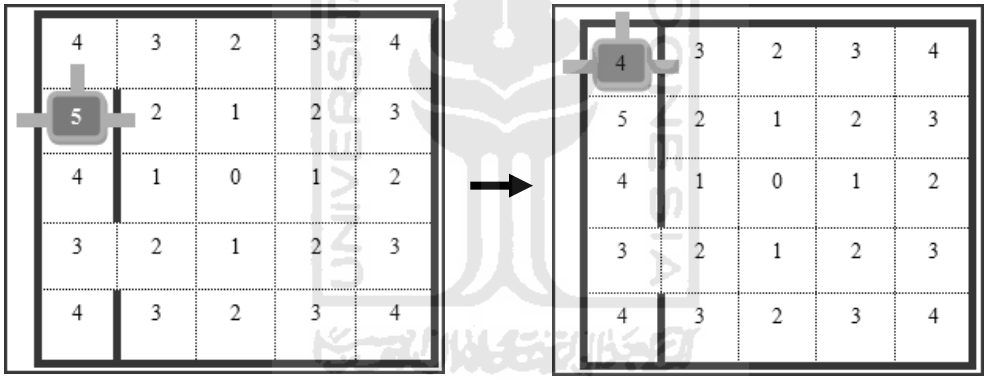
Gambar 2.8 Robot telah melakukan *update* peta labirin

Sekarang, robot bisa bergerak kembali karena dua sel tetangga yang tidak terhalang oleh dinding bernilai lebih kecil, yaitu sel utara dan selatan. Untuk mempersingkat waktu maka robot memilih bergerak maju.



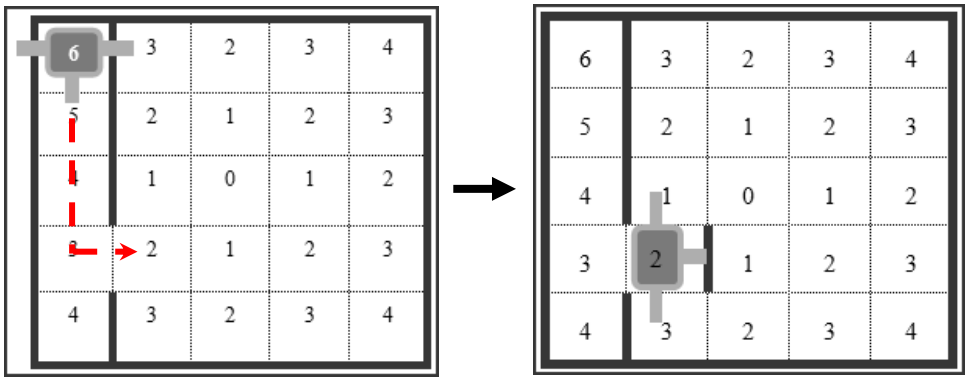
Gambar 2.9 Robot maju ke utara dan kembali menemukan dinding, *re-flood* kembali

Robot kembali menemukan dinding baru, dan menemukan kondisi yang sama seperti pada sel sebelumnya, maka robot melakukan penomoran ulang *(re-flood)* kembali dan berjalan menuju sel dengan jarak terendah.



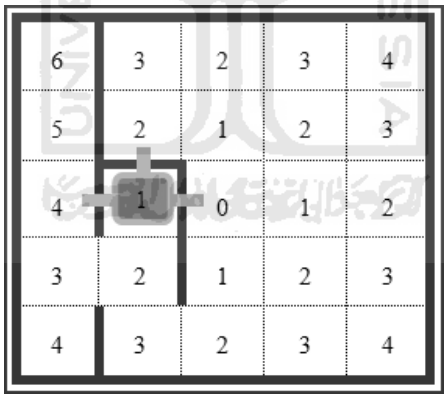
Gambar 2.10 Setelah melakukan *update* peta labirin, robot ke bergerak utara dan menemukan dinding

Robot menemukan jalan buntu, karena sel di utara, timur dan barat terdapat dinding. Karena dinding yang terbuka hanya dinding di selatan dan nilai sel di dinding selatan lebih besar dari nilai sel tempat robot berdiri maka robot kembali *re-flood* dan memutar arah 180 derajat, kemudian kembali berjalan mengikuti sel dengan nomor terkecil.



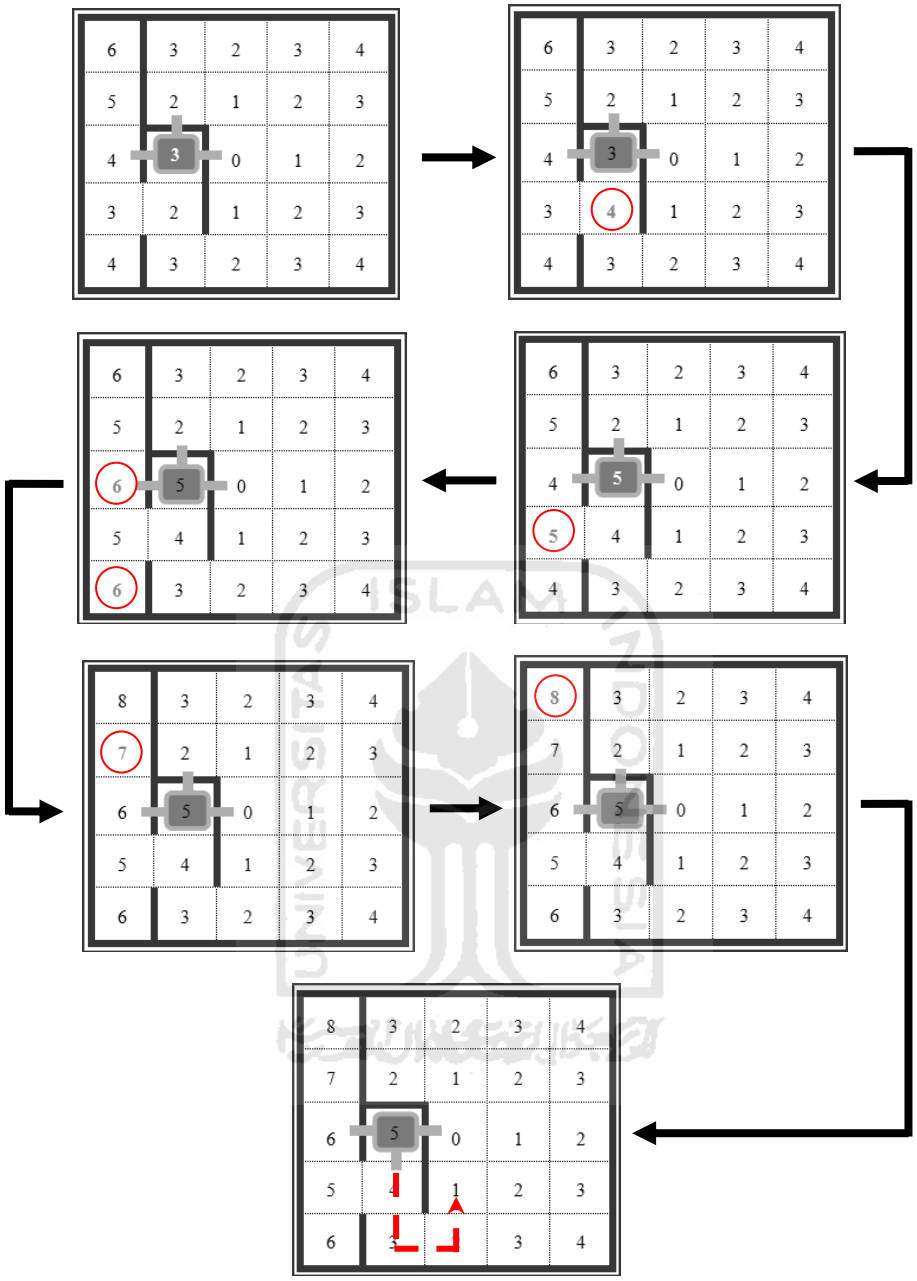
Gambar 2.11 *Update*, maju mengikuti nomor terkecil dan menemukan dinding

Robot kembali menemukan dinding dan segera dipetakan. Karena sel di utara memiliki nilai yang lebih kecil, maka robot berputar 90 derajat ke utara dan bergerak menuju sel tersebut. Robot kembali menumukan jalan buntu, dan kembali melakukan *re-flood* sel tempat robot berdiri saat ini.



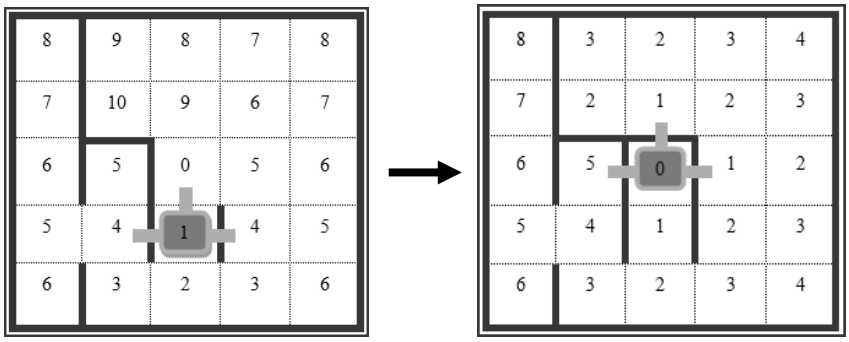
Gambar 2.12 Robot menemukan dinding dan harus berbalik arah

Tetapi itu menyebabkan sel ke selatan menjadi tidak benar, maka robot harus kembali memperbaiki petanya. Nilai sel tempat robot berada kembali di-*update* agar menjadi benar:



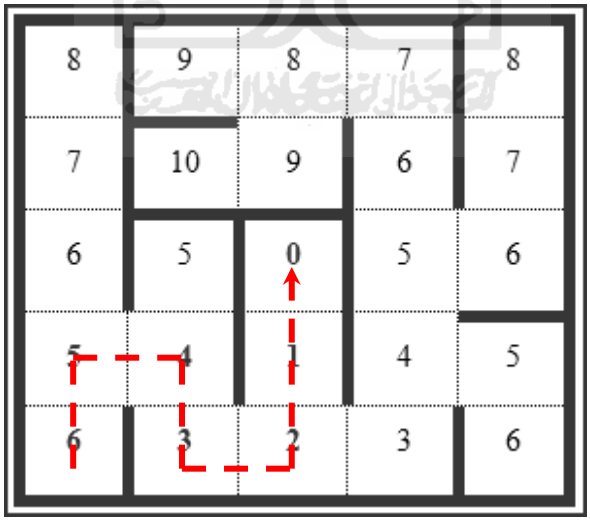
Gambar 2.13 Robot melakukan update peta labirin (*re-flood*)

Robot kembali menemukan dinding dan segera dipetakan. Karena sel di utara memiliki nilai yang lebih kecil, maka robot bergerak lurus menuju utara.



Gambar 2.14 Robot mengikuti nomor terkecil labirin dan menemukan *GOAL*

Walaupun bagain kanan dari labirin (maze) belum dieksplorasi, robot telah berhasil menemukan jalan terpendek menuju sel tujuan dengan menggunakan algoritma *flood-fill.* Sekarang yang dilakukan adalah kembali menuju tempat sel semula dan siap untuk menelusuri sel-sel dengan nilai terkecil sesuai dengan peta yang telah dibuat dengan kecepatan yang lebih tinggi dan waktu yang lebih cepat dari waktu eksplorasi.



Gambar 2.15 Peta jalur terpendek yang diperoleh dari hasil eksplorasi dengan menggunakan algoritma *flood-fill*

2.4 Mikrokontroler AVR ATMEGA32

Mikrokontroler adalah sebuah sistem komputer fungsional dalam sebuah chip. Di dalamnya terkandung sebuah inti prosesor, memori, dan perlengkapan input output. Dengan kata lain, mikrokontroler adalah suatu alat elektronika digital yang mempunyai masukan dan keluaran serta kendali dengan program yang bisa ditulis dan dihapus dengan cara khusus.

Mikrokontroler seri AVR pertama kali diperkenalkan ke pasaran sekitar tahun 1997 oleh perusahaan ATMEL, yaitu sebuah perusahaan yang sangat terkenal dengan produk mikrokontroler seri AT89S51/52-nya yang sampai saat ini masih banyak digunakan di lapangan.

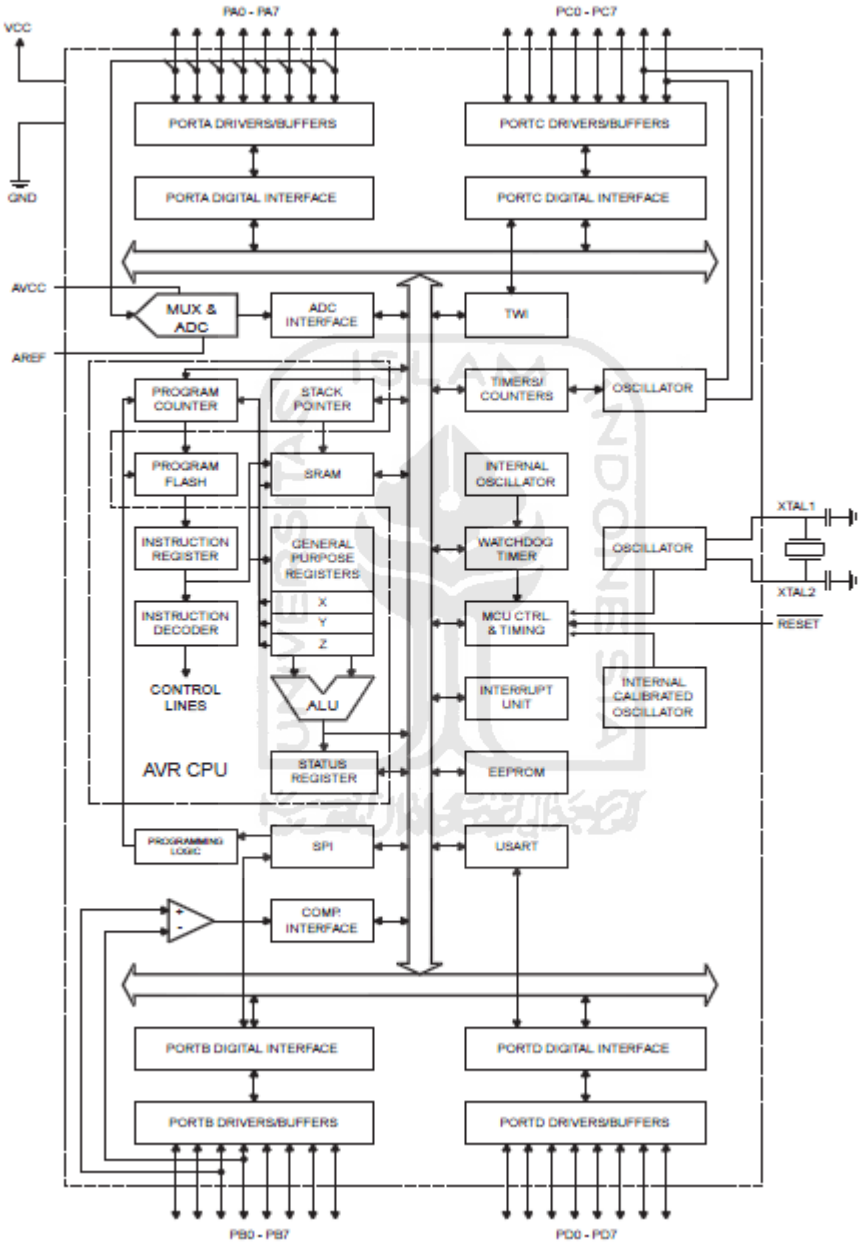
AVR (*Alf and Vegard's Risc processor*) merupakan mikrokontroler dengan arsitektur RISC (*Reduced Instruction Set Computer*) dengan lebar bus data 8 bit. Frekuensi kerja mikrokontroler AVR ini pada dasarnya sama dengan frekuensi osilator sehingga hal tersebut menyebabkan kecepatan kerja AVR untuk frekuensi osilator yang sama akan dua belas kali lebih cepat dibandingkan dengan mikrokontroler keluarga AT89S51/52.

Dengan instruksi yang sangat variatif (mirip dengan sistem CISC-*Complex Instruction Set Computer*) serta jumlah register serba guna (*General Purpose Register*) sebanyak 32 buah yang semuanya terhubung secara langsung ke ALU (*Arithmetic Logic Unit*), kecepatan operasi mikrokontroler AVR ini dapat mencapai 16 MIPS (enam belas juta instruksi per detik), sebuah kecepatan yang sangat tinggi untuk ukuran mikrokontroler 8 bit yang ada dipasaran sampai saat ini.

Untuk memenuhi kebutuhan dan aplikasi industri yang sangat beragam, mikrokontroler keluarga AVR ini muncul dipasaran dengan tiga seri utama, yaitu tinyAVR, ClassicAVR (AVR), dan megaAVR. Mikrokontroler keluarga AVR ini telah dilengkapi dengan memori *flash* sebagai memori program. Tergantung serinya, kapasitas memori *flash* yang dimiliki bervariasi dari 1KB sampai dengan 128 KB. Secara teknis, memori jenis ini dapat di program melalui saluran antar muka yang dikenal dengan nama *Serial Peripheral Interface* (SPI) yang terdapat pada setiap seri AVR tersebut. Dengan menggunakan perangkat lunak programmer (*downloader*) yang tepat, pengisian memori *flash* dengan menggunakan saluran SPI ini dapat dilakukan, bahkan ketika chip AVR telah terpasang pada sistem akhir (*end system*), sehingga pemogramannya sangat fleksibel dan tidak merepotkan pengguna (Secara praktis metode ini dikenal dengan istilah ISP (*In System Programming*), sedangkan perangkat lunaknya dinamakan *In System Programmer*.

2.4.1 Arsitektur ATMEGA32

Berikut blok diagram fungsional dari sebuah ATMEGA32 yang memiliki fitur-fitur yang cukup lengkap.



Gambar 2.16 Blok diagram ATMEGA32

Fitur-fitur yang dimiliki oleh ATMEGA32 antara lain:

a. Performa tinggi, mikrokontroler daya rendah.
b. Mikrokontroler dengan arsitektur RISC 8 bit.
    - 131 kode instruksi dalam bahasa assembly-hampir semuanya membutuhkan 1 clock untuk eksekusi.
    - Mempunyai 32 x 8 bit register kerja kegunaan umum.
    - Pengoperasian full statik, artinya clock dapat diperlambat, bahkan dihentikan sehingga chip berada dalam kondisi sleep. CMOS juga lebih tahan terhadap noise.
    - Kecepatan mengeksekusi sampai dengan 16 mega instuksi per detik pada saat diberikan osilator sebesar 16 MHz.
    - Terdapat rangkaian pengali 2 kali untuk siklus kerjanya dalam chip.
c. Flash EEPROM (*Electrically Eraseable Programmable Read Only Memory*) sebesar 32 *kilobyte* yang dapat diprogram ulang dan dengan kemampuan *Read While Write*.
d. Ketahanan hapus-tulis Flash ROM adalah 10.000 kali dengan pengaturan pilihan kode boot dan *Lock Bit* yang independen.
e. Memori SRAM sebesar 2 *kilobyte* yang dapat dihapus-tulis 100.000 kali.
f. Penguncian kode program untuk keamanan perangkat lunak agar tidak dapat dibaca.
g. Memori yang *non-volatile* EEPROM sebesar 1024 *byte*.

h. Memiliki 2 buah *timer/counter* 8 bit sebanyak 2 buah dan sebuah *timer/counter* 16 bit dengan opsi PWM sebanyak 4 kanal.

i. Memiliki 8 kanal *Analog to Digital Converter* 10 bit dengan jenis *single ended*.

j. Untuk kemasan TQFP ADC dapat diatur 7 buah kanal jenis diferensial dan khusus 2 kanal dengan penguatan yang dapat diatur melalui registernya sebesar 1x, 10x, atau 20x.

k. Antarmuka komunikasi serial USART yang dapat diprogram dengan kecepatan maksimal 2,5 Mbps.

l. Antarmuka SPI master/slave.

m. *Watchdog timer* dengan osilator didalam chip yang dapat diprogram.

n. Komparator analog di dalam chip.

o. Pendeteksian tegangan gagal yang dapat diprogram (*brownout detection*).

p. Osilator RC internal yang terkalibrasi.

q. Sumber interupsi internal dan eksternal.

r. Pilihan Mode *sleep : idle* , pereduksian noise ADC, penghematan daya konsumsi, penurunan daya, kondisi *standby*.

s. 32 pin masukan dan keluaran terprogram.

t. Terdapat pilihan kemasan PDIP 40 pin, TQFP 44 kaki, QFN/MLF 44 titik.

u. Tegangan pengoperasian

- 2,7 – 5,5 Volt untuk ATMEGA32(L)
- 4,5 – 5,5 Volt untuk ATMEGA32

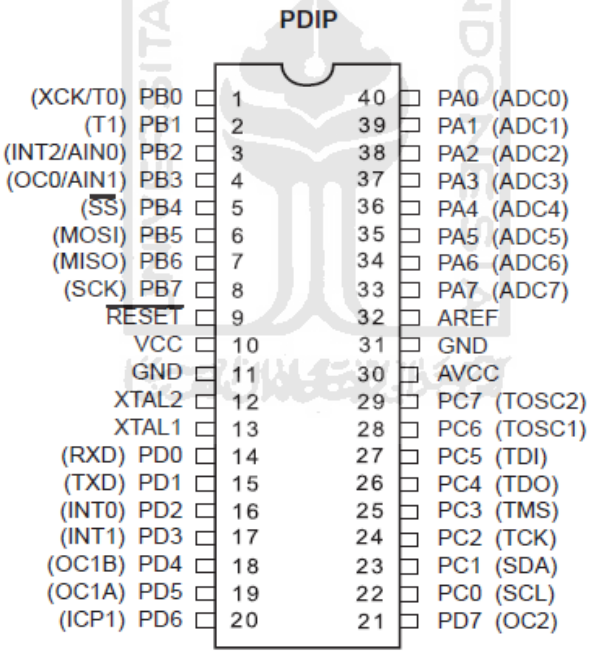
v. Kecepatan

- 0 – 8 MHz untuk ATMEGA32(L)
- 0-16 MHz untuk ATMEGA32

w. Konsumsi daya pada 1 MHz, 3 Volt, suhu 25°C untuk ATMEGA32(L)

- Aktif → 1,1 miliampere
- Mode idle → 0,35 miliampere
- Mode power down → kurang dari 1 mikroampere

ATMEGA32 mempunyai konfigurasi pin sebagai berikut:



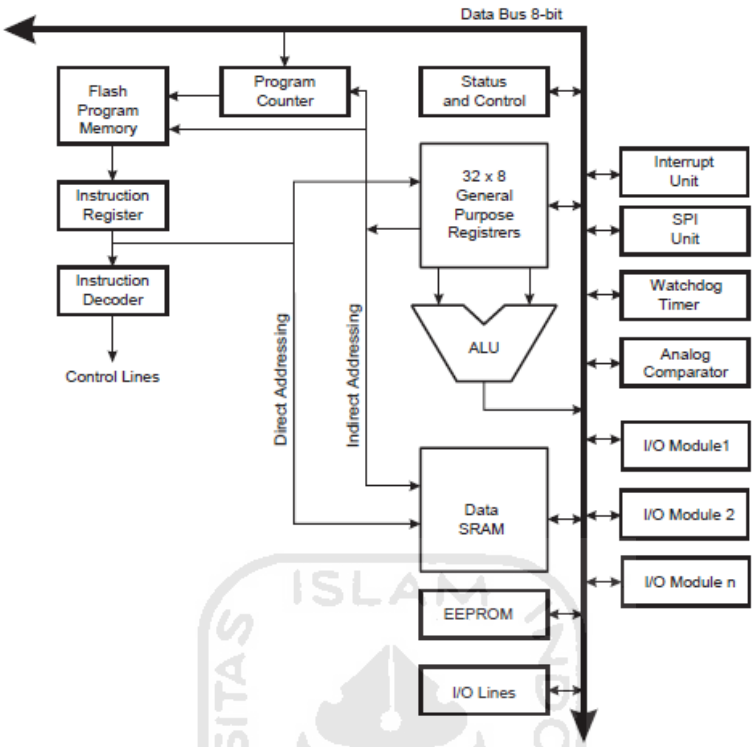
Gambar 2.17 Konfigurasi pin ATMEGA32

1. VCC pada pin 10, berfungsi sebagai catu daya positif.
2. GND pada pin 11 dan 31, berfungsi sebagai catu daya negatif/ground.

3. AVCC pada pin 30, berfungsi sebagai catu daya positif untuk tegangan ADC internal.
4. AREF pada pin 32, merupakan pin masukan untuk tegangan referensi eksternal ADC.
5. PA7 ..... PA0 pada pin 33 – 40, merupakan pin masukan dan keluaran PORTA dan berfungsi khusus sebagai pin masukan ADC.
6. PB0 ..... PB7 pada pin 1 – 8, merupakan pin masukan dan keluaran PORTB dan fungsi khusus sebagai pin Timer/Counter, komparator analog dan SPI.
7. PC0 ..... PC7 pada pin 22 – 29, merupakan pin masukan dan keluaran PORTC dan fungsi khusus.
8. PD0 ..... PD7 pada pin 14 – 21, merupakan pin masukan dan keluaran PORTD dan fungsi khusus.
9. RESET pada pin 9, merupakan pin masukan untuk melakukan reset mikrokontroler (*Active low*).
10. XTAL2 dan XTAL1 pada pin 12 dan 13, merupakan pin masukan untuk osilator eksternal.

2.4.2 Inti CPU ATMEGA32

Inti CPU diperlihatkan pada gambar 2.18 yang menunjukkan kedudukan masing-masing modul dalam CPU. Modul-modul tersebut tergabung menjadi satu dengan sebuah jalur data sebesar 8 bit sehingga pada chip mikrokontroler dengan fitur yang lebih banyak tidak perlu mengubah jalur data, tetapi dapat diatasi dengan menambah fungsi modul *status and control* seiring dengan bertambahnya modul.



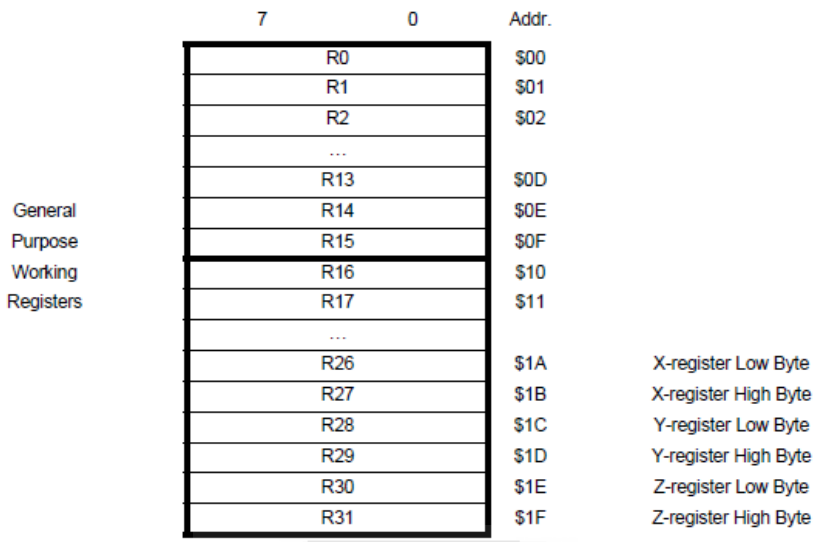
Gambar 2.18 Inti CPU ATMEGA32

2.4.3 Peta Memori ATMEGA32

A. Register Kegunaan Umum R0 - R31

Untuk keperluan umum, AVR memiliki 32 register bantu (R0-R31), masing-masing sebesar 8 bit yang menempati alamat memory data $0000 - $001F. Semua register bantu ini dapat digunakan langsung karena terhubung langsung dengan ALU, tetapi juga memiliki fungsi yang berbeda-beda. Pasangan R1 dan R0 berfungsi untuk menampung hasil perhitungan aritmetika. R16 - R31 dapat digunakan untuk mengambil data secara "*immediate*".

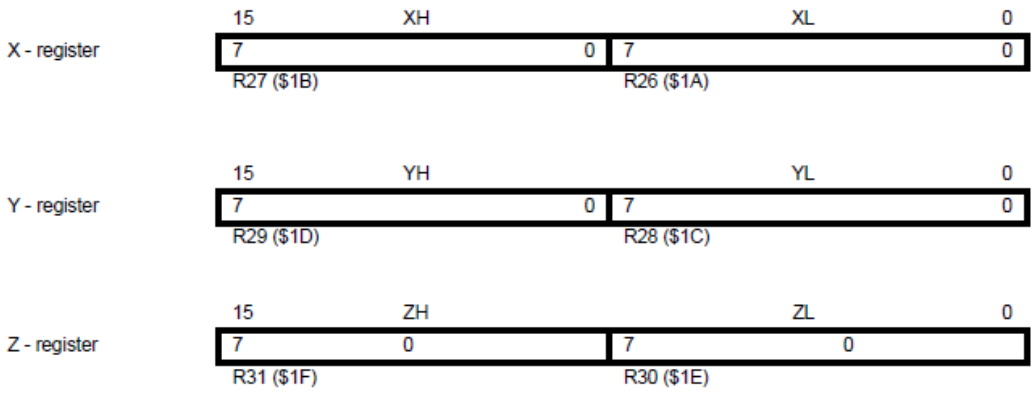
Memori yang ada di dalam mikrokontroler ATMEGA32 ada beberapa macam, misalnya *General Working Register* (Register Kegunaan Umum) sebanyak 32 *byte*, seperti yang ditunjukkan pada gambar 2.19 berikut :



Gambar 2.19 Register fungsi umum

Register tersebut dinamai R1 sampai dengan R31 dari alamat $00 sampai dengan $1F. Untuk pengalamatan secara *indirect* dapat digunakan pasangan *pointer register* sebesar 16 bit yang dirangkai dari 2 buah *register*, yaitu :

- *Pointer* X ⟶ XH = R27, XL = R26
- *Pointer* Y ⟶ YH = R29, YL = R28
- *Pointer* Z ⟶ ZH = R31, ZL = R30



Gambar 2.20 Register X, Y dan Z

B. Stack Pointer Register

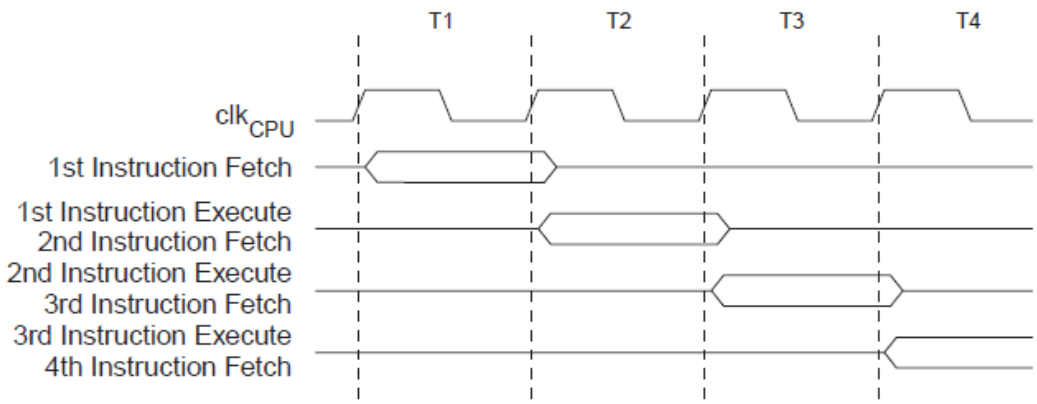
Sebagian besar *stack* digunakan untuk menyimpan data temporer, variabel lokal, dan untuk menyimpan alamat memori setelah terjadi interupsi dan *subroutine*. Alamat SP harus dimulai di atas $60. Alamatnya akan dikurangi satu ketika ada data masuk ke dalam *stack* , didorong dengan instruksi *PUSH* dan dikurangi sebanyak 2 ketika dimasukkan alamat untuk kembali saat menerima instruksi interupsi atau operasi memanggil subrutin. SP akan dinaikkan satu ketika data diambil dari *stack* dengan instruksi POP, dan akan dinaikkan 2 ketika data diambil dari *stack* dengan instruksi RET dari sebuah subrutin, atau RETI dari subrutin interupsi.

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Gambar 2.21 *Stack pointer register*

C. Diagram Instruction Fetch dan Instruction Execute

Gambar 2.22 dibawah menjelaskan konsep waktu pengaksesan secara umum untuk pengeksekusian sebuah intruksi. Kerja CPU AVR digerakkan oleh *clock* CPU (^{clk}CPU), yang juga sebagai sumber *clock* untuk keseluruhan kerja piranti dalam chip. Gambar 2.22 menunjukkan proses paralel (bersamaan) dari pengambilan kode instruksi dan eksekusinya datur oleh arsitektur Harvard dan konsep akses cepat file register.

