PERANCANGAN DAN IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR BERBASIS MIKROKONTROLER BASIC STAMP

TESIS

Oleh

GEDE INDRAWAN
0606003410

PROGRAM STUDI TEKNIK ELEKTRO
PROGRAM PASCA SARJANA BIDANG ILMU TEKNIK
UNIVERSITAS INDONESIA
GANJIL 2007/2008

PERANCANGAN DAN IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR BERBASIS MIKROKONTROLER BASIC STAMP

TESIS

Oleh

GEDE INDRAWAN
0606003410

TESIS INI DIAJUKAN UNTUK MELENGKAPI SEBAGIAN PERSYARATAN MENJADI MAGISTER TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO
PROGRAM PASCA SARJANA BIDANG ILMU TEKNIK
UNIVERSITAS INDONESIA
GANJIL 2007/2008

PERNYATAAN KEASLIAN TESIS

Saya menyatakan dengan sesungguhnya bahwa tesis dengan judul :

PERANCANGAN DAN IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR BERBASIS MIKROKONTROLER BASIC STAMP

yang dibuat untuk melengkapi sebagian persyaratan menjadi Magister Teknik pada Kekhususan Aplikasi Mikroprosesor Program Studi Teknik Elektro Program Pascasarjana Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari tesis yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau Instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, Januari 2008

Gede Indrawan

0606003410

PENGESAHAN

Tesis dengan judul

PERANCANGAN DAN IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR BERBASIS MIKROKONTROLER BASIC STAMP

dibuat untuk melengkapi sebagian persyaratan menjadi Magister Teknik pada Kekhususan Aplikasi Mikroprosesor Program Studi Teknik Elektro Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia. Tesis ini telah diujikan pada sidang ujian tesis pada tanggal 2 Januari 2008 dan dinyatakan memenuhi syarat/sah sebagai tesis pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, Januari 2008

Dosen Pembimbing

Prof. Dr-Ing. Ir. Harry Sudibyo S., DEA

NIP 130891668

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada :

Prof. Dr-Ing. Ir. Harry Sudibyo S., DEA

selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan serta persetujuan sehingga tesis ini dapat selesai dengan baik.

Gede Indrawan
NPM 0606003410
Departemen Teknik Elektro

Dosen Pembimbing
Prof. Dr-Ing. Ir. Harry Sudibyo S., DEA

PERANCANGAN DAN IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR BERBASIS MIKROKONTROLER BASIC STAMP

ABSTRAK

Kecerdasan-buatan dalam bidang robotika adalah suatu algoritma (yang dipandang) cerdas yang diprogramkan ke dalam kontroler robot. Kecerdasan-buatan ini diperlukan robot untuk membantu manusia menjalankan suatu fungsi tertentu secara otomatis dan mandiri. Fungsi yang dilakukan oleh robot melalui kecerdasan-buatan dalam penelitian ini adalah melakukan pencarian jalur sekaligus pemetaan dari satu tempat awal menuju tempat tujuan dalam suatu lingkungan terkontrol berupa labirin.

Perancangan robot di mana kecerdasan-buatan itu akan diberikan meliputi dua aspek, yaitu 1) prototipe robot itu sendiri, dan 2) lingkungan di mana prototipe robot itu akan beroperasi, sedangkan implementasi mengikuti dua aspek rancangan tersebut. Untuk implementasi prototipe robot meliputi tiga aspek, yaitu implementasi 1) masukan (sensor untuk menerima informasi dari lingkungan), 2) pemrosesan (prosesor dan pendukungnya untuk mengolah informasi dari masukan), dan 3) keluaran (motor penggerak robot).

Kecerdasan-buatan diimplementasikan menggunakan bahasa pemrograman PBASIC dengan target mikrokontroller BASIC Stamp BS2px. Pemrograman *multi-bank* digunakan pada target mikrokontroler ini untuk memanfaatkan kapasitas penyimpanan kode program di EEPROM sebanyak 16 KB, yang terdiri atas delapan *bank* memori masing-masing dengan kapasitas 2 KB. Kode program mendukung fungsi robot pencari jalur di lingkungan labirin dengan menggunakan algoritma *Flood-Fill* dan algoritma *modified Flood-Fill* sebagai algoritma utama. Fleksibilitas dan skalabilitas menjadi konsep kecerdasan-buatan robot untuk mengakomodasi fitur tambahan dan mengantisipasi lingkungan yang lebih kompleks.

