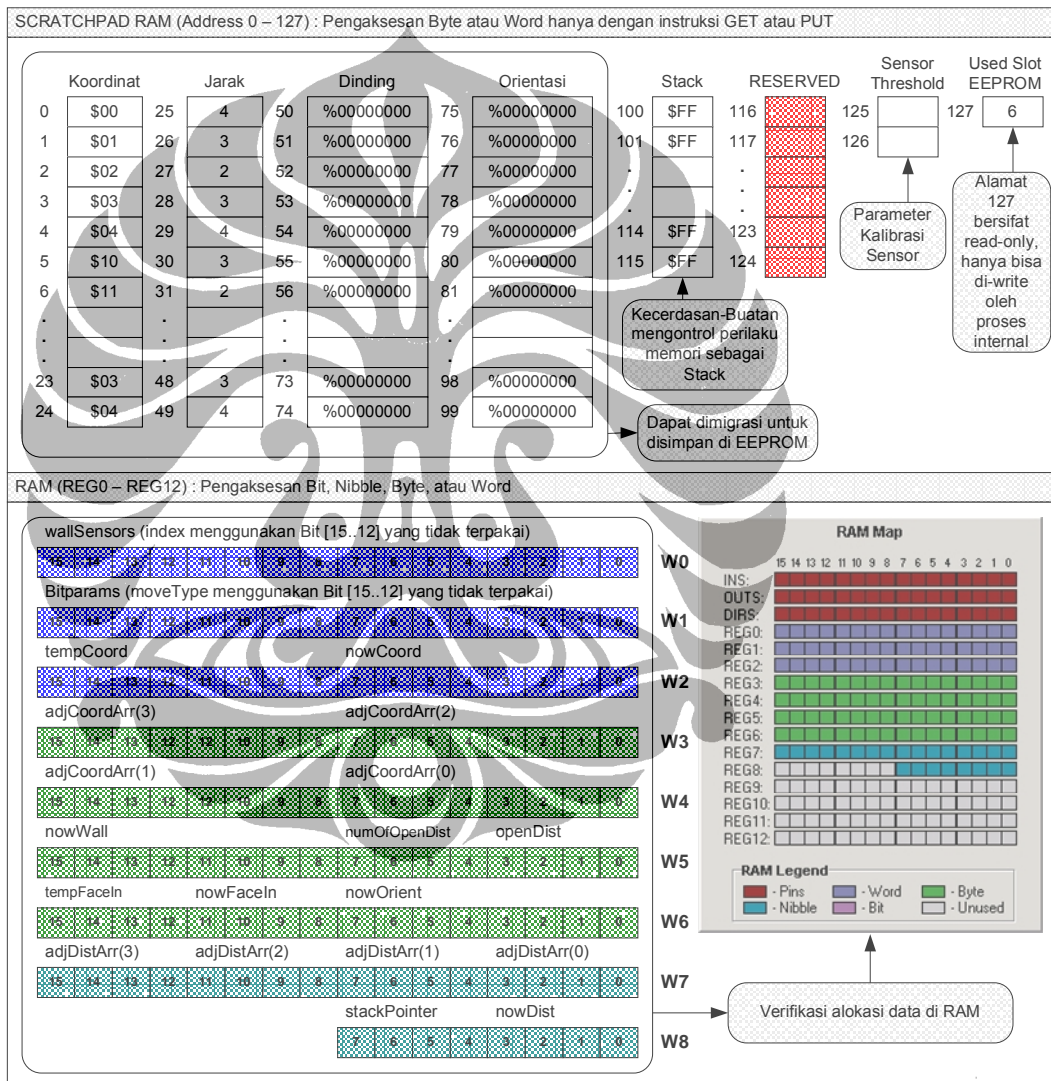


BAB IV IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR

Implementasi kecerdasan-buatan robot pencari jalur berfokus pada algoritma pemrograman untuk menangani rancangan struktur data yang dibuat seperti diperlihatkan Gambar 4.1 di bawah ini.



Gambar 4.1. Implementasi Rancangan Data Kecerdasan-Buatan Robot Pencari Jalur

Berdasarkan Gambar 4.1, secara umum data ditempatkan di dua jenis memori yang berbeda, yaitu:

1. Scratchpad RAM (SP RAM)

Dengan karakteristik hanya bisa diakses menggunakan instruksi PUT dan GET, SP RAM menyimpan jenis data kecerdasan-buatan sebagai berikut:

- a. Koordinat Cell, berfungsi untuk menyimpan informasi koordinat cell di mana robot pencari jalur berada di dalam labirin. Lebar data koordinat cell adalah 8 bit, dengan high nibble (4 bit atas) digunakan untuk menyimpan nilai koordinat x, dan low nibble (4 bit bawah) digunakan untuk menyimpan nilai koordinat y. Jadi nilai maksimum yang bisa disimpan untuk masing-masing nilai x dan y adalah 15.
- b. Jarak Cell, terkait dengan data Koordinat Cell, berfungsi untuk menyimpan informasi jarak cell tersebut dari cell tujuan. Lebar data jarak cell adalah 8 bit. Jadi nilai maksimum yang bisa disimpan adalah 255.
- c. Dinding Cell, terkait dengan data Koordinat Cell, mempunyai lebar data 8 bit, di mana bit 6 diset jika cell tersebut merupakan path untuk sampai ke cell tujuan, bit 5-4 diset jika updating dinding cell telah dilakukan, dan low nibble (bit 3-0) untuk menyimpan informasi dinding cell sebelah barat, selatan, timur, dan utara.
- d. Orientasi Cell, terkait dengan data Koordinat Cell, mempunyai lebar data 8 bit, berfungsi untuk menyimpan informasi orientasi pergerakan robot pencari jalur, yaitu orientasi ke arah barat, selatan, timur, atau utara.
- e. Stack 8 bit, berfungsi sebagai penyimpanan sementara data Koordinat Cell dalam pemrosesan data Jarak Cell berbasis stack menggunakan algoritma Flood-Fill dan algoritma modified Flood-Fill.
- f. Sensor Threshold, dengan ukuran 1 word (2 byte) pada alamat memori 125 dan 126 dari SP RAM, digunakan untuk menyimpan nilai parameter terkait dengan routine kalibrasi sensor proximity robot pencari jalur.

Karena labirin tersusun atas 25 cell (5 cell x 5 cell), keempat jenis data yang pertama ini masing-masing dialokasikan sebanyak 25 alamat memori untuk penyimpanan data.

Dapat dilihat pada Tabel 4.1, pengalokasian alamat memori untuk ketiga jenis data ini adalah sebagai berikut:

Tabel 4.1. Alokasi Alamat Memori Data Cell

Data	Lokasi Memori di Scratchpad RAM
Koordinat Cell	0 - 24
Jarak Cell	25 - 49
Dinding Cell	50 - 74
Orientasi Cell	75 - 99

Inisialisasi data awal di alamat stack (100 - 115) dilakukan oleh kecerdasan-buatan dengan memberikan nilai \$FF = %11111111 (ukuran 1 byte).