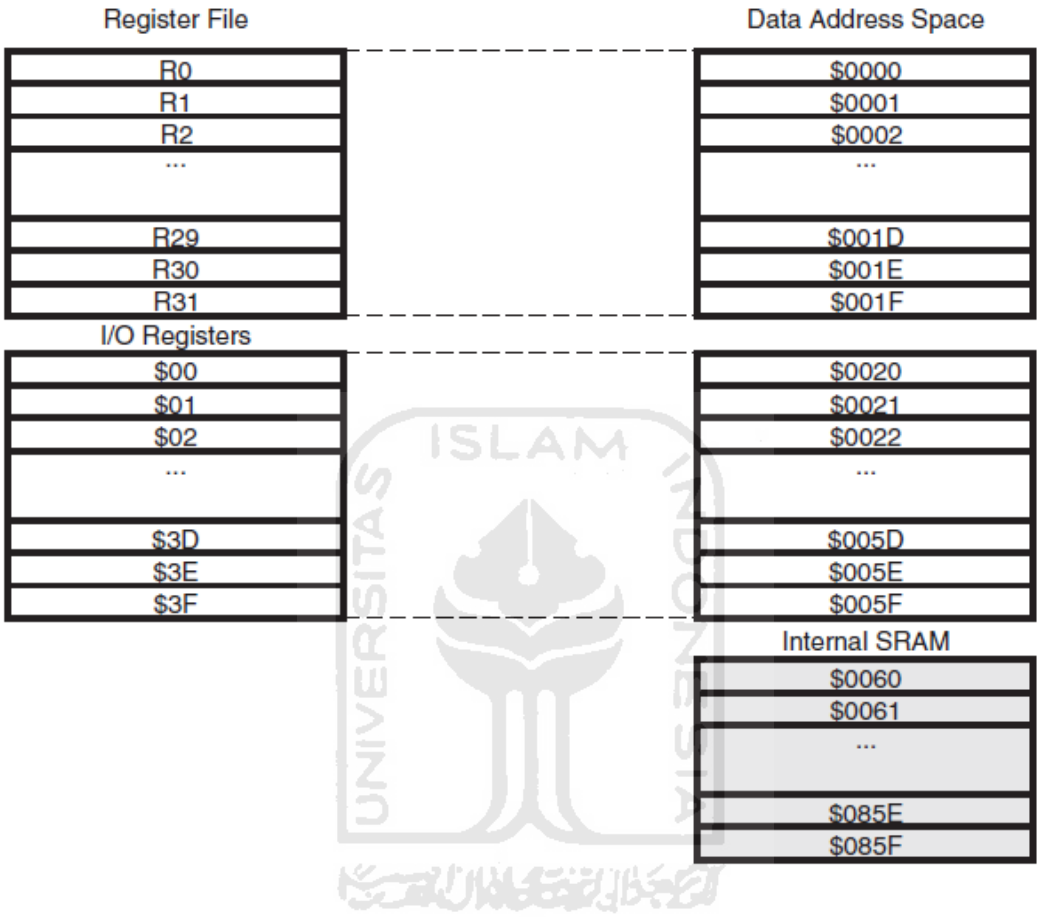


Gambar 2.22 Diagram waktu pengambilan kode instruksi dan eksekusinya secara paralel

D. Register I/O (Register Input/Output)

Register I/O, yaitu register yang berfungsi mengatur fungsi modul-modul pada mikrokontroler, menempati 64 alamat mulai dari $0020 sampai dengan $005F. Kita dapat membayangkan register I/O ini sebagai SFR (*Special Function Register*) yang terdapat pada AT89S52/51. Alamat berikutnya, yaitu $0060 sampai $085F sebesar 2 *kilobyte* berfungsi sebagai SRAM internal. Alamat $085F adalah akhir dari alamat SRAM internal atau disebut RAMEND. Kelompok register ini berfungsi sebagai pengatur kerja fitur-fitur dan peripheral yang terdapat pada AVR, misalnya untuk mengatur kerja interupsi maka digunakan register GICR pada alamat $3B ($5B), PORTA sampai dengan PORTD juga terdapat pada register I/O. Untuk lebih lengkap, register apa saja yang terdapat pada register I/O beserta alamatnya dapat dilihat pada Gambar 2.23 yang berupa tabel dari peta memori yang diambil dari *datasheet* ATMEGA32/8535. Register ini tidak dapat kita set langsung dengan data immediate. Perpindahan data antar register I/O pun tidak dapat dilakukan langsung melalui perintah MOV. Untuk

pengisian register I/O harus melalui register serbaguna (R0-R31), baru diisikan ke register I/O yang bersangkutan.

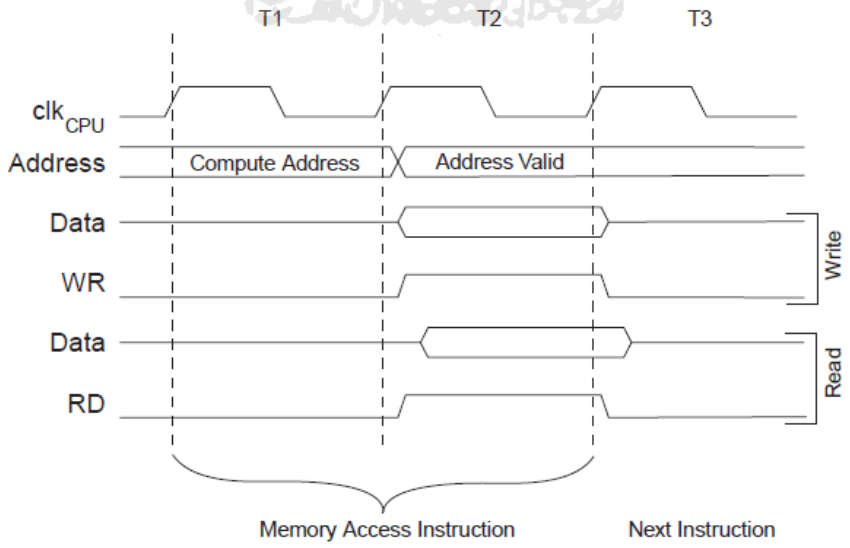


Gambar 2.23 Peta data memori

SRAM adalah data space kosong yang dapat kita gunakan sebagai tempat penyimpanan variabel, data, stack, dan keperluan lainnya. Alamat SRAM dimulai pada alamat $0060 dan berakhir pada alamat $085F untuk ATMEGA32 dan alamat $025F untuk ATMEGA8535. SRAM ini tidak terhubung pada ALU sehingga untuk operasi yang menggunakan data pada SRAM harus melalui register umum R0-R31. Akan tetapi data pada SRAM dapat diakses secara *direct* (langsung) maupun *indirect* (melalui Pointer Register (XYZ)).

Untuk penyimpanan data, mikrokoktroler AVR menyediakan dua jenis memori yang berbeda, yaitu EEPROM (*Electrically Eraseable Programmable Only Memory*) dan SRAM (*Static Random Access Memory*). EEPROM umumnya digunakan untuk menyimpan data-data program yang bersifat permanen, sedangkan SRAM digunakan untuk menyimpan data variabel yang dimungkinkan berubah setiap saatnya. Kapasitas simpan data kedua memori ini bervariasi, tergantung pada jenis AVR-nya. Untuk seri AVR yang tidak memiliki SRAM, penyimpanan data variabel dapat dilakukan pada register serbaguna yang terdapat pada CPU mikrokontroler tersebut.

Dalam siklus *clock* tunggal pada operasi ALU menggunakan dua register *operand* dieksekusi, dan hasilnya direkam kembali ke register tujuan. Terlihat juga proses pembacaan dan penulisan memori dapat dilakukan hampir bersamaan, sehingga ini adalah salah satu fitur AVR, yaitu *read and write*. Gambar 2.24 berikut menunjukkan konsep pewaktuan internal register file :

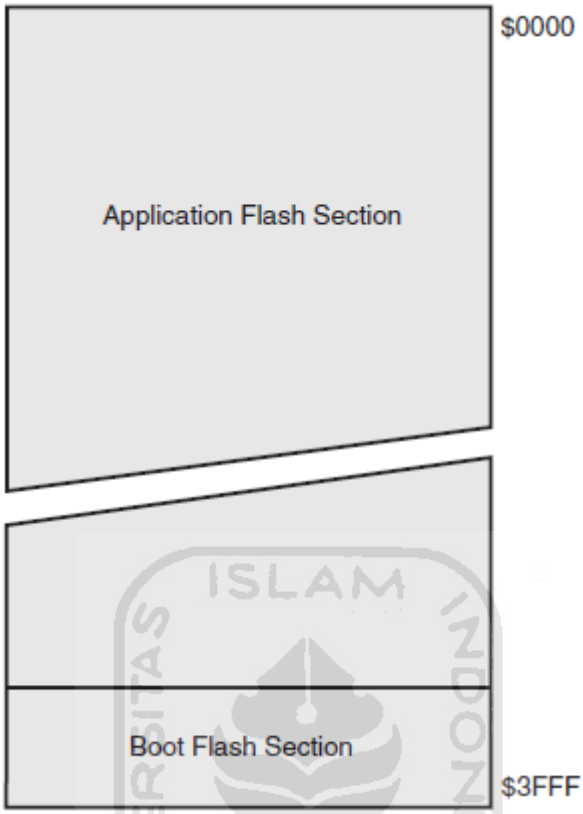


Gambar 2.24 Operasi ALU satu siklus

E. Flash Memory

Memori ini mempunyai kegunaan menyimpan kode-kode instruksi sehingga merupakan memori dengan kapasitas terbesar di antara memori yang ada didalam sebuah chip mikrokontroler. Memori program yang terletak pada memori jenis ini tersusun dalam 1 *word* atau 2 *byte* dengan lebar kode instruksi sebesar 16 bit atau 32 bit. ATMEGA32 memiliki 32 *kilobyte* x 16 bit dengan alamat dari $000 sampai dengan $3FFF. Mode pengalamatan memori ini ditangani oleh *Program Counter* (PC) sebesar 12 bit.

Untuk keperluan keamanan perangkat lunak, lahan memori Flash dibagi menjadi 2 bagian, bagian *Boot Program* dan bagian *Application Program*. Memori ini mampu bertahan hingga 10.000 kali hapus/tulis. Program *counter* ATMEGA32 mempunyai lebar 14 bit sehingga dapat mengalamati 16K lokasi memori program. Tabel konstanta dapat di alokasikan ke setiap alamat dalam memori program tanpa ketentuan tertentu.



Gambar 2.25 Peta memori program ATMEGA32

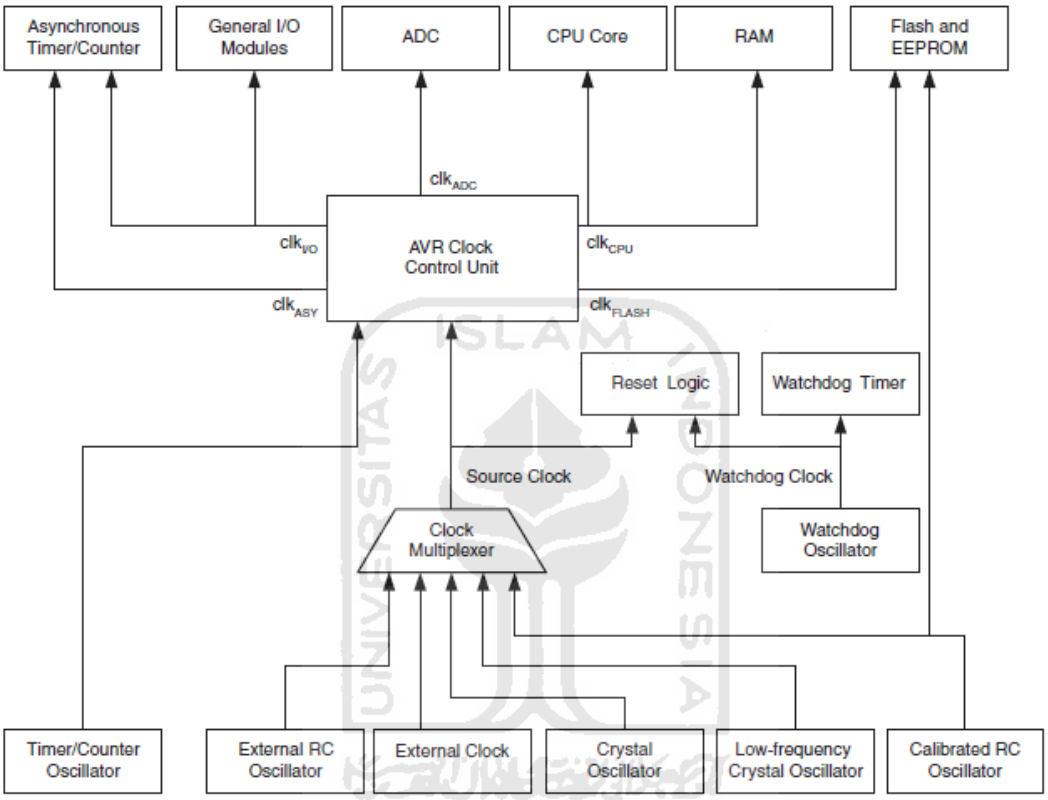
F. EEPROM (Electrically Eraseable Programmable Only Memory)

ATMEGA32 berisi 1024 *byte* memori data EEPROM atau memori yang dapat ditulis dan dihapus secara elektrik. Memori ini diorganisasikan agar dapat diakses baca dan tulis dalam satu *byte*. EEPROM mempunyai ketahanan, sedikitnya 100.000 kali hapus/tulis.

G. Sistem Penditribusian Clock

Clock diperlukan oleh piranti-piranti didalam mikrokontorler untuk dapat bekerja. *Clock* pada tubuh manusia dapat diibaratkan sebagai jantung, yang berfungsi memompa darah keseluruh jaringan tubuh. Jika salah satu bagian tubuh terganggu maka akan mengganggu kerja keseluruhan sistem tubuh metabolisme

manusia. *Clock* juga berfungsi menyinkronkan kerja seluruh piranti dalam chip, seperti piranti *Timer/Counter*, ADC, RAM, *Flash memory*, EEPROM, inti CPU, dan modul antarmuka masukan/keluaran.



Gambar 2.26 Diagram pendistribusian *clock*

Sumber pembangkit *clock* mempunyai beberapa opsi, antara lain :

1. *Calibrated RC Oscillator*

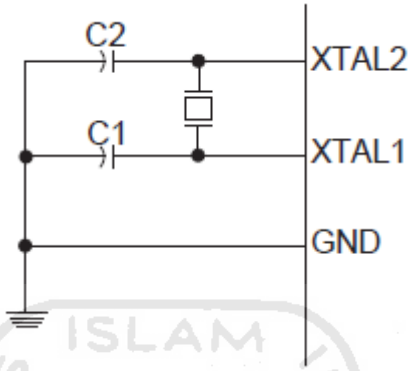
   Berupa jaringan resistor dan kapasitor dengan frekuensi rendah sekitar 1 MHz.

2. *Low Frequency Crystal Oscillator*

   Berupa osilator kristal dengan frekuensi rendah yang ditambahkan pada pin XTAL1 dan XTAL2.

3. *Crystal Oscillator*

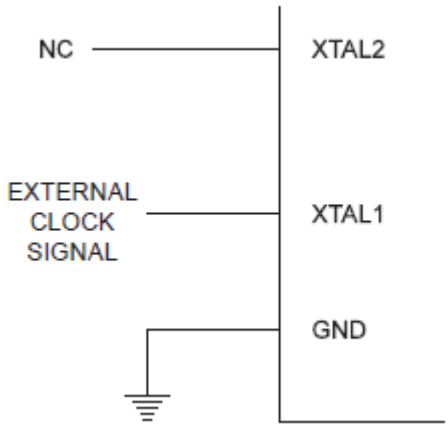
Berupa osilator kristal dengan frekuensi tinggi yang ditambahkan pada pin XTAL1 dan XTAL2.



Gambar 2.27 Koneksi *Crystal Oscillator*

4. *External Clock*

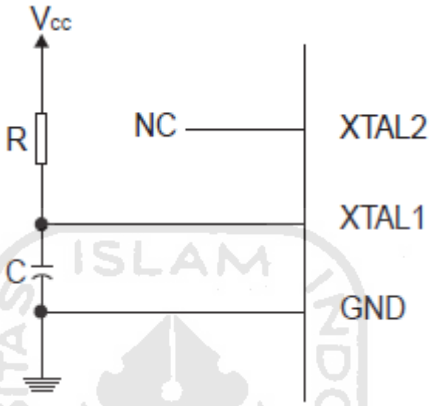
Clock diperoleh dari luar, dapat berupa rangkaian penghasil pulsa periodik, *Voltage to Frequency Converter*, dan sebagainya, yang diberikan pada pin XTAL1.



Gambar 2.28 Konfigurasi *External Clock*

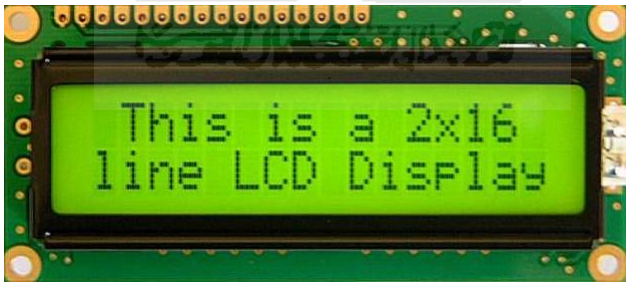
5. *External RC Oscillator*

Berupa jaringan resistor dan kapasitor yang ditambahkan pada pin XTAL1 dengan persamaan estimasi frekuensi keluaran f = 1/(3RC). Nilai C minimal harus 22 pF.



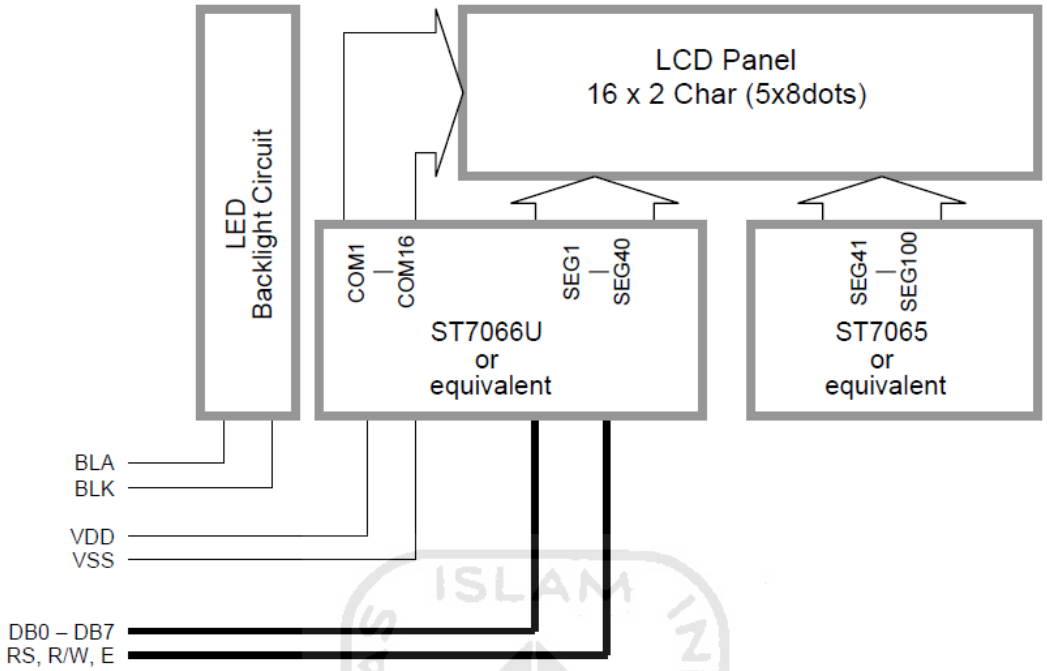
Gambar 2.29 Konfigurasi *External RC Oscillator*

2.5 LCD TOPWAY LMB162ABC



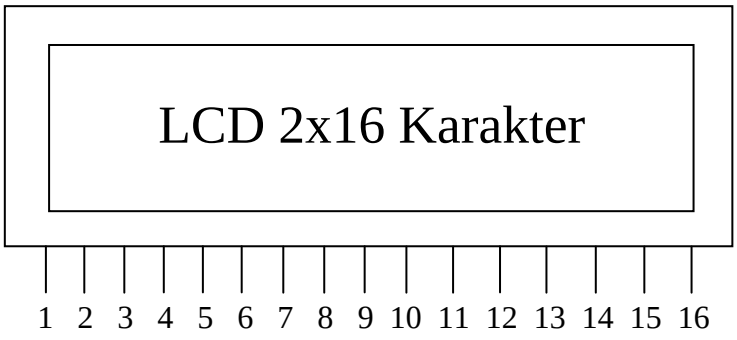
Gambar 2.30 LCD dengan 2x16 karakter

LCD (*Liquid Crystal Display*) adalah suatu penampil dari bahan cairan kristal yang pengoperasiannya menggunakan sistem *dot matriks*. LCD banyak digunakan sebagai penampil dari alat-alat elektronka seperti kalkulator, multimeter digital, jam digital, aplikasi robot, dan sebagainya.



Gambar 2.31 Blok diagram LCD dengan 2x16 karakter

LCD dapat dengan mudah dihubungkan dengan mikrokontroler AVR ATMEGA32. LCD yang digunakan dalam penelitian ini adalah LCD 2x16, yaitu LCD yang memiliki lebar *display* 2 baris 16 kolom, yang mempunyai 16 pin konektor seperti pada gambar 2.32 berikut :



Gambar 2.32 Konfigurasi pin keluaran LCD dengan 2x16 karakter

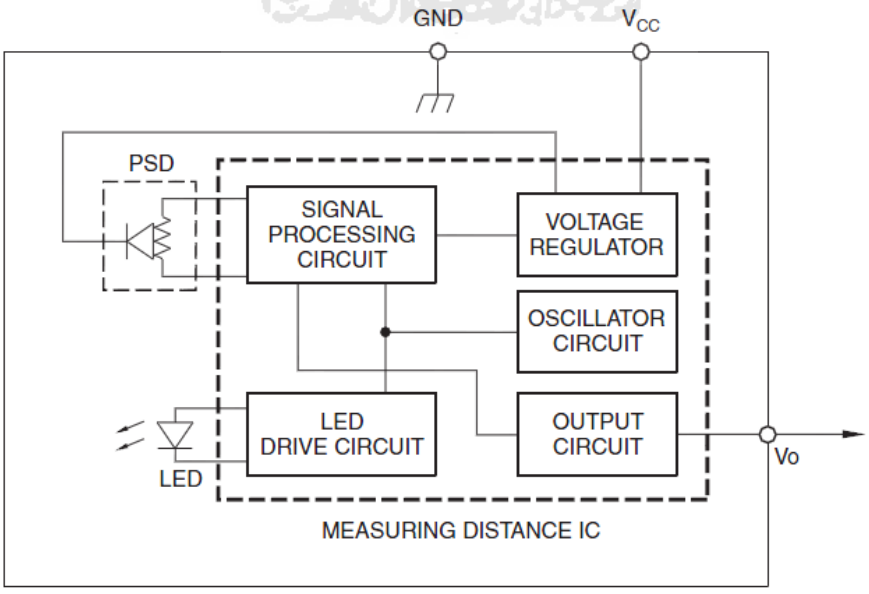
Tabel 2.1 Tabel konfigurasi pin keluaran LCD dengan 2x16 karakter

No	Nama Pin	Deskripsi
1	VSS (*Power*)	Sumber tegangan, *Ground* (0V).
2	VDD (*Power*)	Sumber tegangan positif, +5V.
3	V0 (*Power*)	Tegangan referensi untuk mengatur cahaya LCD.
4	RS (*Input*)	*Register Select*, berfungsi sebagai indikator atau uang menentukan jenis data yang masuk, apakah berupa data atau perintah. Logika *low* (0) menunjukkan yang masuk adalah perintah, sedangkan logika *high* (1) menunjukkan yang masuk adalah data.
5	R/W (*Input*)	*Read/Write*, berfungsi sebagai instruksi, jika *low* berarti menulis data dan *high* berarti membaca data.
6	E (*Input*)	*Enable Clock* LCD, logika 1 setiap kali pengiriman atau pembacaan data.
7	DB0 (I/O)	*Bi-directional tri-state data bus*. • DB0 – DB7 digunakan dalam mode 8 bit. • DB4 – DB7 digunakan dalam mode 4 bit, DB0 – DB3 tetap terbuka.
:	:	
14	DB7 (I/O)	
15	BLA (*Power*)	*Backlight Anode*, yaitu tegangan positif *backlight*.
16	BLK (*Power*)	*Backlight Katode*, yaitu tegangan negatif *backlight*.

2.6 SHARP GP2D120

Gambar 2.33 Sharp GP2D120

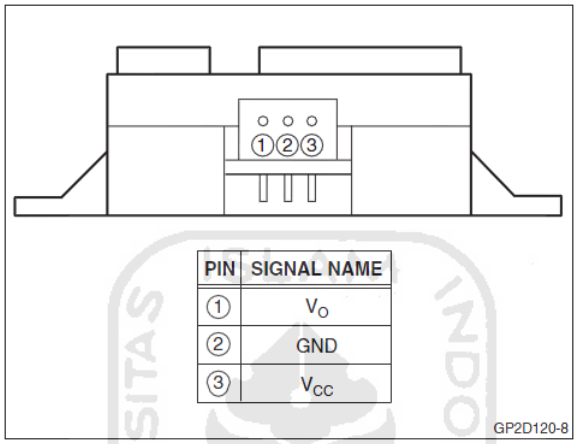
Sensor GP2D120 ini merupakan sensor deteksi jarak dengan pemrosesan sinyal yang terintegrasi dan keluaran berupa tegangan analog. Berikut blok diagram dari Sharp GP2D120 yang terdiri dari LED pemancar dan penerima yang memiliki rangkaian pemrosesan sinyal, rangkaian pengemudi LED, rangkaian osilator, dan rangkaian keluaran analog :



Gambar 2.34 Blok diagram Sharp GP2D120

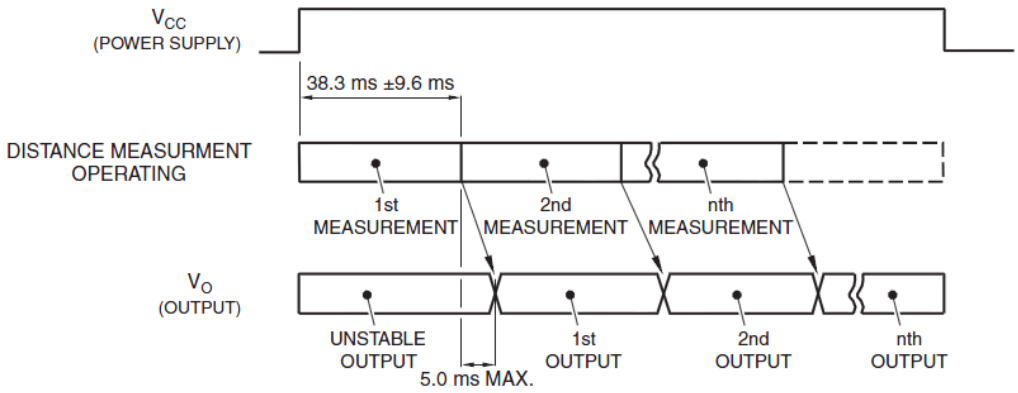
Sensor ini mempunyai keluaran 3 kabel, yang terdiri dari :

1. Pin 1 sebagai tegangan keluaran.
2. Pin 2 sebagai *Ground.*
3. Pin 3 sebagai tegangan masukan sensor.



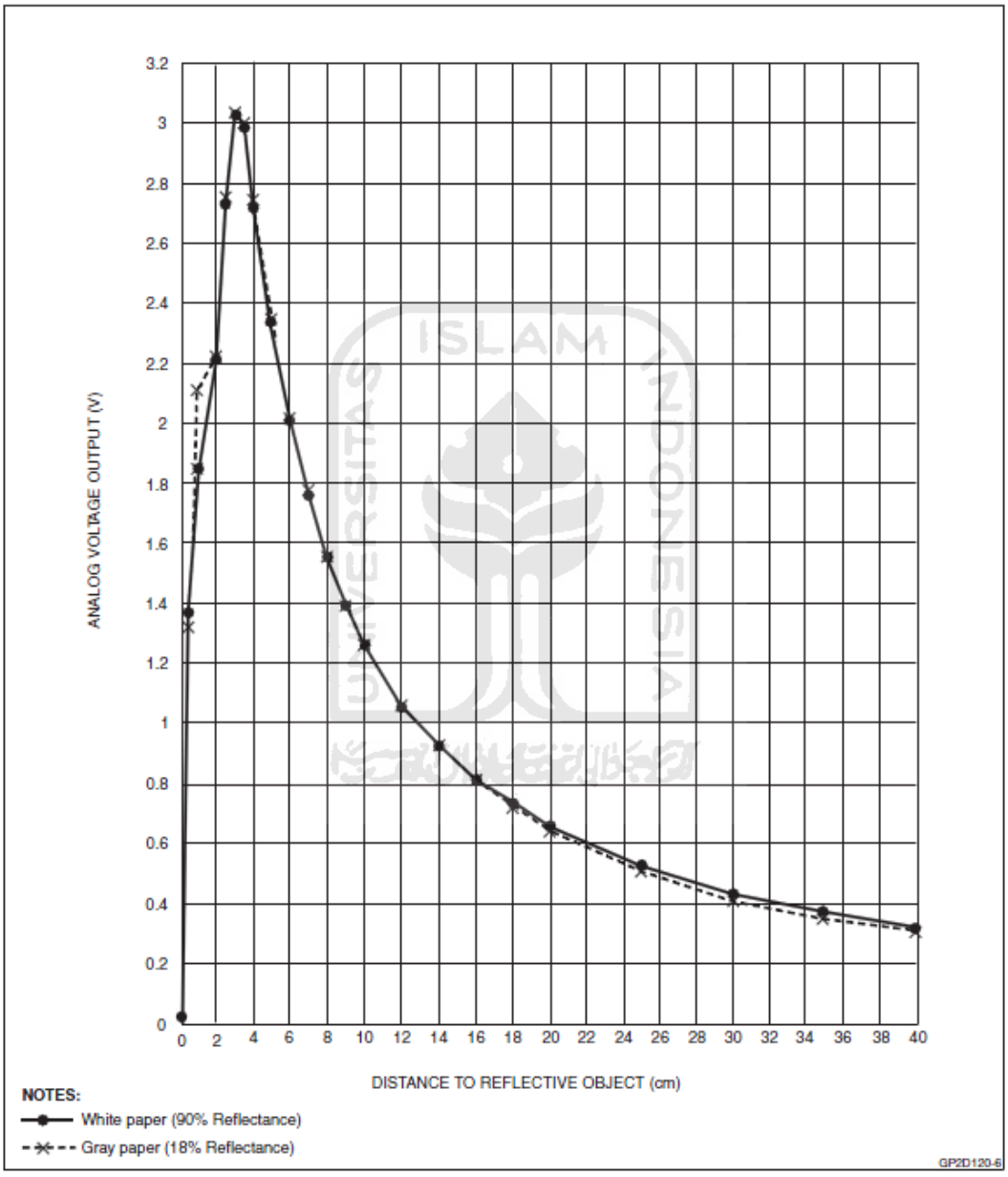
Gambar 2.35 Pin keluaran Sharp GP2D120

GP2D120 ini akan mendeteksi bacaan secara terus menerus ketika diberi daya. Keluarannya berupa tegangan analog yang sesuai dengan jarak yang diukur. Nilai tersebut terus diperbarui setiap 38.3ms $\pm$ 9.6 ms dengan jeda pembacaan berikutya maksimal 5 ms, seperti pada diagram waktu GP2D120 berikut :



Gambar 2.36 Diagram waktu pembacaan Sharp GP2D120

Sensor Sharp GP2D120 ini memiliki kemampuan membaca jarak dari minimal 4cm – maksimal 30 cm. Berikut karakteristik keluaran tegangan analog terhadap jarak dari sensor Sharp GP2D120 ini :



Gambar 2.37 Keluaran tegangan analog terhadap jarak

2.7 Baterai *Lithium Polymer* (LiPo)

Baterai *Litihium Polymer* atau yang bisa disebut dengan LiPo merupakan salah satu jenis baterai yang sering digunakan dalam dunia R/C (*Radio Controlled*). Utamanya untuk R/C tipe pesawat dan helikopter, tapi sekarang sudah banyak dipakai untuk aplikasi robot-robot karya mahasiswa di Indonesia, khususnya D.I. Yogyakarta.

Ada tiga kelebihan utama yang ditawarkan oleh baterai berjenis LiPo dari baterai jenis lain seperti NiCad atau NiMH, yaitu :

- Baterai LiPo memiliki bobot yang ringan dan tersedia dalam berbagai bentuk dan ukuran.
- Baterai LiPo memiliki kapasitas penyimpanan energi listrik yang besar.
- Baterai LiPo memiliki tingkat *discharge rate* energi yang tinggi, dimana hal ini sangat berguna sekali dalam bidang R/C.

Selain keuntungan yang dimilikinya, baterai jenis ini kuga memiliki beberapa kelemahan yaitu :

- Harga baterai LiPo masih tergolong mahal jika dibandingkan dengan baterai jenis NiCad dan NiMH.
- Performa yang tinggi dari baterai LiPo harus dibayar dengan umur yang lebih pendek. Usia baterai LiPo sekitar 300-400 kali siklus pengisian ulang. Sesuai perlakuan yang diberikan pada baterai.
- Baterai LiPo menggunakan bahan elektrolit yang mudah terbakar (Alasan keamanan).

- Baterai LiPo membutuhkan penanganan khusus agar dapat bertahan lama. *Charging, discharging,* maupun penyimpanan dapat mempengaruhi usia dari baterai jenis ini.

Baik dunia R/C maupun perkembangan robotika di Indonesia pada saat ini telah banyak didominasi oleh baterai jenis LiPo daripada Li-Ion. Kedua baterai ini pada dasarnya dibuat menggunakan bahan kimia yang sama dan membutuhkan perhatian yang sama. Perbedaannya adalah pada pemaketan sel (*cell*) dan tipe elektronik yang digunakan. Berikut penjelasan lebih lanjut mengenai baterai Li-Ion dan baterai LiPo :

1. Li-Ion

   Baterai ini menggunakan cairan organik sebagai elektrolit. Elektrolit ini bertanggung jawab terhadap pertukaran ion antar elektroda (*anoda* dan *katoda*) sama seperti yang berlaku pada baterai biasa. Pelarut organik ini bersifat sangat mudah terbakar dan alasan mengapa baterai jenis ini sangat sensitif adalah karena selain dapat terbakar beterai ini juga dapat meledak jika tidak diperlakukan dengan benar. Baterai jenis Li-Ion biasanya dibungkus oleh metal yang keras (sekali lagi sama seperti baterai biasa) yang mengakibatkan bertambahnya bobot dan hanya tersedia dalam bentuk yang terbatas.