Kata kunci: Kecerdasan Buatan, Robot, Mikrokontroler, Basic Stamp, Algoritma *Flood-Fill*, Algoritma *Modified Flood-Fill*

Gede Indrawan
NPM 0606003410
Electrical Engineering Departement

Counsellor
Prof. Dr-Ing. Ir. Harry Sudibyo S., DEA

**DESIGN AND IMPLEMENTATION
ARTIFICIAL INTELLIGENCE OF PATH SEARCHING ROBOT
BASED ON MICROCONTROLLER BASIC STAMP**

ABSTRACT

Artificial intelligence in robotics, as a smart algorithm programmed into the robot, is needed by the robot to help human to do some work automatically and in autonomous way. In this research, the implanted artificial intelligence is designed for path searching, included in mapping activity, from starting point at corner to destination point at center, in controlled environment like maze. General speaking, this research want to contribute in knowledge development under robotics domain of autonomous position-sensing and navigation.

Robot design for this artificial intelligence consists of two aspects, i.e. robot prototype itself, and maze environment where path-searching robot will run. Implementation of robot involves three aspects, i.e. input (sensor to capture information from the environment), process (processor and its supporting system as robot brain for data processing), and output (as result of data processing, it can be signal for controlling the motor, etc).

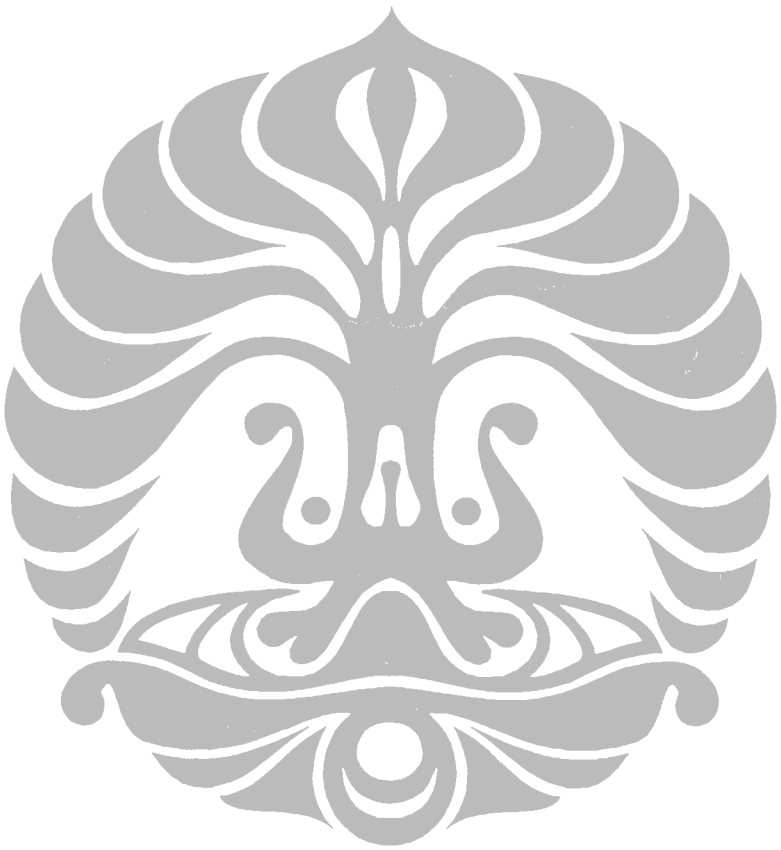
The artificial intelligence in this research is implemented using PBASIC programming language, with BASIC Stamp BS2px from Parallax as targeted microcontroller. Multi bank programming style is used to utilize 16 KB internal EEPROM resource, comprise of eight memory bank with 2 KB capacity respectively, to save program code. This code supports robot function for path searching in maze, by using Flood-Fill algorithm and modified Flood-Fill algorithm as main algorithms. Flexibility and scalability are two concepts of this artificial intelligence to accommodate features addition and to anticipate more complex maze environment.