2. RAM

Dengan karakteristik pengaksesan seperti variabel, yaitu bisa langsung diberikan suatu nilai tertentu dengan terlebih dahulu dideklarasikan, RAM menyimpan jenis data sebagai berikut:

- a. *wallSensors* (1 Word = 2 Byte), berfungsi untuk menyimpan data 12 sensor (bit 11-0). Sisa 4 bit (bit 15-12) digunakan untuk menyimpan data *index* untuk keperluan looping di dalam program.
- b. *bitparams* (1 Word), berfungsi untuk menyimpan bit-bit kontrol kecerdasan-buatan, yaitu:

Tabel 4.2. Bit-Bit Kontrol di RAM

Bit	Parameter	Fungsi
0	<i>isBrokenRule</i>	status memenuhi rule algoritma dalam perhitungan jarak cell
1	<i>isCheckpoint</i>	status validitas pencapaian checkpoint (dinding depan & belakang)
2	<i>isFinish</i>	status pencapaian cell tujuan
3	<i>isNoWall</i>	status updating dinding cell
4	<i>isSideWall</i>	status validitas dinding samping
5	<i>isSetPath</i>	status pengesetan jalur ke cell tujuan
6	<i>isVisitedCell</i>	status kunjungan ke suatu cell
7	<i>scanResult</i>	status scan dinding cell setelah transisi antar cell
8	<i>task</i>	switching task di internal program
9	RESERVED	
10	RESERVED	
11	RESERVED	
12-15	<i>moveType</i>	jenis manuver robot (maju, rotasi 90 ⁰ kanan atau kiri, rotasi 180 ⁰ kanan)

- c. *tempCoord* (1 Byte), berfungsi sebagai buffer data koordinat dari Scratchpad RAM untuk pemrosesan.

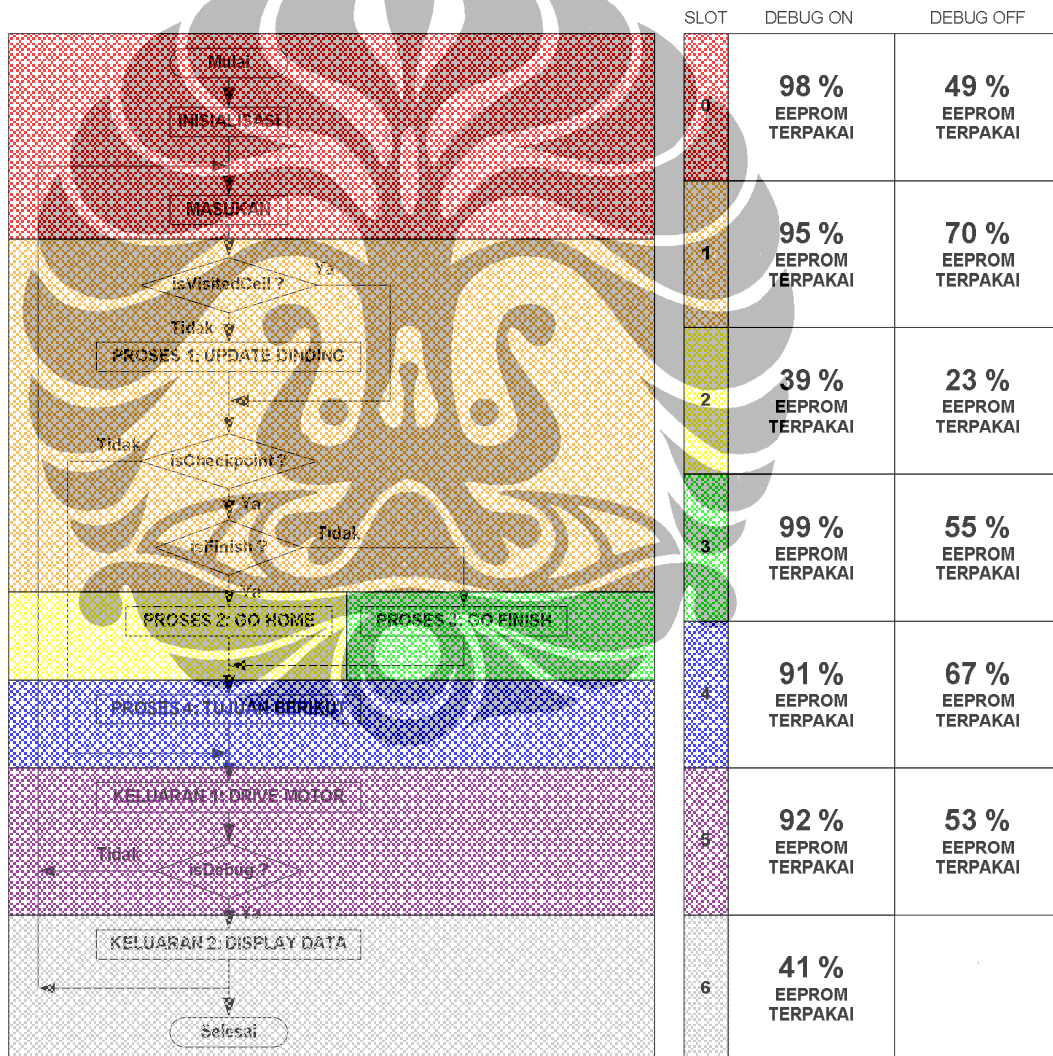
- d. *nowCoord* (1 Byte), berfungsi sebagai buffer data koordinat dari Scratchpad RAM, untuk pemrosesan lebih lanjut.
- e. *adjCoordArray* (array Byte (4)), berfungsi untuk menyimpan data koordinat cell yang bersebelahan tanpa dinding pembatas.
- f. *nowWall* (1 Byte), berfungsi sebagai buffer data dinding dari Scratchpad RAM, untuk pemrosesan lebih lanjut.
- g. *numOfOpenDist* (1 Nibble), berfungsi sebagai informasi jumlah cell bersebelahan tanpa dinding pembatas.
- h. *openDist* (1 Nibble), berfungsi sebagai mask untuk menentukan cell bersebelahan tidak mempunyai dinding dan mempunyai jarak terkecil atau terbesar.
- i. *tempFaceIn* (1 Nibble), berfungsi sebagai penampung orientasi robot pada cell berikut.
- j. *nowFaceIn* (1 Nibble), berfungsi sebagai penampung orientasi robot pada cell di mana sekarang berada.
- k. *nowOrient* (1 Byte), berfungsi sebagai buffer data orientasi dari Scratchpad RAM, untuk pemrosesan lebih lanjut.
- l. *adjDistArray* (array Nibble (4)), berfungsi untuk menyimpan data jarak cell yang bersebelahan tanpa dinding pembatas.
- m. *nowDist* (1 Nibble), berfungsi sebagai buffer data jarak dari Scratchpad RAM, untuk pemrosesan lebih lanjut.
- n. *stackPointer* (1 Nibble), berfungsi sebagai informasi pointer pada stack di Scratchpad RAM.

Verifikasi alokasi data di RAM bisa dilihat pada Gambar 4.1 di mana fasilitas RAM Map dari *Integrated Development Environment* (IDE) pemrograman memberikan visualisasi mapping rancangan data yang bersesuaian.