Gambar 2.38 Baterai Li-Ion

2. LiPo

Baterai LiPo tidak menggunakan cairan sebagai elektrolit melainkan menggunakan elektrolit polimer kering yang berbentuk seperti lapisan plastik film tipis. Lapisan film ini disusun berlapis-lapis diantara *anoda* dan *katoda* yang mengakibatkan pertukaran ion. Dengan metode ini baterai LiPo dapat dibuat dalam berbagai bentuk dan ukuran. Diluar dari kelebihan arsitektur baterai LiPo, terdapat juga kekurangan yaitu lemahnya aliran pertukaran ion yang terjadi melalui elektrolit polimer kering. Hal ini menyebabkan penurunan pada *charging* dan *discharging rate*. Masalah ini sebenarnya bisa diatasi dengan memanaskan baterai sehingga menyebabkan pertukaran ion menjadi lebih cepat, namun metode ini dianggap tidak dapat diaplikasikan pada keadaan sehari-hari. Seandainya para ilmuwan dapat memecahkan masalah ini maka resiko keamanan pada baterai jenis *litihum* akan sangat berkurang.

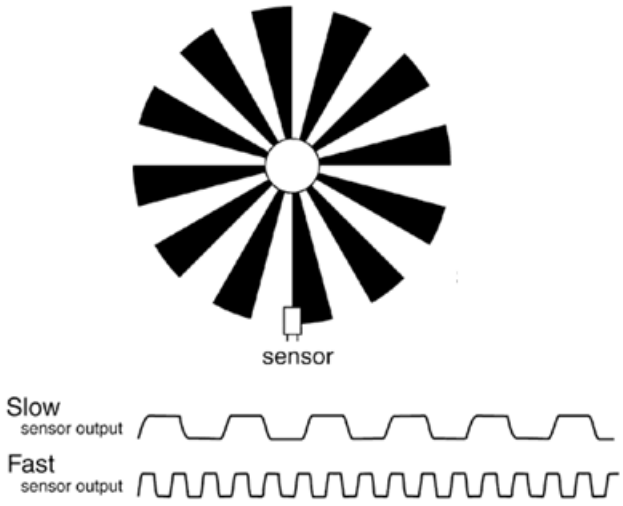
Bisa dikatakan hampir semua baterai jenis LiPo yang beredar diluar sekarang ini (mulai januari, 2011) sebenarnya adalah jenis *Hybrid Litihum Polymer*. Nama yang biasa digunakan untuk baterai ini adalah *Lithium-Ion*

Polymer, namun dunia lebih sering menyebutnya *Lithium Polymer* saja. Padahal baterai jenis ini tidak sepenuhnya menggunakan elektrolit kering seperti yang dijelaskan pada penjelasan tentang baterai LiPo sebelumnya. Dengan menggunakan elektrolit tipe gel terhadap polimer, pertukaran ion yang terjadi meningkat pesat. Elektrolit gel menyebabkan berkurangnya tingkat kebocoran, namun tetap masih mudah terbakar. Baterai jenis ini tidak terlalu berbahaya jika dibandingkan Li-Ion, namun tetap dapat memicu ledakan apabila tidal diperlakukan dengan benar, seperti terbakar api, *overcharge*, hubung singkat, dan lain-lain.

Gambar 2.39 Baterai *Hybrid Lithium Polymer*

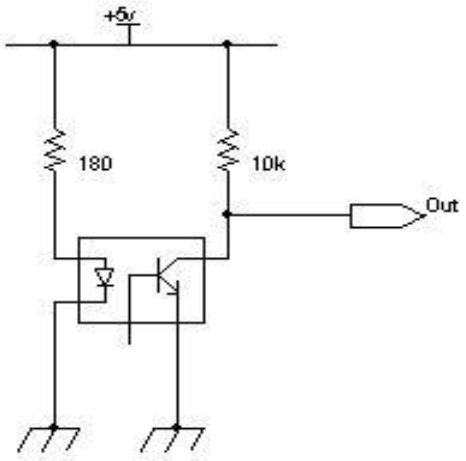
2.8 *Rotary Encoder*

Rotary encoder umumnya menggunakan sensor optik untuk menghasilkan serial pulsa yang dapat diartikan menjadi gerakan, posisi, dan arah. Sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh *Rotary encoder* untuk diteruskan oleh rangkaian kendali. *Rotary encoder* umumnya digunakan pada pengendalian robot, *motor drive*, dan sebagainya.



Gambar 2.40 *Encoder* dengan 1 sensor

Rangkaian penghasil pulsa yang digunakan umumnya memiliki keluaran yang berubah dari +5V menjadi 0.5V ketika cahaya dipantulkan oleh piringan dan ketika diteruskan ke *phototransistor*. Karena alat ini umumnya bekerja dekat dengan motor DC maka banyak *noise* yang timbul sehingga biasanya keluaran akan dimasukkan ke *low-pass filter* dahulu.



Gambar 2.41 Rangkaian penghasil pulsa pada *Rotary encoder*

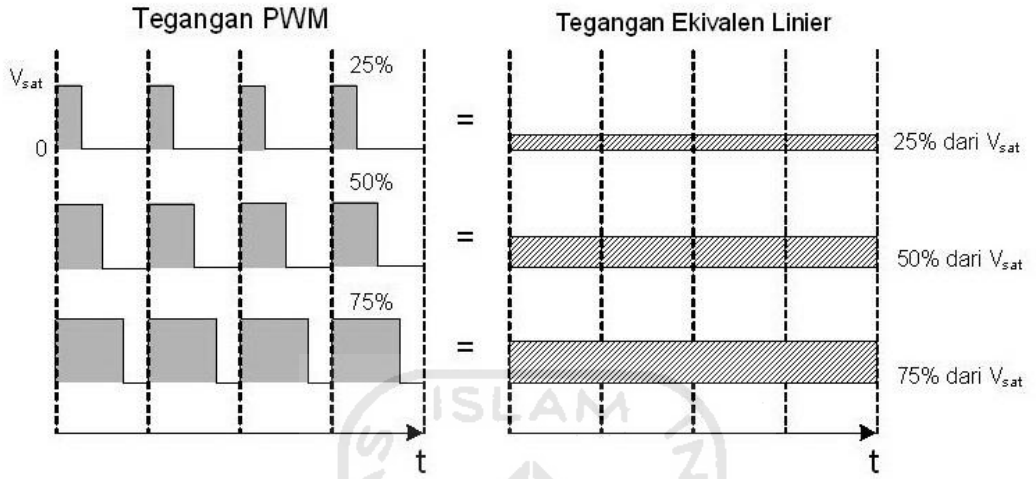
2.9 Motor DC (*Direct Current*)

Motor DC adalah suatu motor penggerak yang dikendalikan dengan arus searah (*Direct Current*). Motor DC merupakan motor yang paling banyak digunakan untuk *mobile* robot, khususnya untuk memutar roda. Motor DC dapat berputar searah jarum jam (CW) maupun berlawanan arah jarum jam (CCW). Selain itu kecepatan putarannya dapat diatur menggunakan PWM. Berikut gambar dari motor DC yang penulis gunakan dalam penelitian ini :

Gambar 2.42 Motor DC dengan *gearbox* langsung

Pulse Widht Modulation (PWM) merupakan suatu metode canggih yang digunakan untuk mengatur kecepatan motor. Pada dasarnya, PWM adalah menyalakan (ON) dann mematikan (OFF) motor DC dengan cepat. Kuncinya adalah mengatur berapa lama waktu ON dan waktu OFF. PWM umumnya digunakan untuk mereduksi daya yang melewati motor DC sehingga menghindarkan motor mengkonsumsi daya berlebih.

Gambar 2.43 berikut merupakan gambaran singkat dari teknik PWM yang sering diterapkan untuk pengendalian motor DC :



Gambar 2.43 Ilustrasi teknik PWM pada pengendalian motor DC

Lebar pulsa PWM dinyatakan dalam *Duty cycle*. Misalkan *duty cycle* 10% berarti lebar pulsa adalah 1/10 dari satu perioda penuh. Semakin kecil pulsa PWM maka tegangan ekivalen liniernya semakin kecil.

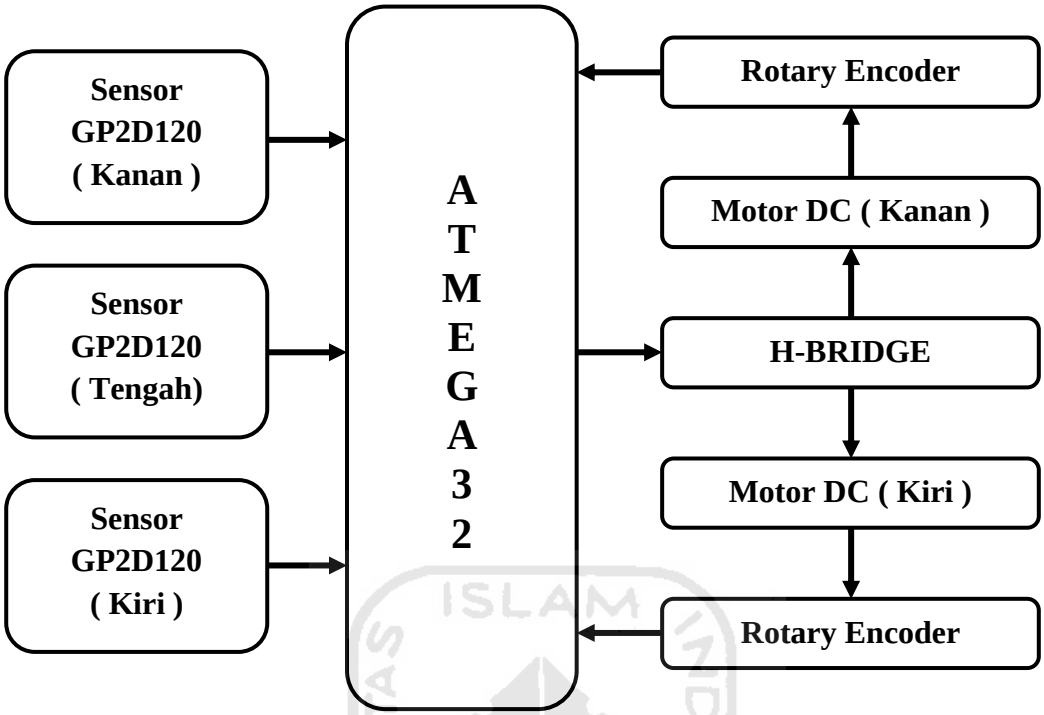
BAB III

PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai perancangan seluruh sistem robot yang meliputi bagian *hardware* dan *software*. Perancangan *hardware* meliputi perancangan desain robot dan pembuatan mekaniknya, juga perancangan dan pembuatan rangkaian elektronik yang diperlukan robot. Perancangan *software* meliputi perancangan *flowchart* yang mendukung berjalannya algoritma *flood-fill* yang diinginkan pada robot.

3.1 Blok Diagram

Perancangan secara umum dari sebuah robot micromouse yang dapat bergerak bebas didalam sebuah labirin terdiri dari sebuah sistem minimum AVR ATMEGA32 sebagai utama dengan beberapa masukan, yaitu 3 buah sensor Infrared SHARP GP2D120 dan 2 buah sensor cahaya untuk menghitung lagkah roda yang dipasang *encoder*. Sedangkan pada keluarannya terdapat 2 buah motor DC *gearbox*. Juga terdapat sebuah LCD 2x16 karakter sebagai penampil yang dapat memudahkan penulis dalam melihat jalannya program. Untuk lebih jelasnya, dapat dilihat pada diagram blok yang terdapat pada Gambar 3.1.



Gambar 3.1 Blok diagram Robot Micromouse

Dari gambar 3.1 diatas dapat dilihat bahwa setiap blok mempunyai fungsi masing-masing yang secara terpusat dikendalikan oleh mikrokontroler ATMEGA32. Adapun fungsi secara umum dari masing-masing bagian blok diagram dapat dijelaskan sebagai berikut :

a. Mikrokontroler ATMEGA32 berfungsi sebagai otak atau pengendali utama yang menerima sinyal masukan dari sensor SHARP GP2D120 untuk mengetahui halangan dan jarak robot terhadap dinding, juga informasi dari sensor pada *encoder* untuk mengetahui langkah roda sehingga dapat diketahui sudah berapa jauh robot bergerak atau berjalan. Selain menerima masukan, mikrokontroler ATMEGA32 ini juga berfungsi

untuk menggerakkan motor DC sesuai informasi yang didapat dari sensor yang telah diproses terlebih dahulu.

b. Sensor *Infrared* SHARP GP2D120 berfungsi sebagai pendeteksi jarak dinding labirin terhadap robot. Pada robot ini digunakan 3 buah sensor SHARP GP2D120 yang terletak pada sisi samping kanan, samping kiri, dan sisi depan robot. Sensor ini terhubung pada PORTA mikrokontroler karena memanfaatkan ADC yang tersedia pada ATMEGA32.

c. *Rotary Encoder* berfungsi untuk mengubah banyaknya langkah pada roda robot menjadi informasi berupa kode digital untuk diteruskan ke mikrokontroler ATMEGA32 sebagai masukan sehingga dapat diketahui sudah sejauh mana robot bergerak atau berjalan. Menggunakan sensor optik *phototransistor* sebagai penerima dan infrared LED sebagai pemancarnya untuk menghasilkan serial pulsa yang dapat diartikan menjadi gerakan, posisi, dan arah.

d. *H-BRIDGE* berfungsi sebagai aktuator untuk mengontrol kecepatan motor DC yang digunakan sebagai penggerak roda robot. Disebut *H-BRIDGE* karena konfigurasi/susunan dari ke-empat MOSFET-nya membentuk seperti huruf H. Nantinya MOSFET ini akan berfungsi sebagai *switching ON/OFF* tegangan dengan frekuensi yang tertentu yang akan memberikan tegangan berbentuk pulsa ke motor DC sehingga nantinya motor dapat berputar, baik searah jarum jam (*clockwise*) ataupun berlawanan arah jarum jam (*counterclockwise*).

3.2 Perancangan Hardware Robot

Robot Micromouse ini dirancang sesederhana mungkin dan dimensi robot sekecil mungkin, disesuaikan dengan tujuan agar memudahkan robot dalam bergerak menelusuri lorong (labirin) yang berukuran tidak lebih dari 16,8 cm x 16,8 cm untuk mencari sel labirin tujuan (*GOAL*).

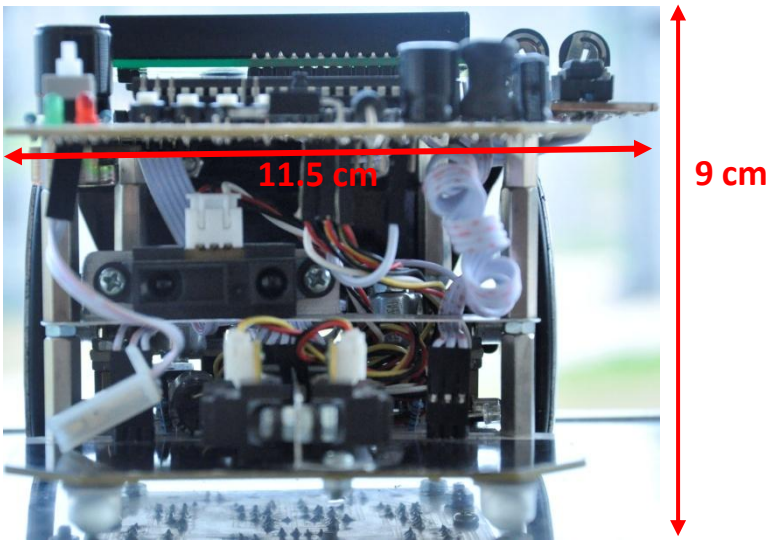
Perancangan robot ini didukung oleh rangkaian-rangkaian elektronik yang membantu kinerja mikrokontroler sebagai pengendali utama untuk mengendalikan seluruh sistem yang ada pada robot. Rangkaian elektronik dirancang sesederhana mungkin sehingga sekaligus menjadi badan robot.

3.2.1 Perancangan Mekanik Robot

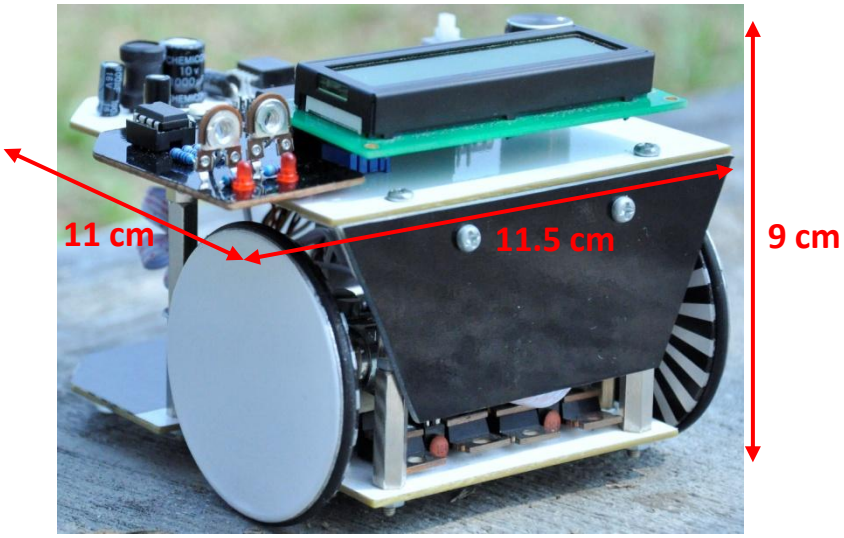
Perancangan mekanik robot micromouse ini meliputi beberapa bagian, yaitu perancangan desain robot dengan ketentuan dimensi yang panjang dan lebarnya tidak boleh melebihi ukuran 16.8cm x 16.8cm atau ukuran 1 sel labirin.

Adapun dimensi dari robot micromouse ini adalah sebagai berikut :

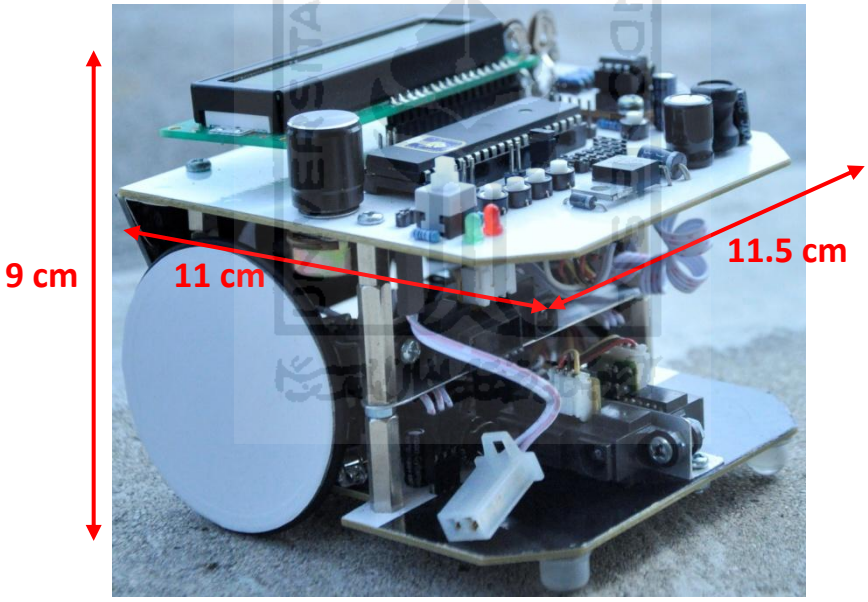
- Panjang : 11 cm x Lebar : 11.5 cm x Tinggi : 9 cm



Gambar 3.2 Robot Micromouse (depan)



Gambar 3.3 Robot Micromouse (belakang)



Gambar 3.4 Robot Micromouse (samping)

3.2.2 Perancangan Elektronik Robot

Dalam perancangannya, robot micromouse ini dibangun dari kumpulan beberapa rangkaian elektronik yang terbagi dalam beberapa bagian, yaitu sistem minimum mikrokontroler ATMEGA32 sebagai pusat pengendali dari semua sistem yang ada terdapat pada robot ini, rangkaian pencatu daya (*power supply*) yang berfungsi mengatur kebutuhan daya dari semua rangkaian yang ada pada robot ini, rangkaian penampil LCD yang berfungsi menampilkan apa saja yang dibutuhkan oleh seorang *programmer* saat program pada mikrokontroler sedang berjalan, rangkaian tombol untuk memilih menu yang telah ditampilkan pada LCD guna mengatur apa saja yang diperlukan dalam melakukan *setting* dan kalibrasi pada robot, rangkaian H-BRIDGE motor sebagai aktuator untuk mengendalikan motor, dan rangkaian penghasil pulsa pada *encoder* roda untuk mengetahui langkah roda pada robot micromouse ini.

A. Sistem Minimum Mikrokontroler ATMEGA32

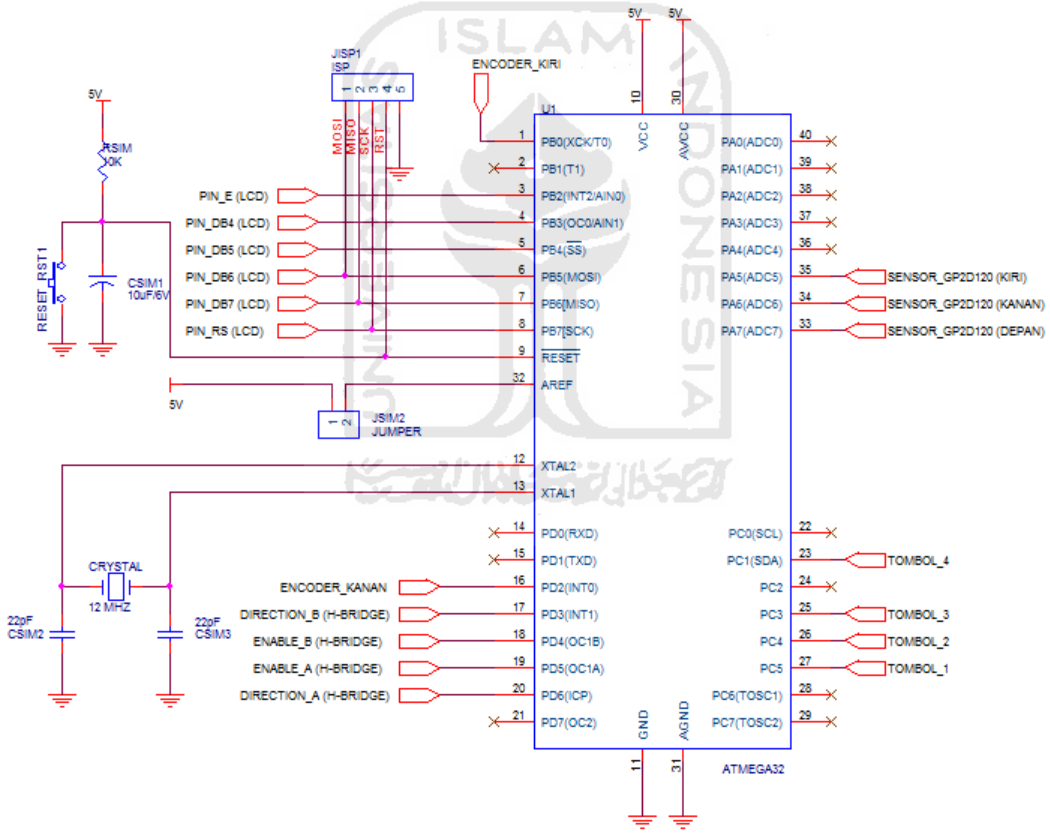
Sistem minimum mikrokontroler ini merupakan pusat pengendali dari semua rangkaian sensor dan aktuator yang terdapat pada robot micromouse ini, karena dari sinilah semuanya di kendalikan sesuai kondisi dan kebutuhan masing-masing. Mikrokontroler yang digunakan adalah ATMEGA32 dari keluarga AVR. Pada ATMEGA32 ini, tersedia 4 *PORT I/O (Input/Output)* yang masing-masingnya terdiri dari 8 pin. Adapun 4 *PORT I/O (Input/Output)* itu terdiri dari PORT.A yang didalamnya tersedia fitur ADC (*Analog to Digital Converter*), PORT.B, PORT.C, dan PORT.D.

PORT.A yang didalamnya terdapat fitur ADC, terhubung dengan 3 sensor jarak SHARP GP2D120 yang keluarannya berupa tegangan analog. 3 sensor ini antara lain sensor kiri yang terhubung dengan PINA.5, sensor kanan yang terhubung dengan PINA.6, dan sensor depan yang terhubung dengan PINA.7. Dipilihnya PORT.A ini sebagai masukan untuk keluaran dari sensor jarak SHARP GP2D120 dikarenakan keluaran SHARP GP2D120 yang berupa data analog sehingga butuh fitur ADC untuk dikonversi ke data digital agar dapat dibaca oleh mikrokontroler, dimana besarnya data analog ini mempresentasikan jarak yang diterima *receiver* sensor ini terhadap cahaya yang dipantulkan oleh *transmitter* SHARP GP2D120 terhadap dinding.

Keluaran dari sensor penghasil pulsa pada *encoder* roda kiri terhubung ke PINB.0 yang didalamnya terdapat fitur *Timer0/Counter0* sehingga pulsa yang masuk ke PINB.0 dapat dicacah dengan cukup akurat. Selain sensor penghasil pulsa pada *encoder* roda kiri, PORT.B juga menjadi keluaran untuk rangkaian penampil LCD yaitu pada PINB.2 – PINB.7. Tidak ada alasan khusus dalam pemilihan PORTB ini sebagai keluaran penampil LCD karena tidak diperlukan fitur khusus untuk menampilkan tampilan pada layar LCD ini.

PORTC digunakan sebagai masukan untuk keluaran dari 4 rangkaian tombol yang bekerja aktif *low* (aktif ketika PIN mikro mikro terhubung ke *Ground*). Tombol 4 terhubung ke PINC.1 dan tombol 1 - tombol 3 terhubung ke PINC.3 - PINC.1. Pada PORT.D terdapat keluaran dari sensor penghasil pulsa pada *encoder* roda kanan yang terhubung dengan PIND.2, karena pada PIND.2 terdapat fitur eksternal interupsi 0 yang dapat dimanfaatkan sebagai pencacah, ini

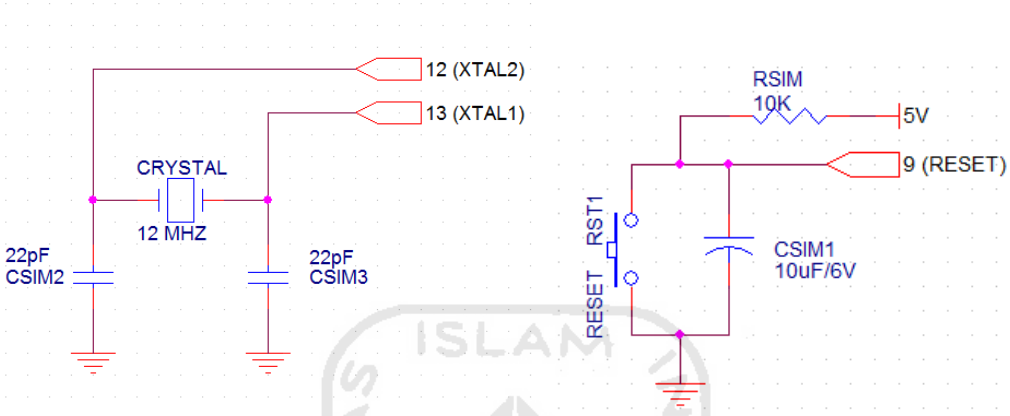
merupakan salah satu alternatif yang di pilih karena fitur *Timer0/Counter0 dan Timer1/Counter1* telah digunakan. Untuk PIND.3 - PIND.6 digunakan sebagai keluaran untuk masukan dari rangkaian *H-BRIDGE* guna mengatur kecepatan dan arah pada motor kanan dan motor kiri pada robot, karena pada PIND.4 dan PIND.5 terdapat fitur OCR1A dan OCR1B yang berfungsi sebagai keluaran PWM (*Pulse Widht Modulation*) yang digunakan untuk mengatur kecepatan kedua motor DC pada robot micromouse ini.



Gambar 3.5 Sistem minimum mikrokontroler ATMEGA32

Rangkaian sistem minimum memerlukan osilator untuk membangkitkan frekuensi. Frekuensi tersebut digunakan untuk menjalankan mikrokontroler ATMEGA32 ini untuk mengeksekusi perintah-perintah yang telah dituliskan

kedalam memori program. Rangkaian osilator pada sistem minimum ini menggunakan *Crystal Oscillator* sebesar 12 MHz dengan 2 kapasitor non-polar yang masing-masing besarnya 22 pF.



Gambar 3.6 Rangkaian osilator dengan *Crystal* 12 MHz (kiri) dan rangkaian RESET (kanan)

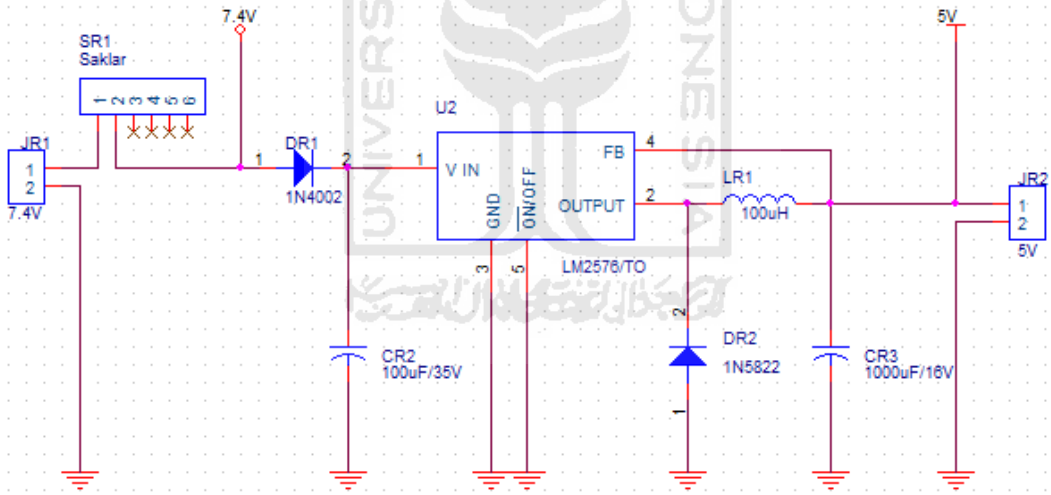
Pada mikrokontroler ATMEGA32 ini, terdapat fitur RESET yang berada pada PIN 9 yang bekerja ketika diberi tegangan 0V (*Active low*). Karena bekerja aktif *low*, maka PIN.9 mikrokontroler ATMEGA32 pada rangkaian sistem minimum dihubungkan dengan keluaran dari rangkaian tombol reset yang terdiri dari sebuah saklar *ON/OFF*, sebuah kapasitor 10uF/6V, masukan tegangan 5V yang diberi resistor 10K ohm.

B. Rangkaian Pencatu Daya (*Power Supply*)

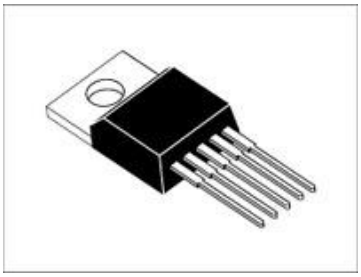
Untuk dapat menjalankan semua sistem yang terdapat pada robot micromouse ini, diperlukan catu daya yang berasal dari baterai. Baterai yang digunakan merupakan jenis baterai LiPo yang mempunyai tengangan 7.4V DC. Rangkaian pencatu daya ini berfungsi menyediakan *supply* tegangan 7.4V dan 5V karena ada

rangkaian yang memerlukan masukan tegangan 7.4V dan ada yang membutuhkan masukan tegangan 5V.