Keywords: Artificial Intelligence, robot, autonomous position-sensing and navigation, microcontroller, PBASIC, Basic Stamp, multi bank programming, Flood-Fill algorithm, modified Flood-Fill algorithm

DAFTAR ISI

		Halaman
PERNYATAAN KEASLIAN TESIS		ii
PENGESAHAN		iii
UCAPAN TERIMA KASIH		iv
ABSTRAK		v
ABSTRACT		vi
DAFTAR ISI		vii
DAFTAR GAMBAR		ix
DAFTAR TABEL		xi
DAFTAR LAMPIRAN		xii
DAFTAR SINGKATAN		xiii
BAB I	PENDAHULUAN	1
	1.1 LATAR BELAKANG	2
	1.2 RUMUSAN MASALAH	2
	1.3 TUJUAN PENELITIAN	2
	1.4 BATASAN MASALAH	2
	1.5 METODOLOGI PENELITIAN	3
	1.6 SISTEMATIKA PENULISAN	4
BAB II	KECERDASAN-BUATAN ROBOT PENCARI JALUR	5
	2.1 KECERDASAN-BUATAN ROBOT	4
	2.2 PERKEMBANGAN ROBOT	6
	2.2 MIKROKONTROLER BASIC STAMP	8
	2.3 ALGORITMA PENCARI JALUR	13
	2.3.1 Algoritma *Wall Following*	13
	2.3.2 Algoritma *Depth Search*	14
	2.3.3 Algoritma *Flood-Fill*	14
	2.3.3 Algoritma *Modified Flood-Fill*	14
BAB III	PERANCANGAN KECERDASAN-BUATAN ROBOT PENCARI JALUR	15
	3.1 PERANGKAT KERAS	17
	3.1.1 Sub Sistem Masukan	17
	3.1.2 Sub Sistem Proses	19
	3.1.3 Sub Sistem Ouput	20
	3.2 PERANGKAT LUNAK	22
	3.2.1 Algoritma *Flood-Fill*	22
	3.2.2 Algoritma *Modified Flood-Fill*	28
BAB IV	IMPLEMENTASI KECERDASAN-BUATAN ROBOT PENCARI JALUR	35
	4.1 IMPLEMENTASI UMUM	39
	4.2 PEMROSESAN INISIALISASI DAN MASUKAN	41
	4.3 PEMROSESAN INFORMASI DINDING CELL	46
	4.4 PEMROSESAN INFORMASI JARAK CELL	47
	4.5 PEMROSESAN STATUS KUNJUNGAN CELL	50
	4.6 PEMROSESAN TUJUAN	51
	4.7 PEMROSESAN MANUVER MOTOR	52
	4.8 PEMROSESAN DISPLAY	52