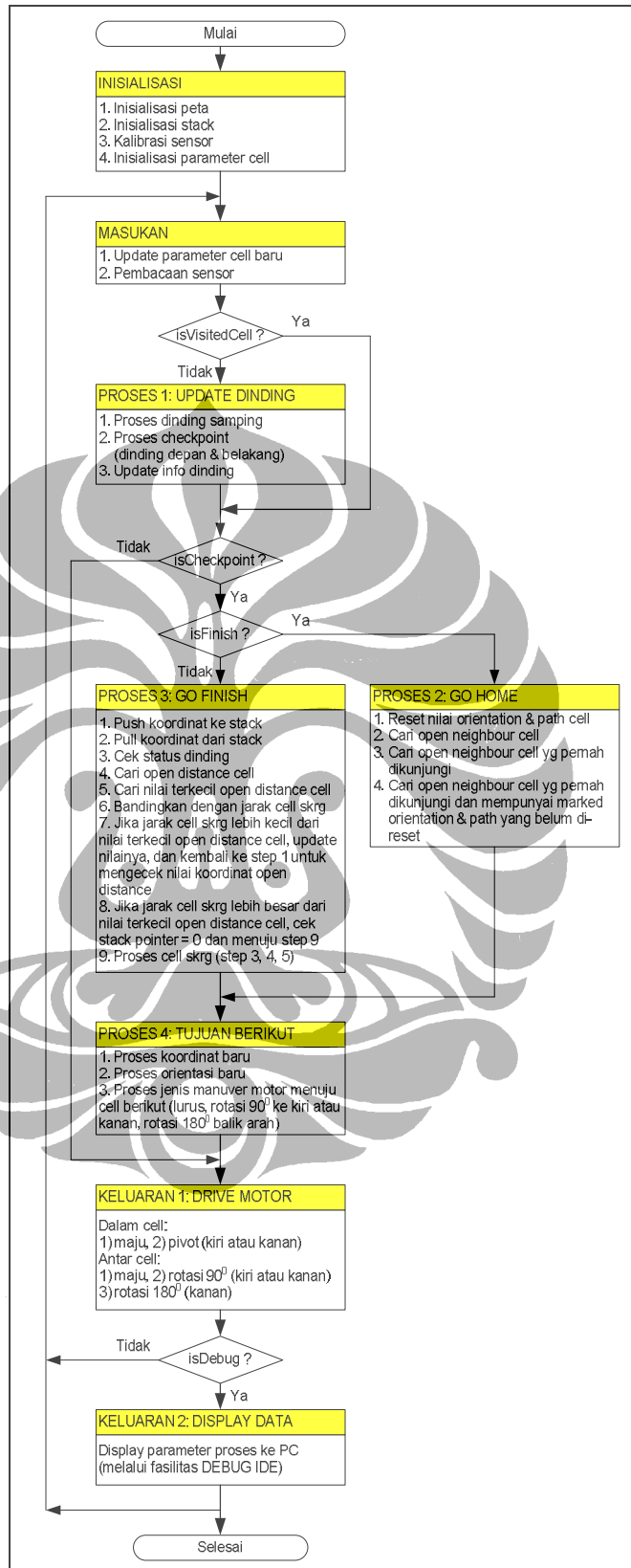
4.1 IMPLEMENTASI UMUM

Implementasi umum kecerdasan-buatan robot pencari jalur bisa dilihat pada diagram alir keseluruhan kerja sistem pada Gambar 4.2, yang merupakan suatu proses looping besar. Looping tidak diperlihatkan untuk menghindari kompleksitas gambar. Di sini hanya diperlihatkan bagian-bagian besar kode pembentuk looping, yang identik dengan lokasi bank EEPROM sebagai targetnya.

Seperti terlihat pada Gambar 4.2, terdapat enam bagian besar kode yang di *mapping* ke bank EEPROM yang bersesuaian. Keenam bagian besar ini diturunkan dari konsep tiga besar penyusun suatu sistem, yaitu masukan, proses, dan keluaran.



Gambar 4.2. Flowchart Sederhana dan Pemakaian Resources EEPROM Kecerdasan-Buatan Robot Pencari Jalur



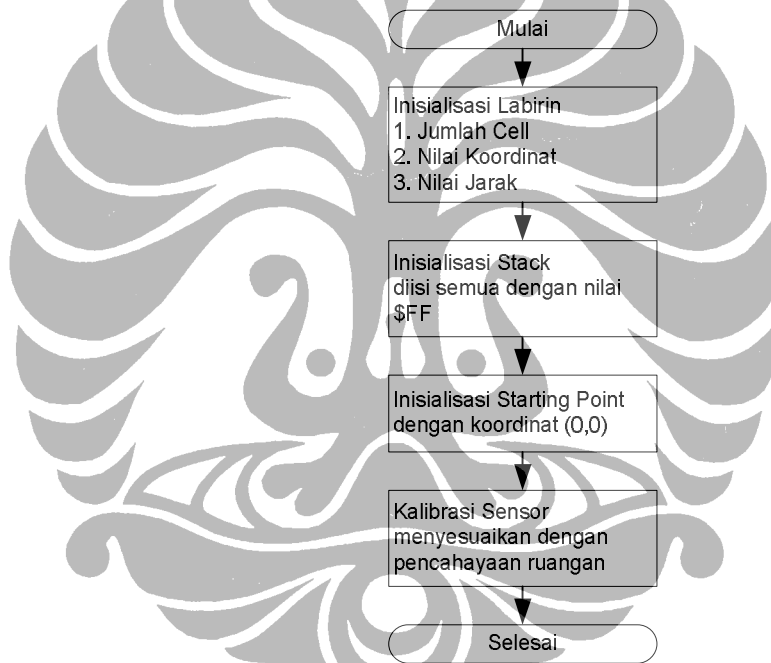
Gambar 4.3. Flowchart Detail Kecerdasan-Buatan Robot Pencari Jalur

4.2 PEMROSESAN INISIALISASI DAN MASUKAN

Berdasarkan Gambar 4.4, Pemrosesan inisialisasi kecerdasan buatan meliputi empat hal, yaitu:

1. Inisialisasi peta labirin
2. Inisialisasi nilai stack
3. Inisialisasi paramater awal pada starting point atau cell
4. Kalibrasi sensor

Di sini hanya akan diuraikan bagian inisialisasi peta labirin dan kalibrasi sensor berhubung inisialisasi nilai stack dan parameter awal hanya merupakan pemberian suatu nilai di lokasi memori tertentu atau variabel.



Gambar 4.4. Diagram Alir Proses Inisialisasi Kecerdasan-Buatan

Gambar 4.5 memperlihatkan bagian kode yang menginisialisasi pembuatan peta labirin dengan nilai jarak awal untuk masing-masing cell tanpa adanya dinding pembatas.

Mekanisme kerja kode adalah mengisi data untuk kordinat dan jarak pada alamat memori yang telah disediakan di Scratchpad RAM. Empat segmen warna memperlihatkan urutan proses yang. Segmen merah diproses pertama, diikuti segmen oranye, kemudian segmen kuning, dan terakhir segmen hijau.

```

B23 = 0

FOR nowX = 0 TO MatrixSizeMinSatu
  FOR nowY = 0 TO MatrixSizeMinSatu
    IF (nowX < (MatrixSizeMinSatu / 2)) THEN
      IF (nowY < (MatrixSizeMinSatu / 2)) THEN
        nowDist = (MatrixSizeMinSatu - nowX) - nowY
      ELSE
        nowDist = nowY - nowX
      ENDIF
    ELSE
      IF (nowY < (MatrixSizeMinSatu / 2)) THEN
        nowDist = nowX - nowY
      ELSE
        nowDist = nowY - (MatrixSizeMinSatu - nowX)
      ENDIF
    ENDIF
    B24 = CoordinateValue + B23
    B25 = nowCoord
    GOSUB Put_Byte
    B24 = DistanceValue + B23
    B25 = nowDist
    GOSUB Put_Byte
    B23 = B23 + 1
  NEXT
NEXT

Put_Byte:
  PUT B24, B25
RETURN