Regulator LM2576 - 5.0 digunakan untuk menghasilkan tegangan 5V pada robot micromouse ini. Selain mampu dilewati arus hingga 3A (LM7805 hanya mampu dilalui arus 1A), keuntungan memakai regulator LM2576 - 5.0 ini dibanding LM7805 yang umumnya dipakai sebagai regulator 5V juga adalah LM2576 - 5.0 ini merupakan regulator bertipe *switching* sehingga tidak menimbulkan panas yang tinggi seperti jika memakai regulator LM7805 yang bertipe *linear*. Berikut gambar rangkaian pencatu daya yang menggunakan LM2576 - 5.0 untuk menghasilkan tegangan 5V pada robot micromouse ini:



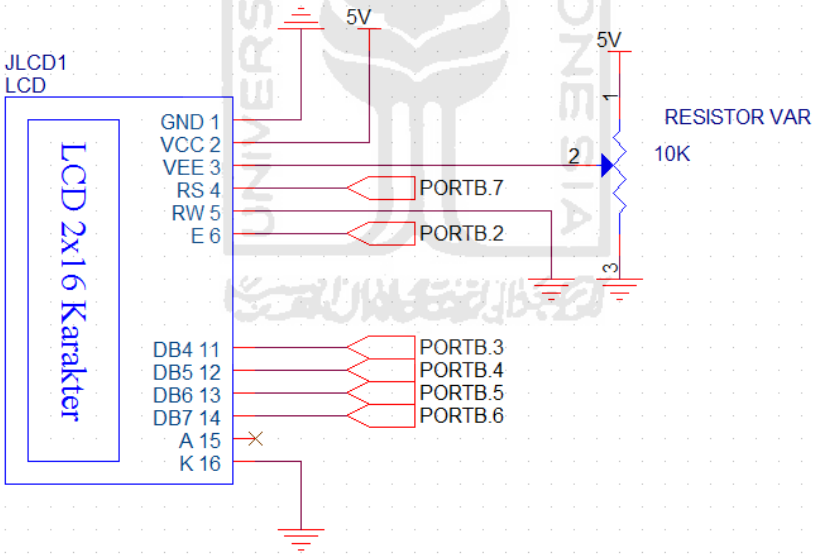
Gambar 3.7 Rangkaian *Power Supply*



Gambar 3.8 Regulator LM2576 – 5.0

C. Rangkaian Penampil LCD (*Liquid Crystal Display*)

LCD (*Liquid Crystal Display*) merupakan suatu penampil dari bahan cairan kristal yang pengoperasiannya menggunakan sistem *dot matriks*. LCD dibutuhkan pada robot micromouse ini untuk menampilkan apa saja yang dibutuhkan oleh seorang *programmer* saat program pada mikrokontroler sedang berjalan, dan dalam pengaturan nilai-nilai yang diperlukan pada saat kalibrasi dilakukan. Pada Robot micromouse ini, LCD digunakan dalam mode 4 bit dengan hanya menggunakan DB4 - DB7 sebagai jalur data. LCD yang digunakan merupakan produk SHENZEN TOPWAY TECHNOLOGY CO.,Ltd dengan seri TOPWAY LMB162ABC.

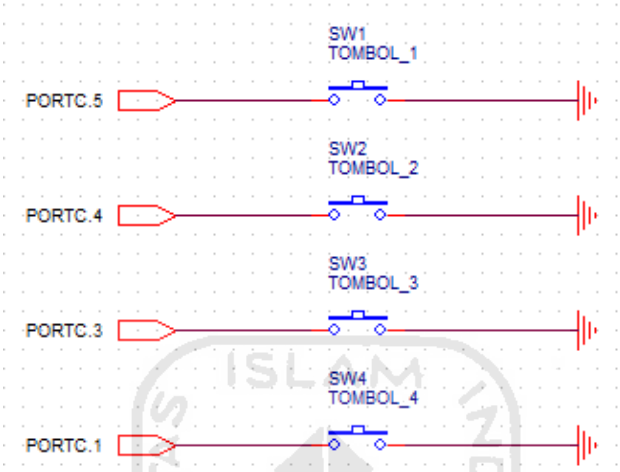


Gambar 3.9 Rangkaian LCD 2x16 Karakter

D. Rangkaian Tombol Menu

Rangkaian tombol menu itu berfungsi untuk memilih menu yang telah ditampilkan pada LCD guna mengatur apa saja yang diperlukan dalam melakukan *setting* dan kalibrasi pada robot. Tombol yang digunakan merupakan tombol

push-buttton. Digunakan 4 tombol pada robot micromouse ini yang terhubung ke PINC.1 dan PINC.3 – PINC.5 dengan mode aktif *low*, aktif jika diberi tegangan 0V (terhubung ke *GROUND*).

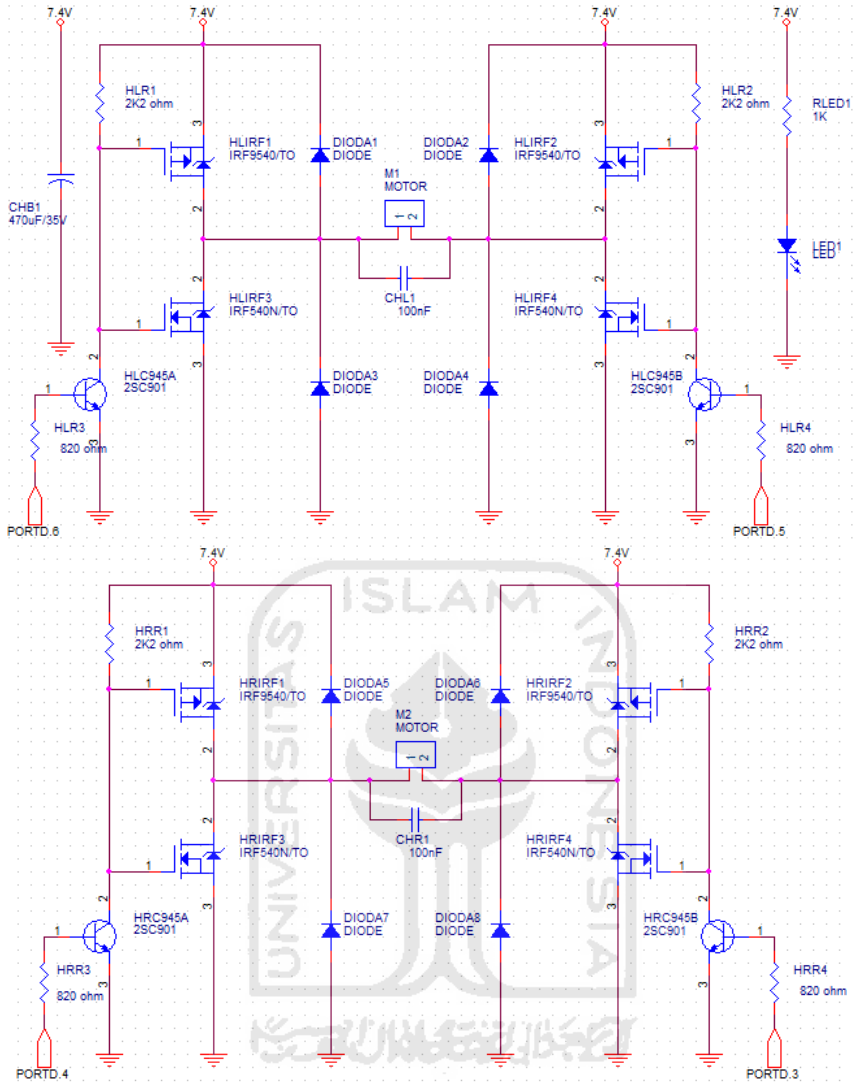


Gambar 3.10 Rangkaian tombol pada robot

E. Rangkaian *H-BRIDGE* Motor

Rangkaian *driver* motor DC ini disebut dengan *H-BRIDGE* dikarenakan konfigurasi/susunan dari ke-empat MOSFET-nya membentuk seperti huruf H. Nantinya MOSFET ini akan berfungsi sebagai *switching ON/OFF* tegangan dengan frekuensi yang tertentu yang akan memberikan tegangan berbentuk pulsa ke motor DC sehingga nantinya motor dapat berputar, baik searah jarum jam (*clockwise*) ataupun berlawanan arah jarum jam (*counterclockwise*).

Untuk mengatur 2 buah motor DC, maka diperlukan 2 buah *H-BRIDGE* sebagai aktuatornya. Sepasang *H-BRIDGE* ini memiliki 4 PIN keluaran yang masing-masingnya sebagai masukan pada PIND.3 - PIND.4 untuk motor kanan dan PIND.5 - PIND.6 untuk motor kiri.



Gambar 3.11 Rangkaian *H-BRIDGE* motor robot

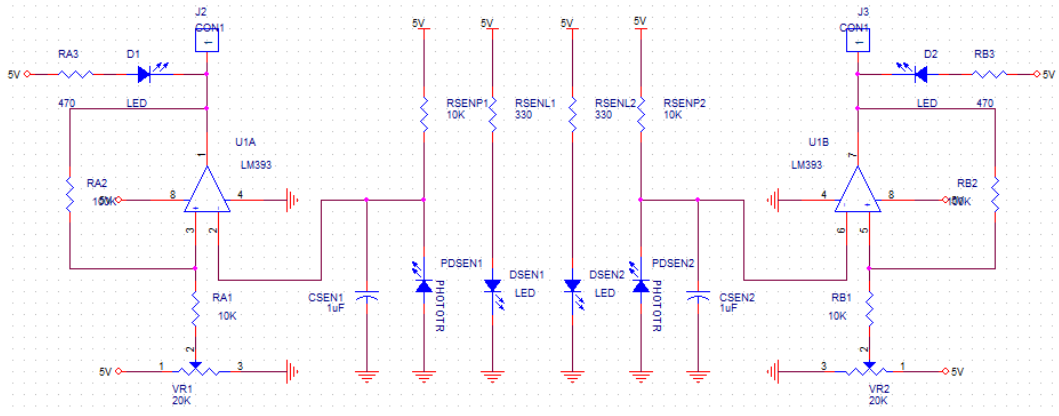
Tabel 3.1 Tabel *logic control H-BRIDGE* pada motor robot

MOTOR	PIND.3	PIND.4	OUTPUT
KANAN	0	0	Berhenti
	0	1	Maju
	1	0	Mundur
	1	1	Berhenti

MOTOR	PIND.5	PIND.6	OUTPUT
KIRI	0	0	Berhenti
	0	1	Maju
	1	0	Mundur
	1	1	Berhenti

F. Rangkaian Penghasil Pulsa pada Encoder Roda

Rangkaian penghasil pulsa yang digunakan pada robot ini, 1 rodanya terdiri dari sebuah *infrared* sebagai pemancar, sebuah *phototransistor* sebagai penerima dan sebuah komparator dengan *hysterisis* sebagai penguat keluarannya . Pantulan dari cahaya *infrared* pada *encoder* roda akan diterima oleh sebuah *phototransistor* yang memiliki keluaran yang berubah dari sekitar 0.5V (terkena warna hitam / tidak ada pantulan cahaya) menjadi sekitar 3V (terkena warna putih / ada pantulan cahaya). Agar lebih yakin bahwa keluaran dari sensor penghasil pulsa ini sudah bisa dibaca oleh mikrokontroler (5V = Berlogika 1, 0V = Berlogika 0) maka keluaran dari sensor penghasil pulsa ini akan dikuatkan dengan komparator dengan *hysterisis* agar nantinya dihasilkan tegangan keluaran 0V saat berlogika 0 dan 5V saat berlogika 1 sehingga keluaran dari sensor penghasil pulsa ini akan berupa kode digital. Keluaran dari komparator dengan *hysterisis* ini diteruskan ke PINB.0 untuk masukan data *encoder* roda kiri dan PIND.2 pada masukan data *encoder* roda kanan.



Gambar 3.12 Rangkaian penghasil pulsa pada Encoder Roda

3.2.3 Perancangan Perangkat Lunak (*Software*) Robot

Perancangan perangkat lunak meliputi rancangan dari program algoritma *flood-fill* hingga pembuatan diagram alir (*flowchart*) dari setiap proses yang terjadi pada robot,misalnya:

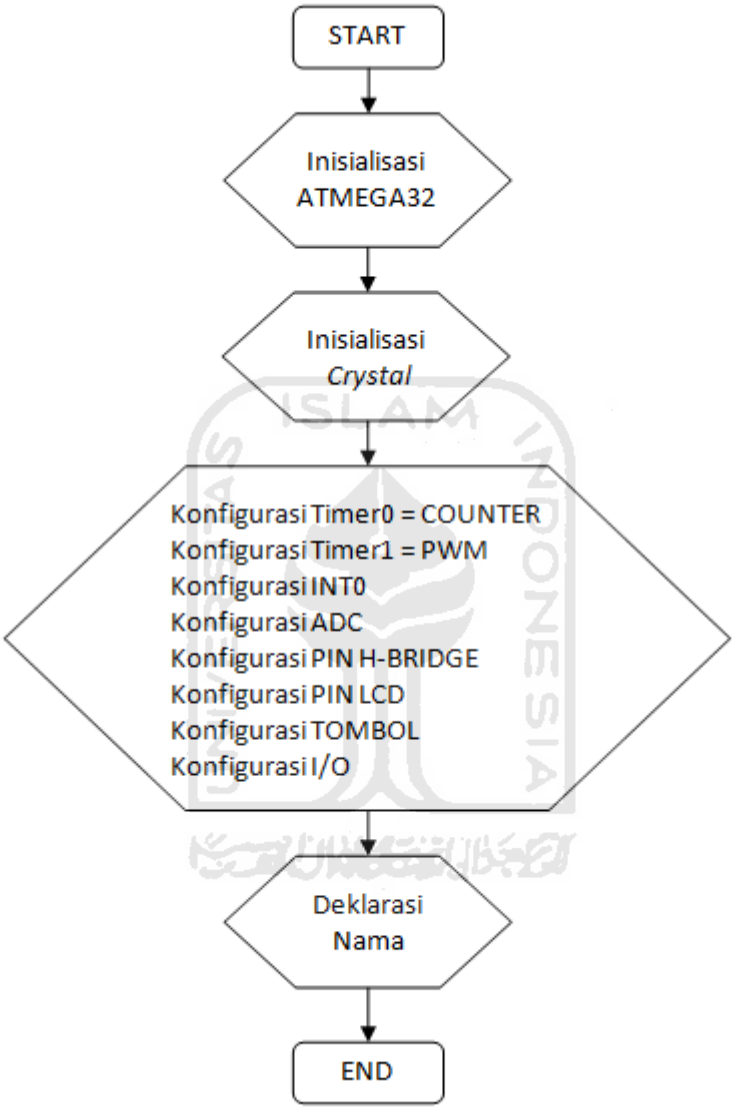
- Proses pemilihan lapangan mana yang akan dieksplorasi oleh robot.
- Proses pemilihan jalur terpendek oleh algoritma *flood-fill* pada mode eksplorasi.
- Proses pemilihan jalan dari *GOAL* menuju *HOME*.
- Proses pemilihan jalan pada mode hapalan dari *HOME* menuju *GOAL*.

Compiler yang digunakan adalah BASCOM (*Basic Compiler*) AVR versi 1.11.9.0 dengan bahasa BASIC. Walaupun mempunyai kelemahan pada pemakaian memori yang cukup besar, alasan dipilihnya *compiler* BASCOM ini karena bahasanya yang lebih sederhana dan mudah dipahami.

Dalam menulis program, ada beberapa hal yang harus dideklarasikan terlebih dahulu, diantaranya:

- Jenis dari mikrokontroler yang digunakan.
- *Crystal oscillator* yang digunakan.
- Konfigurasi fitur-fitur pada mikrokontroler jika digunakan, seperti *TIMER*, ADC, PWM, LCD, dan lain-lain.
- Deklarasi dari variabel yang digunakan.
- Pengaturan dari PORT mikrokontroler, apakah akan digunakan sebagai masukan, keluaran, atau fungsi khusus.
- Penginisialisasian nama dan konstanta yang digunakan

Berikut *flowchart* dari deklarasi variabel dan konfigurasi awal dari robot micromouse ini:



Gambar 3.13 Diagram alir proses inisialisasi mikrokontroler

Proses inisialisasi awal mikrokontroler pada gambar 3.13, dituliskan pada BASCOM AVR pada program berikut ini:

```
'=====================================================
$regfile = "m32def.dat"
$crystal = 12000000
'=====================================================
'================= Konfigurasi LCD ===================
Config Lcdpin=Pin,Rs=Portb.7,E=Portb.2,Db4=Portb.3
Config Lcdpin = Pin,Db5=Portb.4,Db6=Portb.5,Db7=Portb.6
Config Lcd = 16 * 2
Cursor Off
'=====================================================
'================= Konfigurasi PWM ===================
Config Timer1 = Pwm , Pwm = 8 , Prescale = 64 , Compare
A Pwm = Clear Up , Compare B Pwm = Clear Up
'=====================================================
'============== Konfigurasi interupsi ================
Config Int0 = Falling
On Int0 Int_ext0
Enable Interrupts
Enable Int0
Ddrd.2 = 0
Portd.2 = 1
'=====================================================
'========= Konfigurasi timer0 sebagai counter =========
Config Timer0 = Counter , Edge = Falling
Ddrb.0 = 0
Portb.0 = 1
Langkah_kiri Alias Tcnt0
'=====================================================
'================ Konfigurasi H-Bridge ===============
Dir_kanan Alias Portd.3
Dir_kiri Alias Portd.6
'----------------------
Config Portd.3 = Output
Config Portd.6 = Output
'----------------------
Set Dir_kanan
Set Dir_kiri
'=====================================================
'============ Konfigurasi ADC untuk Sensor ============
Config Adc = Single , Prescaler = 8 , Reference = Avcc
Start Adc
'=====================================================
```

```
'============== Konfigurasi Tombol ===================
Tombol1 Alias Pinc.5
Tombol2 Alias Pinc.4
Tombol3 Alias Pinc.3
Tombol4 Alias Pinc.1
'----------------------
Config Pinc.5 = Input
Config Pinc.4 = Input
Config Pinc.3 = Input
Config Pinc.1 = Input
'----------------------
Set Portc.5
Set Portc.4
Set Portc.3
Set Portc.1
=====================================================
```

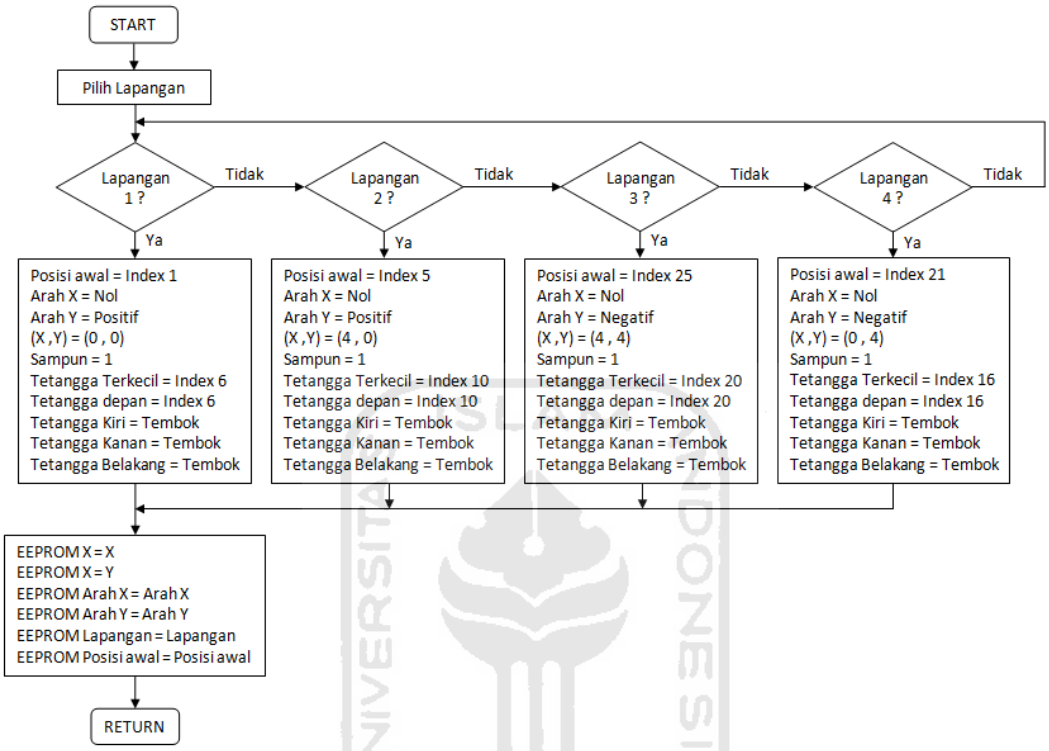
A. Perancangan Inisialisasi Pemilihan Lapangan yang akan di Eksplorasi

Sebelum mulai melakukan eksplorasi, robot terlebih dahulu harus diberitahu di lapangan mana dia akan memulai eksplorasi ini guna posisi titik koordinat mana robot akan memulai perjanan (*HOME*). Terdapat 4 pilihan lapangan pada menu yang "Pilih Lapangan" yang akan tampil di LCD jika kita memilih "Mode Eksplorasi", antara lain:

- Lapangan 1 : Memulai eksplorasi dari koordinat (0 , 0)
- Lapangan 2 : Memulai eksplorasi dari koordinat (4 , 0)
- Lapangan 3 : Memulai eksplorasi dari koordinat (4 , 4)
- Lapangan 4 : Memulai eksplorasi dari koordinat (0 , 4)

Setelah salah satu dari lapangan dipilih, data yang terdapat pada lapangan tersebt akan langsung disimpan pada memori EEPROM (*Electrically Eraseable Programmable Read Only Memory*). Untuk lebih lengkap mengenai data apa saja

yang akan dimasukkan ketika salah satu dari lapangan dipilih dan bagaimana proses pemilihannya dapat dilihat pada diagram alir berikut:

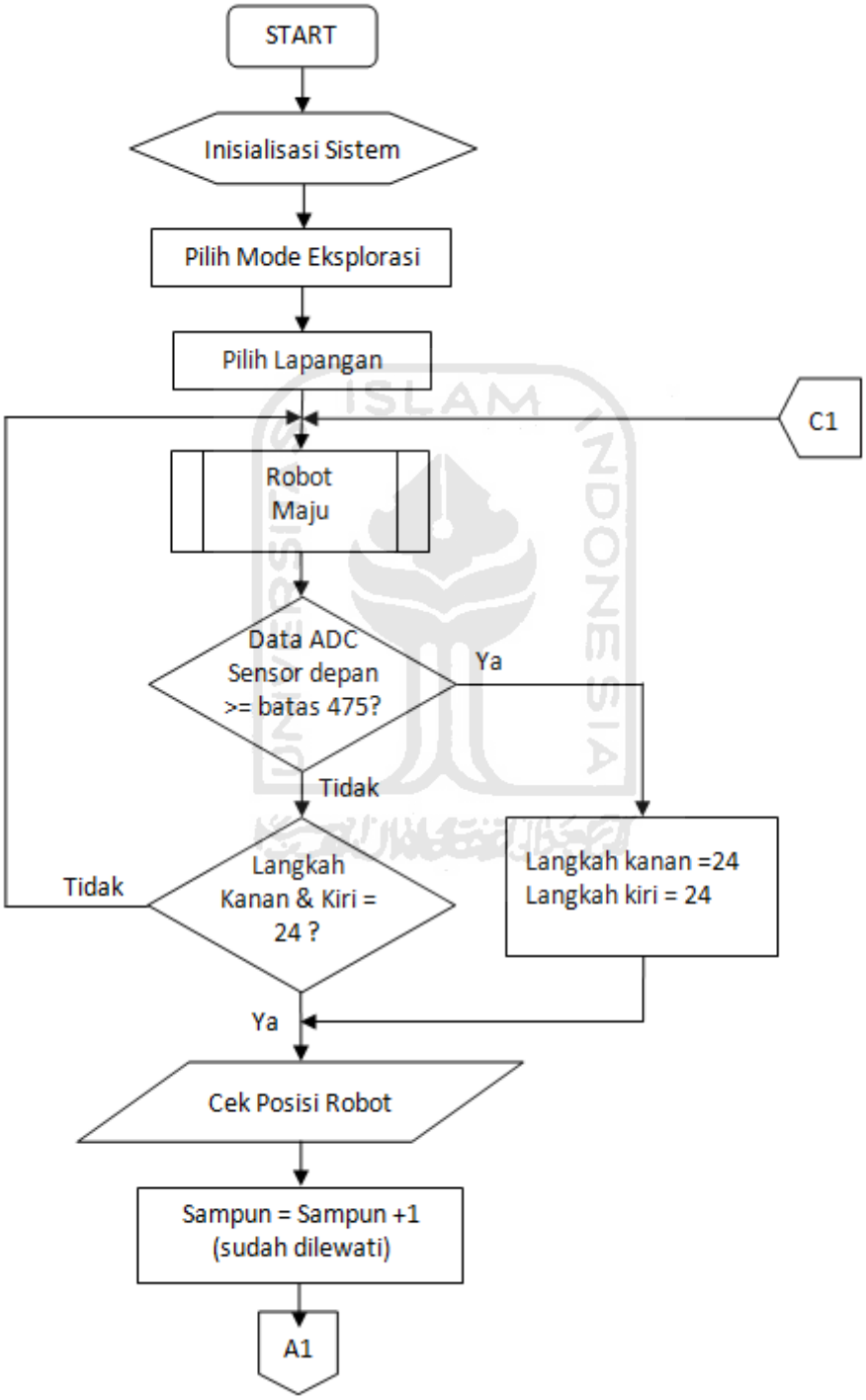


Gambar 3.14 Diagram alir proses pemilihan lapangan

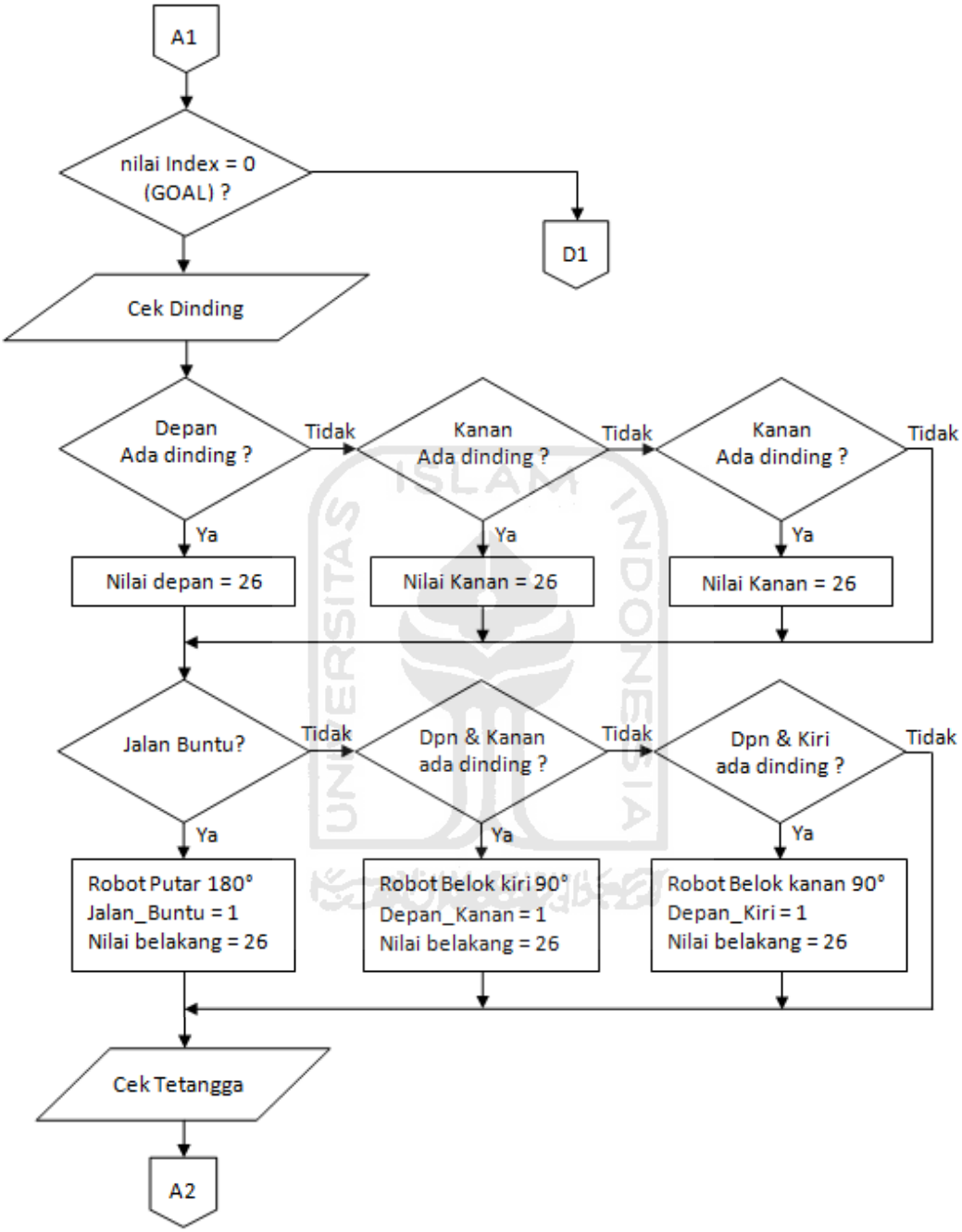
B. Perancangan Algoritma *Flood-fill* pada Pemilihan Jalur Terpendek dalam Mode Eksplorasi dari *HOME* menuju *GOAL*

Algoritma *Flood-Fill* dapat dianalogikan seperti membanjiri labirin dengan air yang banyak, terus mengalir hingga mencapai lantai tempat tujuan. Jalur yang dilalui oleh tetesan air pertama di tempat tujuan merupakan jalur terpendek untuk mencapai tempat tujuan tersebut. Melibatkan proses penomoran setiap sel dalam labirin dimana nomor-nomor ini merepresentasikan jarak setiap sel dengan sel tujuan. Sel tujuan yang ingin dicapai diberi nomor 0. *GOAL* dapat dicapai dengan

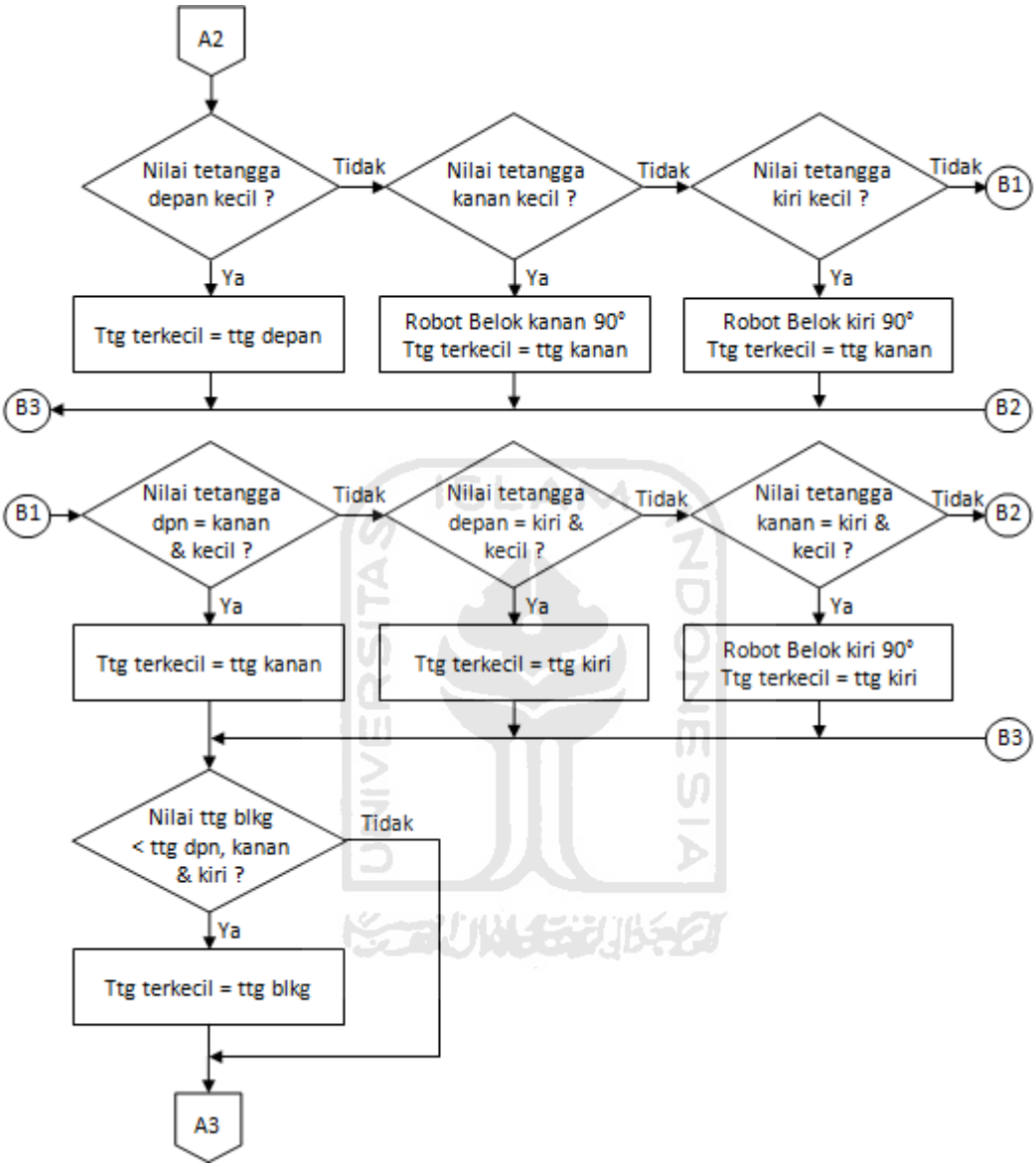
cara robot terus mengikuti sel dengan nomor terkecil. Berikut rancangan diagram alir dari algoritma *flood-fill* ini dalam mencari jalan terpendek menuju *GOAL*:



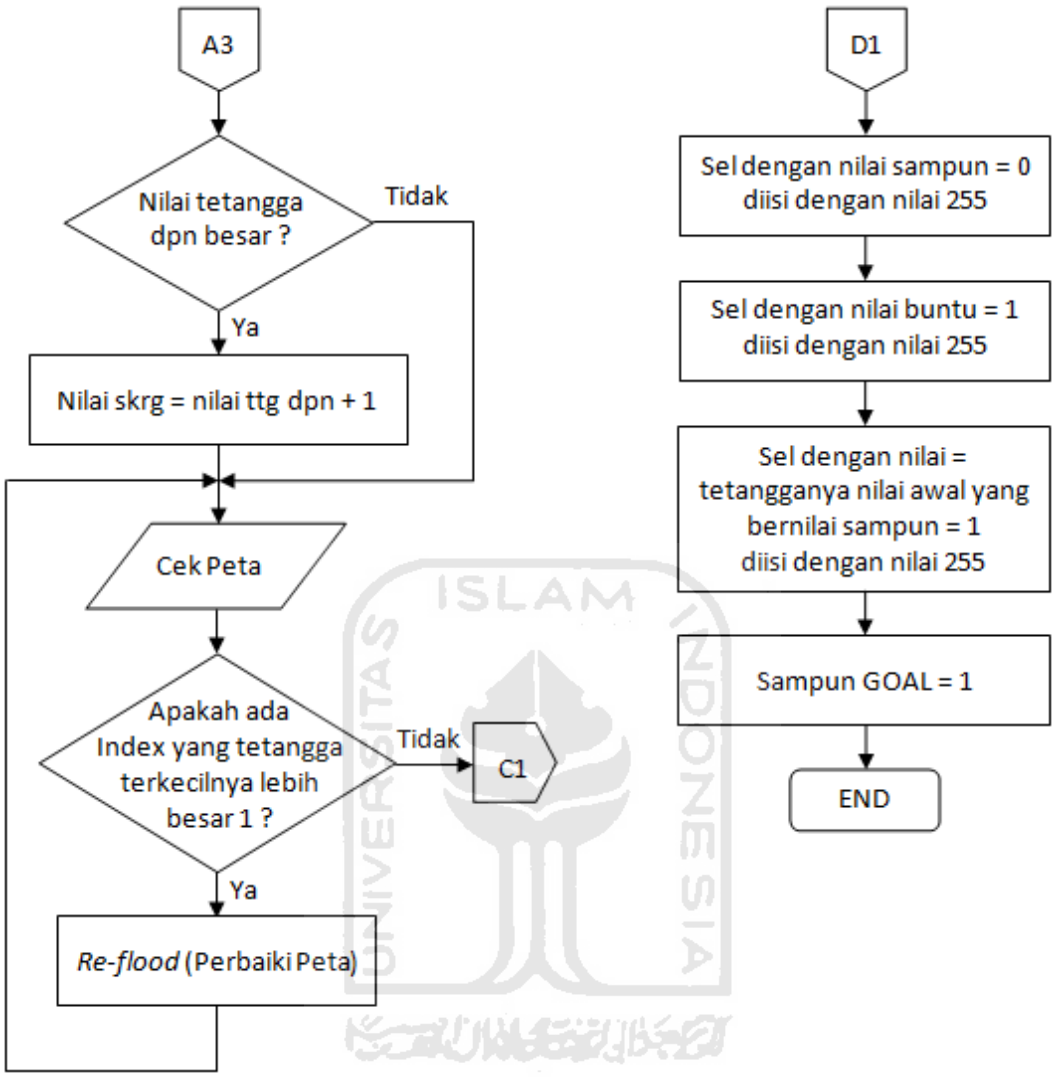
Gambar 3.15 Diagram alir proses Algoritma *flood-fill* menemukan *GOAL*



Gambar 3.16 Diagram alir proses Algoritma *flood-fill* menemukan *GOAL*



Gambar 3.17 Diagram alir proses Algoritma *flood-fill* menemukan *GOAL*



Gambar 3.18 Diagram alir proses Algoritma *flood-fill* menemukan *GOAL*

Penjelasan dari diagram alir proses Algoritma *flood-fill* menemukan *GOAL* ini dimulai pada diagram alir terdapat pada Gambar 3.15. Pada gambar 3.15, diagram alir dimulai dengan *START* yang menandakan awal mulainya program dan robot siap untuk melakukan eksplorasi, tetapi sebelumnya robot harus melakukan proses insialisasi sistem terlebih dahulu. Setelah proses inisialisasi

dilakukan sistem masuk ke pemilihan mode eksplorasi sebagai mode yang akan dijalankan robot dan lapangan mana yang akan di eksplorasi.