BAB V KESIMPULAN 53
DAFTAR ACUAN 54
DAFTAR PUSTAKA 55
LAMPIRAN 56



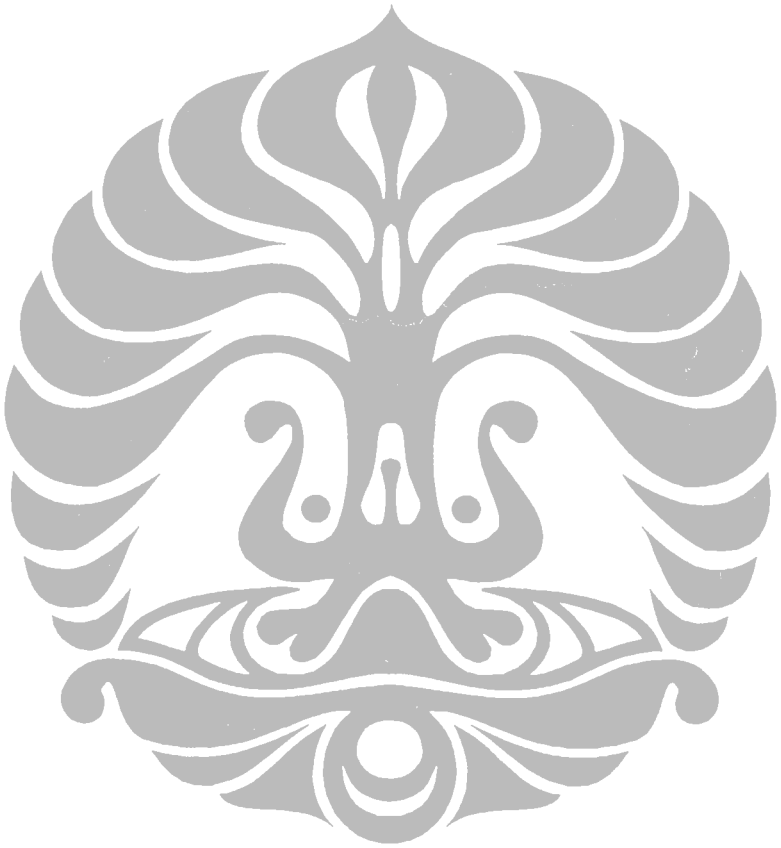
DAFTAR GAMBAR

		Halaman
Gambar 1.1	Salah Satu Varian Tahapan SDLC Untuk Pengembangan Sistem	3
Gambar 2.1	Kontrol robot loop tertutup berbasis AI	4
Gambar 2.2a	Robot Pencari Jalur	6
Gambar 2.2b	Robot Bedah	6
Gambar 2.2c	Robot *Vacuum Cleaner*	7
Gambar 2.2d	Nanocar dari Molekul Tunggal	7
Gambar 2.3	Modul Internal BASIC Stamp	9
Gambar 2.4	*Pinout* BASIC Stamp	9
Gambar 2.5	Lingkungan Pengembangan BASIC Stamp	11
Gambar 2.6	Koneksi *Download* Program dari PC ke BASIC Stamp	11
Gambar 2.7	*Editor* Program BASIC Stamp	12
Gambar 2.8	Algoritma dan Diagram Alir *Wall Follower*	13
Gambar 2.9	Lingkungan Labirin di mana Algoritma *Wall Following* Tidak Bekerja	14
Gambar 3.1	Lingkungan Robot Pencari Jalur	15
Gambar 3.2	Lingkungan Labirin 25 *Cell*	15
Gambar 3.3a	Diagram Blok Sistem	16
Gambar 3.3b	Diagram Blok Masukan	16
Gambar 3.3c	Diagram Blok Proses	16
Gambar 3.3d	Diagram Blok Keluaran	16
Gambar 3.4	Robot dengan Sensor *Proximity*	18
Gambar 3.5a	IR LED Sensor *Proximity*	18
Gambar 3.5b	Phototransistor Sensor *Proximity*	18
Gambar 3.5c	Sensor *Proximity* (Satu Kemasan)	18
Gambar 3.6	Sensor *Proximity* Berbasis QRD1114	18
Gambar 3.7	*Board of Education* (BOE) Rev. C *Carrier Board* untuk Modul Mikrokontroler BASIC Stamp	19
Gambar 3.8	*Continuous Rotation Servo*	
Gambar 3.9	Koneksi *Board of Education* dan *Servo*	20
Gambar 3.10	*Pinout Servo* (+5 V, ground, dan signal)	21
Gambar 3.11	Kode Program Pengontrol *Servo*	21
Gambar 3.12	Pergerakan Awal Robot dalam Algoritma *Flood-Fill*	22
Gambar 3.13	Algoritma dan Diagram Alir Menentukan *Cell* dengan Nilai Jarak Terendah	23
Gambar 3.14	*Byte* Merepresentasi Status Dinding *Cell*	
Gambar 3.15	Algoritma dan Diagram Alir *Updating* Peta Dinding	23
Gambar 3.16	Penemuan Dinding Labirin dalam Algoritma *Flood-Fill*	25
Gambar 3.17	Simulasi Algoritma *Flood-Fill*	26

Gambar 3.18	Algoritma dan Diagram Alir Flood Labirin dengan Nilai Jarak Baru	27
Gambar 3.19	Pergerakan Awal Robot dalam Algoritma *Modified Flood-Fill*	28
Gambar 3.20	Penemuan Dinding Labirin dalam Algoritma *Modified Flood-Fill*	29
Gambar 3.21	*Updating* Nilai Jarak *Cell* dalam Algoritma *Modified Flood-Fill*	29
Gambar 3.22	Algoritma dan Diagram Alir *Update* Nilai Jarak (*Modified Flood-Fill*)	30
Gambar 3.23	Simulasi Algoritma *Modified Flood-Fill*	33
Gambar 3.24	Inisialisasi Awal Peta Labirin Pada Program PBASIC	34
Gambar 4.1	Implementasi Rancangan Data Kecerdasan-Buatan Robot Pencari Jalur	35
Gambar 4.2	Flowchart Sederhana dan Pemakaian Resources EEPROM Kecerdasan-Buatan Robot Pencari Jalur	39
Gambar 4.3	Flowchart Detail Kecerdasan-Buatan Robot Pencari Jalur	40
Gambar 4.4	Diagram Alir Proses Inisialisasi Kecerdasan-Buatan	41
Gambar 4.5	Kode Inisiasi Peta Awal Labirin	42
Gambar 4.6	Kode Kalibrasi Sensor Pada Pencahayaan Tertentu	43
Gambar 4.7	Diagram Alir Mekanisme Kerja Sensor	44
Gambar 4.8	Sensor Proximity dari Robot Pencari Jalur	45
Gambar 4.9	Pemrosesan Informasi Dinding Cell	46
Gambar 4.10	Pemrosesan Informasi Jarak Cell Berbasis Stack	47
Gambar 4.11	Kode Kecerdasan-Buatan untuk Push dan Pull	47
Gambar 4.12	Simulasi Pemrosesan Informasi Jarak Cell Berbasis Stack	49
Gambar 4.13	Pemrosesan Status Kunjungan Cell Berbasis Stack	50
Gambar 4.14	Pemrosesan Tujuan	51
Gambar 4.15	Pemrosesan Keluaran	52
Gambar 4.16	Display Parameter Internal Proses ke PC	52