```

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

- Proses Pertama
- Proses Kedua
- Proses Ketiga
- Proses Keempat

B23 = variabel byte sementara

MatrixSize = ukuran matrik = 5
 MatrixSizeMinSatu = MatrixSize - 1

cellCoord = variabel koordinat
 x = cellCoord.HIGHNIB
 y = cellCoord.LOWNIB

cellDist = variabel nilai jarak

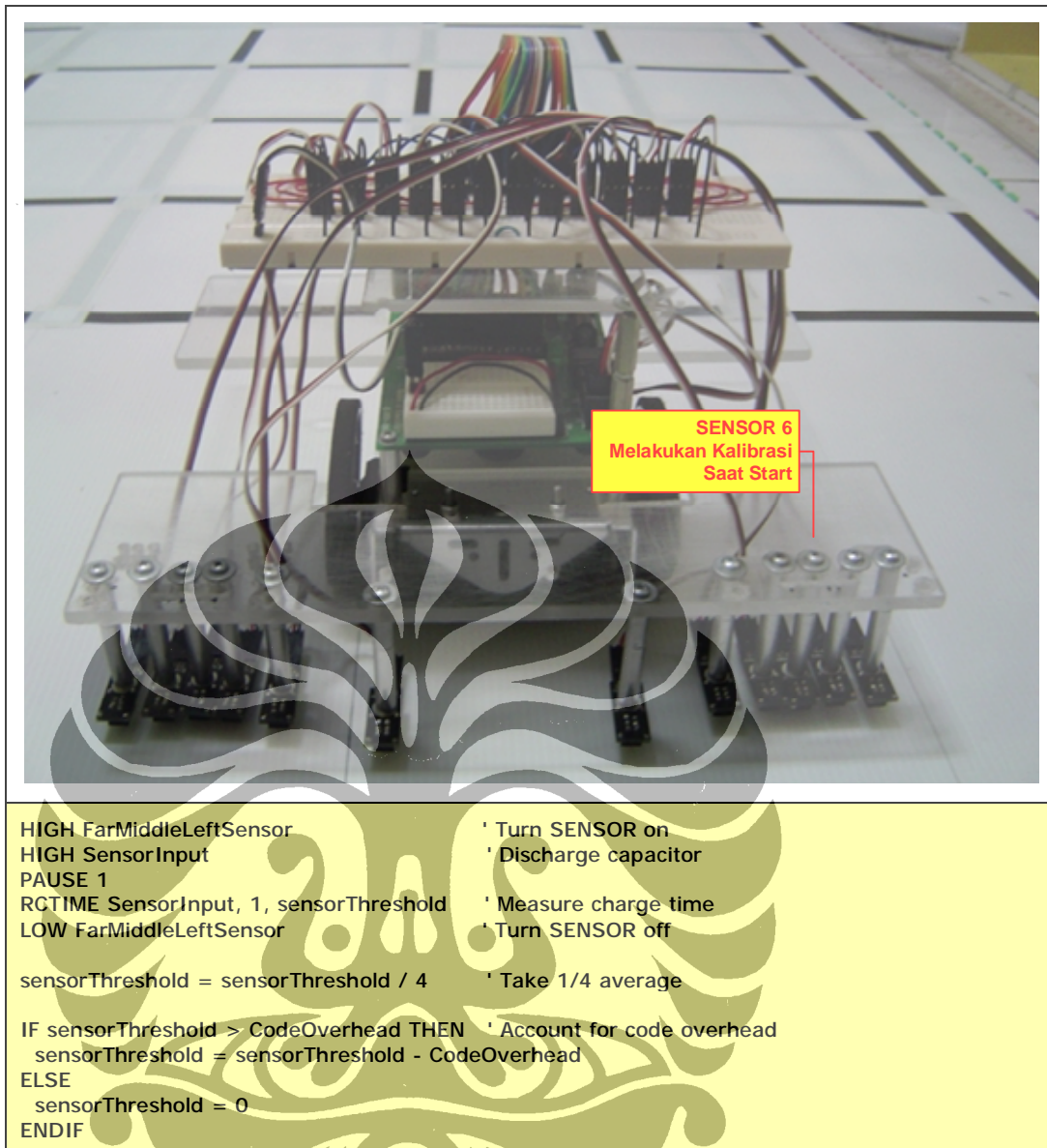
Write_Data = rutin menyimpan data ke Scratchpad RAM

Gambar 4.5. Kode Inisiasi Peta Awal Labirin

Gambar 4.6 memperlihatkan bagian kode kalibrasi sensor. Tujuan dikalibrasi adalah memberikan suatu nilai ambang batas (*sensorThreshold*) yang relatif tidak terpengaruh oleh intensitas cahaya suatu tempat di mana robot berada. Nilai ambang batas itu sendiri digunakan untuk pembedaan pembacaan permukaan berwarna hitam dan putih oleh sensor infra merah.

Mekanisme kerja kode kalibrasi, yaitu pada start awal robot pencari jalur, sensor kalibrasi, yaitu sensor 6, harus diposisikan menghadap permukaan berwarna hitam. Jika dihadapkan pada permukaan berwarna putih, kalibrasi tidak akan berjalan benar.

Nilai ambang batas yang diperoleh melalui pembacaan sensor 6 kemudian dibagi empat untuk memperoleh nilai ambang batas yang diinginkan. Nilai ini ditampung oleh variabel *sensorThreshold* yang dialokasikan di RAM.

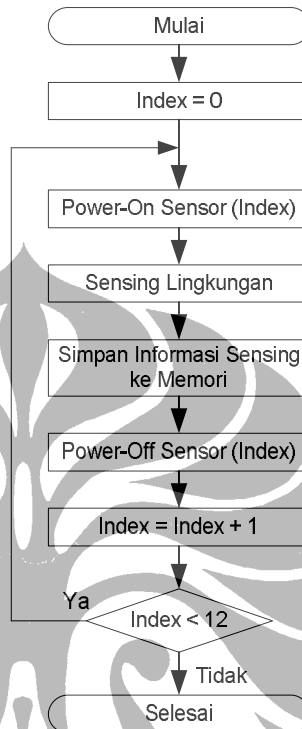


Gambar 4.6. Kode Kalibrasi Sensor Pada Pencahayaan Tertentu

Gambar 4.7 memperlihatkan diagram alir dari proses kerja sensor untuk robot pencari jalur. Terdapat dua belas sensor yang mendeteksi permukaan labirin dua dimensi, apakah berwarna hitam atau putih. Gambar 4.8 memberikan visualisasi posisi sensor pada robot (tampak atas dan tampak bawah), sedangkan Tabel 4.3 memberikan pemakaian pin IO mikrokontroler BASIC Stamp untuk sensor.

Berdasarkan Tabel 4.3, lima belas pin IO (dari enam belas yang tersedia) terpakai semua. Satu pin IO yaitu pin 15 pun tidak bisa digunakan karena adanya pemakaian untuk keperluan timing pembacaan sensor.

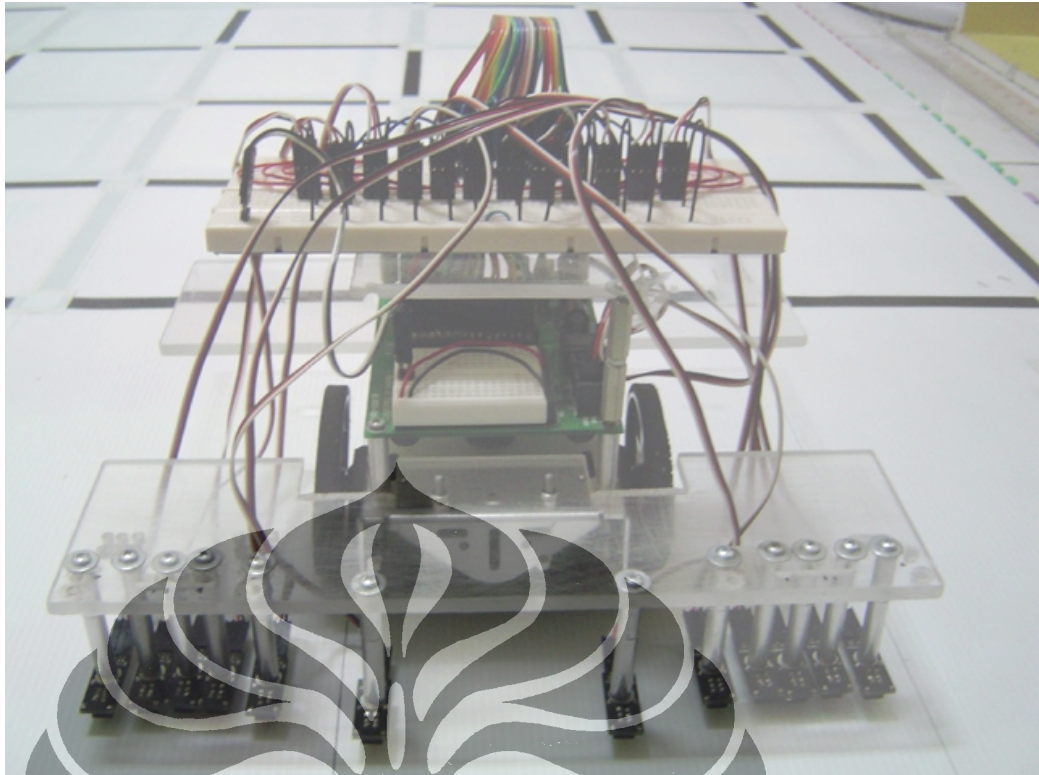
Gambar 4.7 memperlihatkan mekanisme kerja sensor yang hanya diaktifkan pada saat diperlukan saja. Ini menghemat pemakaian arus sehingga robot pencari jalur bisa bergerak lebih lama jika menggunakan sumber energi portabel, seperti baterai.