Setelah semua inisialisasi awal telah dilakukan, juga mode dan lapangan telah di pilih, maka robot akan bergerak maju sampai langkah pada pembacaan *encoder* roda kanan dan kiri mencapai 24 langkah atau jika data pembacaan ADC sensor depan >= batas 475 (Robot sangat dekat dengan tembok) walaupun langkah kanan dan kiri belum mencapai 24, setelah itu robot berhenti untuk melakukan cek posisi sekarang guna mengetahui di sel nomor berapa robot berada. Untuk sel yang telah di tinggalkan robot sebelumnya ditandai dengan menambah variabel sampun yang awalnya 0 dengan angka 1 sehingga sampun sel sebelumnya sekarang bernilai 1 yang menandakan sel tersebut telah di eksplorasi dan dilalui sebanyak 1 kali. Selanjutnya perjalanan dari program algoritma *flood-fill* ini berlanjut ke diagram alir yang ada pada gambar 3.16.

Setelah robot mengetahui di posisi mana dia berada sekarang, di koordinat (x,y) mana dan berapa nilai index sel tersebut, maka robot melakukan cek apakah nilai dari index tempat robot berdiri sekarang bernilai 0 atau tidak. Jika nilai dari index sekarang bernilai 0 maka posisi *GOAL* telah dicapai maka sistem akan diteruskan ke diagram alir pada gambar 3.17. Pada diagram alir ini, semua sel yang belum di eksplorasi atau mempunyai nilai sampun = 0 akan diberi nilai yang sangat besar yaitu 255 atau di blok. Selain itu sel dengan nilai buntu = 1 juga akan diberi nilai 255, begitu juga sel dengan nilai yang sama dengan tetangganya nilai awal (*HOME)* setelah semua proses blok sel sebelumnya dilakukan. Variabel

sampun *GOAL* akan diberi nilai 1 untuk menandakan bahwa *GOAL* telah berhasil dicapai.

jika nilai index dari sel sekarang bukan bernilai 0, maka robot akan melakukan cek dinding. Jika didepan ada dinding maka nilai sel depan diisi dengan nilai yang tinggi yaitu 26. Sebenarnya tidak ada keharusan untuk mengisi sel yang mempunyai dinding dengan nilai 26, angka berapapun boleh digunakan untuk menandai sel dengan dinding tersebut asalkan lebih besar dari 25 dan tidak lebih besar dari 255 karena variabel index yang digunakan dalam pemograman ini berukuran 1 byte atau sebesar 0 – 255 jika dalam desimal, maka untuk memudahkan pemograman, penulis memilih angka 26 untuk menandai dinding dan angka 255 untuk menandai sel yang diblok ketika penentuan jalur terpendek dilakukan.

Selanjutnya jika didepan tidak ada dinding maka dilanjutkan ke dinding kanan dan begitu seterusnya. Jika semua proses telah selesai dilakukan dan telah didapat dibagian mana saja dinding terdapat maka proses dilanjutkan, jika diketahui didepan ada dinding, di kanan ada dinding, dan di kiri juga terdapat dinding maka robot menemui jalan buntu dan robot akan berputar 180° lalu menandai sel sekarang dengan cara mengisi variabel Jalan_buntu = 1 juga mengisi nilai belakang sel ini dengan 26 yang menandakan setelah berputar 180° dibelakang dari robot akan terdapat dinding.

Jika diketahui didepan ada dinding, di kanan ada dinding, tetapi di kiri tidak terdapat dinding maka robot akan belok kiri 90° lalu menandai sel sekarang dengan cara mengisi variabel Depan_kanan = 1 juga mengisi nilai belakang sel ini

dengan 26 yang menandakan setelah belok kiri 90° dibelakang dari robot akan terdapat dinding.

Jika diketahui didepan ada dinding, di kiri ada dinding, tetapi di kanan tidak terdapat dinding maka robot akan belok kanan 90° lalu menandai sel sekarang dengan cara mengisi variabel Depan_kanan = 1 juga mengisi nilai belakang sel ini dengan 26 yang menandakan setelah belok kiri 90° dibelakang dari robot akan terdapat dinding.

Setelah semua proses dilakukan, maka robot akan melakukan cek sel mana sajakah yang menjadi tetangganya sekarang dan diagram alir pun berlanjut ke gambar 3.17. Setelah mengetahui sel mana saja yang menjadi tetangga dari sel tempat robot sekarang, dan berapa nilai index dari masing-masing sel tersebut maka robot akan melakukan pengecekan kembali. Disini Algoritma *flood-fill* mulai diterapkan. Jika nilai index dari tetangga depan lebih kecil dari nilai index tetangga lainnya, maka robot akan memilih nilai depan sebagai tujuannya dan menjadikan nilai depan sebagai tetangga terkecil dari sel tempat robot sekarang.
Jika nilai index dari tetangga kanan lebih kecil dari nilai index tetangga lainnya, maka robot akan menjadikan nilai kanan sebagai tetangga terkecil dari sel tempat robot sekarang lalu robot berputar 90° ke kanan. Untuk sementara robot akan memilih nilai depan setelah dilakukan putaran sebagai tujuannya. Begitu juga ketika diketahui nilai index tetangga kiri lebih kecil dari nilai index tetangga lainnya, maka robot akan menjadikan nilai kiri sebagai tetangga terkecil dari sel tempat robot sekarang lalu robot berputar 90° ke kiri dan untuk sementara robot akan memilih nilai depan setelah dilakukan putaran sebagai tujuannya.

Jika nilai index dari tetangga depan sama dengan nilai index dari tetangga kanan, dan keduanya lebih kecil dari nilai index tetangga lainnya maka robot akan menjadikan nilai kanan sebagai nilai tetangga terkecil dari sel tempat robot memilih nilai depan sebagai tujuannya dengan alasan penghematan waktu tempuh karena jika robot memilih tetangga kanan sebagai tujuan berikutnya, robot harus berbelok 90° dahulu ke kanan berjalan maju yang tentunya akan menghabiskan banyak waktu. Proses seperti ini juga dilakukan jika diketahui nilai index tetangga depan sama dengan nilai index tetangga kiri, dan keduanya lebih kecil dari nilai index tetangga lainnya.

Untuk kondisi nilai index dari tetangga kanan sama dengan nilai index dari tetangga kiri, dan keduanya lebih kecil dari nilai index tetangga lainnya maka robot akan menjadikan nilai kanan sebagai nilai tetangga terkecil dari sel tempat robot dan memilih tetangga kiri sebagai tujuan. Penetapan tetangga mana yang akan menjadi tujuan dan terkecil pada kondisi ini tergantung dari susunan lapangan yang akan dilalui yang telah kita katahui sebelumnya.

Setelah itu masih ada satu lagi pengujian setelah semua tetangga terkecil telah di tetapkan pada proses sebelumnya yaitu jika nilai index belakang dari nilai sel sekarang lebih kecil dari nilai index depan, kanan ataupun kiri, maka tetangga belakang akan dijadikan sebagai tetangga terkecil dari sel sekarang. Setelah semua proses telah dilakukan maka sistem diteruskan ke diagram alir yang terdapat pada gambar 3.18.

Pada diagram alir ini, sistem akan dilanjutkan ke pengecekan apakah nilai tetangga depan yang telah ditetapkan sebagai tujuan dari proses sebelumnya lebih

besar dari nilai sel tempat robot sekarang atau tidak, karena jika lebih besar maka robot tidak bisa berjalan menuju ke sel tersebut karena syarat dari algoritma *flood-fill* ini adalah robot mengikuti sel dengan nomor yang lebih kecil dari sel sekarang. Jika nilai index dari tetangga depan tidak lebih besar dari nilai sekarang (lebih kecil) maka sistem dilanjut ke proses selanjutnya yaiut cek peta yang telah dibuat sejauh ini. Namun jika ternyata nilai tetangga depan lebih besar dari nilai sekarang, maka nilai sel sekarang akan dirubah dengan cara menjadikan besar **nilai sekarang = nilai dari tetangga depan + 1**.

Setelah dilakukan perubahan, maka bisa dipastikan nilai sel sekarang lebih besar 1 dari nilai sel tetangga didepannya, dimana tetangga depan merupakan nilai tujuan berikutnya yang akan dilalui oleh robot dalam melakukan eksplorasinya. Selanjutnya proses dilanjutkan, robot akan melakukan cek peta yang ada pada robot guna mengetahui apakah ada perubahan yang terjadi setelah dilakukannya penambahan nilai sekarang yang dilakukan pada proses sebelumnya. Jika terdapat ada nilai sel yang lebih kecil dari tetangga terkecilnya, maka akan dilakukan perbaikan peta (*Re-flood*) sampai tidak ada nilai sel dari peta pada robot yang lebih kecil dari nilai tetangga terkecilnya. Perbaikan peta ini dilakukan dengan cara menjadikan nilai sel yang tetangga terkecilnya lebih besar menjadi **nilai sel = nilai tetangga terkecil + 1**. Sebaliknya jika tidak terjadi perubahan pada peta, maka robot akan kembali ke diagram alir yang ada pada gambar 3.15 untuk bergerak maju sampai menemukan sel dengan nilai index = 0 (*GOAL*).

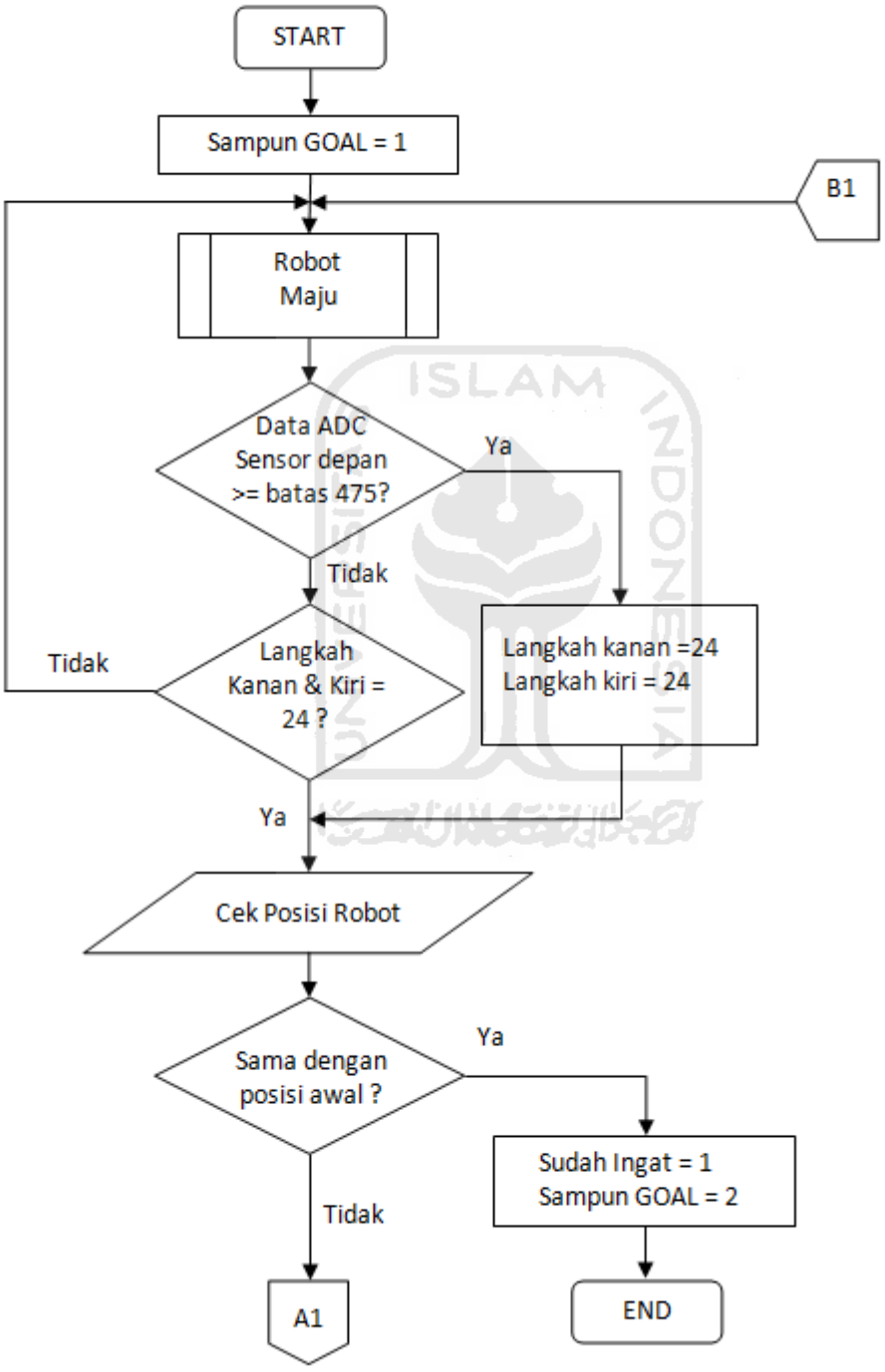
C. Perancangan *flowchart* Pemilihan Jalur dari *GOAL* menuju *HOME*

Ketika *GOAL* dari suatu lapangan telah ditemukan, maka secara bersamaan peta jalur terpendek dari lapangan tersebut telah dibuat. Untuk bisa kembali lagi ke *HOME* maka robot akan menerapkan metode *algoritma flood-fill* dengan cara kebalikan dari ketika melakukan eksplorasi menemukan *GOAL*, yaitu jika pada eksplorasi robot akan berjalan menuju sel yang mempunyai nilai terkecil hingga nilai index = 0 ditemukan, maka pada saat kembali dari *GOAL* menuju *HOME* ini robot akan berjalan menuju sel yang nilainya lebih besar 1 dari nilai sel robot berada hingga robot berada pada *HOME* atau posisi tempat robot melakukan *START*. Perancangan dari diagram alir robot dari *GOAL* menuju *HOME* ini dapat dilihat pada gambar 3.19 - gambar 3.20.

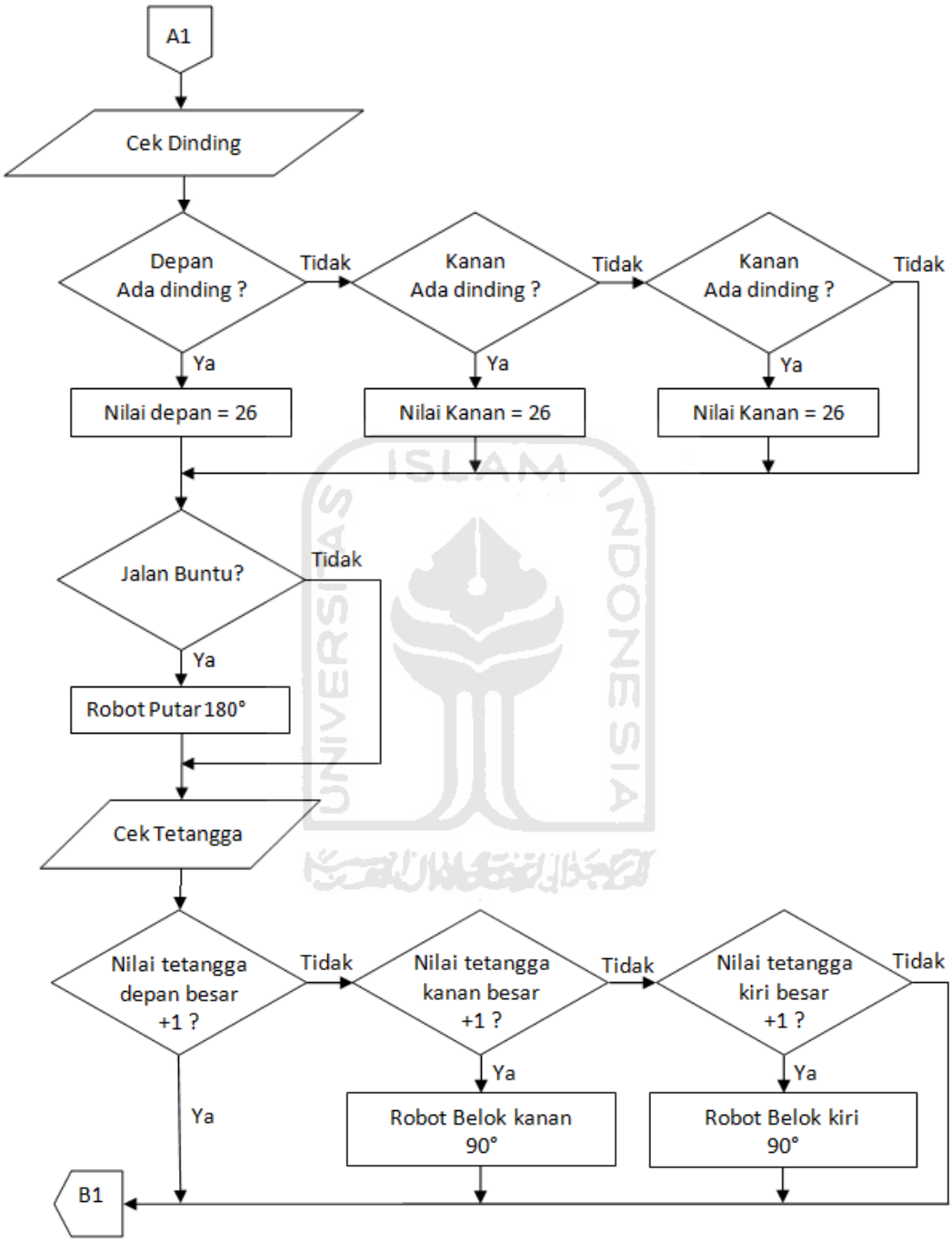
Pada gambar 3.19, diagram alir dimulai dengan *START* yang menandakan awal mulainya program dan sampun GOAL bernilai 1 yang menandakan robot sudah melakukan eksplorasi dan telah menemukan *GOAL* dan sekarang robot berada pada mode kembali ke *HOME*.

Selanjutnya robot akan bergerak maju sampai langkah pada pembacaan *encoder* roda kanan dan kiri mencapai 24 langkah atau jika data pembacaan ADC sensor depan >= batas 475 (Robot sangat dekat dengan tembok) walaupun langkah kanan dan kiri belum mencapai 24, setelah itu robot berhenti untuk melakukan cek posisi sekarang untuk mengetahui di sel nomor berapa robot berada dan melakukan cek apakah koordinat (x,y) dari tempat robot berdiri sekarang sama dengan koordinat (x,y) awal (*HOME*) atau tidak. Jika koordinat (x,y) dari tempat robot berdiri sekarang sama dengan koordinat (x,y) awal, maka

HOME telah dicapai. Mengisi nilai variabel sudah_ingat = 1 & variabel sampun_*GOAL* = 2 yang menandakan mode hapalan sudah bisa dijalankan.



Gambar 3.19 Diagram alir proses penentuan jalan dari *GOAL* menuju *HOME*



Gambar 3.20 Diagram alir proses penentuan jalan dari *GOAL* menuju *HOME*

Jika Posisi awal (*HOME*) belum ditemukan, maka sistem akan berlanjut ke diagram alir pada gambar 3.20 dan robot akan melakukan cek dinding. Jika didepan ada dinding maka nilai sel didepan diisi dengan nilai yang tinggi yaitu 26. Jika didepan tidak ada dinding maka dilanjutkan ke dinding kanan dan begitu seterusnya. Jika semua proses telah selesai dilakukan dan telah didapat dibagian mana saja dinding terdapat maka proses dilanjutkan, jika diketahui didepan ada dinding, di kanan ada dinding, dan di kiri juga terdapat dinding maka robot menemui jalan buntu dan robot akan berputar 180°, jika tidak ditemui jalan buntu maka proses akan diteruskan ke cek tetangga.

Robot akan melakukan cek sel mana sajakah yang menjadi tetangganya sekarang. Setelah mengetahuinya, robot akan melihat peta jalur terpendek yang telah dibuat. Jika nilai index dari tetangga depan lebih besar 1 dari nilai sel robot sekarang, maka robot akan menjadikan tetangga depan sebagai tujuan berikutnya. Tetapi jika nilai index dari tetangga depan tidak lebih besar 1 dari nilai sel sekarang, maka proses dilanjutkan ke kondisi beikutnya.

Jika nilai index dari tetangga kanan lebih besar 1 dari nilai sel robot sekarang, maka robot akan berputar 90° ke kanan dan menetapkan nilai sel depan setelah dilakukan putaran sebagai tujuan berikutnya. Begitu juga ketika diketahui nilai index tetangga kiri lebih besar 1 dari nilai sel robot sekarang, maka robot akan berputar 90° ke kiri dan menetapkan nilai sel depan setelah dilakukan putaran sebagai tujuan berikutnya. Selanjutnya sistem kembali ke diagram alir yang ada pada gambar 3.19 untuk bergerak maju sampai robot berhasil menemukan *HOME*.

D. Perancangan *flowchart* Pemilihan Jalur pada Mode Hapalan

Mode hapalan merupakan mode dimana robot dapat langsung berjalan dari *HOME* menuju *GOAL* tanpa harus melakukan eksplorasi lapangan terlebih dahulu, dengan syarat variabel sudah_ingat sudah bernilai satu dan variabel sampun *GOAL* sudah bernilai 2. Variabel sudah_ingat ini dapat di cek pada menu hapalan yang ditampilkan LCD sebelum robot diperintahkan untuk berjalan.

Pada mode hapalan ini, robot akan kembali menerapkan metode algoritma *flood-fill* dengan mengikuti nilai sel terkecil pada pencariannya menuju *GOAL*, tentunya dengan menggunakan peta yang telah dibuat setelah eksplorasi dilakukan. Perancangan dari diagram alir robot pada mode hapalan ini dapat dilihat pada gambar 3.21 - gambar 3.22.

Pada gambar 3.21, diagram alir dimulai dengan *START* yang menandakan awal mulainya program. Sudah_ingat bernilai 1 dan sampun GOAL bernilai 2 yang menandakan robot sudah melakukan eksplorasi hingga menemukan *GOAL* dan robot telah berhasil kembali ke *HOME* dengan sukses.

Selanjutnya robot akan bergerak maju sampai langkah pada pembacaan *encoder* roda kanan dan kiri mencapai 24 langkah atau jika data pembacaan ADC sensor depan >= batas 475 (Robot sangat dekat dengan tembok) walau pun langkah kanan dan kiri belum mencapai 24, setelah itu robot berhenti untuk melakukan cek posisi sekarang untuk mengetahui di sel nomor berapa robot berada.

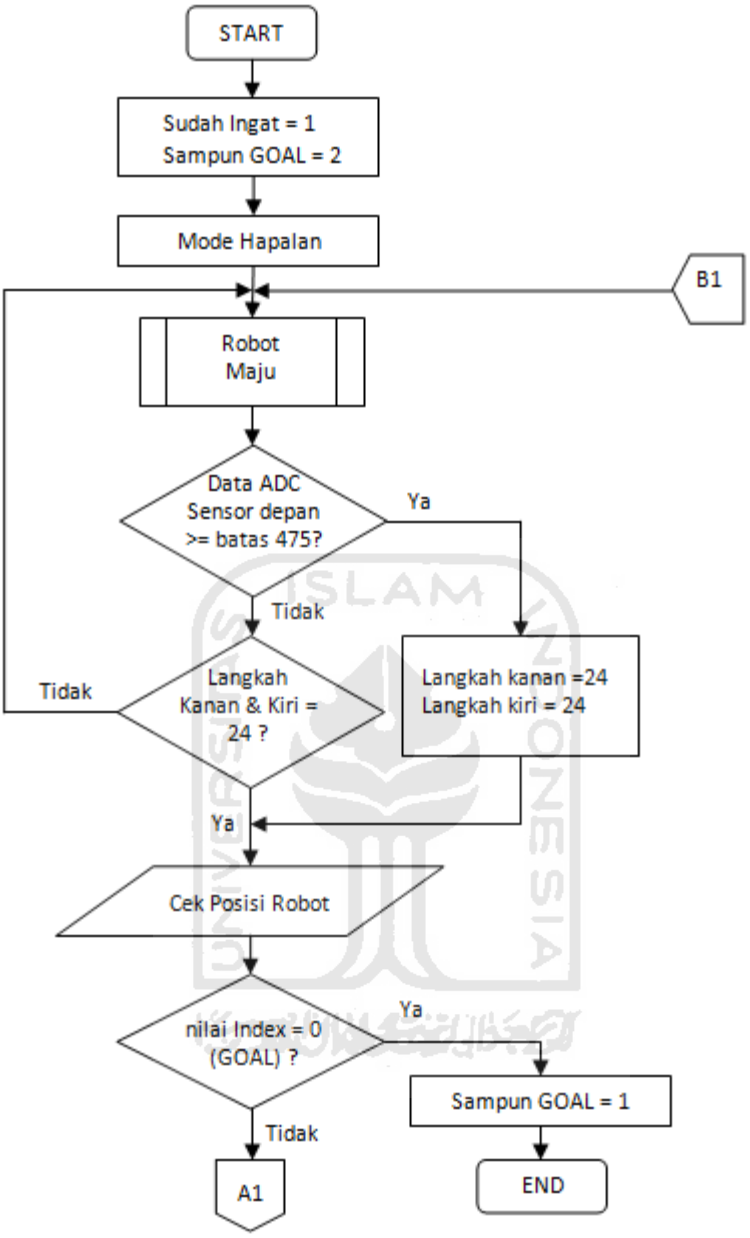
Setelah robot mengetahui di posisi mana dia berada sekarang, di koordinat (x,y) mana dan berapa nilai index sel tersebut, maka robot melakukan cek apakah

nilai dari index tempat robot berdiri sekarang bernilai 0 atau tidak. Jika nilai dari index sekarang bernilai 0 maka posisi *GOAL* telah dicapai dan nilai variabel sampun *GOAL* akan diberi nilai 1 yang menandakan bahwa *GOAL* telah berhasil dicapai.

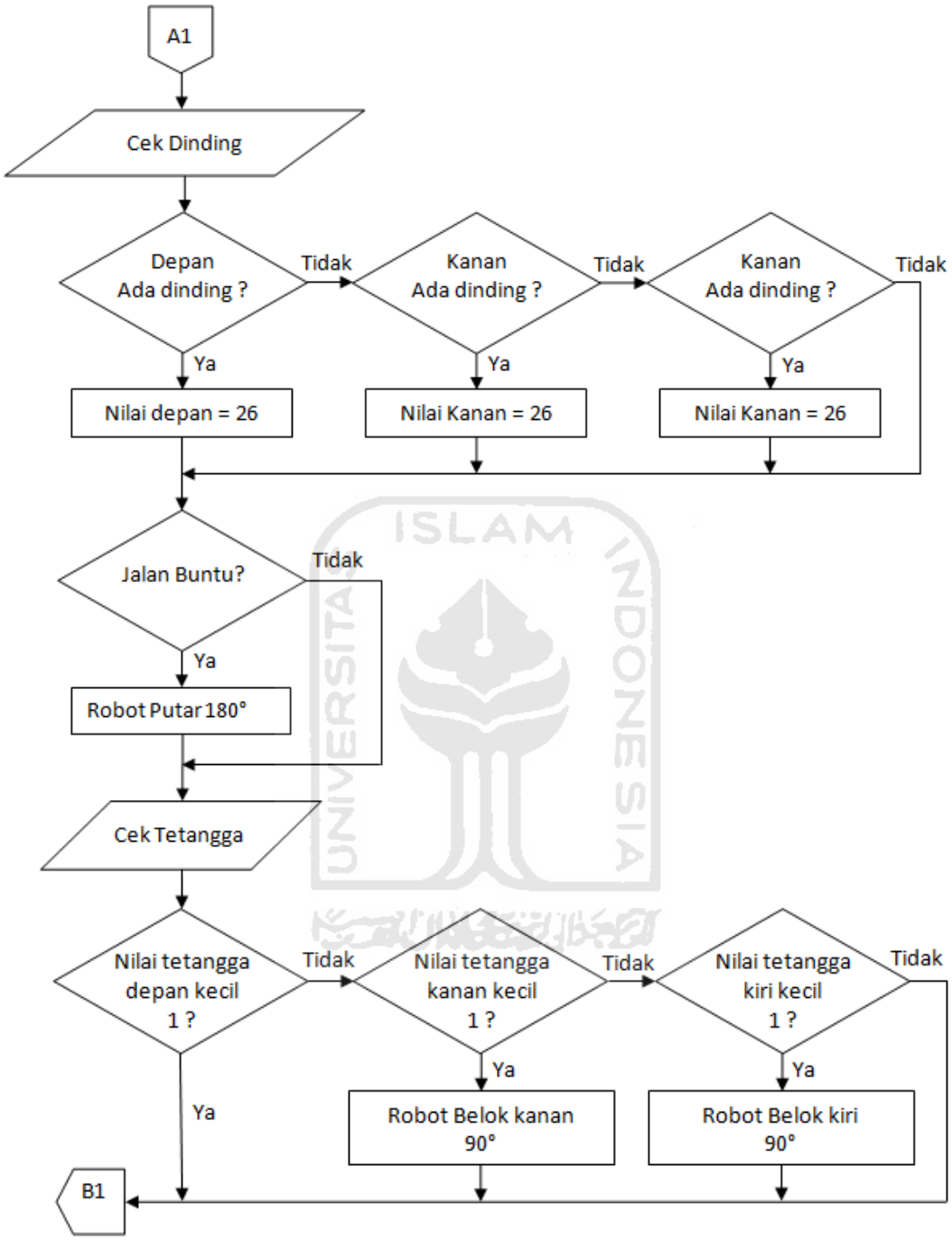
Jika *GOAL* belum ditemukan, maka sistem akan berlanjut ke diagram alir pada gambar 3.22 dan robot akan melakukan cek dinding. Jika didepan ada dinding maka nilai sel didepan diisi dengan nilai yang tinggi yaitu 26. Jika didepan tidak ada dinding maka dilanjutkan ke dinding kanan dan begitu seterusnya. Jika semua proses telah selesai dilakukan dan telah didapat dibagian mana saja dinding terdapat maka proses dilanjutkan, jika diketahui didepan ada dinding, di kanan ada dinding, dan di kiri juga terdapat dinding maka robot menemui jalan buntu dan robot akan berputar 180°, jika tidak ditemui jalan buntu maka proses akan diteruskan ke cek tetangga.

Robot akan melakukan cek sel mana sajakah yang menjadi tetangganya sekarang. Setelah mengetahuinya, robot akan melihat peta jalur terpendek yang telah dibuat. Jika nilai index dari tetangga depan lebih kecil 1 dari nilai sel robot sekarang, maka robot akan menjadikan tetangga depan sebagai tujuan berikutnya. Tetapi jika nilai index dari tetangga depan tidak lebih kecil 1 dari nilai sel sekarang, maka proses dilanjutkan ke kondisi beikutnya.