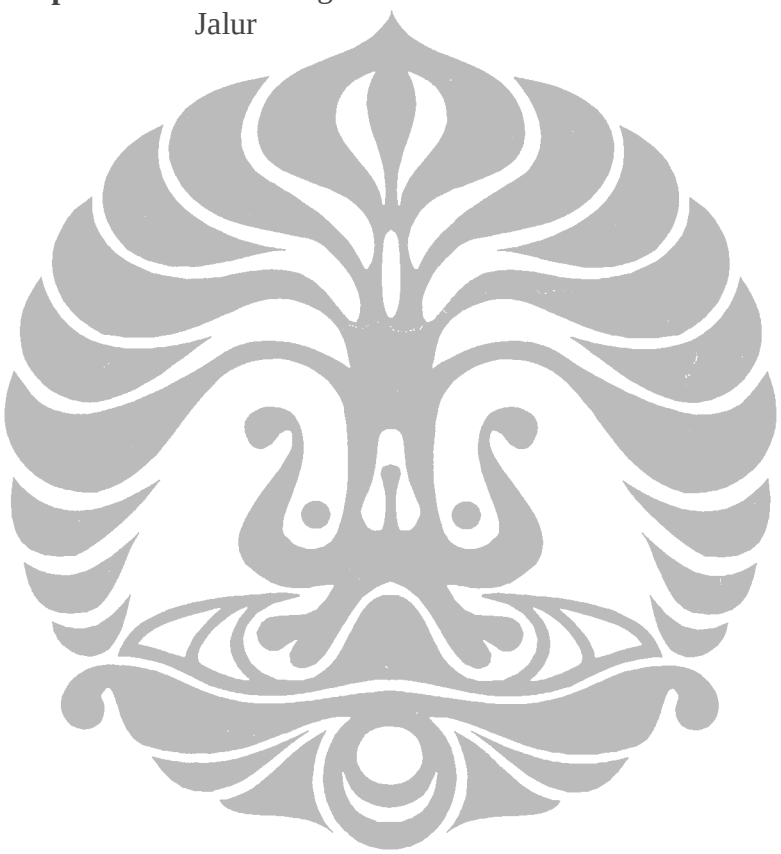
DAFTAR TABEL

		Halaman
Tabel 2.1	*Resource* I/O pada Basic Stamp	10
Tabel 4.1	Alokasi Alamat Memori Data Cell	37
Tabel 4.2	Bit-Bit Kontrol di RAM	37
Tabel 4.3	Pemakaian Pin Robot Pencari Jalur	44



DAFTAR LAMPIRAN

		Halaman
Lampiran 1	Keluarga Mikrokontroler Basic Stamp	57
Lampiran 2	Foto-Foto Robot Pencari Jalur	58
Lampiran 3	Paper pada "The 10th Quality In Research (QIR) International Conference", University of Indonesia, December 4-6, 2007	60
Lampiran 4	Kode Program Kecerdasan-Buatan Robot Pencari Jalur	66



DAFTAR SINGKATAN

b	*bit*
B	*Byte*
CPU	*Central Processing Unit*
EEPROM	*Electrical Erase Programmable Read Only Memory*
IO	*Input Output*
IR	*Infra Red*
K	*Kilo*
RAM	*Random Access Memory*
ROM	*Read Only Memory*
SPRAM	*Scratchpad Random Access Memory*
V	*Volt*

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Teknologi robot berkembang pesat dan membantu manusia dalam berbagai aspek kehidupan. Sebagai 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 biasanya digunakan untuk tugas yang berat, berbahaya, pekerjaan yang berulang dan kotor. Biasanya kebanyakan robot industri digunakan dalam bidang produksi. Penggunaan robot lainnya termasuk untuk pembersihan limbah beracun, penjelajahan bawah air dan luar angkasa, pertambangan, pekerjaan "cari dan tolong" *(search and rescue)*, dan untuk pencarian tambang. Belakangan ini robot mulai memasuki pasaran konsumen di bidang hiburan, dan alat pembantu rumah tangga, seperti penyedot debu, dan pemotong rumput [1].

