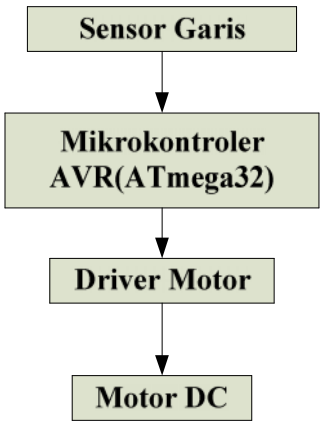
# BAB III

# PERANCANGAN SISTEM

## 3.1. Diagram Blok Sistem

Bab ini menjelaskan mengenai komponen yang membangun suatu sistem kendali dan algoritma *mobile robot* untuk studi kasus *maze solving* berdasarkan landasan teori. Gambar 3.1 merupakan gambar diagram blok keseluruhan dari sistem.




Gambar 3.1 Diagram Blok Sistem

Fungsi dan cara kerja masing-masing blok sistem:

1. Blok Sensor Garis: blok ini berfungsi mendeteksi garis yang berwarna hitam atau putih. Pada blok ini terdiri dari 2 komponen yang berfungsi sebagai pemancar dan penerima. Komponen yang berfungsi sebagai pemancar yaitu led *superbright*, dan komponen yang berfungsi sebagai penerima yaitu phototransistor. Cara kerjanya yaitu led akan di pancarkan ke garis hitam atau putih selanjutnya cahaya yang dipantulkan tersebut akan diterima oleh phototransistor, ketika led dipancarkan ke garis hitam maka cahaya akan diserap sedangkan pada garis putih cahaya akan dipantulkan.
2. Blok Mikrokontroler: blok ini berfungsi sebagai kendali yang akan mengatur jalannya *mobile* robot. Cara kerjanya yaitu mikrokontroler akan mendapatkan data dari sensor garis berupa sinyal analog selanjutnya data tersebut akan di konfersi ke sinyal digital melalui fitur ADC (*Analog to Digital Converter*)

*internal* yang terdapat di mikrokontroler. Setelah dikonversi ke sinyal digital, mikrokontroler akan memerintahkan *driver* motor untuk menggerakan motor DC melalui bahasa pemrograman tingkat tinggi yang menyebabkan robot bergerak.

3. Blok *Driver* Motor: blok ini berfungsi untuk mengatur arah putaran motor DC, baik secara *one-directional* (satu arah) ataupun *bidirectional* (dua arah/bolak-balik). Cara kerjanya yaitu motor akan berputar searah jarum jam ataupun kebalikannya tergantung dari intruksi yang diberikan mikrokontroler dengan cara memberikan logika *high* atau *low* pada pin IC *driver* motor.
4. Blok Motor DC: blok ini berfungsi untuk memutarkan roda pada *mobile robot*.

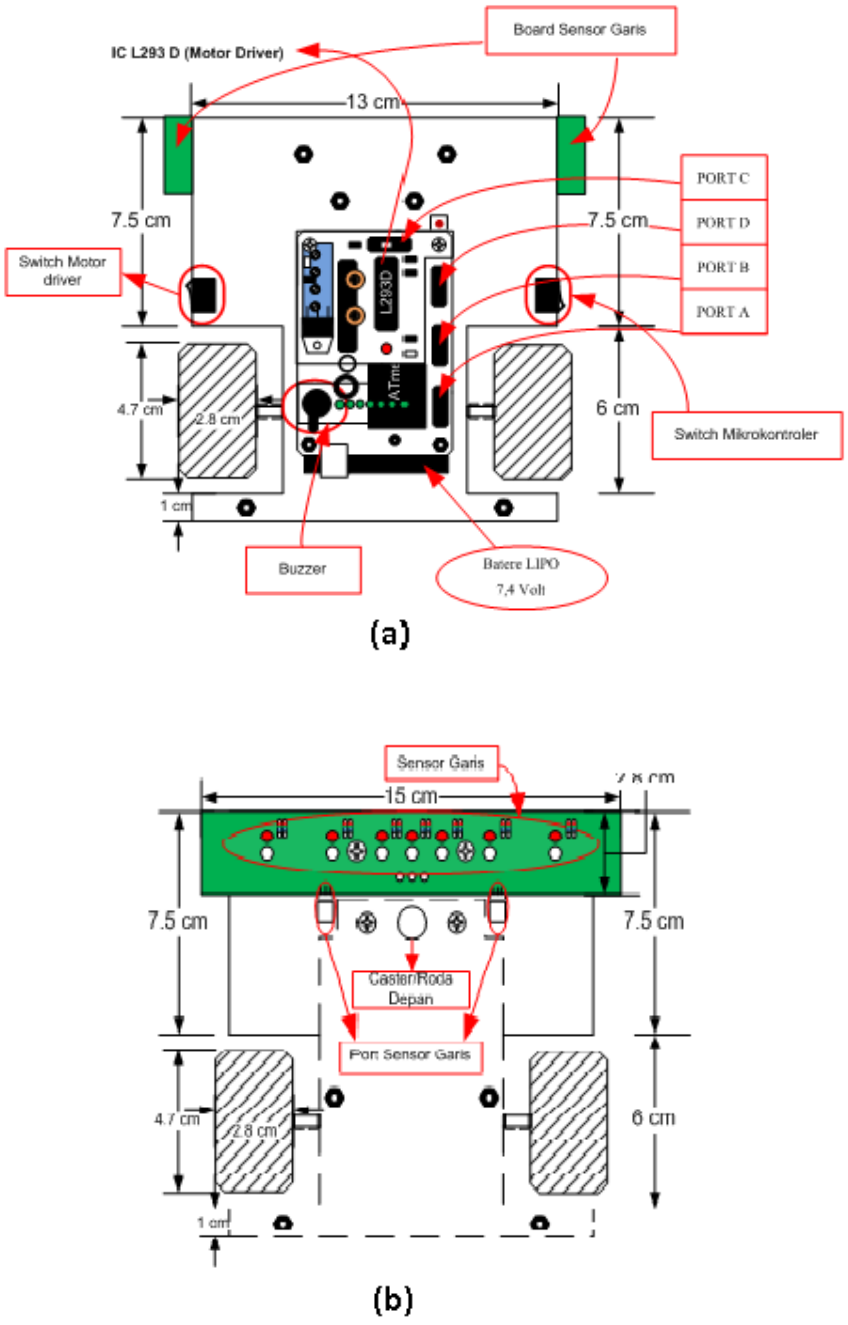
## 3.2. Perangkat Keras (Hardware)

### 3.3.1. Sistem Mekanik Robot

Berikut ini adalah robot yang dipakai dengan spesifikasi sebagai berikut:

1. Jenis robot merupakan robot beroda yang memiliki 1 buah roda *caster* di depan dan 2 buah roda belakang yang digerakkan dengan 2 motor dc di roda bagian kiri dan kanan.
2. *Body* robot berdimensi 12,2 cm x13 cm yang terbuat dari akrilik.
3. Sensor garis dipasang di bawah bagian depan.
4. Roda belakang terbuat dari karet dan berdimensi diameter 4,5 cm, lebar 3 cm.

Pada gambar 3.2 memperlihatkan bentuk desain robot yang akan dibuat.




Gambar 3.2 (a) Robot Tampak Atas (b) Robot Tampak Bawah

### 3.3.2. Mikrokontroler (ATmega32)

Mikrokontroler yang digunakan adalah mikrokontroler jenis *AVR* Atmega32 dengan dilengkapi 32 *pin* yang bisa digunakan sebagi *input* atau pin *output*, *pin* tersebut dibagi menjadi 4 port yaitu

1. Port A terdiri dari A.0 sampai A.7
2. Port B terdiri dari B.0 sampai B.7
3. Port C terdiri dari C.0 sampai C.7
4. Port D terdiri dari D.0 sampai D.7

Port yang digunakan pada perancangan sistem ini adalah port A, port B, Port D.

Pemilihan mikrokontroler jenis ini didasari kemampuannya yang cukup handal, permrograman yang tidak terlalu sulit dan harganya yang relatif tidak terlalu mahal. Konfigurasi *port* yang digunakan dapat dilihat pada tabel 3.2.

Tabel 3.1 Konfigurasi port yang digunakan pada ATmega32

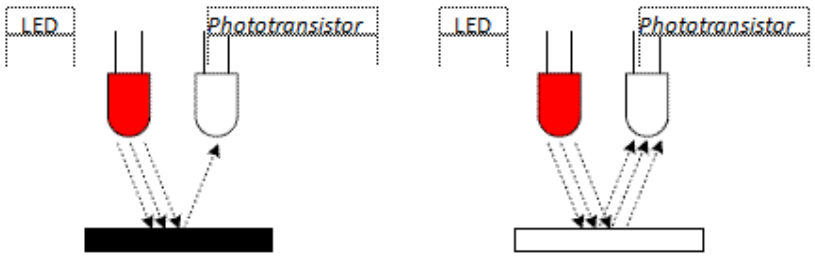
| Nama Pin | No. Pin | Fungsi | Keterangan |
|---|---|---|---|
| PORT A | 0 | Input / ADC 0 | Sensor garis 1 |
| PORT A | 1 | Input / ADC 1 | Sensor garis 2 |
| PORT A | 2 | Input / ADC 2 | Sensor garis 3 |
| PORT A | 3 | Input / ADC 3 | Sensor garis 4 |
| PORT A | 4 | Input / ADC 4 | Sensor garis 5 |
| PORT A | 5 | Input / ADC 5 | Sensor garis 6 |
| PORT B | 2 | Output | Driver motor (A +) |
| PORT B | 3 | Output | Driver motor (A -) |
| PORT B | 4 | Output | Driver motor (EN B) |
| PORT B | 5 | Output | Driver motor (EN A) |
| PORT D | 6 | Output | Driver motor (B +) |

| | | | |
|---|---|---|---|
| PORT D | 7 | Output | Driver motor (B -) |

### 3.3.3. Sensor Garis

Sensor garis berfungsi untuk mendeteksi warna dari permukaan yang berada dibawah robot *line follower* dengan maksud agar sensor garis ini dapat mengasilkan logika posisi dari robot *line follower* terhadap garis tepat berada dibawah robot.

Logika posisi yang dihasilkan oleh sensor garis ini kemudian akan dijadikan *input* ke mikrokontroler pada robot. Pada sensor garis, komponen yang digunakan yaitu *phototransistor* sebagai pendeteksi garis hitam dengan dasar putih yang menjadi jalur robot dan LED ( *Light Emitting Diode* ) *superbright* sebagai pemancar cahaya ke lantai yang kemudian dipantulkan dan diterima oleh *phototransistor*. Untuk lebih jelasnya memahami prinsip kerja sensor garis dapat dilihat pada gambar 3.5.