Jika nilai index dari tetangga kanan lebih kecil 1 dari nilai sel robot sekarang, maka robot akan berputar 90° ke kanan dan menetapkan nilai sel depan setelah dilakukan putaran sebagai tujuan berikutnya.



Gambar 3.21 Diagram alir proses penentuan jalan pada mode Hapalan



Gambar 3.22 Diagram alir proses penentuan jalan pada mode Hapalan

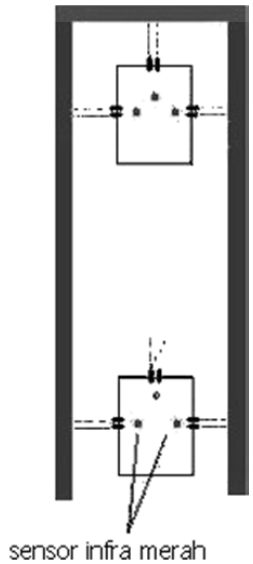
Begitu juga ketika diketahui nilai index tetangga kiri lebih kecil 1 dari nilai sel robot sekarang, maka robot akan berputar 90° ke kiri dan menetapkan nilai sel depan setelah dilakukan putaran sebagai tujuan berikutnya. Selanjutnya sistem kembali ke diagram alir yang ada pada gambar 3.21 untuk bergerak maju sampai robot berhasil menemukan *GOAL*.

BAB IV

ANALISA DAN PEMBAHASAN

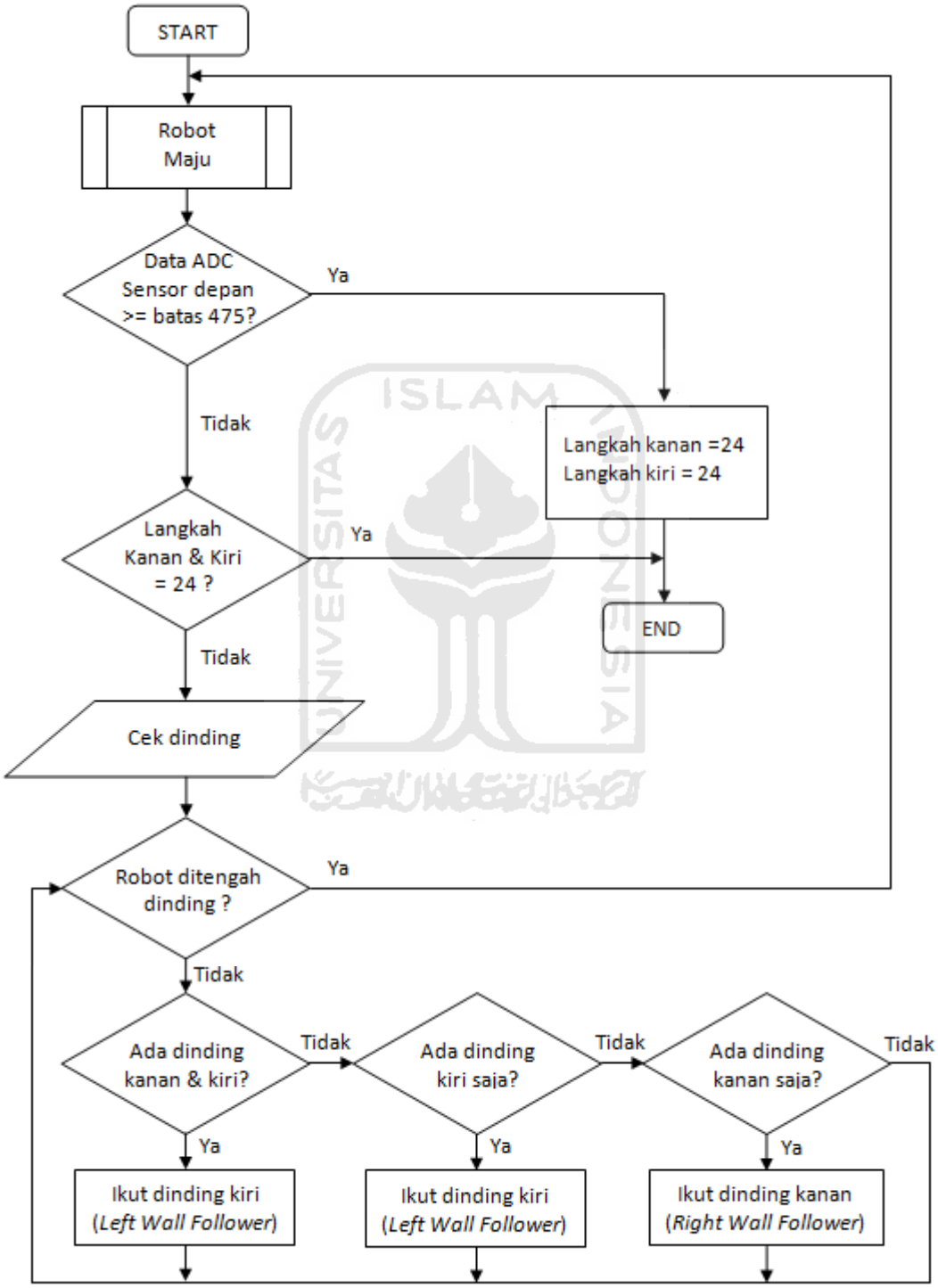
4.1 Metode Pembacaan Dinding Labirin

Robot micromouse pada penelitian ini memiliki 3 sensor SHARP GP2D120 yang terletak di sisi depan, sisi kanan dan sisi kiri robot yang berfungsi mendeteksi dinding dan rintangan lainnya pada labirin. Sensor ini akan mengembalikan nilai yang sebanding dengan jarak robot dengan dinding kiri, dinding kanan, atau dinding di depan robot yang akan dibaca oleh PINA dengan memanfaatkan fitur ADC pada mikrokontroler ATMEGA32. Selanjutnya, data dari sensor SHARP GP2D120 ini akan disimpan oleh robot untuk pemetaan dinding labirin dan juga diproses oleh robot ketika robot berjalan mengikuti dinding sampai langkah roda kanan dan langkah roda kiri bernilai 24.



Gambar 4.1 Metode pembacaan dinding labirin

4.2 Penggunaan Metode *Wall Follower* untuk Membuat Robot Berjalan Lurus pada Labirin



Gambar 4.2 Diagram alir *wall follower* pada waktu robot jalan berpindah 1 sel

Robot micromouse dalam penelitian ini memanfaatkan metode *wall follower* ketika berjalan dari 1 sel ke sel lainnya. Hal ini dilakukan, karena jika hanya mengandalkan perhitungan *encoder* roda kanan dan *encoder* roda kiri saja, maka robot akan bergerak miring atau bahkan menabrak dinding sehingga pada hitungan langkah ke 24 robot belum mencapai 1 sel, ini disebabkan kedua motor DC yang memiliki kecepatan berbeda walaupun jenisnya sama. Diagram alir dari metode wall follower yang digunakan pada robot ini dapat dilihat pada gambar 4.2 sebelumnya.

Diagram alir dimulai dengan *START* yang menandakan awal mulainya program dan diteruskan dengan robot bergerak maju sampai langkah pada pembacaan *encoder* roda kanan dan kiri mencapai 24 langkah atau jika data pembacaan ADC sensor depan >= batas 475 (Robot sangat dekat dengan tembok) walaupun langkah kanan dan kiri belum mencapai 24. Jika langkah roda kanan dan kiri sudah bernilai 24, maka program berakhir ditandai dengan *END* yang artinya robot sudah berpindah sejauh 1 sel.

Jika langkah 24 belum terpenuhi maka robot akan melakukan cek dinding secara kontinu. Selama posisi robot masih berada ditengah labirin maka robot akan terus maju lurus, tetapi jika dalam perjalanannya berjalan lurus robot bergeser akibat motor kanan dan kiri yang tidak sama kecepatannya maka jika robot mendeteksi ada dinding di kiri dan kanan, robot akan mengikuti dinding kiri sampai robot kembali berada ditengah, begitu juga jika hanya dinding kiri yang terdeteksi. Jika robot hanya mendeteksi dinding kanan, maka robot akan mengikuti dinding kanan sampai robot kembali berada ditengah. Begitu

seterusnya sampai langkah roda kiri dan roda kanan bernilai 24 atau sudah berpindah sejauh 1 sel.

4.3 Pengujian Robot dalam Melakukan Eksplorasi Lapangan

Sebelum mengeksplorasi suatu lapangan, robot micromouse terlebih dahulu telah dilengkapi dengan peta buta, yaitu peta informasi sementara mengenai seberapa jauh robot berada dari *GOAL* pada saat melakukan eksplorasi ini, tetapi peta buta ini tidak dilengkapi informasi mengenai dimana saja dinding-dinding sel berada.

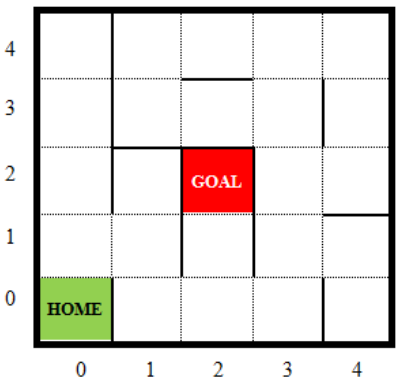
Peta buta ini dibuat dengan cara memberi nilai 0 pada sel labirin yang akan menjadi tujuan pada eksplorasi ini, lalu setelah itu masing-masing sel labirin yang ada disekitar sel tujuan diberi nilai dengan cara n + 1 hingga semua sel diberi nomor. Peta buta ini lah yang nantinya akan menjadi panduan robot dalam melakukan eksplorasi, dan yang akan dirubah oleh robot micromouse seiring ditemukannya dinding-dinding sel selama melakukan perjalanan. Untuk memudahkan pemograman, masing - masing sel juga telah diberi index yang mana nilai masing masing index pada tiap sel ini merupakan nilai peta buta pada tiap – tiap sel.

4	4	3	2	3	4
3	3	2	1	2	3
2	2	1	0	1	2
1	3	2	1	2	3
0	4	3	2	3	4
	0	1	2	3	4

4	Index 21	Index 22	Index 23	Index 24	Index 25
3	Index 16	Index 17	Index 18	Index 19	Index 20
2	Index 11	Index 12	Index 13	Index 14	Index 15
1	Index 6	Index 7	Index 8	Index 9	Index 10
0	Index 1	Index 2	Index 3	Index 4	Index 5
	0	1	2	3	4

Gambar 4.3 Peta buta (kiri) dan nilai index dari masing-masing sel (kanan)

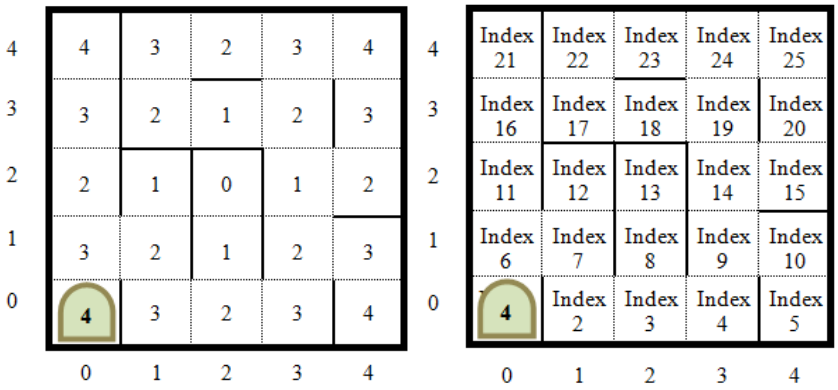
Lapangan pertama yang akan di eksplorasi pada pengujian ini dapat dilihat pada gambar 4.4 berikut:



Gambar 4.4 Lapangan yang akan di eksplorasi robot micromouse

Diketahui robot akan mengawali eksplorasi dari koordinat (0,0), dan *GOAL* terletak pada koordinat (2,2). Menurut peraturan dari lomba robot micromouse, sebuah *HOME* harus memiliki 3 dinding di setiap sisinya, masing - masing di sisi kanan, sisi kiri, dan sisi belakang. Ini dilakukan agar robot berjalan lurus kedepan hingga berpindah 1 sel ketika mengawali eksplorasinya.

Karena hanya sel didepan robot yang terbuka, dan nilai sel didepan robot lebih kecil dari nilai sel sekarang, maka robot akan bergerak maju sejauh 1 sel menuju index 6 dan menjadikan nilai dari index 6 sebagai tetangga terkecil dari sel sekarang yaitu index 1.

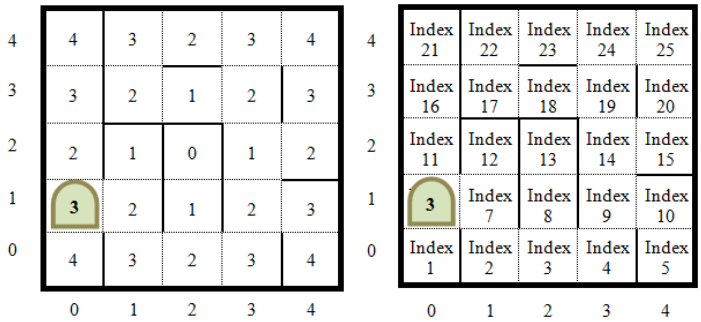


Gambar 4.5 Robot berada di *HOME* dan siap untuk mengeksplorasi lapangan

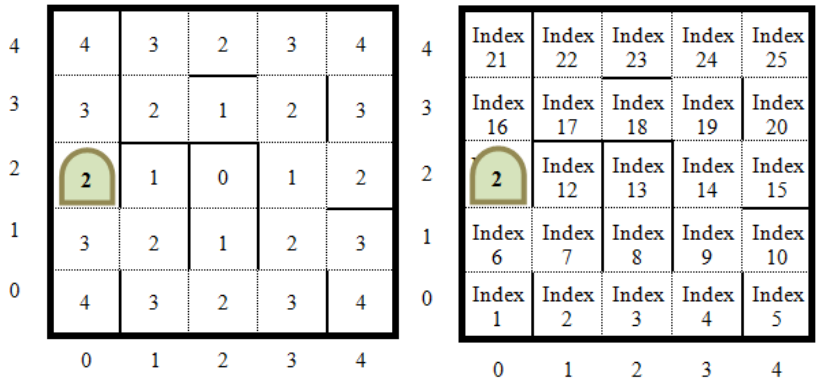
Variabel sampun pada index 1 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali. Setelah robot berpindah sejauh 1 sel, maka robot akan melakukan cek apakah nilai dari index sel sekarang 0 atau tidak, jika tidak berarti *GOAL* belum dicapai dan robot melanjutkan eksplorasinya. Pengecekan ini dilakukan secara kontinu setiap robot baru berpindah 1 sel hingga robot menemukan *GOAL*.

Karena bukan *GOAL*, maka robot melakukan cek dinding untuk mengetahui siapa saja tetangga yang terbuka dan berapa nilainya agar robot bisa menentukan ke sel mana ia harus bergerak selanjutnya.

Diketahui di sisi kiri robot terdapat dinding, maka tetangga kiri diabaikan. Disisi kanan robot terbuka tetangga dengan index 7 yang bernilai 2, dan didepan robot terbuka index 11 yang juga bernilai 2. Karena Sel robot sekarang mempunyai index 6 dengan nilai 3, dan masing - masing nilai index dari tetangganya bernilai lebih kecil dan sama, maka robot lebih memilih untuk maju ke index 11 yang ada didepannya dengan alasan robot akan menghabiskan banyak waktu jika memilih index 7 yang ada dikanannya karena harus berbelok dulu baru maju 1 sel. Robot maju 1 sel ke index 11 dan menjadikan index 7 sebagai tetangga terkecil dari index 6 tempat robot berada sekarang. Variabel sampun pada index 6 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali.

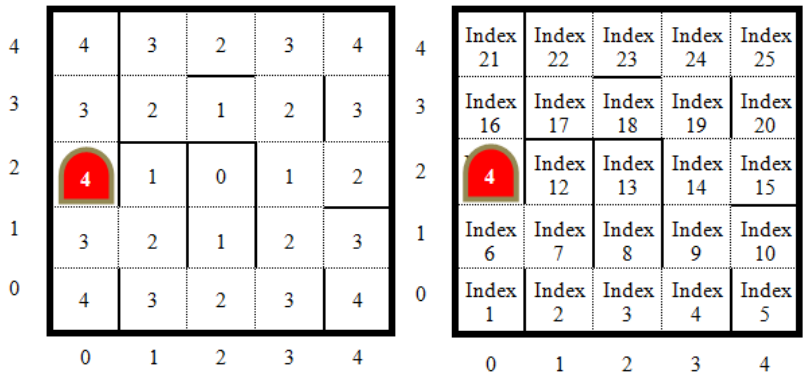


Gambar 4.6 Robot berada pada sel dengan index 6 yang bernilai 3

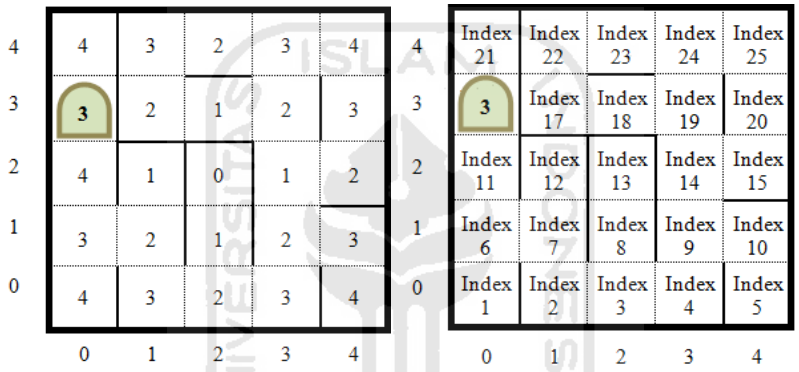


Gambar 4.7 Robot berada pada sel dengan index 11 yang bernilai 2

Setelah sampai pada sel dengan Index 11, maka robot melakukan cek dinding dan mendapat informasi bahwa di sisi kanan dan sisi kirinya adalah dinding, maka sisi kanan dan kiri pun diabaikan. Sisi depan mempunyai index 16 dengan nilai 3, dan sisi belakang yang baru saja di tinggalkan mempunyai index 6 dan juga bernilai 3, sementara nilai index 11 pada sel sekarang adalah 2. Karena sel didepannya lebih besar, maka syarat dari metode algoritma *flood-fill* yang diterapkan yaitu robot hanya bisa bergerak maju ke sel yang nomornya lebih kecil dari sel sekarang tidak terpenuhi. Jika berada pada kondisi ini, maka robot harus melakukan *re-flood* guna mengubah peta buta yang ada sekarang dengan cara **nilai sekarang = nilai sel tetangga depan + 1**. Maka sel index 11 tempat sel sekarang yang awalnya bernilai 2 berubah menjadi 4 **(3 + 1= 4)**. Karena sel didepan robot sudah menjadi lebih kecil dari sel sekarang, maka robot pun bergerak maju kedepan menuju sel dengan index 16 dan menjadikan Index 6 yang berada dibelakang sel sekarang sebagai tetangga terkecil dari index sel sekarang yaiut index 11. Variabel sampun pada index 11 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali.

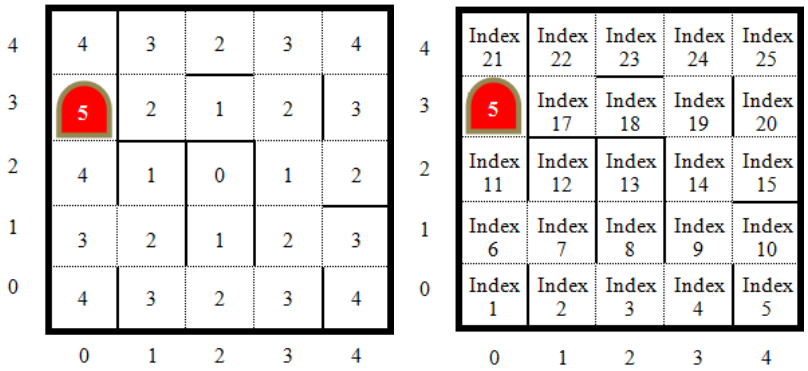


Gambar 4.8 Nilai Index 11 berubah menjadi 4 setelah dilakukan *re-flood*



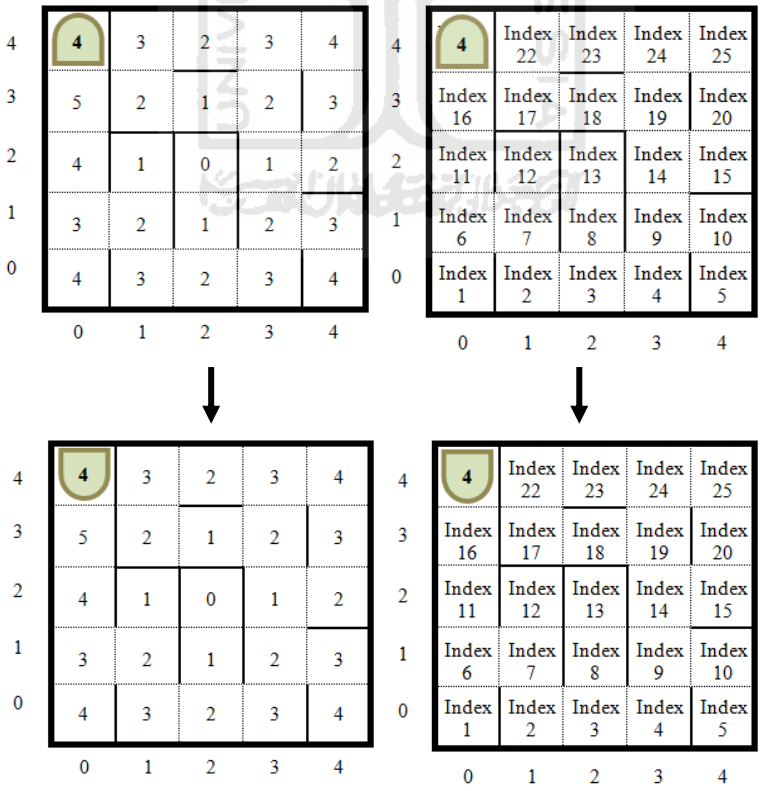
Gambar 4.9 Robot berada pada sel dengan index 16 yang bernilai 2

Setelah sampai pada sel dengan Index 16, maka robot melakukan cek dinding dan mendapat informasi bahwa di sisi kanan dan sisi kirinya adalah dinding, maka sisi kanan dan kiri pun diabaikan. Sisi depan mempunyai index 21 dengan nilai 4, dan sisi belakang yang baru saja di tinggalkan mempunyai index 11 dan juga bernilai 4, sementara nilai index 16 pada sel sekarang adalah 3 atau lebih kecil dari kedua tetangganya, maka robot kembali melakukan *re-flood* lalu maju 1 sel ke tetangga depannya dan menjadikan tetangga belakang yaitu index 11 sebagai tetangga terkecil dari index sel sekarang yaitu index 16. Variabel sampun pada index 16 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali.



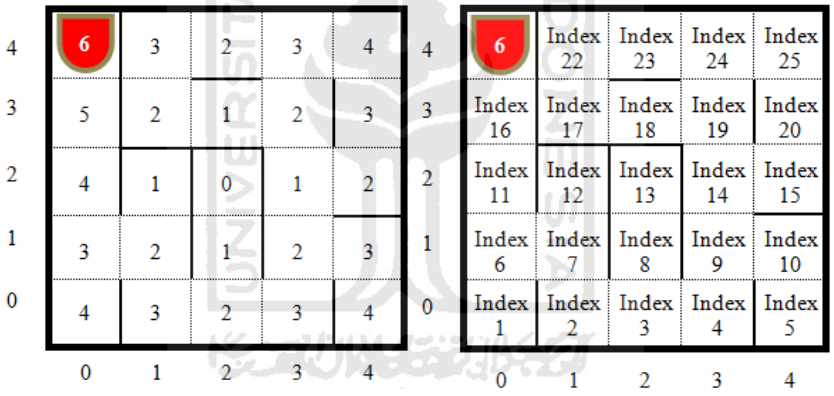
Gambar 4.10 Nilai Index 16 berubah menjadi 5 setelah dilakukan *re-flood*

Setelah sampai pada sel dengan Index 16, maka robot melakukan cek dinding dan mendapat informasi bahwa di 3 sisinya terdapat dinding atau robot menumukan jalan buntu, maka robot langsung berputar 180° dan menandai sel ini dengan variabel jalan_buntu = 1 .



Gambar 4.11 Robot menemukan jalan buntu dan berputar 180°

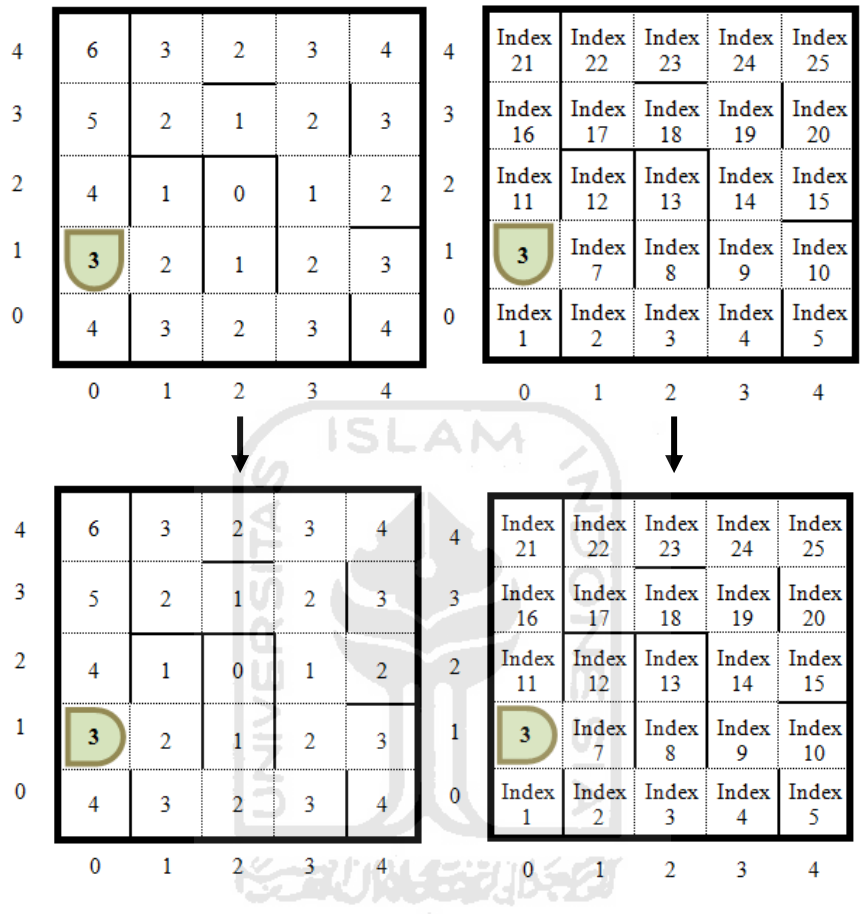
Setelah selesai berputar, maka robot kembali melakukan cek dinding dan mendapat informasi bahwa di sisi kanan, sisi kiri dan sisi belakangnya adalah dinding, maka sisi kanan,sisi kiri dan sisi belakang pun diabaikan. Sisi depan mempunyai index 16 dengan nilai 5, sementara nilai index 21 pada sel sekarang adalah 4 atau lebih kecil dari tetangganya yang terbuka, maka robot kembali melakukan *re-flood* dan maju 1 sel ke tetangga depannya dan juga menjadikan tetangga depan yaitu index 16 sebagai tetangga terkecil dari index sel sekarang yaitu index 21. Variabel sampun pada index 21 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali.



Gambar 4.12 Nilai Index 21 berubah menjadi 6 setelah dilakukan *re-flood*

Karena pada index 21 tadi telah ditandai dengan variabel Jalan_buntu = 1, maka ketika sampai di index 16, sisi belakangnya yaitu index 21 akan dianggap sebagai dinding dan hal ini akan mengakibatkan index 16 di tandai dengan Jalan_buntu = 1 juga karena pada robot telah di masukkan informasi jika disisi kanan, sisi kiri, dan sisi belakang terdapat dinding maka sel tersebut akan ditandai dengan variabel Jalan_buntu = 1. Hal ini agar sel ini tidak dilalui robot lagi karena

beujung pada jalan buntu. Secara peta sel dengan variabel buntu ini belum di blok, tetapi jika robot berada pada sisi ini dia akan menganggapnya sebagai dinding.

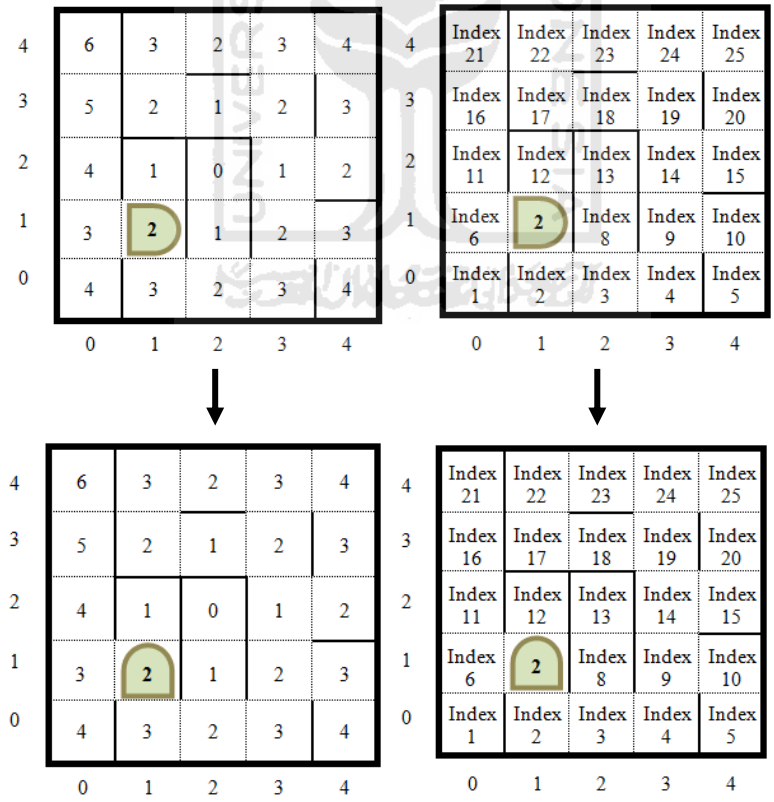


Gambar 4.13 Robot belok kiri 90° karena nilai index tetangga kiri lebih kecil

Sekarang robot telah sampai pada sel dengan nilai index 6. Robot melakukan cek dinding dan mendapat informasi bahwa di sisi kanan adalah dinding, maka sisi kanan diabaikan. Sisi depan mempunyai index 1 dengan nilai 4, dan sisi belakang yang baru saja di tinggalkan dianggap sebagai dinding karena variabel Jalan_buntunya bernilai 1, sisi kiri mempunyai index 7 dengan nilai 2. Karena nilai index dari tetangga kiri lebih kecil maka robot berbelok 90° lalu maju ke index 7 dan menjadikan index 7 sebagai tetangga terkecil dari index 6 yang

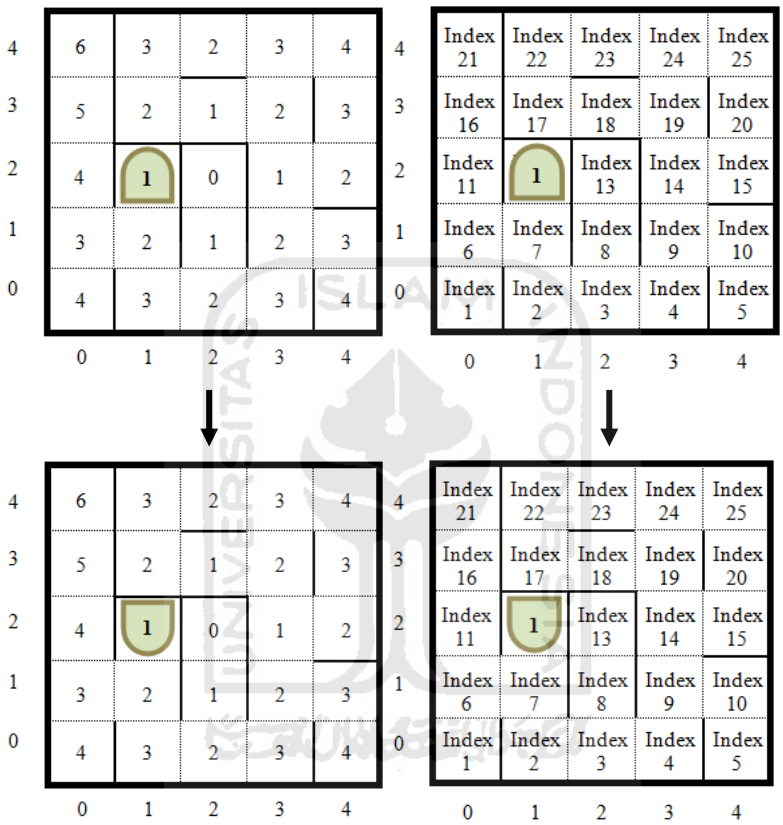
terbaru. Variabel sampun pada index 6 diberi nilai +1 menjadi 2 yang menandakan sel ini telah dilewati 2 kali selama eksplorasi ini.