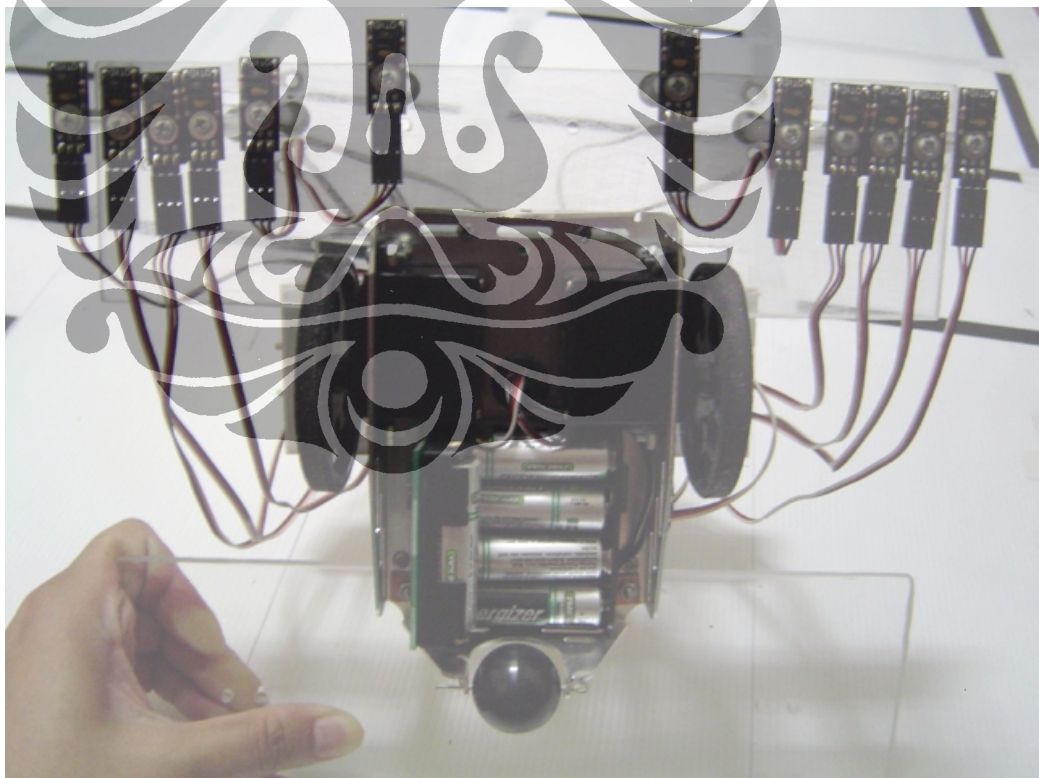
Gambar 4.7. Diagram Alir Mekanisme Kerja Sensor

Tabel 4.3. Pemakaian Pin Robot Pencari Jalur

NOMOR PIN	PEMAKAIAN
0	Keluaran ke catu daya sensor kanan luar
1	Keluaran ke catu daya sensor kanan tengah
2	Keluaran ke catu daya sensor kanan tengah
3	Keluaran ke catu daya sensor kanan dalam
4	Keluaran ke catu daya sensor kiri dalam
5	Keluaran ke catu daya sensor kiri tengah
6	Keluaran ke catu daya sensor kiri tengah
7	Keluaran ke catu daya sensor kiri luar
8	Keluaran ke catu daya sensor kanan belakang luar (penempatan tidak di belakang)
9	Keluaran ke catu daya sensor kanan belakang dalam (penempatan tidak di belakang)
10	Keluaran ke catu daya sensor kiri belakang luar (penempatan tidak di belakang)
11	Keluaran ke catu daya sensor kiri belakang dalam (penempatan tidak di belakang)
12	Keluaran ke kontrol motor kanan
13	Keluaran ke kontrol motor kiri
14	Masukan data dari semua sensor (<i>time-switching</i>)
15	Penggunaan internal untuk pewaktuan



(a)