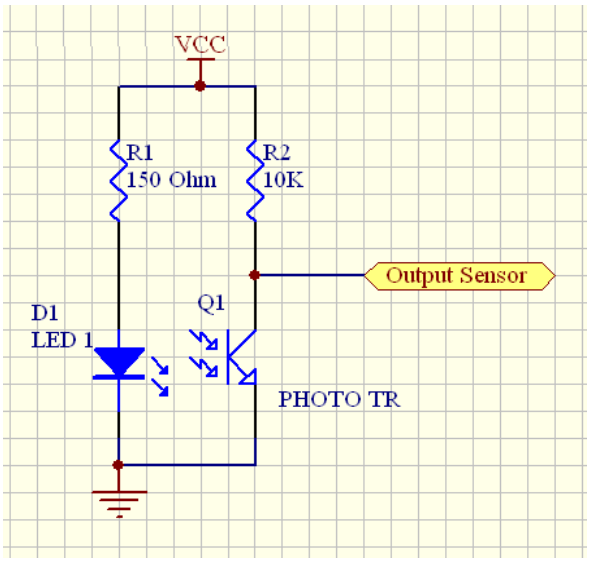



Gambar 3.3 Prinsip Kerja Sensor Garis

*Phototransistor* bekerja dengan cara menangkap emisi cahaya yang dikeluarkan oleh LED. Prinsip kerja dari *phototransistor* adalah ketika basis menangkap cahaya maka *collector* akan terhubung dengan *emitter* dalam hal ini transistor bekerja. *Phototransistor* memiliki *mode* aktif artinya transistor akan menghasilkan reaksi yang sebanding dengan besaran cahaya yang diterima sampai dengan tingkatan tertentu.

*Output* dari *phototransistor* yaitu berupa tegangan. Semakin besar intensitas cahaya yang didapat maka makin besar pula tegangan yang dihasilkan. Dengan memanfaatkan fitur internal ADC (*Analog to Digital*

*Converter*) yang terdapat pada IC ATmega32, sinyal *analog* yang berasal dari sensor garis dapat dibubah menjadi sinyal *digital*. Hal ini dilakukan karena mikrokontroler hanya bisa bekerja dengan data *digital*. ADC tersebut terdiri dari 8 *chanel* yang terletak pada *port* A0 sampai *port* A7 pada IC ATmega32. Pada gambar 3.4 yang memperlihatkan rangkaian sensor garis yang digunakan.




Gambar 3.4 Rangkaian Sensor Garis

Jumlah sensor yang digunakan adalah 7 sensor yang masing-masing output sensornya dihubungkan ke port A pada mikrokontroler. Tegangan yang menuju led dihubungkan dengan resistor 150 Ω yang dihubungkan dari Vcc dan kaki anoda led, pemilihan resistor tersebut bertujuan untuk menghasilkan intensitas cahaya yang cukup yang nantinya cahaya tersebut akan ditangkap oleh phototransistor. Untuk mendapatkan nilai resistor pada led dapat menggunakan rumus 3.1.

$$R_{LED} = \frac{V_{in} - V_f}{I_f} \quad \text{.....3.1 [4, 11]}$$

$$R_{LED} = \frac{5\ Volt - 0{,}5 Volt}{30\ mA}$$

$$R_{LED} = \frac{4{,}5\ Volt}{30\ mA}$$

$$R_{LED} = 150\ \Omega$$

Keterangan:

$R_{LED}$ = Resistor LED

$V_{in}$ = Tegangan input

$V_f$ = Tegangan pada saat forward bias (0,5 Volt)

$I_f$ = Arus pada saat forward bias (dapat dilihat di datasheet)

Selain itu, tegangan juga dihubungkan ke resistor 10 KΩ yang selanjutnya dihubungkan ke phototransistor, fungsi resistor tersebut yaitu sebagai penghambat arus yang akan mengalir ke mikrokontroler pada saat kondisi *phototransistor OFF* (tidak terkena cahaya) sehingga didapat logika *high* yang tepat pada keluaran sensor ($V_{ce}$) yaitu sebesar 4 volt dan tegangan ini akan dikonversi oleh ADC yang terdapat pada mikrokontroler. Untuk mendapatkan nilai resistor pada led dapat menggunakan rumus 3.2

$$R_{Phototransistor} = \frac{V_{in} - V_{CE}}{I_C} \quad \text{..... 3.2 [4, 10]}$$

$$10\ K\Omega = \frac{5\ Volt - V_{CE}}{100\ \mu A}$$

$$5\ Volt - V_{CE} = 10\ K\Omega \times 100\ \mu A$$

$$5\ Volt - V_{CE} = 1\ Volt$$

$$V_{CE} = 5\ Volt \quad 1\ Volt$$

$$V_{CE} = 4\ Volt$$

Keterangan:

$R_{Phototransistor}$ = Resistor Phototransistor

$V_{in}$ = Tegangan input

$V_{CE}$ = Tegangan *collector emiter*

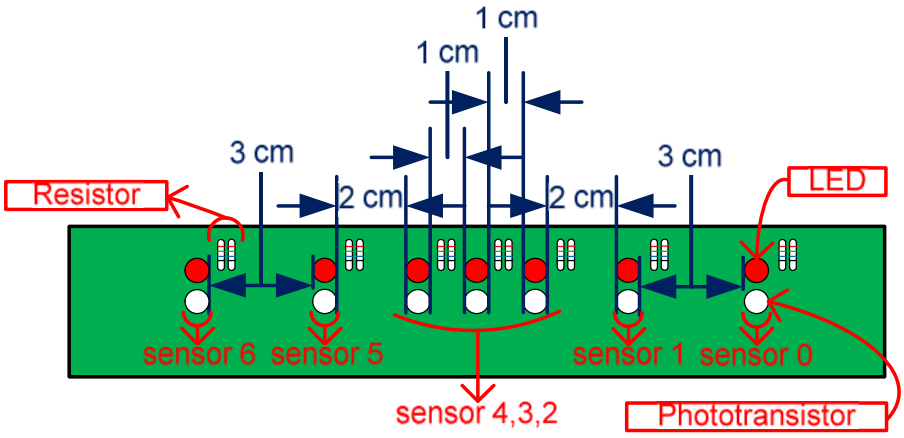
$I_C$ = Arus *collector* (dapat dilihat di datasheet)

Tujuh sensor yang digunakan memiliki fungsi masing-masing, yaitu:

1. Sensor 0 dan sensor 1 berfungsi untuk mendeteksi adanya belok kanan.
2. Sensor 2, sensor 3, sensor 4 berfungsi untuk mendeteksi garis lurus.

3. Sensor 5 dan sensor 6 berfungsi untuk mendeteksi adanya belok kiri.

Jarak antar sensor pun tidak sama, hal ini bertujuan agar memudahkan dalam mengenali bentuk persimpangan. Untuk lebih jelasnya dapat dilihat pada gambar 3.5.




Gambar 3.5 Design Bord Sensor Garis

### 3.3.4. Aktuator (Penggerak Robot)

#### 3.3.3.1. Driver Motor

*Driver* motor berfungsi untuk mengatur kecepatan dan arah putaran motor DC. IC *driver* motor yang digunakan adalah IC L293D, pemilihan IC jenis ini selain karena ukurannya yang lebih kecil dan lebih murah, IC L293D juga berisi empat buah *driver-H* berarus tinggi. IC ini didesain guna menyediakan pengaturan (*driver*) arus listrik secara dua arah (*bidirectional*) hingga mencapai lebih dari 1 A pada tegangan dari 4,5 V sampai dengan 36 V. Selain itu, IC L293D juga didesain untuk mengendalikan beban induktif seperti relay, solenoid, motor DC, dan motor stepper bipolar.

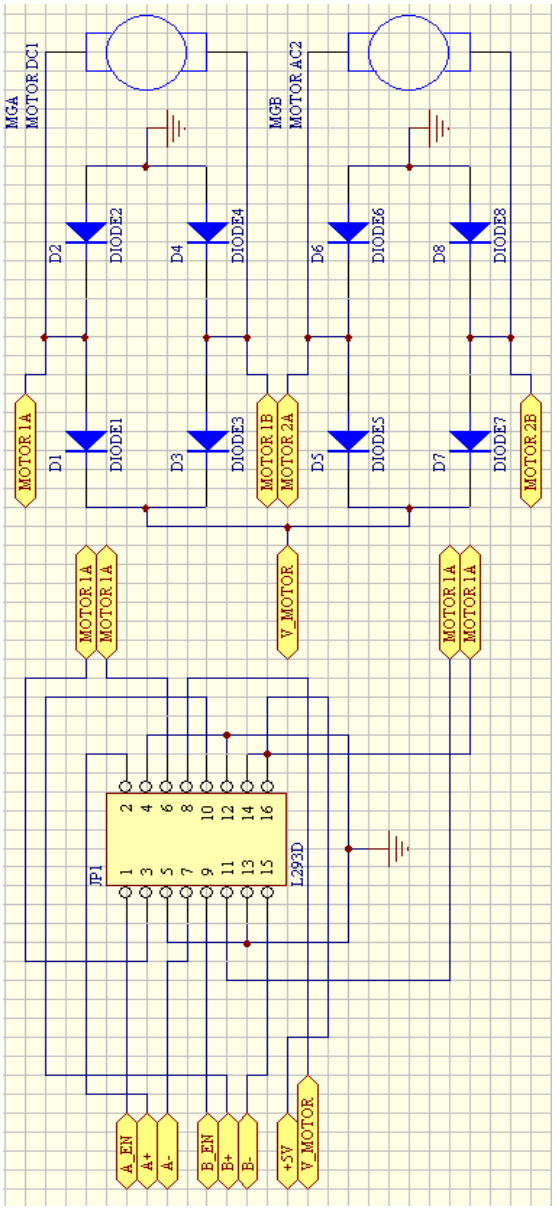
Satu motor dapat dikontrol dengan 3 pin yaitu A*enable*, A+ dan A-. Berikut ini adalah tabel kebenaran untuk salah satu motor.

Tabel 3.2 Tabel Kebenaran untuk Salah Satu Motor

| $A_{EN}$ | A+ | A- | Reaksi Motor |
|---|---|---|---|
| L | L | L | Motor off |
| L | L | H | Motor off |
| L | H | L | Motor off |
| L | H | H | Motor off |
| H | L | L | Stop |
| H | H | L | Berputar searah jarum jam |
| H | L | H | Berputar berlawanan arah jarum jam |
| H | H | H | Stop |

**Ket:** H = *High* = 1 dan L = *Low* = 0

Pengontrolan kedua motor kiri dan kanan dibutuhkan 6 pin yaitu untuk motor kiri masing-masing $A_{EN}$,A+ , A-dan motor kanan $B_{EN}$, B+, B-. Pin *output* yang menuju motor DC dihubungkan dengan dioda 1N4002 yang dihubungkan ke sumber Vcc dan *ground* motor DC. Pemilihan dioda ini berdasarkan kemampuannya untuk menangani arus tinggi yang mampu mengurangi *spike* yang terjadi akibat perubahan arah gerak robot. Berikut gambar 3.6 yang memperlihatkan gambar rangkaian *motor driver*.




Gambar 3.6 Rangkaian Motor Driver L293D

### 3.3.3.2. Motor DC

Motor DC berfungsi untuk memutarkan roda pada sebuah *mobile* robot. Motor DC kebanyakan memiliki bentuk fisik bulat,

sehingga tidak mudah untuk memasangnya pada chassis. Kecepatan putarnya (RPM) tinggi, namun torsinya rendah sehingga perlu dilengkapi dengan *gearbox*. *Gearbox* yang digunakan adalah jenis *double gearbox* yang sudah terpaket dengan satu pasang motor DC yang dapat bekerja pada tegangan 3 Volt hingga 6 Volt. Pemilihan *gearbox* ini didasarkan karena kecepatan maksimum yang dapat dihasilkan adalah 1.040 rpm dan torsi maksimum hingga 2,3 kgf.cm. Kecepatan dan torsi dapat diatur dengan mengatur *gear ratio*. Pada gambar 3.7 memperlihatkan bentuk fisik *gearbox* yang digunakan.