Sekarang robot telah sampai pada yang mempunyai index 7 dengan nilai 2. Robot melakukan cek dinding dan mendapat informasi bahwa di sisi depan adalah dinding, maka sisi depan diabaikan. Sisi kanan mempunyai index 2 dengan nilai 3, sisi belakang mempunyai index 6 dengan nilai 3, sisi kiri mempunyai index 12 dengan nilai 1. Karena nilai index dari tetangga kiri lebih kecil maka robot berbelok 90° lalu maju ke index 12 dan menjadikan index 12 sebagai tetangga terkecil dari index 7. Variabel sampun pada index 7 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali selama eksplorasi ini.



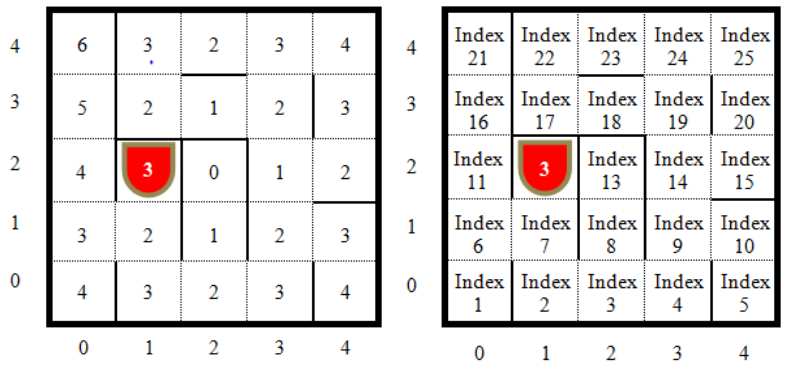
Gambar 4.14 Robot belok kiri 90° karena nilai index tetangga kiri lebih kecil

Setelah sampai pada sel dengan Index 12, maka robot melakukan cek dinding dan mendapat informasi bahwa di 3 sisinya terdapat dinding atau robot menumukan jalan buntu, maka robot langsung berputar 180° dan menandai sel ini dengan variabel jalan_buntu = 1



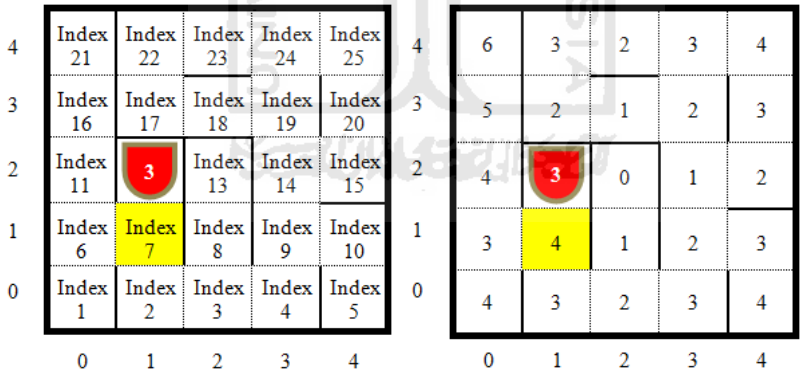
Gambar 4.15 Robot menemukan jalan buntu dan berputar 180°

Setelah selesai berputar, maka robot kembali melakukan cek dinding dan mendapat informasi bahwa di sisi kanan, sisi kiri dan sisi belakangnya adalah dinding, maka sisi kanan,sisi kiri dan sisi belakang pun diabaikan. Sisi depan mempunyai index 7 dengan nilai 2, sementara nilai index 12 pada sel sekarang adalah 1 atau lebih kecil dari tetangganya yang terbuka, maka robot kembali melakukan *re-flood* sehingga nilai index 12 menjadi 3.



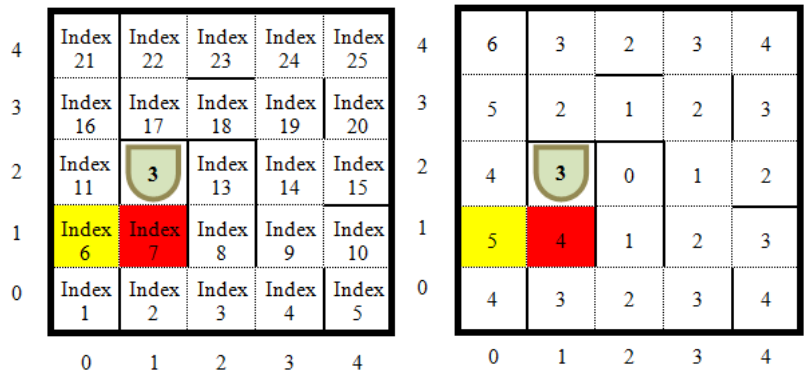
Gambar 4.16 Nilai Index 12 berubah menjadi 3 setelah dilakukan *re-flood*

Tetapi perubahan mengakibatkan peta yang ada sementara menjadi rusak, hal ini karena index 12 merupakan tetangga terkecil dari index 7, dan syarat yang telah diberikan adalah suatu sel harus lebih besar 1 dari tetangga terkecilnya. Karena index 12 lebih besar dari index 7, maka nilai index 7 berubah menjadi **nilai tetangga terkecil + 1**, sehingga diperoleh nilai index 7 sebesar 4 **(3 + 1 = 4)**.



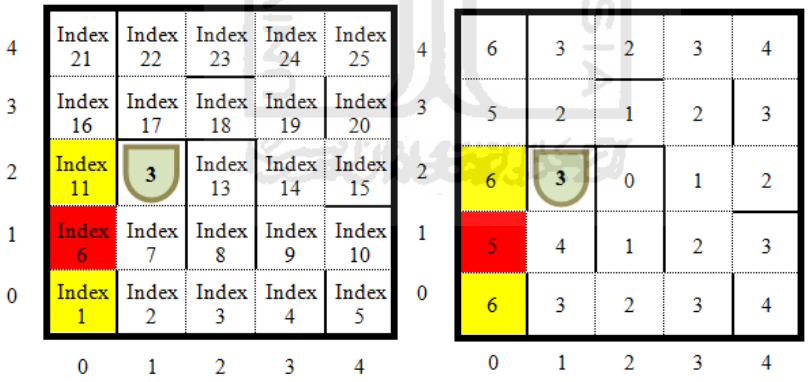
Gambar 4.17 Robot memperbaiki peta karena terjadi perubahan akibat *re-flood*

Namun, karena index 7 merupakan tetangga terkecil dari index 6, dan index 7 lebih besar dari index 6, maka nilai index 6 berubah menjadi **nilai tetangga terkecil + 1**, sehingga diperoleh nilai index 6 sebesar 4 **(4 + 1 = 5)**.



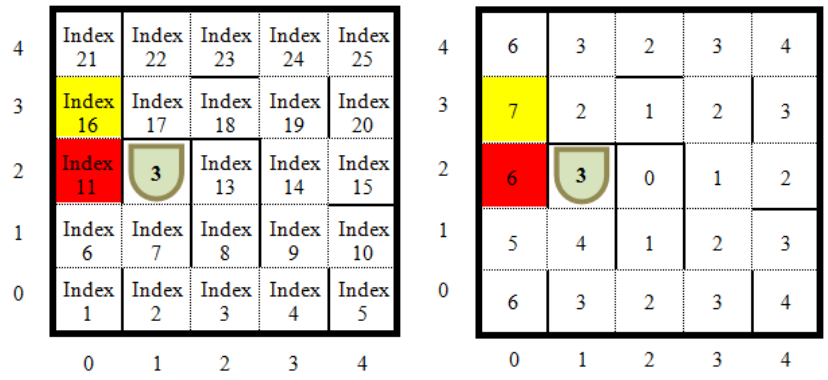
Gambar 4.18 Robot memperbaiki peta karena terjadi perubahan akibat *re-flood*

Karena index 6 merupakan tetangga terkecil dari index 1 dan index 11, dan index 5 lebih besar dari keduanya, maka nilai index 1 dan index 11 berubah menjadi **nilai tetangga terkecil + 1**, sehingga diperoleh nilai index 1 sebesar 6 **(5 + 1 = 6)** dan index 11 sebesar 6 **(5 + 1 = 6)**.



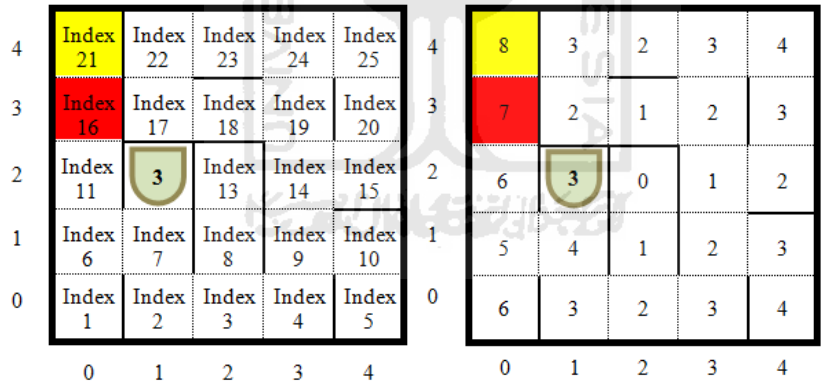
Gambar 4.19 Robot memperbaiki peta karena terjadi perubahan akibat *re-flood*

Karena index 11 merupakan tetangga terkecil dari index 16, dan index 11 lebih besar dari index 16, maka nilai index 16 berubah menjadi **nilai tetangga terkecil + 1**, sehingga diperoleh nilai index 16 sebesar 4 **(6 + 1 = 7)**.



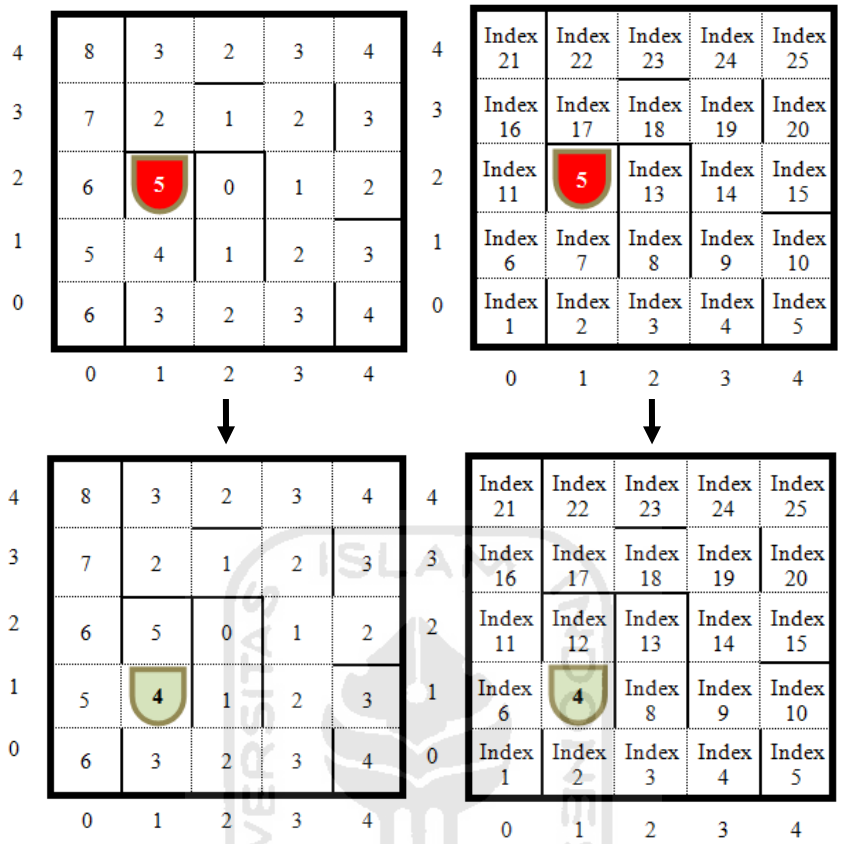
Gambar 4.20 Robot memperbaiki peta karena terjadi perubahan akibat *re-flood*

Karena index 16 merupakan tetangga terkecil dari index 21, dan index 16 lebih besar dari index 21, maka nilai index 21 berubah menjadi **nilai tetangga terkecil + 1**, sehingga diperoleh nilai index 16 sebesar 4 **(7 + 1 = 8)**.



Gambar 4.21 Robot memperbaiki peta karena terjadi perubahan akibat *re-flood*

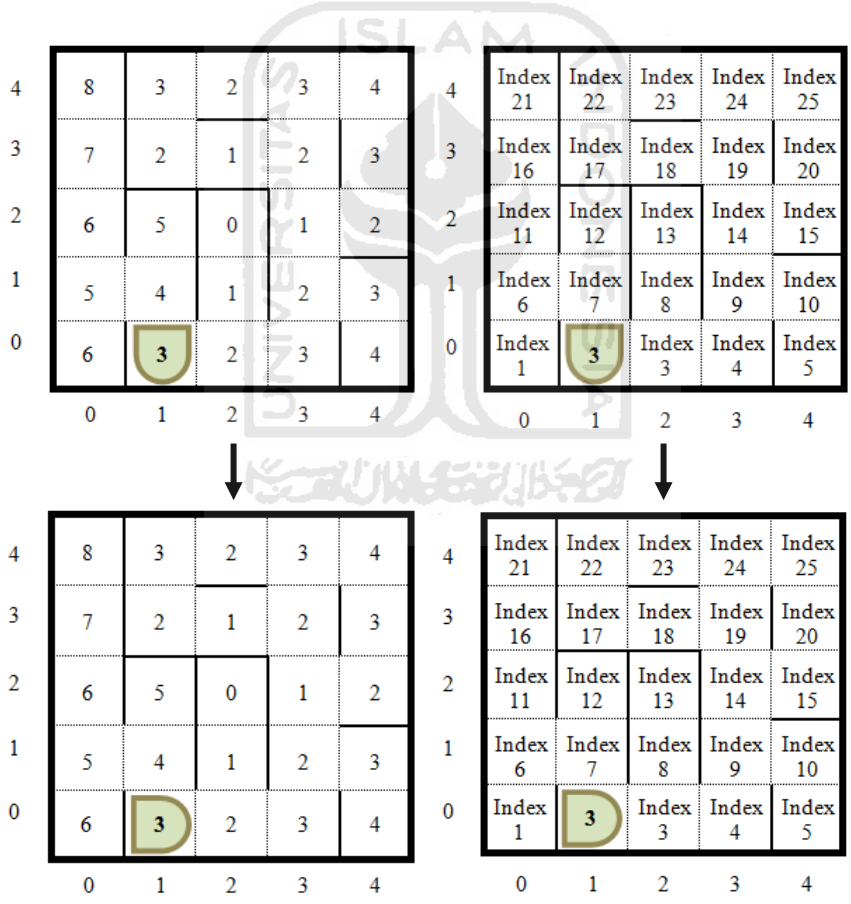
Karena nilai tetangga depan dari index 12 lebih besar, maka robot kembali melakukan *re-flood* lalu setelah nilai tetangga depan lebih kecil maka robot maju kedepan dan menjadikan index 7 sebagai tetangga terkecil dari index 7. Variabel sampun pada index 12 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali selama eksplorasi ini.



Gambar 4.22 Robot melakukan *re-flood* dan maju 1 sel ke depan

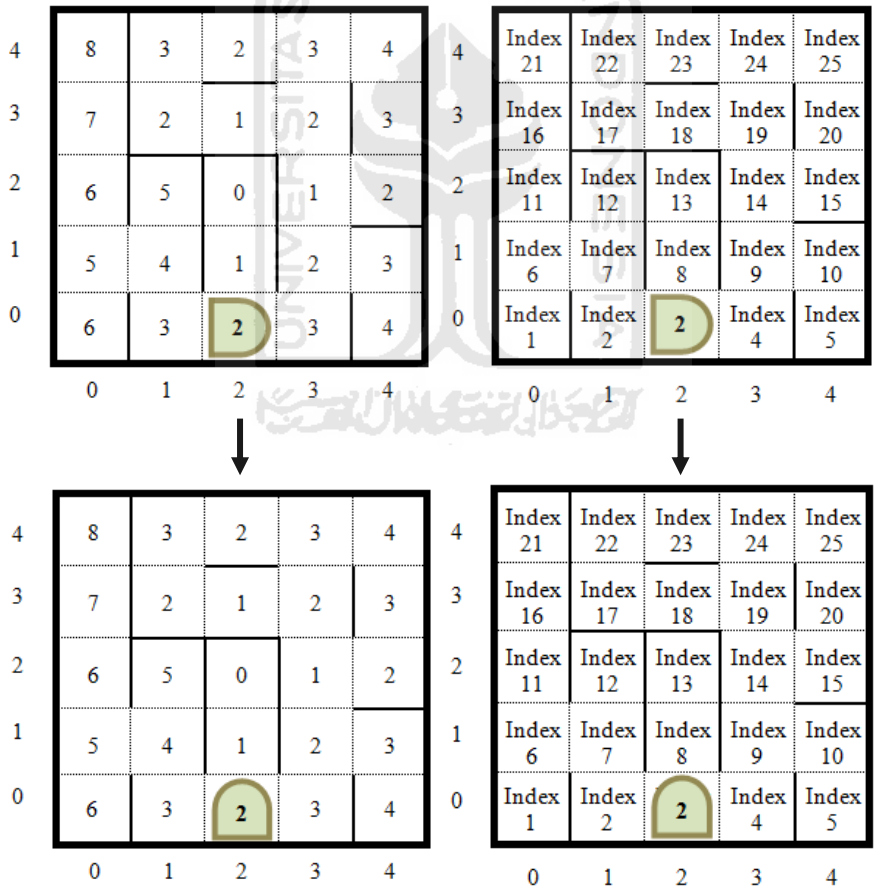
Sekarang robot telah sampai pada sel berikutnya yang mempunyai index 7 bernilai 4 . Robot melakukan cek dinding dan mendapat informasi bahwa di sisi kiri adalah dinding, maka sisi kiri diabaikan. Sisi kanan mempunyai index 6 dengan nilai 5, sisi belakang mempunyai index 12 dengan nilai 5, sisi depan mempunyai index 2 dengan nilai 3. Karena nilai index dari tetangga depan lebih kecil maka robot maju ke index 3 dan menjadikan index 3 sebagai tetangga terkecil dari index 7 yang terbaru. Variabel sampun pada index 7 diberi nilai +1 menjadi 2 yang menandakan sel ini telah dilewati 2 kali selama eksplorasi ini.

Setelah sampai pada sel dengan index 2, maka robot melakukan cek dinding dan mendapat informasi bahwa di sisi depan dan sisi kanan adalah dinding, maka sisi depan dan sisi kanan diabaikan. Sisi belakang mempunyai index 7 dengan nilai 4, sisi kiri mempunyai index 3 dengan nilai 2. Karena nilai index dari tetangga kiri lebih kecil maka robot berbelok 90° lalu maju ke index 3 dan menjadikan index 3 sebagai tetangga terkecil dari index 2. Variabel sampun pada index 2 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali selama eksplorasi ini.



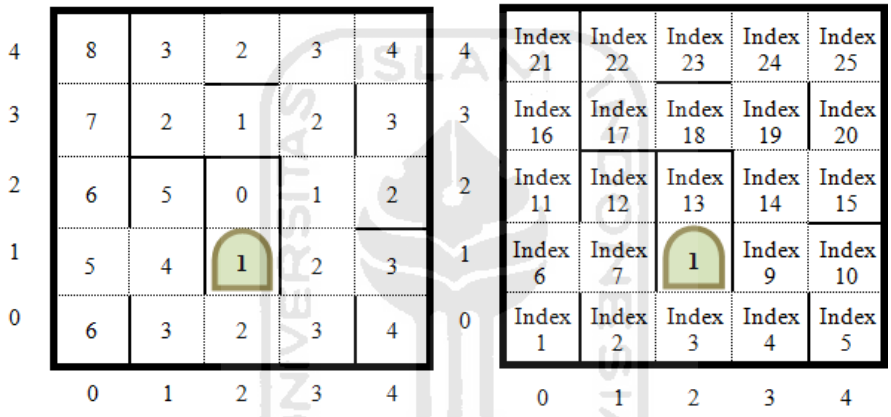
Gambar 4.23 Robot belok kiri 90° karena nilai index tetangga kiri lebih kecil

Setelah sampai pada sel dengan index 3, maka robot melakukan cek dinding dan mendapat informasi bahwa sisi kanan adalah dinding, maka sisi kanan diabaikan. Sisi belakang mempunyai index 2 dengan nilai 3, sisi depan mempunyai index 4 dengan nilai 3, sisi kiri mempunyai index 8 dengan nilai 1. Karena nilai index dari tetangga kiri lebih kecil maka robot berbelok 90° lalu maju ke index 8 dan menjadikan index 8 sebagai tetangga terkecil dari index 3. Variabel sampun pada index 3 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali selama eksplorasi ini.



Gambar 4.24 Robot belok kiri 90° karena nilai index tetangga kiri lebih kecil

Setelah sampai pada sel dengan index 8, maka robot melakukan cek dinding dan mendapat informasi bahwa sisi kanan dan sisi kiri adalah dinding, maka sisi kanan dan sisi kiri diabaikan. Sisi belakang mempunyai index 3 dengan nilai 2, sisi depan mempunyai index 13 dengan nilai 0. Karena nilai index dari tetangga kiri lebih kecil maka robot maju ke index 13 dan menjadikan index 13 sebagai tetangga terkecil dari index 8. Variabel sampun pada index 8 diberi nilai +1 yang menandakan sel ini telah dilewati 1 kali selama eksplorasi ini.

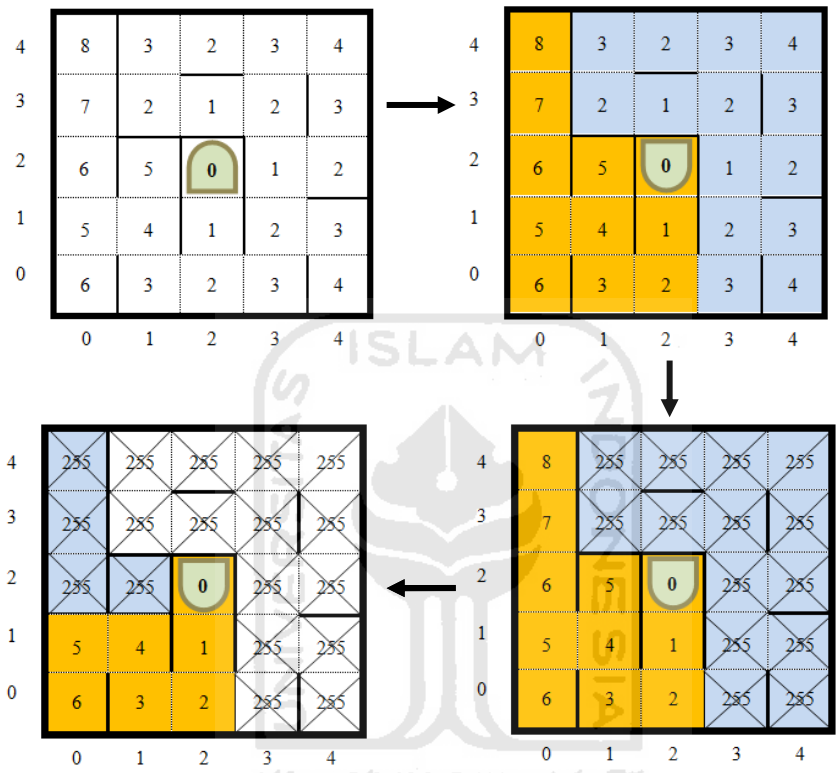


Gambar 4.25 Robot bergerak maju menuju sel dengan index 13

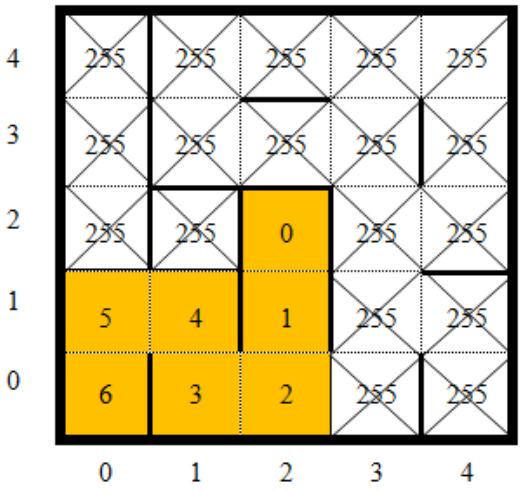
Setelah sampai pada sel dengan Index 13, maka robot menemukan bahwa nilai index 13 adalah 0, maka *GOAL* telah ditemukan, tetapi setelah melakukan cek dinding robot mendapat informasi bahwa di 3 sisinya terdapat dinding atau robot menumukan jalan buntu, maka robot langsung berputar 180°. Setelah berputar, karena *GOAL* telah ditemukan maka robot pun segera melakkan pemblokiran jalur pada peta.

Semua jalur yang belum dilalui yang ditandai dengan nilai sampun masih berisi 0, maka sel dengan nilai sampun = 0 akan diisi dengan nilai 255. Selanjutnya sel yang mempunyai nilai 1 pada variabel Jalan_buntu nya juga diberi

nilai 255 sehingga didapat peta jalur terpendek dari lapangan yang baru saja di eksplorasi oleh robot. Variabel Sampun_GOAL juda diisi dengan nilai 1 yang menandakan robot akan masuk pada mode perjalanan dari *HOME* menuju *GOAL*.



Gambar 4.26 Robot menemukan *GOAL* dan membuat peta jalur terpendek

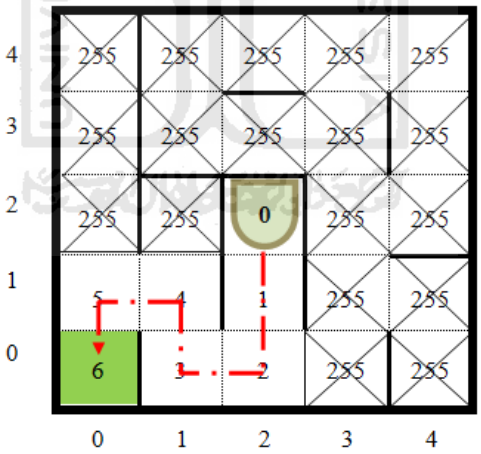


Gambar 4.27 Peta jalur terpendek hasil dari metode algoritma *flood-fill*

4.4 Pengujian Robot dalam Melakukan Perjalanan dari *GOAL* ke *HOME*

Pada saat robot melakukan perjalanan pulang dari *GOAL* ke *HOME*, robot akan menerapkan algoritma *flood-fill* dengan cara yang berbeda dari saat eksplorasi dilakukan, yaitu robot akan mengikuti sel yang mempunyai nomor lebih besar 1 dari sel robot sekarang. Robot akan terus berpindah 1 sel dan hanya akan menuju ke sel yang lebih besar 1 dari sel sekarang, sampai robot berada di sel yang mempunyai koordinat (x,y) yang sama dengan sel awal tempat robot memulai eksplorasi (*HOME*).

Pada saat robot melakukan perjalanan pulang dari *GOAL* ke *HOME* ini, robot hanya mengandalkan peta jalur terpendek yang telah dibuat setelah lapangan berhasil di eksplorasi, sehingga tidak akan ada proses *re-flood* pada mode ini.



Gambar 4.28 Jalur perjalanan pulang robot dari *GOAL* ke *HOME*

Ketika robot telah kembali menemukan *HOME*, maka robot akan mengisi Sudah_ingat dengan nilai 1 yang menandakan mode hapalan sudah bisa digunakan. Robot juga mengisi nilai variabel Sampun_GOAL dengan nilai 2 yang

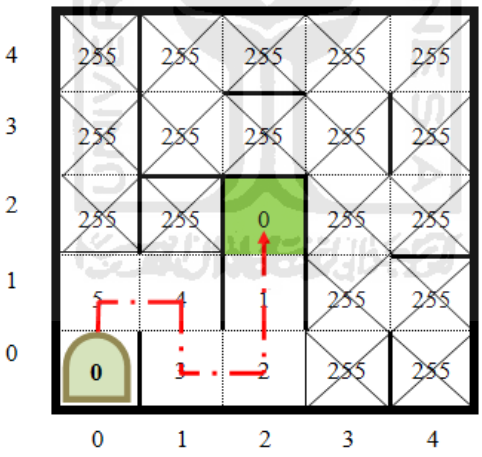
menandakan robot akan masuk pada mode perjalanan hapalan dari *GOAL* menuju *HOME*. Berikut perjalanan robot dari *HOME* menuju *GOAL*:

Gambar 4.29 Rangkaian Perjalanan robot dari *GOAL* menuju *HOME*

4.5 Pengujian Robot dalam Mode Hapalan

Pada saat robot menggunakan mode hapalan, maka robot akan menerapkan algoritma *flood-fill* dengan cara awal seperti saat eksplorasi dilakukan, yaitu robot akan mengikuti sel yang mempunyai nomor lebih kecil 1 dari sel robot sekarang. Robot akan terus berpindah 1 sel dan hanya akan menuju ke sel yang lebih kecil 1 dari sel sekarang, sampai robot berada di sel yang mempunyai nilai index = 0 (*GOAL*).

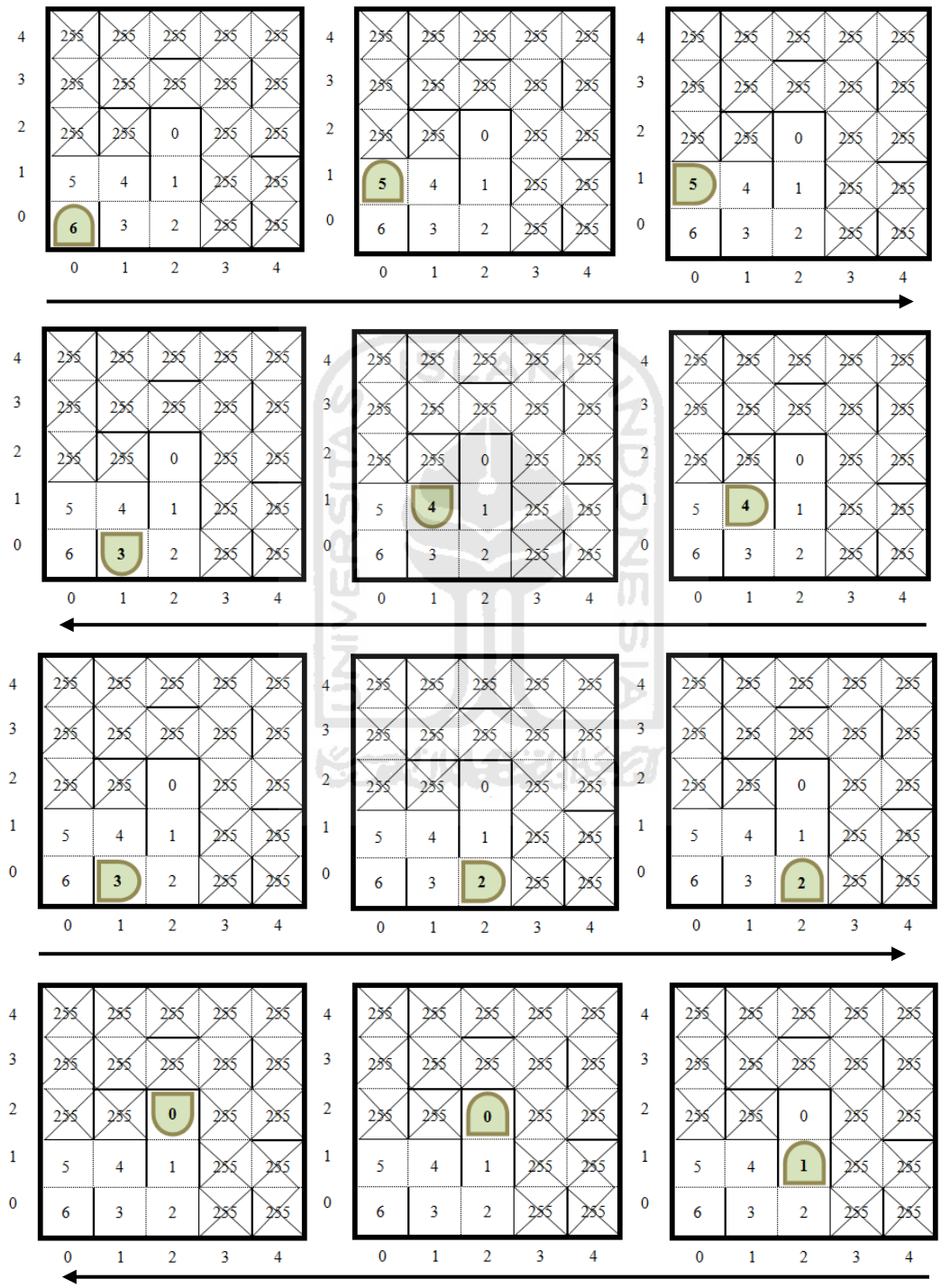
Pada mode hapalan, robot hanya mengandalkan peta jalur terpendek yang telah dibuat setelah lapangan berhasil di eksplorasi, sehingga tidak akan ada proses *re-flood* pada mode ini.



Gambar 4.30 Jalur perjalanan robot dari *HOME* ke *GOAL* pada mode hapalan

Ketika robot telah kembali menemukan *GOAL*, maka robot akan mengisi kembali mengisi nilai variabel Sampun_GOAL dengan nilai 1 yang menandakan robot akan kembali masuk pada mode perjalanan dari *HOME* menuju *GOAL*.