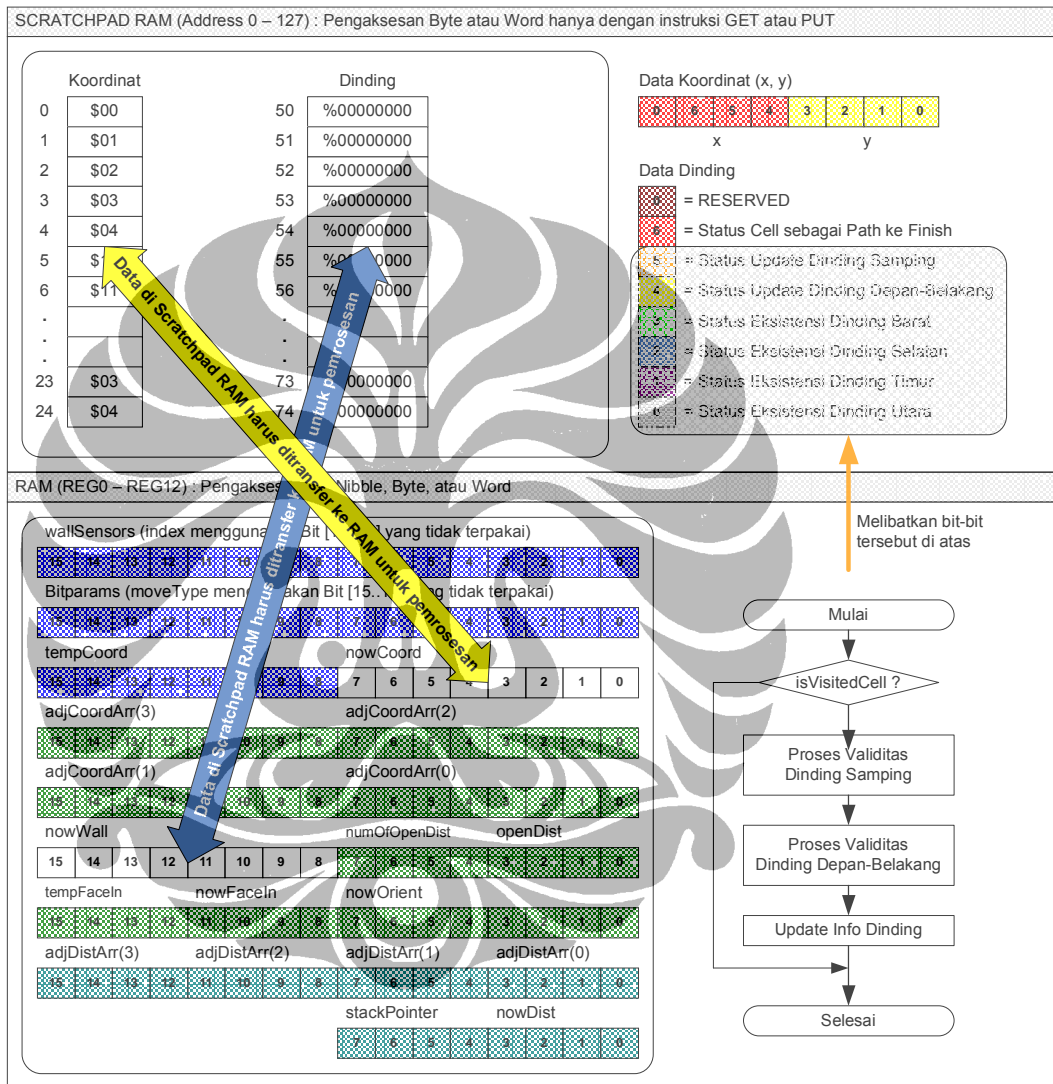


(b)

Gambar 4.8. Sensor Proximity dari Robot Pencari Jalur
(a) Tampak Atas (b) Tampak Bawah

4.3 PEMROSESAN INFORMASI DINDING CELL

Bagian besar kedua sesuai dengan Gambar 4.2 adalah kode pemroses eksistensi dinding cell. Setiap kali robot pencari jalur memasuki cell dengan status belum pernah dikunjungi (bit 5-4 = 00b pada Data Dinding di Gambar 4.9) , maka kecerdasan-buatan akan mengecek eksistensi dinding cell.

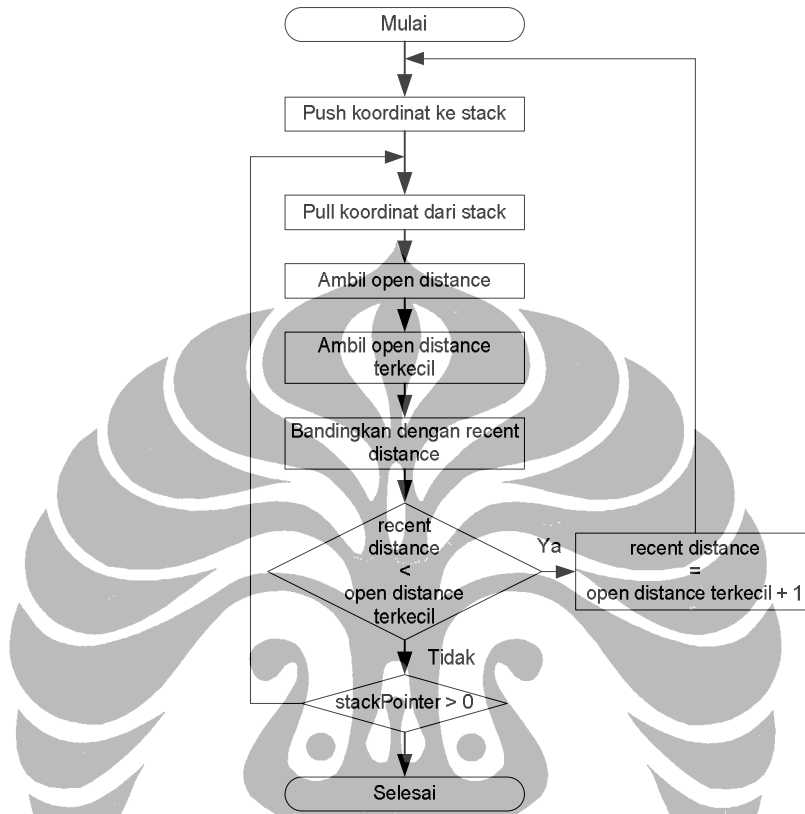


Gambar 4.9. Pemrosesan Informasi Dinding Cell

Berdasarkan Gambar 4.9, dalam proses ini di mana Data Jarak dan Orientasi tidak terlibat, diperlihatkan bit-bit Data Dinding yang akan ter-update, beserta buffer-buffer yang digunakan untuk menampung data dari Scratchpad RAM.

4.4 PEMROSESAN INFORMASI JARAK CELL (GO FINISH)

Bagian besar ketiga sesuai dengan Gambar 4.2 adalah kode pemroses jarak cell. Kode ini berbasis pemrosesan stack dengan diagram alir diperlihatkan pada Gambar 4.10.

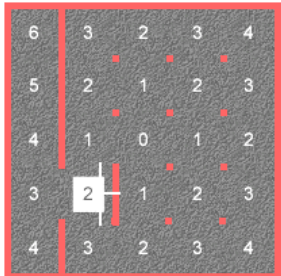
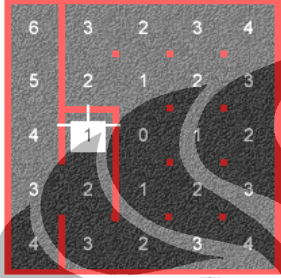
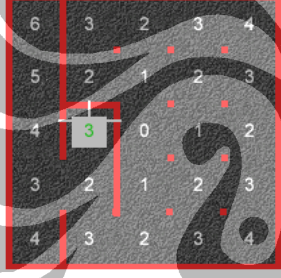


Gambar 4.10. Pemrosesan Informasi Jarak Cell Berbasis Stack

Push_Coordinate:	Pull_Coordinate:
<pre> IF (stackPointer > 0) THEN FOR B23 = stackPointer-1 TO 0 B24 = StackValue + B23 GOSUB Get_Byte B24 = StackValue + (B23 + 1) GOSUB Put_Byte NEXT ENDIF B24 = StackValue B25 = tempCoord GOSUB Put_Byte stackPointer = stackPointer + 1 RETURN </pre>	<pre> B24 = StackValue GOSUB Get_Byte tempCoord = B25 IF (stackPointer > 0) THEN FOR B23 = 0 TO stackPointer-1 B24 = StackValue + (B23 + 1) GOSUB Get_Byte B24 = StackValue + B23 GOSUB Put_Byte NEXT ENDIF stackPointer = stackPointer - 1 RETURN </pre>

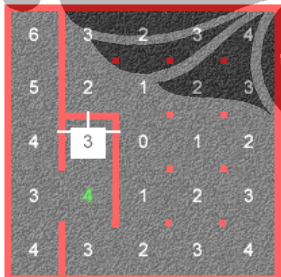
Gambar 4.11. Kode Kecerdasan-Buatan untuk Push dan Pull

Simulasi kerja kode pemroses jarak cell ini diperlihatkan pada Gambar 4.12 yang mengambil sebagian simulasi deskriptif pada bagian sebelumnya.