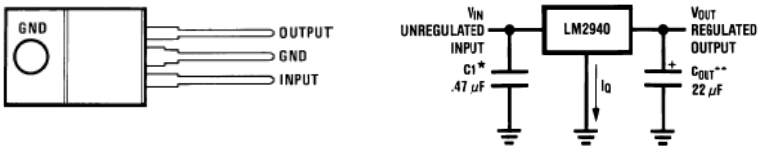



Gambar 3.7 Double gearbox

### 3.3.3.3. Catu Daya

Catu daya merupakan faktor pendukung terpenting, karena hampir keseluruhan komponen *hardware* pada robot membutuhkan tegangan untuk mengoperasikannya. Catu daya yang dibutuhkan sebanyak 2 buah, pertama untuk mencatu mikrokontroler dan sensor garis. Kedua, tegangan actuator berupa motor dc.

Tegangan yang digunakan berasal dari 1 buah baterai *rechargeable* berjenis *Lithium polymer* yang mempunyai nilai tegangan 7,4 volt dengan arus 1300 mAh. Untuk supply tegangan ke aktuator berupa motor dc, baterai dihungkan langsung ke motor dc sedangkan untuk supply tegangan ke mikrokontroler dan sensor garis diperlukan IC regulator untuk mendapatkan tegangan 5 volt, IC tersebut yaitu IC LM2940.



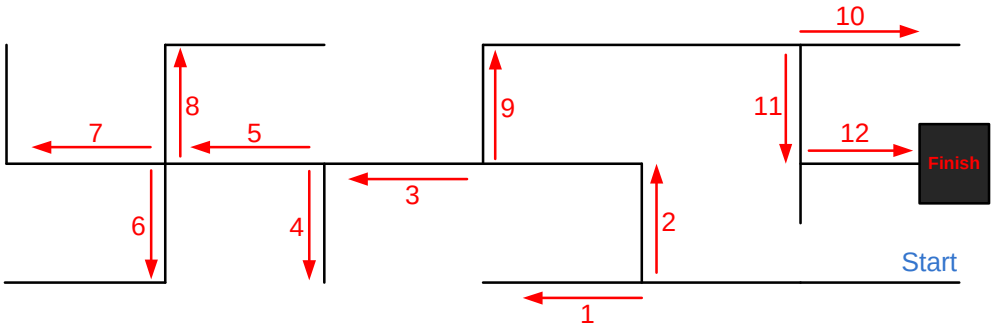

Gambar 3.8 (a) IC LM2940 (b) Skematik regulator 5 volt

Gambar 3.8 (a) menunjukkan gambar IC regulator untuk tegangan *output* 5 Vdc (yang terukur 4,9 volt) yang dihasilkan dari tegangan sumber Vin sebesar 7,4 volt. Pada Gambar 3.10 (b) terdapat kapasitor, digunakan untuk menghilangkan *ripple* yang berasal dari tegangan.

## 3.3. Perangkat Lunak (Software)

### 3.3.1. Perancangan Pohon Penelusuran Berdasarkan Algoritma Depth First Search

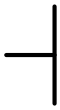
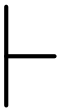
Berdasarkan teori algoritma depth first search dan bentuk track yang dirancang, maka dapat dirancang sebuah pohon penelusuran untuk mengetahui jalur mana sajakah yang akan ditelusuri robot dan jalur yang ditelusuri tersebut akan disederhanakan menjadi rangkaian jalur yang lebih efektif dan efisien. Gambar 3.9 memperlihatkan bentuk *track* yang digunakan dan arah jalur yang akan ditelusuri robot.



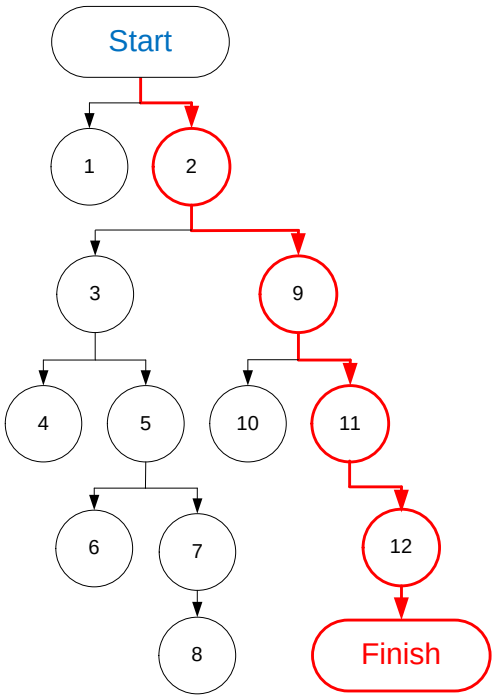

Gambar 3.9 Bentuk track dan arah jalur yang dilewati

Pada gambar *track* di atas terdapat beberapa bentuk persimpangan yang terdapat pada tabel 3.3.

Tabel 3.3 Bentuk-bentuk persimpangan

| Nama | Bentuk track | Data Yang tersimpan | Keterangan |
|---|---|---|---|
| Left junction |  | Karekter “L” | Belok kiri |
| T-junction |  | Karakter “L” | Belok kiri |
| Right junction |  | Karakter “S” | Lurus |
| Cross junction |  | Karakter “L” | Belok kiri |
| Dead End |  | Karakter “B” | Putar kiri |

Berdasarkan gambar 3.9 maka dapat dibuat sebuah pohon penelusurannya yang diperlihatkan pada gambar 3.10.




Gambar 3.10 Pohon penelusuran

Berikut penjelasan pohon penelusuran:

1. Awalnya robot bergerak dari titik start selanjutnya saat robot menemui bentuk persimpangan *right junction*, robot akan bergerak lurus ke arah nomer 1 maka akan disimpan karekter “S” yang artinya robot bergerak lurus, setelah bergerak lurus ternyata jalur tersebut merupakan jalur buntu dan robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan sisimpan karakter “B” yang berarti robot telah menemui jalur buntu. Maka :

   Data langkah yang tersimpan = “SB”
2. Selanjutnya robot bertemu bentuk persimpangan *left junction*, robot akan bergerak ke arah nomer 2 maka akan disimpan karakter “L” yang artinya robot belok kiri, pada tahap ini sudah 3 langkah yang tersimpan pada ingatan robot yaitu “SBL”. Dari 3 data langkah tersebut akan di konversi menjadi karakter “R”. Maka

   Data langkah yang tersimpan = “SBL”

   Dikonversi menjadi = “R”

3. Selanjutnya robot bertemu bentuk persimpangan *right junction,* robot akan bergerak ke arah nomer 3 maka akan disimpan karekter “S” yang artinya robot bergerak lurus. Maka :
   Data langkah yang tersimpan = “RS”
4. Robot bertemu bentuk persimpangan *left junction*, robot bergerak ke arah nomer 4 maka akan disimpan karakter “L” yang artinya robot belok kiri, setelah belok kiri ternyata jalur tersebut merupakan jalur buntu dan robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan disimpan karakter “B” yang berarti robot telah menemui jalur buntu. Maka :
   Data langkah yang tersimpan = “RSLB”
5. Selanjutnya robot bertemu bentuk persimpangan *T-junction,* robot akan bergerak ke arah nomer 5 maka akan disimpan karekter “L” yang artinya robot belok kiri. Maka :
   Data langkah yang tersimpan = “RSLBL”
   Dikonversi menjadi = “RSS”
6. Selanjutnya robot bertemu bentuk persimpangan *cross junction,* robot akan bergerak ke arah nomer 6 maka akan disimpan karekter “L” yang artinya robot belok kiri. Setelah belok kiri ternyata jalur tersebut merupakan jalur buntu maka robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan sisimpan karakter “B” yang berarti robot telah menemui jalur buntu. Maka :
   Data langkah yang tersimpan = “RSSLB”
7. Selanjutnya robot bertemu bentuk persimpangan *cross junction,* robot akan bergerak ke arah nomer 7 maka akan disimpan karekter “L” yang artinya robot belok kiri. Setelah bergerak lurus ternyata jalur tersebut merupakan jalur buntu dan robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan sisimpan karakter “B” yang berarti robot telah menemui jalur buntu. Maka :
   Data langkah yang tersimpan = “RSSLBLB”
   Dikonversi menjadi = “RSSSB”

8. Selanjutnya robot bertemu bentuk persimpangan *cross junction,* robot akan bergerak ke arah nomer 8 maka akan disimpan karekter “L” yang artinya robot belok kiri. setelah bergerak lurus ternyata jalur tersebut merupakan jalur buntu dan robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan disimpan karakter “B” yang berarti robot telah menemui jalur buntu. Setelah berputar kembali ke persimpangan sebelumnya robot bertemu bentuk persimpangan *cross junction,* robot akan belok ke kiri maka akan disimpan karekter “L”. Selanjutnya robot kembali ke persimpang sebelumnya lagi dan pada titik ini robot bertemu bentuk persimpangan *right junction*, robot akan bergerak lurus dan karakter yang disimpan yaitu “S” Maka :

   Data langkah yang tersimpan = “RSSSBLBLS”

   Dikonversi menjadi = “RSSRBL”

   Dikonversi lagi menjadi = “RSSBS”

   Dikonversi lagi menjadi = “RSB

9. Pada titik ini robot bertemu bentuk persimpangan *left junction*, robot bergerak ke arah nomer 9 maka akan disimpan karakter “L” yang artinya robot belok kiri,. Maka :

   Data langkah yang tersimpan = “RSBL”

   Dikonversi menjadi = “RR”

10. Selanjutnya robot bertemu bentuk persimpangan *right junction,* robot akan bergerak ke arah nomer 10 maka akan disimpan karekter “S” yang artinya robot bergerak lurus. Setelah bergerak lurus ternyata jalur tersebut merupakan jalur buntu dan robot akan berputar kembali ke persimpangan sebelumnya. Pada saat berputar akan sisimpan karakter “B” yang berarti robot telah menemui jalur buntu.Maka :

    Data langkah yang tersimpan = “RRSB”

11. Selanjutnya robot bertemu bentuk persimpangan *left junction*, robot akan bergerak ke arah nomer 11 maka akan disimpan karakter “L” yang artinya robot belok kiri. Maka:

    Data langkah yang tersimpan = “RRSBL”

Dikonversi menjadi = “RRR”

12. Selanjutnya robot bertemu bentuk persimpangan *left junction*, robot akan bergerak ke arah nomer 11 maka akan disimpan karakter “L” yang artinya robot belok kiri. Maka:

    Data langkah yang tersimpan = “RRRL” (*right, right, right, left*)

Dari hasil penelusuran jalur maka didapat rangkangkaian jalur yang lebih efektif dan efisien yaitu *right* (kanan), *right* (kanan), *right* (kanan), *left* (kiri). Jika diambil dari pohon penelusuran, jalur yang efektif dan efisien yaitu 2, 9, 12, 11.

### 3.3.2. Perancangan Prosedural

Perancangan prosedural merupakan salah satu bagian terpenting dalam proses perancangan robot, karena pada perancangan prosedural terdiri beberapa algoritma yang disusun secara terprosedur.

Pada perancangan prosedural ini terdiri dari 2 algoritma yang digunakan untuk pergerakan robot dan penelusuran *track* yaitu:

1. Algoritma pemetaan (*maping*)
2. Algoritma penyederhanaan (simplifikasi)

#### 3.3.3.1. Algoritma Pemetaan (Maping)

Algoritma ini bertujuan agar robot dapat bergerak dari titik *start* ke *finish* sambil melakukan penyimpanan data setiap langkah dan algoritma ini juga merupakan program utama. Data langkah tersebut akan dikonversikan menjadi sebuah kode berupa karakter.