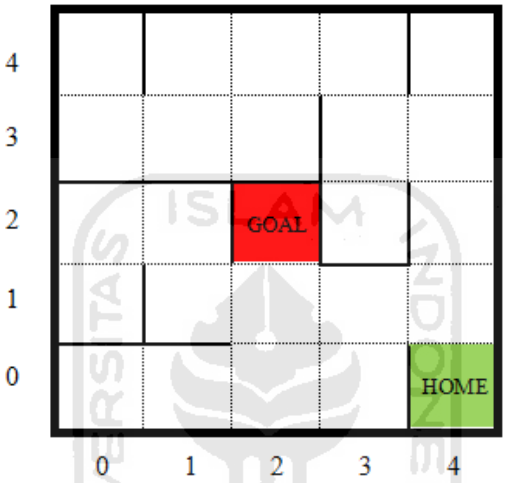
Serangkaian perjalanan robot dari *HOME* menuju *GOAL* pada mode hapalan ini dapat dilihat pada gambar 4.31 berikut:



Gambar 4.31 Rangkaian Perjalanan robot pada mode hapalan

4.6 Contoh Kasus yang Tidak Berhasil dipecahkan dengan Baik

Walaupun sudah bisa membuat peta jalur terpendek setelah melakukan eksplorasi pada beberapa lapangan, masih ada beberapa lapangan yang belum bisa dibuat jalur terpendeknya oleh robot micromouse selama ujicoba. Diantaranya lapangan pada gambar 4.32 berikut:



Gambar 4.32 Contoh lapangan yang belum bisa dibuat jalur terpendeknya

Pada lapangan ini, robot melakukan *START* dari koordinat (4,4). Sesuai peta buta yang telah diberikan terlebih dahulu, pada ujicoba robot melakukan eksplorasi keatas terlebih dahulu sampai ke sel dengan koordinat (4,4). Karena menemui jalan buntu, maka robot berputar dan maju 1 sel lalu berbelok ke kanan, begitu seterusnya sampai semua sel tereksplorasi yang berujung pada sel (0,4). Karena pada sel dengan koordinat (0,4) robot menemui jalan buntu dan semua sel diatas telah dieksplorasi tetapi belum menemukan *GOAL* juga, maka robot kembali ke sel didepan *HOME* untuk kembali mengeksplorasi sel - sel yang terdapat pada bagian kiri dari *HOME*. Begitu seterusnya sampai *GOAL* ditemukan.

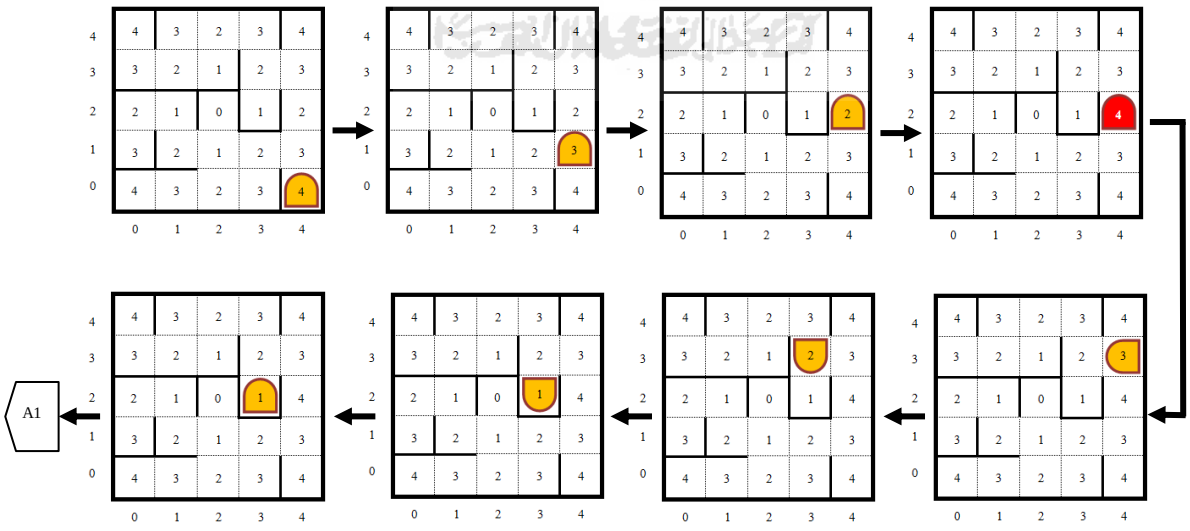
Kesalahan ini terletak pada saat robot mengambil keputusan pada mode eksplorasi setelah GOAL ditemukan. Gagalnya robot membuat peta jalur terpendek dari robot ini, disebabkan kurangnya parameter-parameter dan informasi yang diberikan kepada robot terhadap masalah – masalah yang mungkin timbul (tidak terduga) seperti pada kasus ini terdapat beberapa sel yang tidak mempunyai dinding di salah satu sisinya, seperti sel pada koordinat (1,3), (1,4) , (2,3) dan (2,4) juga sel pada koordinat (2,0), (2,1), (3,0) dan (3,1) yang mana kondisi ini belum penulis masukkan ke dalam robot pada saat pemetaan dilakukan karena pada awal penelitian ini telah ditentukan lapangan mana yang akan digunakan sehingga penulis hanya berkonsentrasi pada lapangan tersebut.

Perlu diperhatikan, pada penelitian ini Algoritma flood-fill hanya digunakan robot dalam pengambilan keputusan ketika robot berjalan menentukan sel mana yang akan menjadi tujuannya selama perjalanan, yaitu robot hanya bisa berjalan mengikuti sel dengan nomor terkecil atau mengikuti nomor terbesar khusus ketika robot berada pada mode pulang dari *GOAL* ke *HOME*. Untuk pembuatan peta jalur terpendeknya, robot hanya melakukan pemblokan jalur dari peta yang telah didapat ketika dilakukan eksplorasi dengan hanya mengandalkan persyaratan yang telah diberikan sebelumnya. Oleh sebab itu, jika kasus dari lintasan labirin ini dirubah, maka jika persyaratan yang sebelumnya tidak mampu untuk membuat jalur terpendek dari peta hasil eksplorasi yang dibuat, maka diperlukan tambahan persyaratan yang disesuaikan dengan kasus dari lintasan tersebut. Labirin pada gambar 4.32 merupakan contoh dari perlunya tambahan syarat untuk lintasan/kasus yang berbeda.

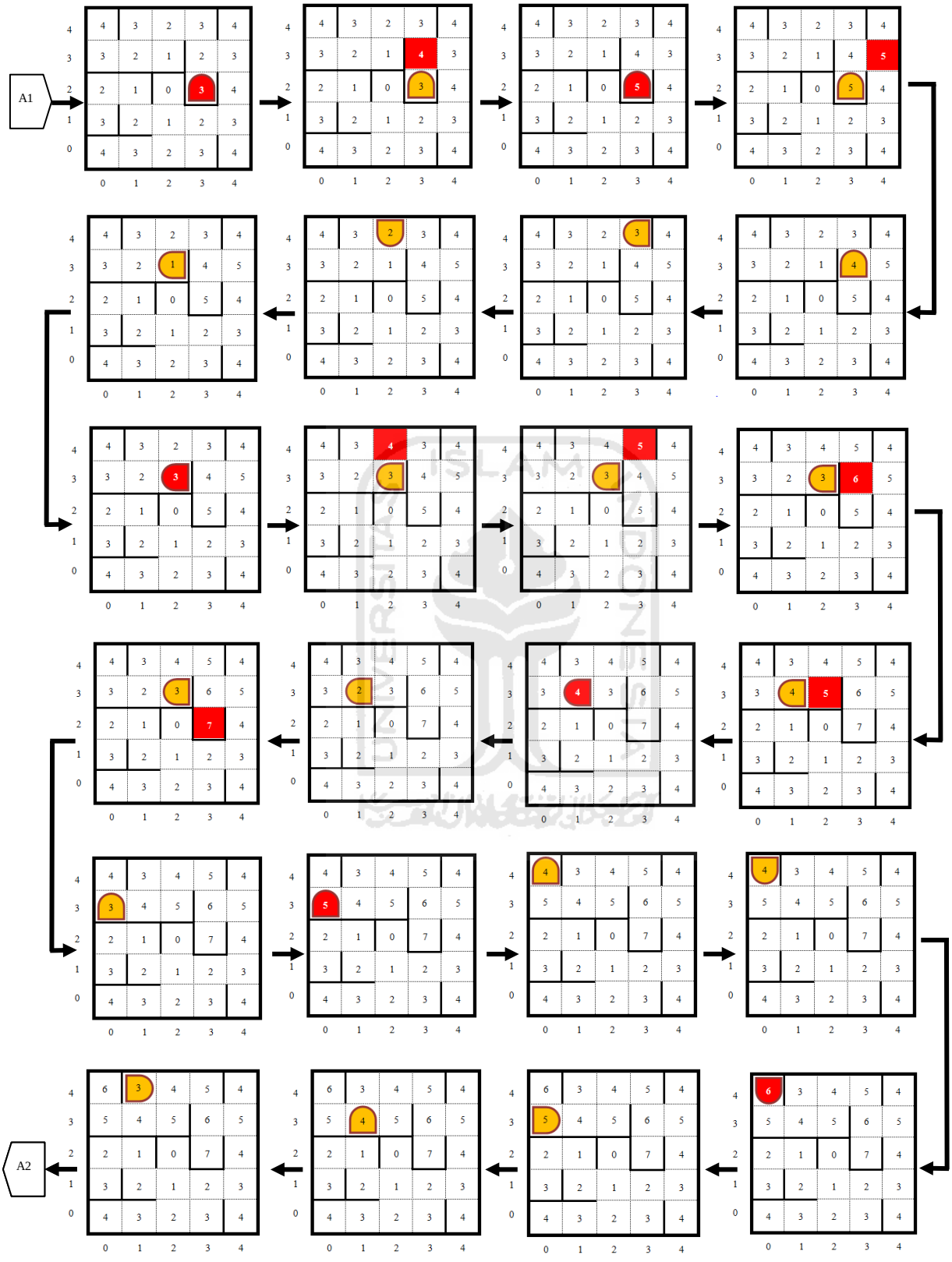
Pada kasus sebelumnya, yaitu pada lapangan seperti pada gambar 4.4, persyaratan yang penulis berikan untuk penentuan jalur terpendek dari peta yang dihasilkan pada saat eksplorasi adalah sebagai berikut;

1. Memblokir semua sel yang belum dilewati dengan mengisi nilai sel tersebut dengan nilai 255.
2. Memblokir semua sel yang mempunyai varibel jalan_buntu = 1 dengan mengisi sel tersebut dengan nilai 255.

Persyaratan diatas bekerja dengan baik pada kasus dengan bentuk lapangan seperti pada gambar 4.4, tetapi persyaratan tersebut masih kurang jika diimplementasikan pada kasus dengan bentuk lapangan seperti pada gambar 4.32 karena masih terdapat percabangan yang menyebabkan robot bingung dalam mengambil keputusan. Berikut penjelasan rangkaian perjalanan robot saat melakukan eksplorasi menuju sel tujuan:

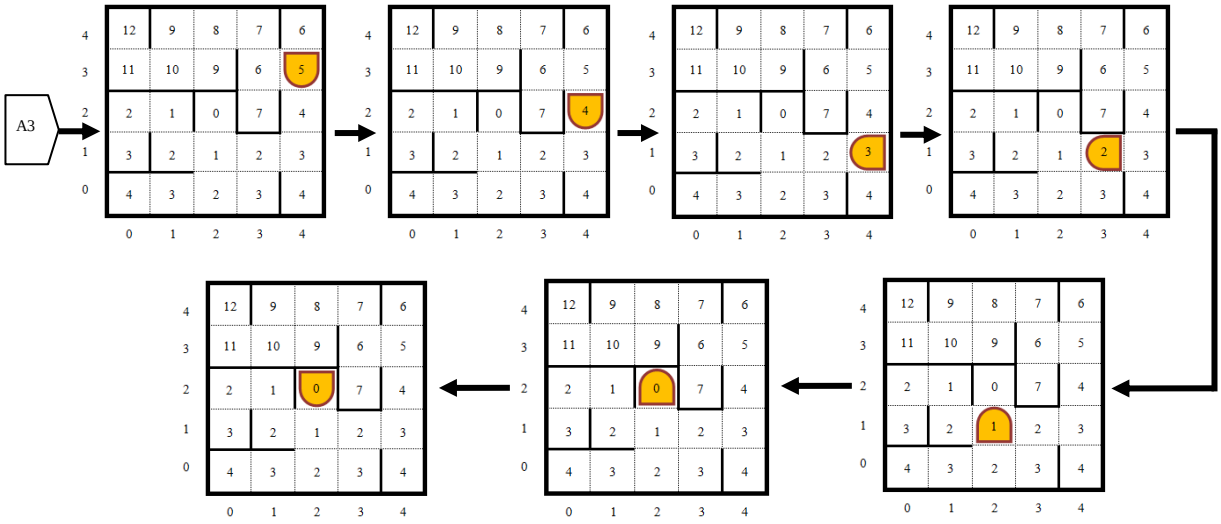


Gambar 4.33 Serangkaian perjalanan eksplorasi robot



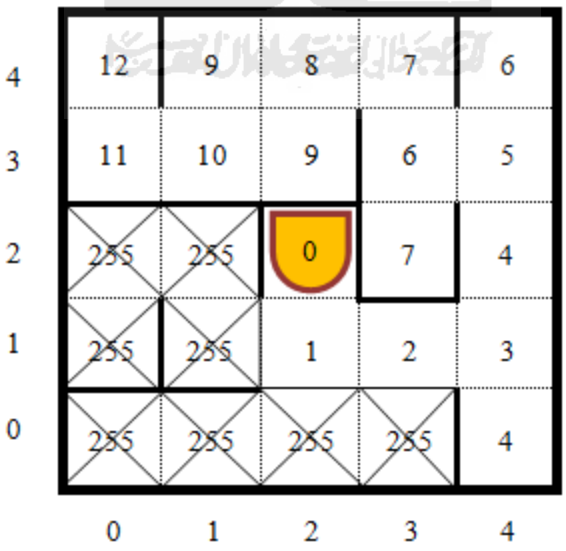
Gambar 4.34 Serangkaian perjalanan eksplorasi robot

Gambar 4.35 Serangkaian perjalanan eksplorasi robot



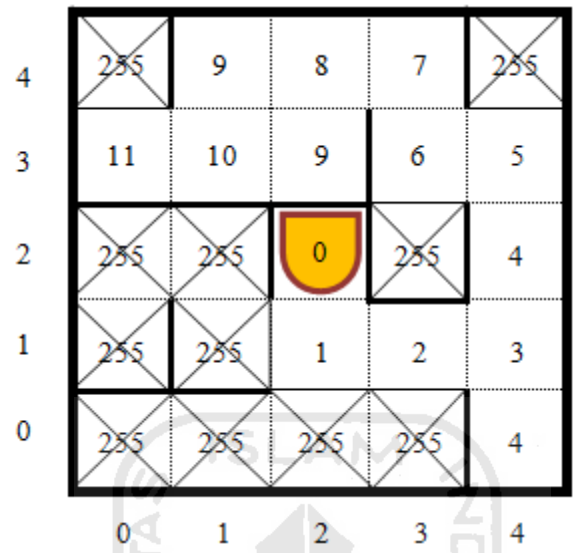
Gambar 4.36 Serangkaian perjalanan eksplorasi robot

Setelah *GOAL* ditemukan oleh robot, maka didapat peta jalur labirin dari hasil eksplorasi. Sesuai persyaratan yang telah penulis masukkan, maka robot akan segera melakukan pemblokiran pada semua sel yang belum pernah dilewati seperti pada gambar 4.37 berikut:



Gambar 4.37 Pemblokiran semua sel yang belum dilewati

Selanjutnya, semua sel yang merupakan jalan buntu juga dilakukan pemblokiran seperti pada gambar 4.38 berikut:

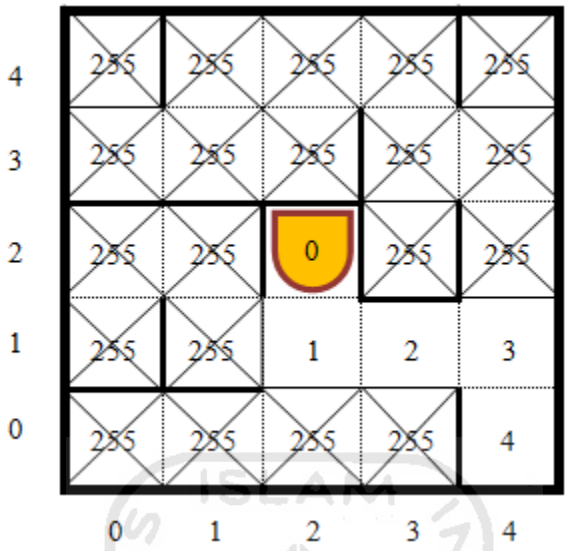


Gambar 4.38 Pemblokiran semua sel yang merupakan jalan buntu

Pada gambar diatas, dapat dilihat walaupun telah dilakukan pemblokiran sesuai dengan syarat yang telah penulis masukkan sebelumnya ternya masih terdapat percabangan pada sel dengan koordinat (4,1) yang menyebabkan robot akan bingung ketika akan melakukan perjalanan pulang dan ketika nantinya menjalankan mode hapalan. Oleh karena itu, untuk kasus seperti ini diperlukan tambahan syarat untuk pemblokiran sel agar didapat jalur terpendek dari labirin dengan menambahkan syarat berikut:

- Sel yang nilainya sama dengan sel awal dan juga yang nilainya lebih besar dari sel awal diberi nilai dengan 255.

Sehingga peta labirin akan menjadi seperti gambar 4.39 berikut:



Gambar 4.39 Peta jalur terpendek setelah dilakukan penambahan syarat

Jadi untuk beberapa kasus labirin, agar didapat jalur terpendek yang sebenarnya maka diperlukan penambahan asumsi persyaratan baru dalam pemblokiran selnya.

BAB V

PENUTUP

5.1 Kesimpulan

Selama melakukan pengujian eksplorasi pada beberapa lapangan, robot selalu berhasil menemukan *GOAL* dengan metode algoritma *flood-fill* ini. Tetapi, tidak semua lapangan yang peta jalur terpendeknya berhasil dibuat oleh robot dari peta buta yang telah diberikan, hal ini terjadi mungkin disebabkan ada beberapa kondisi yang ditemui robot di beberapa lapangan yang perlu tindakan khusus. Karena terdapat parameter-parameter tertentu yang dilihat robot dalam membuat peta yang telah dieksplorasi menjadi sebuah peta dengan jalur terpendek.

Berhubung dalam penelitian ini awalnya sudah ditentukan dahulu lapangan mana yang harus dibuat peta jalur terpendeknya, maka untuk kasus lapangan yang telah ditentukan dapat dinyatakan bahwa algoritma *flood-fill* dapat berfungsi dengan baik karena robot sudah mampu menemukan jalur terpendek dan membuat peta terpendeknya. Penentuan lapangan ini mengacu pada jika kita ingin mengikuti sebuah lomba, lapangannya tentu sudah diberi tahu terlebih dahulu, sehingga ada kemungkinan jika pada lapangan lain ada kondisi yang tidak terpenuhi yang dapat menyebabkan algoritma flood-fill belum bisa menunjukkan performa terbaiknya dalam membuat peta jalur terpendek karena pentingnya dilakukan pembelajaran lapangan terlebih dahulu sehingga kita bisa memberikan

banyak informasi kepada robot guna tercapainya metode algoritma *flood-fill* yang baik dalam pemecahan masalah jalur terpendek ini.

Penuhnya memori pada ATMEGA32 ini menjadi salah satu penghambat penulis dalam melakukan penelitian ini, sehingga program harus diringkas sedemikian rupa agar memori program yang digunakan dapat berkurang. Keterbatasan memori ini juga menjadi hambatan penulis dalam usaha ingin semakin menyempurnakan metode algoritma *flood-fill* ini karena tidak bisa menambah banyak kondisi dan informasi lagi ke dalam robot.

Pemakaian metode *wall follower* ketika robot berjalan dari sel satu ke sel lainnya sangat membantu demi optimalnya algoritma *flood-fill* ini, karena jika tidak menggunakan metode *wall follower* ketika robot berjalan maka robot dapat berbelok menabrak dinding labirin dan juga robot belum berpindah 1 sel ketika langkah roda kanan dan roda kiri sudah mencapai nilai 24. Hal ini disebabkan kecepatan kedua motor DC yang digunakan robot berbeda, padahal jenis dan tipe motor DC-nya sama.

5.2 Saran

Berikut ini merupakan saran - saran yang dapat penulis berikan guna membantu mengembangkan robot micromouse dan metode algoritma *flood-fill* ini menjadi lebih baik kedepannya:

1. Penggunaan motor DC yang dilengkapi dengan *encoder* langsung didalamnya (motor *rotary encoder*) atau penggunaan motor *magnetic encoder* akan sangat membantu dalam menambah ke-presisi-an robot

dalam berjalan lurus dari 1 sel ke sel lainnya, sehingga algoritma *flood-fill* bisa menjadi semakin optimal.

2. Penggunaan mikrokontroler yang mempunyai memori besar (seperti; ATMEGA64, ATMEGA128) sangat diperlukan karena banyak memori yang digunakan dalam pembuatan program algoritma *flood-fill* ini, hal ini disebabkan banyaknya kondisi dan informasi yang harus kita masukkan agar robot dapat mengimplementasikan algoritma *flood-fill* dengan optimal guna mendapatkan peta jalr terpendek dari suatu kasus lapangan.
3. Penggunaan sensor kompas sepertinya diperlukan pada robot micromouse ini agar robot dapat melakukan putaran dengan presisi, seperti putaran 90° dan putaran 180° ketika robot menemui jalan buntu.
4. Penambahan sensor sudut (45°) diperlukan dalam pengembangan robot ini agar dapat menambah kondisi yang diperoleh robot, dan memungkinkan robot bergerak diagonal sehingga waktu tempuh bisa semakin cepat.
5. Robot micomouse pada penelitian ini menggunakan 3 sensor SHARP GP2D120, masing - masing pada sisi depan, sisi kanan dan sisi kiri robot. Penambahan 1 lagi sensor SHARP GP2D120 pada sisi belakang sangat diperlukan guna mengetahui apakah ada nilai index dari tetangga belakag dan apakah dibelakang terdapat dinding atau tidak.
6. Untuk beberapa kasus labirin, agar didapat jalur terpendek yang sebenarnya maka diperlukan penambahan asumsi persyaratan baru dalam pemblokiran selnya.

DAFTAR PUSTAKA

Abdullah M N Rahman, Akhmad Hendriawan dan Reesa Akbar, 2010. *Penerapan Algoritma Flood Fill Untuk Menyelesaikan Maze Pada Line Follower Robot.* Skripsi, tidak diterbitkan. Surabaya : Jurusan Teknik Elektronika PENS-ITS.

Ananian, S. C., & Humphreys, Greg., 2003. *Theseus : A Maze-Solving Robot*. PAPER, published. Department of Electrical Engineering at Princeton University.

Andrianto, Heri., 2008. *Pemrograman Mikrokontroler AVR Atmega16 Menggunakan Bahasa C (CodeVision AVR)*. Bandung : Penerbit INFORMATIKA. ISBN : 978-979-1153-41-6.

Anita, Nur Syadfitri., 2010. *Robot Micromouse Dengan Menggunakan Algoritma Depth-First Search.* Skripsi, tidak diterbitkan. Depok : Fakultas Ilmu komputer Universitas Gunadharma.

Bekti, Samudera Harapan., 2009. *Pencarian Shorthest Path Dinamik Dengan Menggunakan Algoritma Bellman-Based Flood-Fill Dan Implementasinya Pada Robot Micromouse.* Makalah IF2091 Struktur Diskrit Tahun 2009. Bandung : Program Studi Teknik Informatika ITB.

Budiharto, Widodo., 2006. *Belajar Sendiri Membuat Robot Cerdas*. Jakarta : Penerbit PT Elex Media Komputindo. ISBN : 979-20-9178-5.

De, Tondra., & Hall, Drew., 2004. *A Detailed Design And Analysis Of Micromouse*. Honors Thesis I. University of Nevada, Las Vegas.

Pitowarno, Endra., 2006. *Robotika Desain Kontrol, Dan Kecerdasan Buatan*. Yogyakarta : Penerbit Andi Offset Yogyakarta.

Susilo, Deddy., 2010. *48 Jam Kupas Tuntas Mikrokontroler MCS51 & AVR*. Yogyakarta : Penerbit ANDI. ISBN : 978-979-29-1346-0.

Thiang., Khoswanto, Hendry., Pasila, Felix., Ninaber, Ferdi., & Thelly, Hendra., 2009. *Implementasi Metode Steepest Ascent Hill Climbing Pada Mikrokontroller MCS51 Untuk Robot Mobil Pencari Rute Terdekat*. Surabaya: PAPER SNATI.

Willardson, D. M., 2001. *Analysis Of Micromouse Maze Solving Algorithms*. ECE 557 : Learning from Data, Spring 2001.

http://www.micromouseonline.com/book/micromouse-book

http://madan.wordpress.com/2006/07/24/micromouse-maze-solving-algorithm/

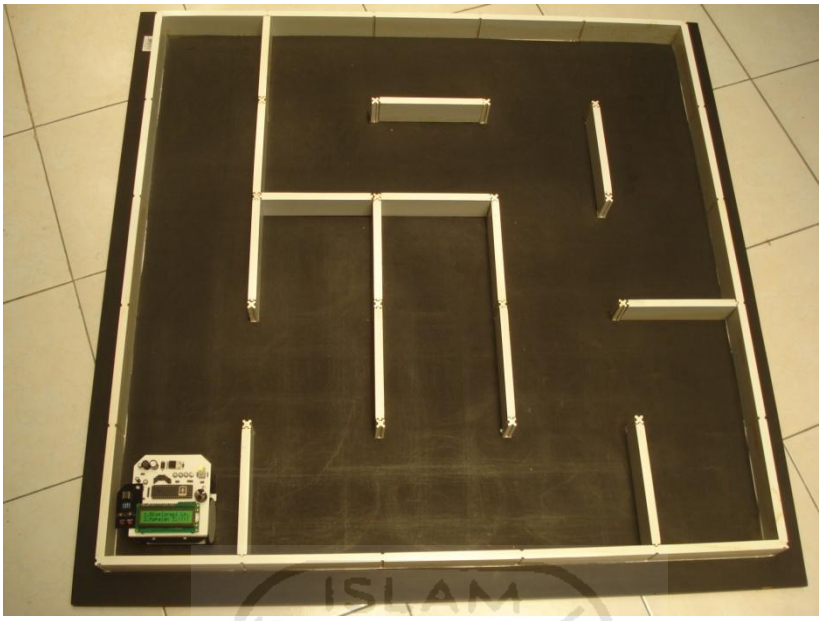
http://www.codecodex.com/wiki/Implementing_the_flood_fill_algorithm#Micromouse

http://www.societyofrobots.com/member_tutorials/node/94

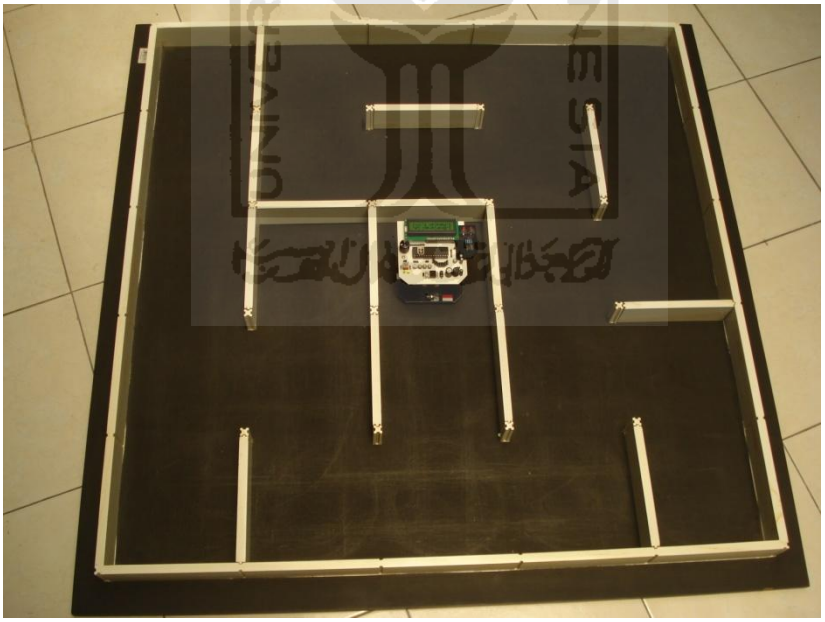
http://micromouse.cs.rhul.ac.uk/mtech/rules_main.shtml

http://www.micromouseinfo.com/introduction/mfloodfill.html

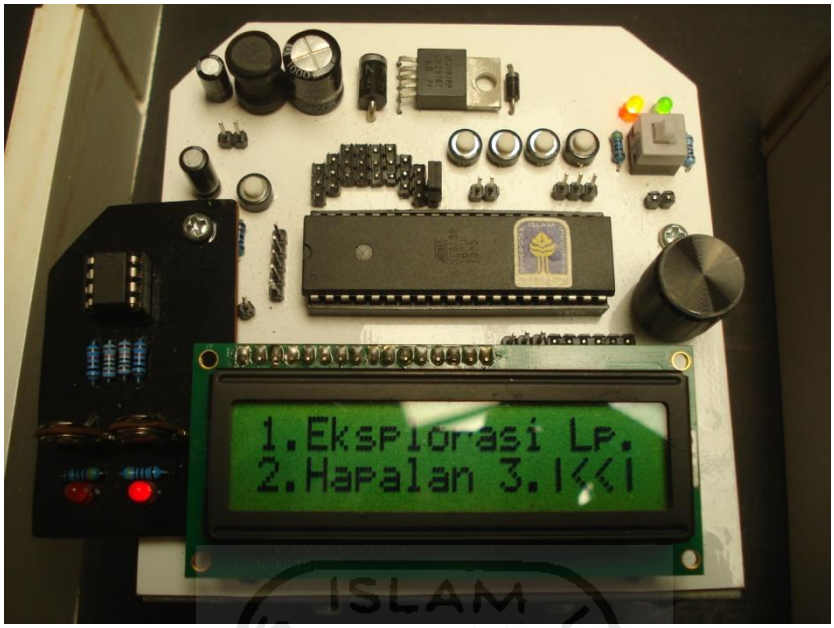
http://feri-klik.blogspot.com/2011/05/teknik-pwm-pulse-width-modulation.html



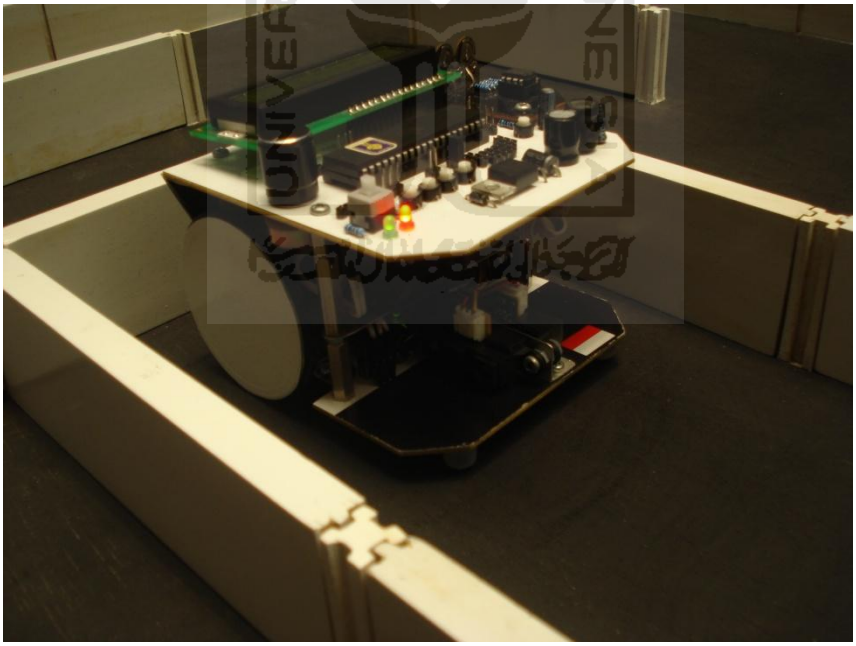
A. Robot pada posisi *HOME*



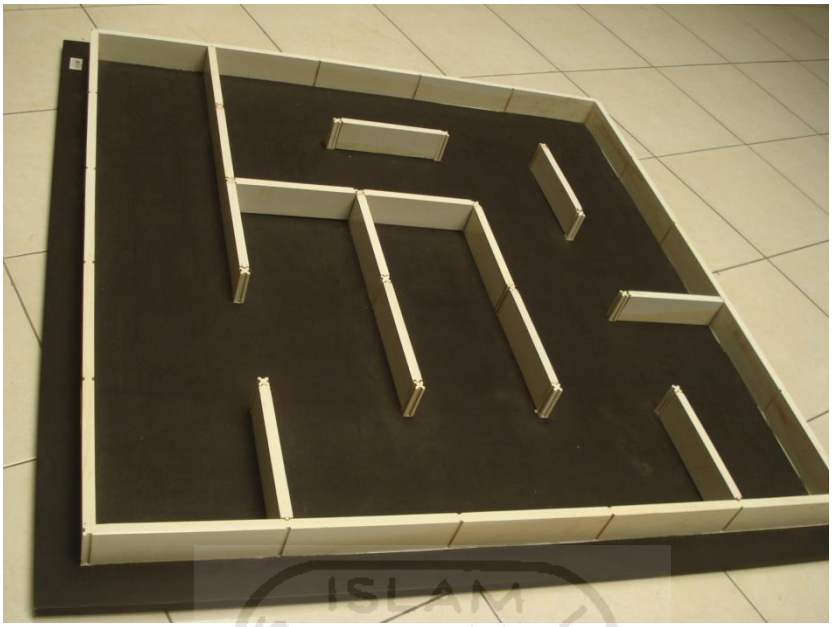
B. Robot pada posisi *GOAL*



C. Tampilan Menu Robot



D. Robot didalam Labirin



E. Labirin yang digunakan berukuran 5x5