9. 
 - Push koordinat (1,1) ke stack
 - Pull koordinat (1,1) dari stack
 - Ambil open distance, yaitu 3 (sisi barat), 3 (sisi selatan), dan 1 (sisi utara)
 - Ambil open distance terkecil, yaitu 1 (sisi utara)
 - Apakah recent distance (2) < open distance terkecil (1) (tidak menyalahi aturan)
 - Tidak, robot pindah ke cell dengan open distance terkecil, yaitu koordinat baru (1,2), sepanjang tidak ada data lagi di stack (stackPointer = 0)
10. 
 - Push koordinat (1,2) ke stack
 - Pull koordinat (1,2) dari stack
 - Ambil open distance, yaitu 2 (sisi selatan)
 - Ambil open distance terkecil, yaitu 2
 - Apakah recent distance (1) < open distance terkecil (2) (menyalahi aturan)
11. 
 - Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 2 + 1 = 3
 - Karena menyalahi aturan, push koordinat open distance, yaitu (1,1), ke stack
 - stackPointer = stackPointer + 1 = 0 + 1 = 1

addr Stack

75	\$11
----	------

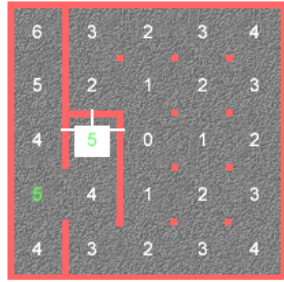
 - Karena stackPointer > 0, pull koordinat (1,1) dari stack
 - stackPointer = stackPointer - 1 = 1 - 1 = 0
 - Ambil open distance, yaitu 3 (sisi barat), 3 (sisi selatan), dan 3 (sisi utara)
 - Ambil open distance terkecil, yaitu 3
 - Apakah recent distance (2) < open distance terkecil (3) (menyalahi aturan)
12. 
 - Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 3 + 1 = 4
 - Karena menyalahi aturan, push koordinat open distance, yaitu (0,1), (1,0), dan (1,2), ke stack
 - 3 kali push maka stackPointer = 3

addr Stack

75	\$12
76	\$10
77	\$01

 - Karena stackPointer > 0, pull koordinat (1,2) dari stack
 - stackPointer = stackPointer - 1 = 3 - 1 = 2
 - Ambil open distance, yaitu 4 (sisi selatan)
 - Ambil open distance terkecil, yaitu 4
 - Apakah recent distance (3) < open distance terkecil (4) (menyalahi aturan)

13.



- Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 4 + 1 = 5
- Karena menyalahi aturan, push koordinat open distance, yaitu (1,1), ke stack
- stackPointer = stackPointer + 1 = 2 + 1 = 3

addr Stack

75	\$11
76	\$10
77	\$01

- Karena stackPointer > 0, pull koordinat (1,1) dari stack
- stackPointer = stackPointer - 1 = 3 - 1 = 2
- Ambil open distance, yaitu 3 (sisi barat)
- Ambil open distance terkecil, yaitu 3
- Apakah recent distance (4) < open distance terkecil (3) (tidak menyalahi aturan)
- Tidak, Karena stackPointer > 0, pull koordinat (1,0) dari stack
- stackPointer = stackPointer - 1 = 2 - 1 = 1
- Ambil open distance, yaitu 4 (sisi utara), dan 2 (sisi timur)
- Ambil open distance terkecil, yaitu 2
- Apakah recent distance (3) < open distance terkecil (2) (tidak menyalahi aturan)
- Tidak, Karena stackPointer > 0, pull koordinat (0,1) dari stack
- stackPointer = stackPointer - 1 = 1 - 1 = 0
- Ambil open distance, yaitu 4 (sisi selatan), 4 (sisi timur), dan 4 (sisi utara)
- Ambil open distance terkecil, yaitu 4
- Apakah recent distance (3) < open distance terkecil (4) (menyalahi aturan)
- Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 4 + 1 = 5
- Karena menyalahi aturan, push koordinat open distance, yaitu (0,0), (1,1), dan (0,2), ke stack
- 3 kali push maka stackPointer = 3

addr Stack

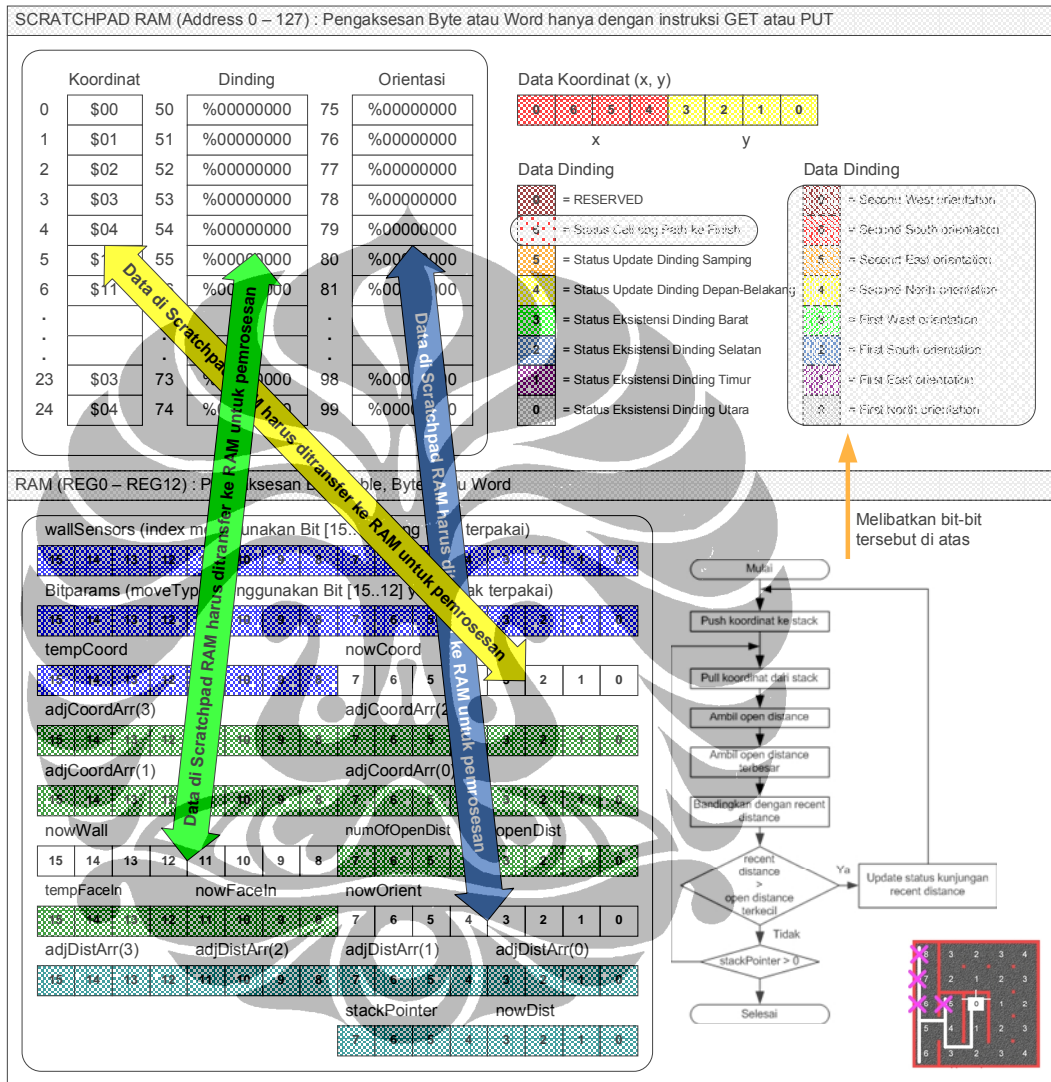
75	\$02
76	\$11
77	\$00

- dan seterusnya, proses berlanjut untuk meng-update nilai jarak

Gambar 4.12. Simulasi Pemrosesan Informasi Jarak Cell Berbasis Stack

4.5 PEMROSESAN STATUS KUNJUNGAN CELL (GO HOME)

Bagian besar keempat sesuai dengan Gambar 4.2 adalah kode pemroses status kunjungan di suatu cell. Data Orientation dan bit 6, 5, dan 4 pada Data Dinding di Gambar 4.13, merupakan data yang akan diproses pada bagian ini.