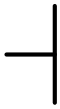
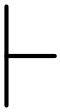
Pada algoritma ini, terdapat beberapa proses yaitu proses pencatatan data langkah, penyimpanan data langkah, penyederhanaan data langkah (simplifikasi). Masing-masing proses tersebut akan diwakili oleh suatu *subroutine* agar memudahkan dalam pembuatan program. Tetapi sebelumnya harus ditentukan kode data langkah dan sensor mana yang digunakan karena ada beberapa bentuk belokan dan persimpangan, tetapi hanya persimpangan saja yang datanya

disimpan. Adapun kode data langkah yang digunakan pada algoritma pemetaan ini adalah :

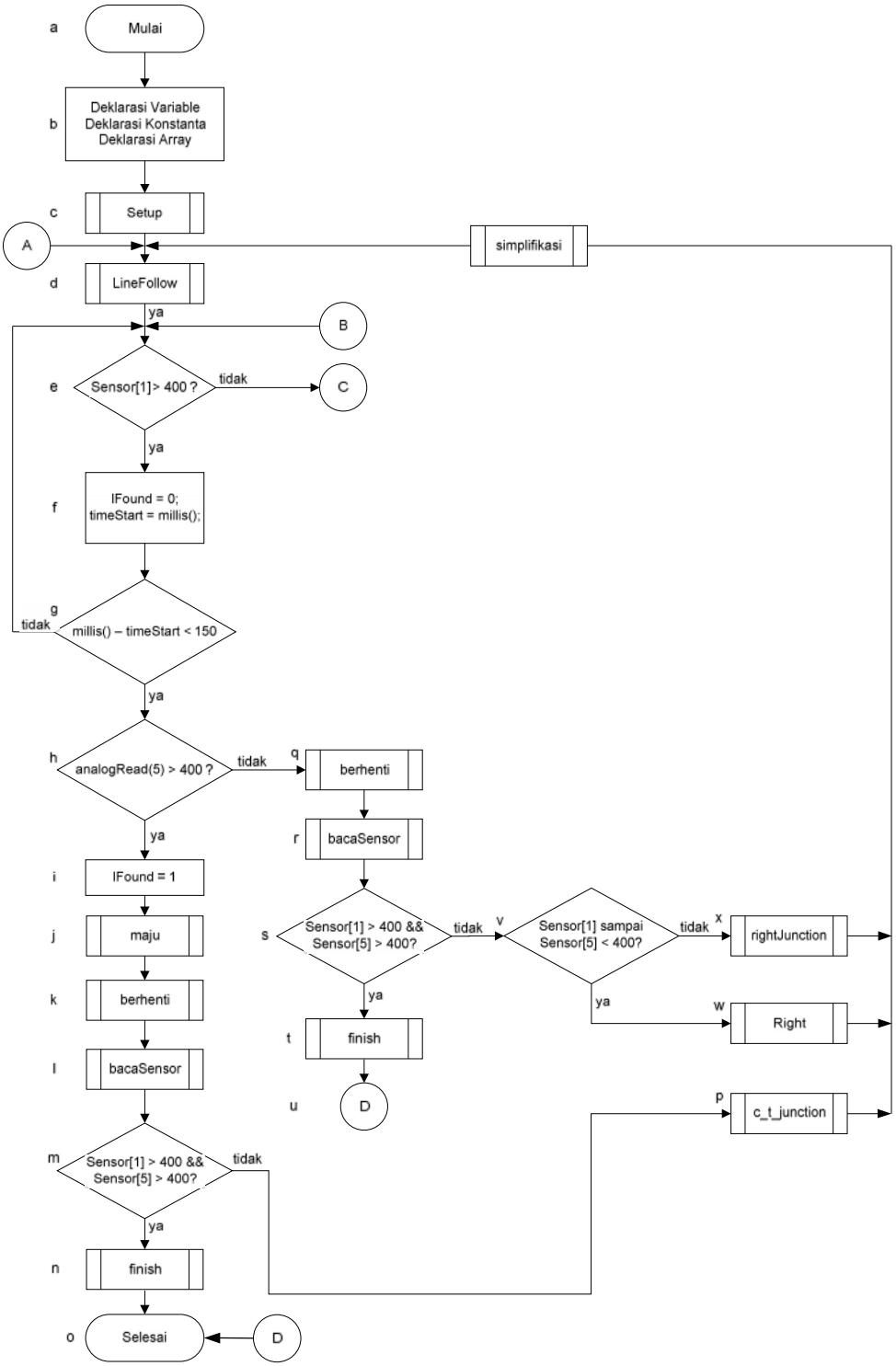
1. “L” berarti *left.* Ini menandakan robot telah melakukan belok kiri karena melewati persimpangan.
2. “S” berarti *Straight. I*ni menandakan robot telah melakukan jalan terus karena bertemu dengan persimpangan tiga dengan pilihan lurus atau belok kanan.
3. “B” berarti *back*. Ini menandakan robot berjumpa jalur buntu dan berjalan kembali ke persimpangan yang terakhir.

Untuk lebih jelasnya dapat dilihat pada tabel 3.4.

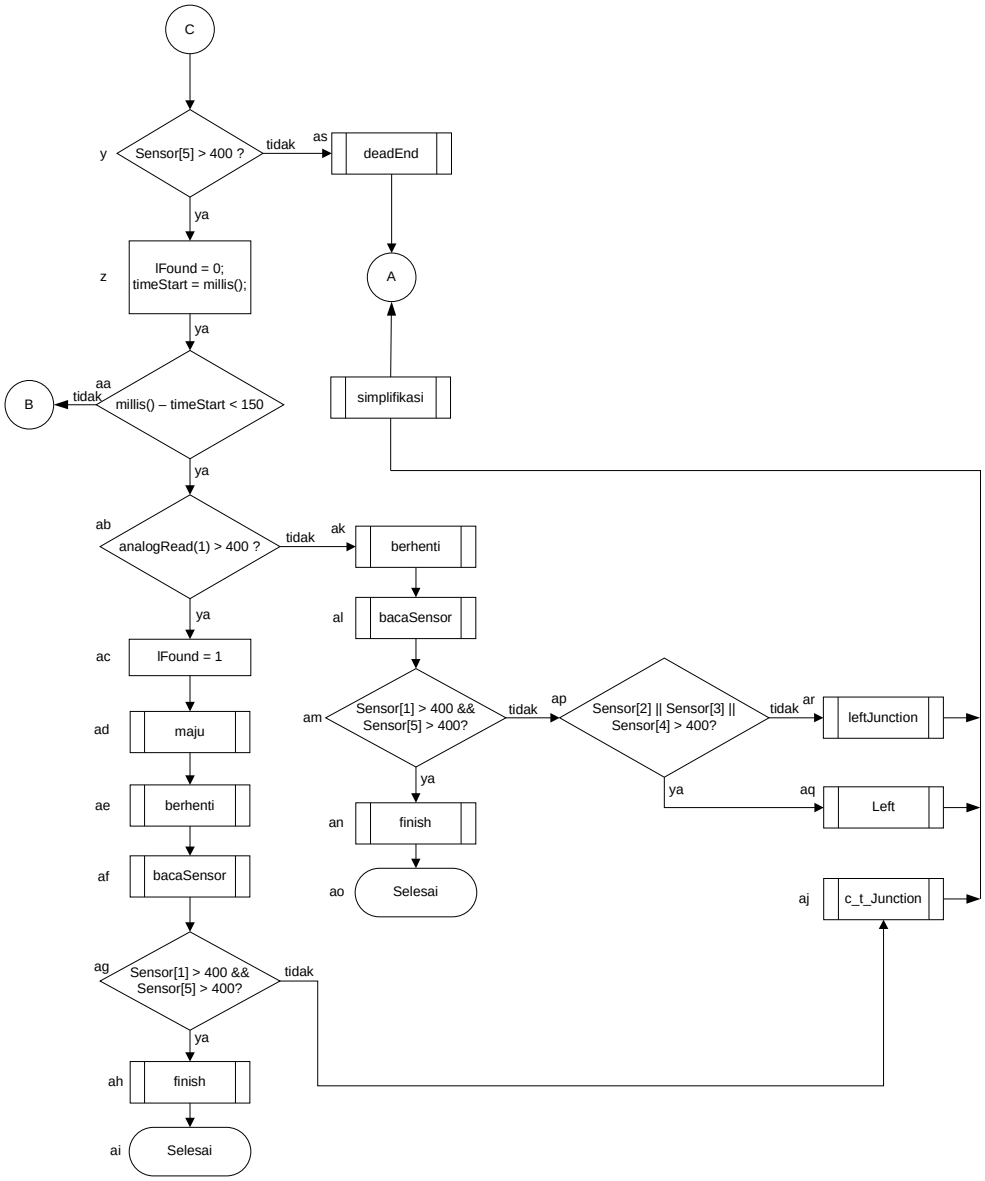
Tabel 3.4 Bentuk belokan dan persimpangan

| Nama | Bentuk track | Sensor yang digunakan | Data Yang disimpan |
|---|---|---|---|
| Left |  | Sensor 5 | - |
| Left junction |  | Sensor 5 | Karekter “L” |
| T-junction |  | Sensor 5 dan Sensor 1 | Karakter “L” |
| Right |  | Sensor 1 | - |
| Right junction |  | Sensor 1 | Karakter “S” |
| Cross junction |  | Sensor 5 dan Sensor 1 | Karakter “L” |
| Straight |  | Sensor 2 sampai sensor 4 | - |
| Dead end | Buntu | Sensor 2 sampai sensor 4 | Karakter “B” |

Dari tabel di atas maka dapat dirancang sebuah algoritma pemetaannya. Algoritma pemetaan dapat dilihat pada gambar 3.11 sampai gambar 3.12.




Gambar 3.11 Diagram alir pemetaan




Gambar 3.12 Diagram alir pemetaan (lanjutan)

Berikut ini penjelasan algoritma pemetaan:

a. Memulai awal program.
b. Pendeklarasian program. Deklarasi variabel, konstanta, array.
c. Panggil *subroutine* setup.
d. Panggil *subroutine* lineFollow agar robot bergerak lurus
e. Periksa apakah sensor 1 lebih besar dari 400 (berada pada garis hitam).

f. Jika ya, isi nilai 0 pada variabel lFound, isi variabel time start dengan nilai millis.

g. Periksa apakah nilai hasil pengurangan millis dengan variabel time start lebih kecil dari 150.

h. Jika ya, apakah sensor 5 juga lebih besar dari 400 (berada pada garis hitam).

i. Isi nilai 1 pada variabel lFound.

j. Jika ya, panggil *subroutine* maju (robot akan bergerak maju sedikit).

k. Panggil *subroutine* berhenti (robot berhenti selama 100 ms).

l. Panggil *subroutine* baca sensor.

m. Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.

n. Jika ya, panggil *subroutine finish*.

o. Program selesai.

p. Jika sensor 1 dan sensor 5 tidak berada pada garis hitam, panggil *subroutine* c_t_junction, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

q. Panggil *subroutine* berhenti (robot berhenti selama 100 ms).

r. Panggil *subroutine* baca sensor.

s. Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.

t. Jika ya, panggil *subroutine finish*.

u. Program selesai.

v. Jika sensor 1 dan sensor 5 tidak berada pada garis hitam, periksa apakah sensor 2 sampai sensor 4 lebih kecil dari 400 (berada pada garis putih).

w. Jika ya, panggil *subroutine* Right, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

x. Jika tidak, panggil *subroutine* rightJunction, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

y. Periksa apakah sensor 5 lebih besar dari 400 (berada pada garis hitam).

z. Jika ya, isi nilai 0 pada variabel lFound, isi variabel time start dengan nilai millis.

a.a Periksa apakah nilai hasil pengurangan millis dengan variabel time start lebih kecil dari 150.

a.b Jika ya, apakah sensor 1 juga lebih besar dari 400 (berada pada garis hitam).

a.c Isi nilai 1 pada variabel lFound.

a.d Jika ya, panggil *subroutine* maju (robot akan bergerak maju sedikit).

a.e Panggil *subroutine* berhenti (robot berhenti selama 100 ms).

a.f Panggil *subroutine* baca sensor.

a.g Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.

a.h Jika ya, panggil *subroutine finish.*

a.i Program selesai.

a.j Jika sensor 1 dan sensor 5 tidak berada pada garis hitam, *subroutine* prosedur c_t_junction, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

a.k Panggil *subroutine* berhenti (robot berhenti selama 100 ms).

a.l Panggil *subroutine* baca sensor.

a.m Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.

a.n Jika ya, panggil *subroutine finish.*

a.o Program selesai.

a.p Jika sensor 1 dan sensor 5 tidak berada pada garis hitam, periksa apakah sensor 2 atau sensor 3 atau sensor 3 lebih besar dari 400 (berada pada garis hitam).

a.q Jika ya, panggil *subroutine* Left, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

a.r Jika tidak, panggil *subroutine* leftJunction, selanjutnya panggil *subroutine* simplifikasi dan kembali ke langkah d.

a.s Jika sensor 5 tidak lebih besar dari 400 panggil *subroutine* dedEnd dan kembali ke langkah d.

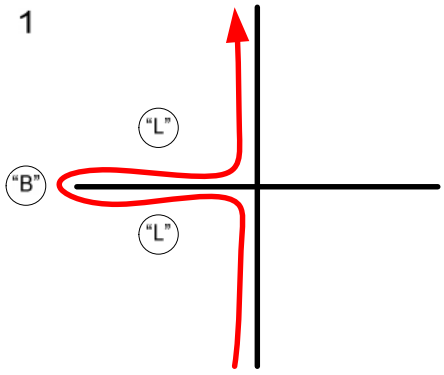
### 3.3.3.2. Algoritma penyederhanaan (simplifikasi)

Algoritma ini merupakan bagian dari algoritma pemetaan. Setiap instruksi yang terdapat pada algoritma pemetaan ini, akan diwakili oleh satu nama *subroutine* yang nantinya akan di panggil ketika banyaknya data langkah yang tersimpan pada array lebih besar atau sama dengan 3 dan data kedua dari data langkah terakhir array adalah karakter ‘B’, ketika syarat tersebut terpenuhi maka data tersebut akan dikonversi menjadi satu karakter tertentu.

Dari proses pemetaan maka akan kode-kode akan tersusun, kode-kode tersebut akan disederhanakan menggunakan formulasi sebagai berikut:

1. Setiap karakter data langkah dikonversi ke sebuah nilai, yaitu “L” = 270, “R” = 90, “S” = 0, “B” = 180.
2. Nilai tersebut akan dijumlahkan dan akan dibagi dengan 360, sisa pembagiannya akan dikonversi menjadi karakter data langkah (Hasil penyederhanaannya.

Berikut adalah proses pemberian kode dan cara penyederhanaannya berdasarkan bentuk persimpangan:

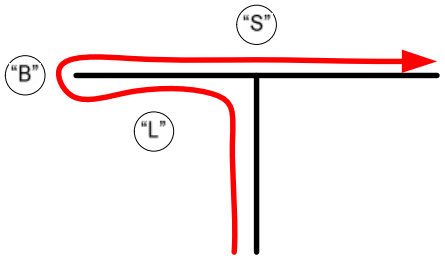



Data yang tersimpan = “LBL” = 270 + 180 + 270 = 720

Hasil penyederhannya = 720 mod 360 = 0

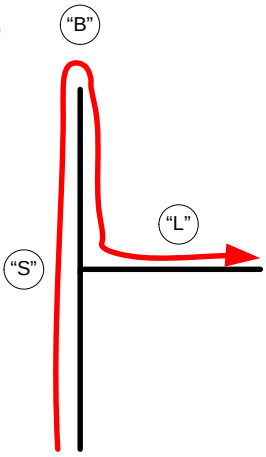
0 dikonversi menjadi “S”.

2



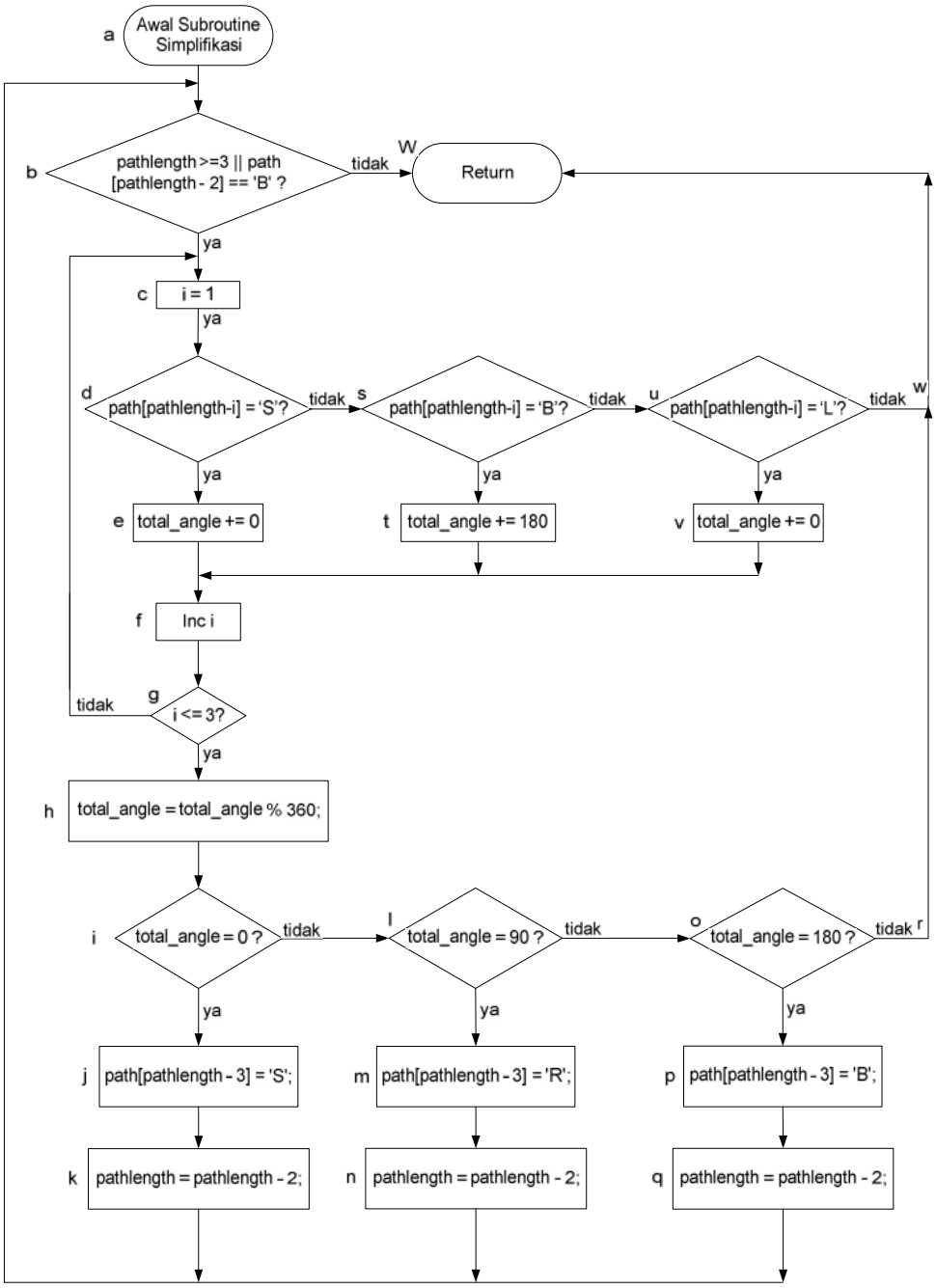

Data yang tersimpan = "LBS" = 270 + 180 + 0 = 450
Hasil penyederhannya = 450 mod 360 = 90
90 dikonversi menjadi "R".

3




Data yang tersimpan = "SBL" = 0 + 180 + 270 = 450
Hasil penyederhannya = 450 mod 360 = 90
90 dikonversi menjadi "R".

Berdasarkan cara di atas maka dapat dirancang sebuah algoritma penyederhanaan (simplifikasi). Algoritma simplifikasi dapat dilihat pada gambar 3.13.




Gambar 3.13 Diagram alir simplifikasi

Berikut ini penjelasan algoritma simplifikasi:

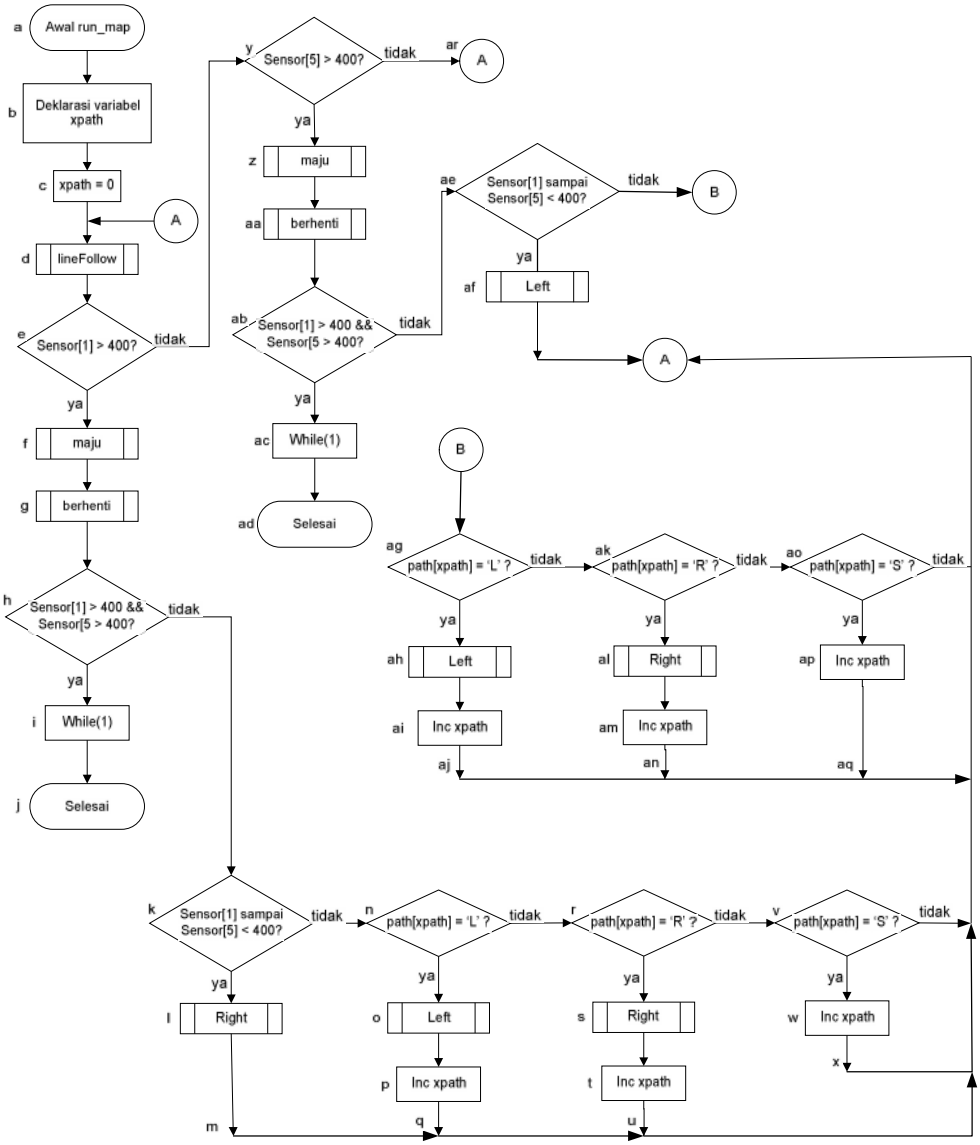
a. Memulai awal simplifikasi.
b. Periksa apakah banyak array lebih besar sama dengan 3 atau data ke dua dari terakhir sama dengan “B”.