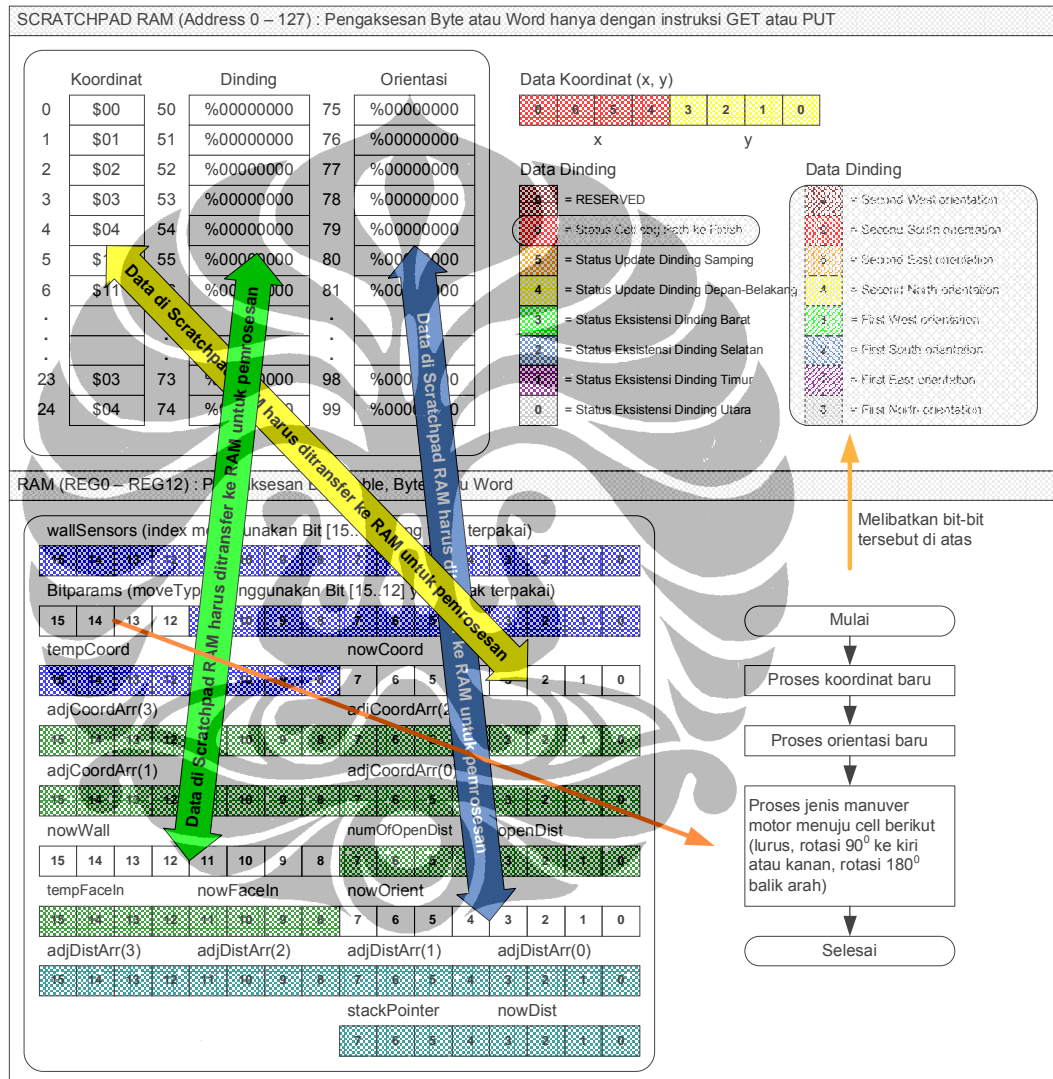


Gambar 4.13. Pemrosesan Status Kunjungan Cell Berbasis Stack

Tujuan kode pada bagian ini adalah untuk menavigasi robot pencari jalur jika sudah sampai di cell tujuan (flag *isFinish* = 1) untuk kembali ke cell awal. Gambar 4.13 memperlihatkan labirin dengan status kunjungan di beberapa cell yang di-reset sehingga robot hanya mempunyai satu jalur pasti untuk kembali.

4.6 PEMROSESAN TUJUAN

Bagian besar kelima sesuai dengan Gambar 4.2 adalah kode pemroses tujuan ke cell berikut. Gambar 4.14 memperlihatkan Data Jarak tidak terlibat dalam proses, sedangkan pada Data Dinding, bit 7-6 merupakan bit-bit yang menentukan orientasi pergerakan robot selanjutnya, apakah ke arah barat, selatan, timur, atau utara.

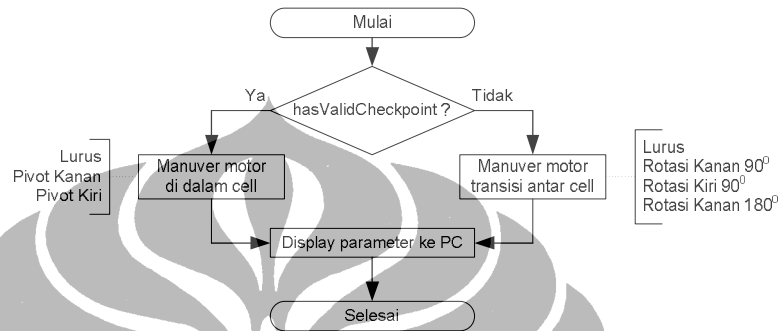


Gambar 4.14. Pemrosesan Tujuan

Gambar 4.14 memperlihatkan juga variabel *moveType* di RAM yang mengatur jenis manuver robot pencari jalur pada saat menuju cell berikutnya.

4.7 PEMROSESAN MANUEVER MOTOR

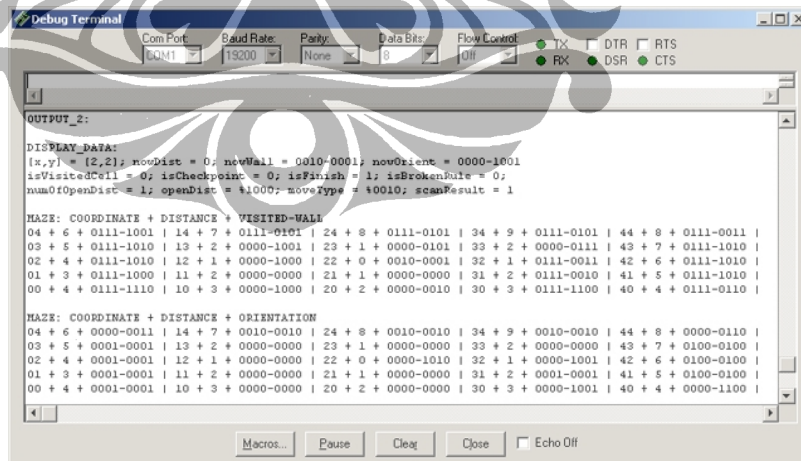
Bagian besar kelima sesuai dengan Gambar 4.2 adalah kode pemroses keluaran. Gambar 4.15 memperlihatkan bit kontrol *isCheckpoint* mengatur manuver robot, yaitu apakah melakukan manuver di dalam cell atau melakukan manuver transisi antar cell. Dua jenis manuver tersebut mempunyai jenis-jenis pergerakan dasar seperti dapat dilihat pada Gambar 4.14.



Gambar 4.15. Pemrosesan Keluaran

4.8 PEMROSESAN DISPLAY

Bagian ini memproses monitoring sistem parameter internal dengan menampilkannya ke layar komputer melalui komunikasi serial antara robot dan komputer. Bagian monitoring ini sangat berguna dalam proses *debugging* untuk mencari *bug-bug* selama *runtime*.



Gambar 4.16. Display Parameter Internal Proses ke PC

Bagian ini merupakan bagian terakhir dari tujuh bagian besar kode, untuk selanjutnya proses berlanjut dengan melakukan looping ke bagian besar pertama.