c. Jika ya, isi variabel i dengan 1.
d. Periksa apakah data pada array karakter ”S”.
e. Jika ya, jumlahkan variabel total_angle dengan 0.
f. Increment variabel i.
g. Perikasa apakah variabel i lebih kecil sama dengan 3, jika tidak kembali ke langkah c.
h. Jika ya, isi variabel total_angle dengan sisa pembagian dari variabel total_angle dengan 360.
i. Periksa apakah variabel total_angle sama dengan 0.
j. Jika ya, ganti data ketiga dari terakhir dengan array dengan karakter ”S”.
k. Kurangi jumlah array dengan 2 selanjutnya kembali ke langkah b.
l. Periksa apakah variabel total_angle sama dengan 90.
m. Jika ya, ganti data ketiga dari terakhir dengan array dengan karakter ”R”.
n. Kurangi jumlah array dengan 2 selanjutnya kembali ke langkah b.
o. Periksa apakah variabel total_angle sama dengan 180.
p. Jika ya, ganti data ketiga dari terakhir dengan array dengan karakter ”B”.
q. Kurangi jumlah array dengan 2 selanjutnya kembali ke langkah b.
r. Jika variabel total_angle nilainya selain 0, 90, 180 maka lanjutkan ke langkah yang terdapat pada algoritma pemetaan.
s. Periksa apakah data pada array karakter ”B”.
t. Jika ya, jumlahkan variabel total_angle dengan 180 dilanjukan ke langkah f.
u. Periksa apakah data pada array karakter ”L”.
v. Jika ya, jumlahkan variabel total_angle dengan 270 dilanjukan ke langkah f.
w. Periksa apakah banyak array tidak lebih besar sama dengan 3 atau data ke dua dari terakhir tidak sama dengan “B”, maka dilanjutkan ke langkah yang terdapat pada algoritma pemetaan.

#### 3.3.3.3. Algoritma Bergerak Berdasarkan Peta (Running Map)

Algoritma ini akan dikerjakan setelah proses penyederhanaan data langkah selesai dan robot dapat bergerak dari titik *start* ke titik *finish* beserta penyimpanan data langkah yang sesuai dengan bentuk persimpangan yang dihadapi.

Algoritma ini bertujuan untuk meninstruksikan robot agar bergerak berdasarkan peta yang telah disederhanakan, maka dalam algoritma ini hanya memanggil data langkah yang tersimpan dalam sebuah array. Untuk lebih jelasnya lihat gambar 3.14.




Gambar 3.14 Diagram alir Run map

Berikut ini penjelasan algoritma pemetaan:

a. Memulai awal program.
b. Pendeklarasian program. Deklarasi variabel.
c. Isi variabel xpath dengan nilai 0.
d. Panggil *subroutine* lineFollow agar robot bergerak lurus.
e. Periksa apakah sensor 1 lebih besar dari 400 (berada pada garis hitam).

f. Jika ya, panggil *subroutine* maju.
g. Panggil *subroutine* berhenti (robot berhenti selama 100 ms).
h. Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.
i. Jika ya, robot berhenti.
j. Program selesai.
k. Periksa apakah sensor 1 sampai 5 berada pada garis putih.
l. Jika ya, panggil *subroutine right*.
m. Jika ya, keluar dari *subroutine* dan kembali mengeksekusi instruksi d.
n. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’L’.
o. Jika ya, panggil *subroutine left*.
p. *Increment* variabel array xpath.
q. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.
r. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’R’.
s. Jika ya, panggil *subroutine right*.
t. *Increment* variabel array xpath.
u. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.
v. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’S’.
w. *Increment* variabel array xpath.
x. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.
y. Periksa apakah sensor 5 lebih besar dari 400 (berada pada garis hitam).
z. Jika ya, panggil *subroutine* maju.
a.a. Panggil *subroutine* berhenti (robot berhenti selama 100 ms).
a.b. Periksa apakah sensor 1 dan sensor 5 berada pada garis hitam.
a.c. Jika ya, robot berhenti.
a.d. Program selesai.
a.e. Periksa apakah sensor 1 sampai 5 berada pada garis putih.

a.f. Jika ya, panggil *subroutine left*, keluar dari *subroutine* dan kembali mengeksekusi instruksi d.

a.g. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’L’.

a.h. Jika ya, panggil *subroutine left*.

a.i. *Increment* variabel array xpath.

a.j. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.

a.k. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’R’.

a.l. Jika ya, panggil *subroutine right*.

a.m. *Increment* variabel array xpath.

a.n. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.

a.o. Jika tidak, ambil data pada array, periksa apakah data array adalah karakter ’S’.

a.p. *Increment* variabel array xpath.

a.q. Keluar dari *subroutine* dan kembali mengeksekusi instruksi d.

a.r. Jika sensor 5 tidak berada pada garis hitam kembali mengeksekusi instruksi d.

### 3.3.3. Perancangan Subroutine

Perancangan *subroutine* bertujuan agar program lebih terstruktur dan sederhana. *Subroutine* - *Subroutine* ini akan dipanggil pada algoritma utama karena dalam suatu *subroutine* terdapat beberapa instruksi program.

Perancangan subroutine pada tugas akhir ini dibagi menjadi 3 bagian, yaitu:

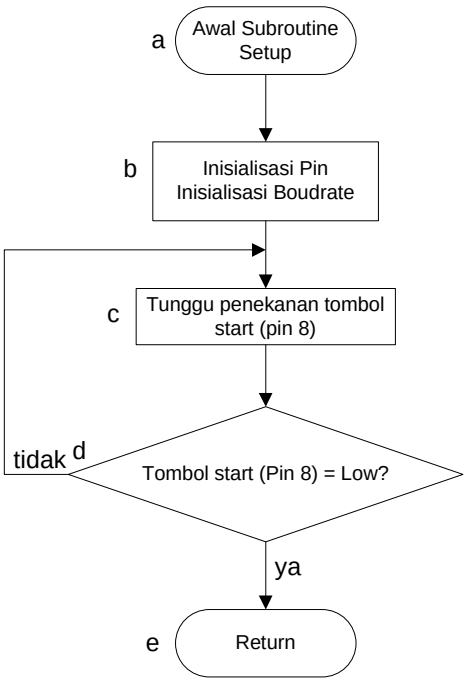
1. Subroutine utama
2. Subroutine penyimpanan data langkah
3. Subroutine pergerakan robot
4. Subroutine display path

### 3.3.3.1. Subroutine Utama

*Subroutine* ini memegang peran penting karena didalamnya terdapat instruksi yang fatal jika terlewat. Berikut yang termasuk pada *subroutine* utama:

1. Setup

   Pada *subroutine* ini terdapat instruksi untuk inisialisasi pin, boudrate, tombol *start*. Gambar 3.15 memperlihatkan diagram alir *subroutine* setup.




Gambar 3.15 Diagram alir subroutine setup

Berikut ini penjelasan diagram alir *subroutine setup*:

a Memulai diagram alir subroutine setup.

b Inisialisasi pin, inisialisasi boudrate.

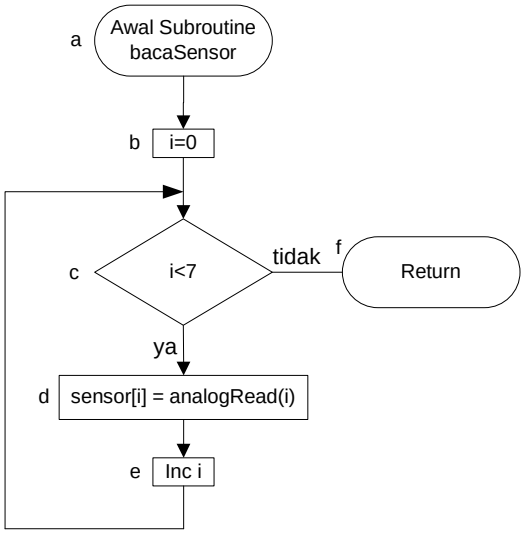
c Tunggu penekanan tombol start.

d Periksa apakah pin 8 (tombol *start*) low.

e Jika ya, keluar dari subroutine dan kembali mengeksekusi instruksi berikutnya pada program utama dan robot akan mulai bergerak.

2. Baca Sensor

Pada subroutine bertujuan untuk pembacaan sensor melalui pembacaan ADC. Gambar 3.16 memperlihatkan diagram alir *subroutine* baca sensor.




Gambar 3.16 Diagram alir subroutine baca sensor

Berikut ini penjelasan diagram alir *subroutine baca sensor*:

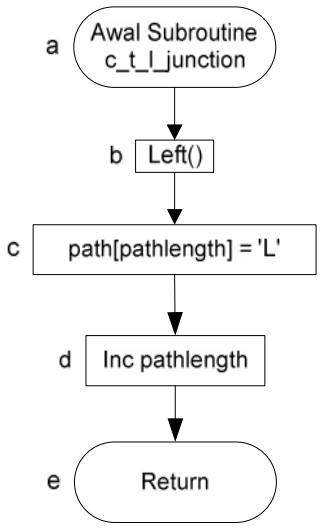
a. Memulai diagram alir subroutine baca sensor.
b. Isi variabel i dengan 0.
c. apakah i kurang dari 7.
d. Periksa Periksa apakah isi variabel sensor ke i sama dengan ADC ke i.
e. Jika ya, increment variabel i dan kembali ke langkah c.
f. Jika tidak, keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

### 3.3.3.2. Subroutine Penyimpanan Data Langkah

Pada *subroutine* ini terdiri dari dua proses yaitu pergerakan dan penyimpanan bentuk pergerakan tersebut, jadi *subroutine* ini dipanggil ketika robot menemui salah satu bentuk persimpangan. Berikut yang termasuk subroutine penyimpanan data langkah:

1. Cross, T, Left junction

   *Subroutine* ini dipanggil ketika robot menemui *cross junction*, *left junction*, dan T-juntion. Data langkah yang tersimpan pada array adalah karakter ‘L’. Gambar 3.17 memperlihatkan diagram alir *subroutine* ini.




Gambar 3.17 Diagram alir subroutine cross junction, left junction, dan T-juntion

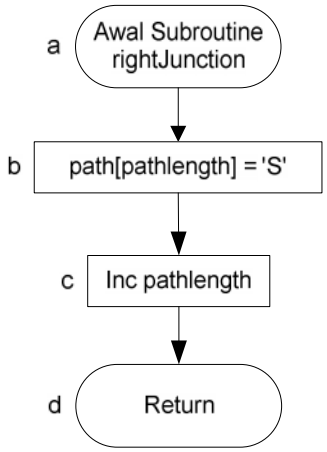
Berikut ini penjelasan diagram alir *subroutine cross, T, left junction*:

a. Memulai diagram alir subroutine c_t_l_junction.
a. Putar kiri sampai sensor 4 berada pada garis hitam.
b. Simpan karakter ‘L’ pada array.
c. Increment array.

d. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

2. Right Junction

*Subroutine* ini dipanggil ketika robot menemui *left junction*. Data langkah yang tersimpan pada array adalah karakter ‘L’. Gambar 3.18 memperlihatkan diagram alir *subroutine* ini.



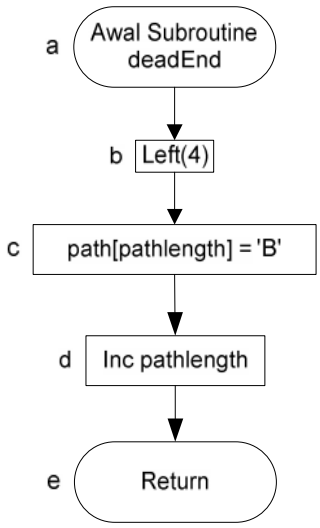

Gambar 3.18 Diagram alir subroutine righ junction

Berikut ini penjelasan diagram alir *right junction*:

a. Memulai diagram alir subroutine right junction.
b. Simpan karakter ‘L’ pada array.
c. Increment array.
d. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

3. Dead End

*Subroutine* ini dipanggil ketika robot menemui *track* buntu. Data langkah yang tersimpan pada array adalah karakter ‘B’. Gambar 3.19 memperlihatkan diagram alir *subroutine* ini.




Gambar 3.19 Diagram subroutine dead end

Berikut ini penjelasan diagram alir *subroutine dead end*:

a. Memulai diagram alir subroutine left junction.
b. Putar kiri sampai sensor 4 berada pada garis hitam.
c. Simpan karakter ‘B’ pada array.
d. Increment array.
e. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

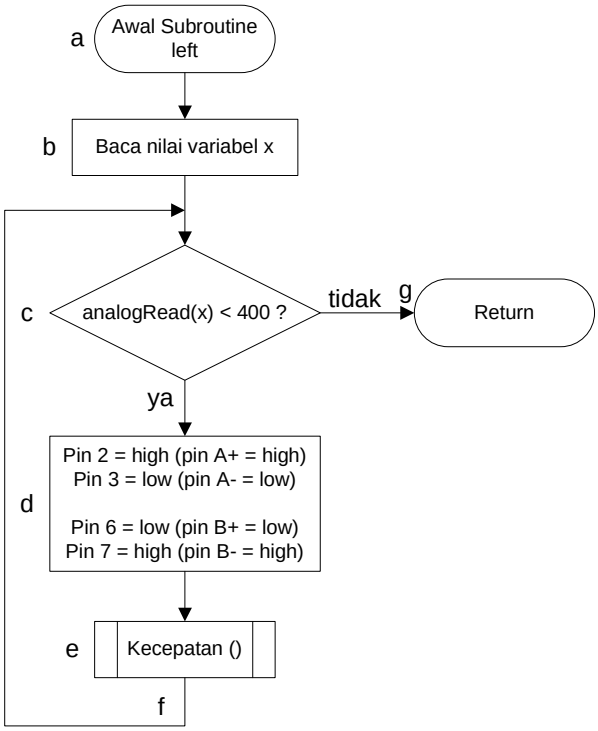
#### 3.3.3.3. Subroutine Pergerakan robot

Pada *Subroutine* ini terdiri beberapa instuksi agar robot dapat berputar atau belok, maju dan berhenti. *Subroutine* ini dipanggil ketika robot menemui belokan dan persimpangan. Berikut yang termasuk *subroutine* pergerakan robot:

1. Left

   *Subroutine left* berfungsi untuk memberi instruksi putar kiri. *Subroutine* ini dipanggil ketika robot menemui belokan kiri 90 derajat, cross junction, left junction, T-junction, dan dead end (buntu). Pada saat pemanggilan *subroutine* ini, putaran atau belokannya dapat dibatasi dengan menggunakan sensor

pembatas dan sensor pembatasnya dapat dipilih sesuai dengan pergerakan robotnya. Untuk pemilihan sensor pembatasnya dapat digunakan sebuah variabel yang nilainya dapat diganti. Gambar 3.20 memperlihatkan diagram alir *subroutine* ini.




Gambar 3.20 Diagram alir subroutine left

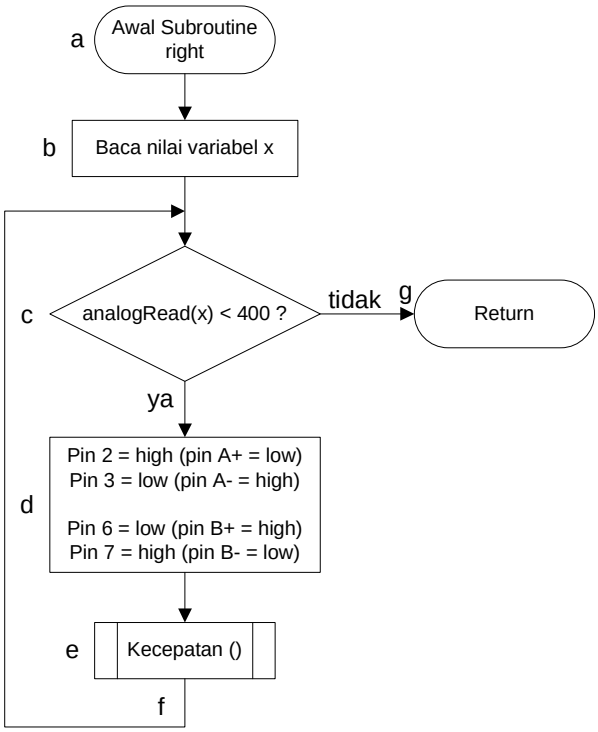
Berikut ini penjelasan diagram alir *subroutine left*:

a. Memulai diagram alir *subroutine left*.
b. Baca nilai variabel x.
c. Periksa apakah ADC ke x lebih kecil dari 400.
d. Inisialisasi pin yang dihubungkan ke A+ dan A- agar roda berputar maju dan Inisialisasi pin yang dihubungkan ke B+ dan B- agar roda berputar dengan arah berlawanan sehingga robot berputar ke kiri.
e. Panggil *subroutine* kecepatan untuk memberi nilai PWM kiri dan kanan.

f. Jika ya, kembali ke langkah c.
g. Jika tidak, lanjutkan ke langkah yang terdapat pada algoritma pemetaan.

2. Right

*Subroutine right* berfungsi untuk memberi instruksi putar kanan. *Subroutine* ini dipanggil ketika robot menemui belokan kanan 90 derajat. Pada saat pemanggilan *subroutine* ini, putaran atau belokannya dapat dibatasi dengan menggunakan sensor pembatas dan sensor pembatasnya dapat dipilih sesuai dengan pergerakan robotnya. Untuk pemilihan sensor pembatasnya dapat digunakan sebuah variabel yang nilainya dapat diganti. Gambar 3.21 memperlihatkan diagram alir *subroutine* ini.



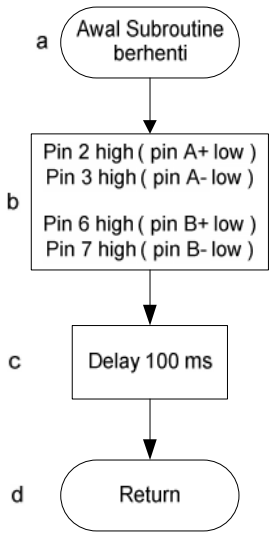

Gambar 3.21 Diagram alir subroutine right

Berikut ini penjelasan diagram alir *subroutine right*:

a. Memulai diagram alir *subroutine right*.
b. Baca nilai variabel x.
c. Periksa apakah ADC ke x lebih kecil dari 400.
d. Inisialisasi pin yang dihubungkan ke A+ dan A- agar roda berputar maju dan Inisialisasi pin yang dihubungkan ke B+ dan B- agar roda berputar dengan arah berlawanan sehingga robot berputar ke kanan.
e. Panggil *subroutine* kecepatan untuk memberi nilai PWM kiri dan kanan.
f. Jika ya, kembali ke langkah c.
g. Jika tidak, lanjutkan ke langkah yang terdapat pada algoritma pemetaan.

3. Berhenti

*Subroutine* berhenti berfungsi untuk memberi instruksi berhenti selama 100 ms kepada robot. Setelah robot berhenti, robot akan melanjutkan instrusi selanjutnya. Gambar 3.22 memperlihatkan diagram alir *subroutine* ini.



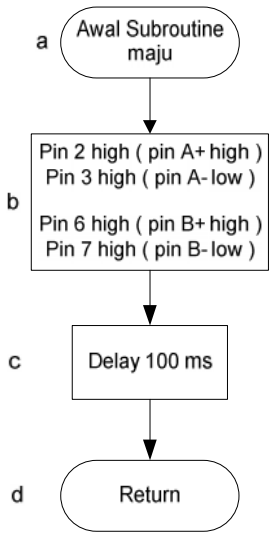

Gambar 3.22 Diagram alir subroutine berhenti

Berikut ini penjelasan diagram alir *subroutine berhenti*:

a. Memulai diagram alir *subroutine berhenti*
b. Inisialisasi pin yang dihubungkan ke A+ dan A- agar roda berputar mundur dan Inisialisasi pin yang dihubungkan ke B+ dan B- agar roda berputar mundur sehingga robot berhenti.
c. Delay 100 *milisecond.*
d. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

4. Maju

*Subroutine* maju berfungsi untuk memberi instruksi maju selama 100 ms kepada robot. Setelah robot maju, robot akan melanjutkan instrusi selanjutnya. Gambar 3.23 memperlihatkan diagram alir *subroutine* ini.



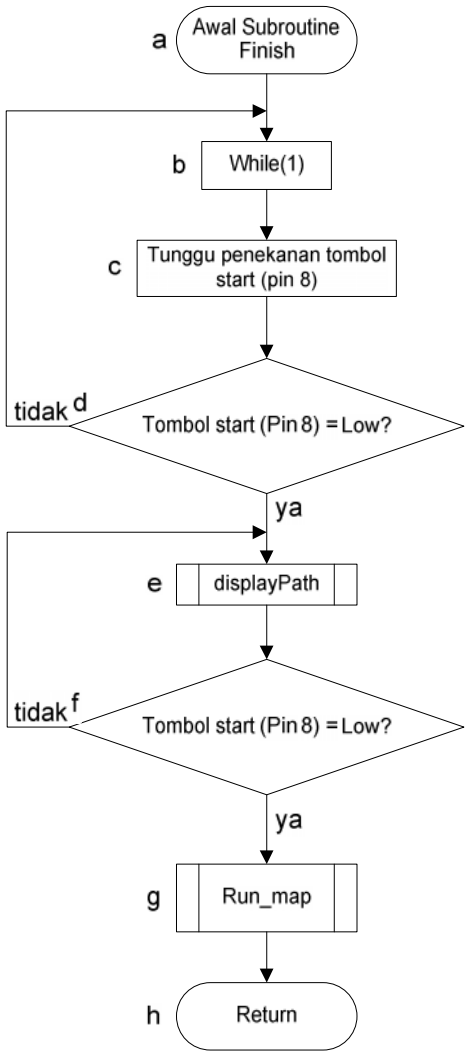

Gambar 3.23 Diagram alir subroutine maju

Berikut ini penjelasan diagram alir *subroutine maju*:

a. Memulai diagram alir *subroutine maju*
b. Inisialisasi pin yang dihubungkan ke A+ dan A- agar roda berputar maju dan Inisialisasi pin yang dihubungkan ke B+ dan B- agar roda berputar maju sehingga robot maju.
c. Delay 100 *milisecond*.
d. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

2. Finish

*Subroutine finish* berfungsi untuk memberi instruksi maju berhenti sampai ada penekanan tombol *start*. Setelah tombol *start* ditekan maka, robot akan mengeksekusi instruksi yang ada pada *subroutine run map*. Gambar 3.24 memperlihatkan diagram alir *subroutine* ini.



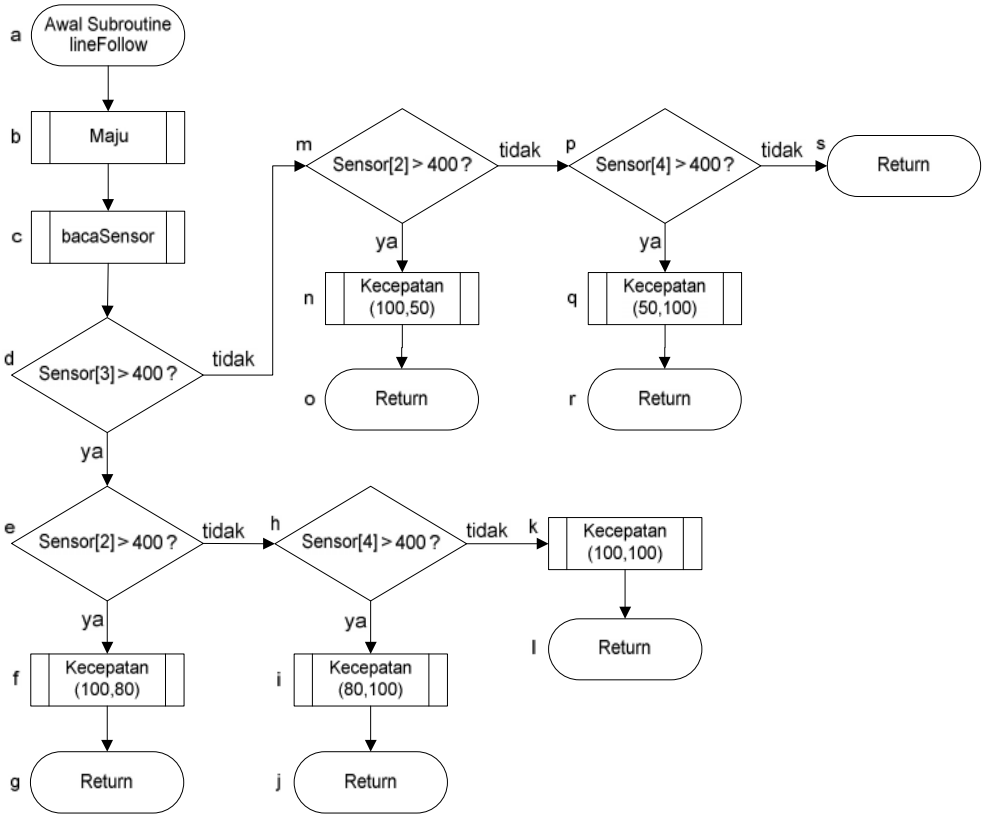

Gambar 3.24 Diagram alir subroutine finish

Berikut ini penjelasan diagram alir *subroutine finish*:

a. Memulai diagram alir *subroutine finish.*
b. Robot berhenti.
c. Set pin 8 *low* (tombol *start*) pada high.
d. Tunggu penekanan tombol start.
e. Jika ya, panggil *subroutine displayPath.*
f. Tunggu penekanan tombol start.
g. Jika ya, panggil *subroutine Run_map.*
h. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

3. Line Follow

*Subroutine line follow* berfungsi untuk menelusuri garis lurus. Gambar 3.25 memperlihatkan diagram alir *subroutine* ini.




Gambar 3.25 Diagram alir subroutine line follow

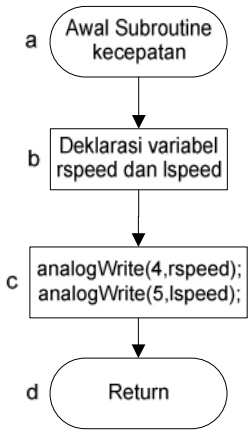
Berikut ini penjelasan diagram alir *subroutine line follow*:

a. Memulai diagram alir *subroutine line follow.*
b. Panggil *subroutine* maju.
c. Panggil *subroutine* baca sensor.
d. Periksa apakah sensor 3 lebih besar dari 400 (berada pada garis hitam).
e. Jika ya, periksa apakah sensor 2 lebih besar dari 400 (berada pada garis hitam).
f. Jika ya, nilai PWM kiri 100, nilai PWM kanan 80

g. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.
h. Jika ya, periksa apakah sensor 4 lebih besar dari 400 (berada pada garis hitam).
i. Jika ya, nilai PWM kiri 80, nilai PWM kanan 100
j. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.
k. Jika sensor 3 saja yang lebih besar dari 400 (berada pada garis hitam), nilai PWM kiri 100, nilai PWM kanan 100.
l. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.
m. Periksa apakah sensor 2 lebih besar dari 400 (berada pada garis hitam).
n. Jika ya, nilai PWM kiri 100, nilai PWM kanan 50.
o. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.
p. Periksa apakah sensor 4 lebih besar dari 400 (berada pada garis hitam).
q. Jika ya, nilai PWM kiri 50, nilai PWM kanan 100.
r. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.
s. Jika sensor 2, 3, 4, tidak ada yang lebih besar dari 400 (berada pada garis hitam) Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

4. Kecepatan

   *Subroutine* kecepatan berfungsi member nilai PWM atau member nilai kecepatan motor. Gambar 3.26 memperlihatkan diagram alir *subroutine* ini.



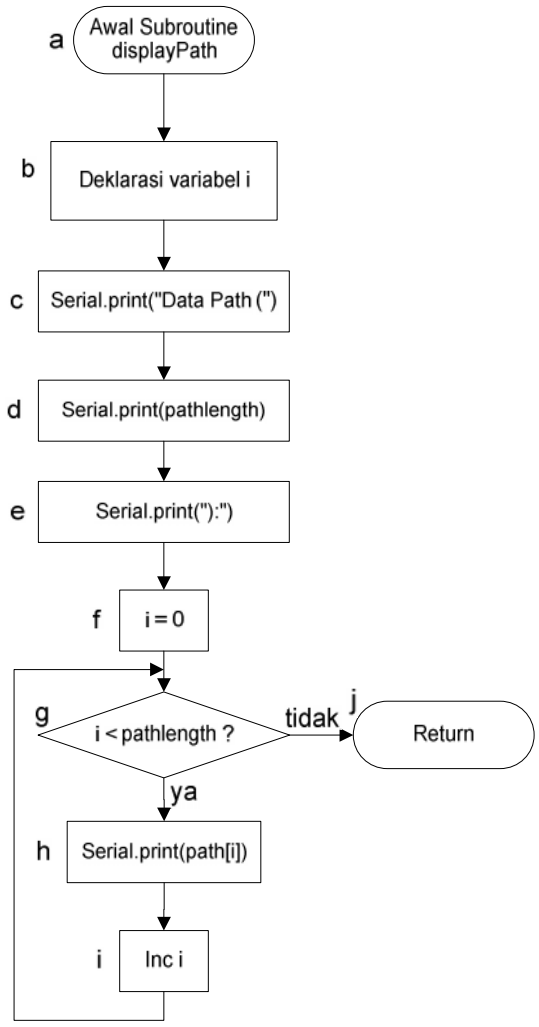

Gambar 3.26 Diagram alir subroutine kecepatan

Berikut ini penjelasan diagram alir *subroutinekecepatan*:

a. Memulai diagram alir *subroutine finish*
b. Deklarasi variabel.
c. Inisialisasi pin 4 (kanan) dan nilai PWM yang diwkili oleh variabel rspeed, Inisialisasi pin 5 (kiri) dan nilai PWM yang diwkili oleh variabel lspeed.
d. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

#### 3.3.3.4. Subroutine displayPath

*Subroutine* ini berfungsi untuk menampilkan data langkah robot setelah disimplifikasi (disederhanakan) pada monitor. Data langkah tersebut disimpan pada sebuah *array* berupa karakter (L, R, S) yang akan mewakili pergerakan robot pada saat robot melakukan running pada mode ke 2.




Gambar 3.27 Diagram alir subroutine displayPath

Berikut ini penjelasan diagram alir *subroutinekecepatan*:

a. Memulai diagram alir *subroutine displayPath.*
b. Deklarasi variabel i.
c. Cetak kalimat “Data Path (“ pada monitor.
d. Cetak nilai yang terdapat pada variabel *pathlength* pada monitor.
e. Cetak karakter “):” pada monitor.
f. Isi variabel I dengan nilai 0.
g. Periksa apakah nilai variabel i lebih kecil daripada variabel pathlength.

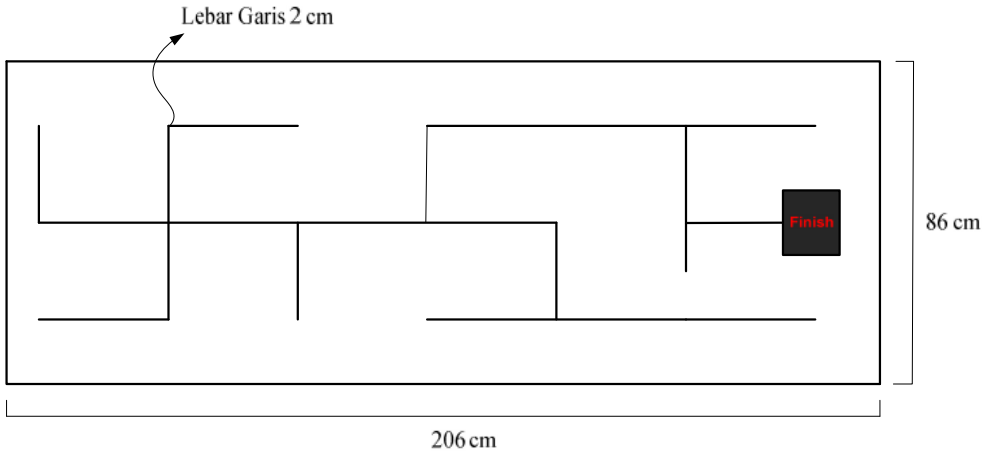
h. Cetak karakter yang terdapat pada *array* path ke i.
i. Keluar dari *subroutine* dan kembali mengeksekusi instruksi berikutnya pada program utama.

### 3.4. Maket Lapangan

Berikut ini adalah bentuk *track* yang akan digunakan dengan spesifikasi sebagai berikut:

1. Bentuk *track* berdimensi 206 cm x 86 cm yang bahannya terbuat dari papan putih.
2. Lebar garisnya 2 cm yang terbuat dari isolasi warna hitam.

Gambar 3.25 memperlihatkan miniatur *track* yang digunakan.




Gambar 3.28 Miniatur Track