Salah satu aspek yang sangat menarik dan penting dalam bidang robotika adalah teknologi pengikut jalur *(line following)*. Robot *line following* membantu mengotomatisasi pabrik, melakukan pengantaran surat, paket, atau material secara cepat dan efisien. Teknologi *line following* tidak hanya untuk robot-robot kecil atau sejenisnya [2].

Jenis robot ini digunakan untuk kendaraan pengeruk salju atau juga kendaraan penumpang yang dapat mengikuti jalur magnetik pada jalan raya pintar (*smart highways*). Jenis kendaraan robot ini dapat mendeteksi jalan, rintangan, dan lain-lain, menghilangkan kemacetan lalu lintas, sehingga jalan raya lebih aman dan lebih mudah untuk dilalui. Suatu saat, kita dapat secara sederhana menginstruksikan kendaraan untuk membawa kita dan rangkaian *line following* akan membantu ke suatu tujuan tertentu dengan lebih aman dengan usaha yang minimalis.

Secara umum, hal-hal tersebut di atas berkaitan dengan kecerdasan-buatan dalam pencarian suatu jalur secara otomatis yang dilakukan robot untuk memudahkan kegiatan-kegiatan manusia dalam kehidupan.

Berkaitan dengan hal tersebut di atas, penelitian ini mencoba untuk merancang dan mengimplementasikan suatu jenis kecerdasan-buatan robot *line following* yang digunakan dalam pencarian jalur menuju ke tujuan yang diinginkan di suatu lingkungan. Lingkungan yang digunakan mengadaptasi lingkungan yang digunakan pada kontes robot *MicroMouse*, yaitu lingkungan labirin *(maze)* dengan start awal robot di salah satu sudut labirin dan tujuan berada di tengah-tengan labirin tersebut.

1.2 RUMUSAN MASALAH

Rumusan masalah penelitian meliputi dua hal, yaitu:

1. Rancangan prototipe robot pencari jalur

   Rancangan meliputi dua aspek, yaitu 1) prototipe robot itu sendiri, dan 2) lingkungan di mana prototipe robot itu akan beroperasi.
2. Implementasi prototipe robot pencari jalur

   Implementasi mengikuti dua aspek rancangan di atas. Untuk implementasi prototipe robot meliputi tiga aspek, yaitu implementasi 1) masukan (sensor untuk menerima informasi dari lingkungan), 2) pemrosesan (prosesor dan pendukungnya untuk mengolah informasi dari masukan), dan 3) keluaran (motor penggerak robot).

1.3 TUJUAN PENELITIAN

Merancang dan mengimplementasikan prototipe robot dengan kecerdasan-buatan untuk pencarian jalur sekaligus pemetakan lingkungan labirin di mana robot berada.

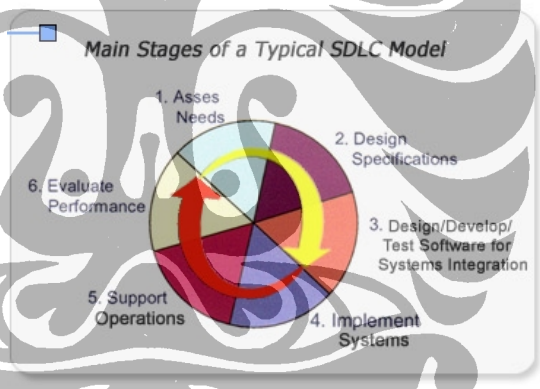
1.4 BATASAN MASALAH

Batasan masalah perancangan dan implementasi prototipe robot pencari jalur mengadopsi aturan-aturan dari kontes robot MicroMouse [3][4], diantaranya: 1) Robot bersifat *autonomous*, *self-contained* (mandiri), dan tidak menerima asistensi dari luar dalam menjalankan fungsinya, 2) Berbasis prosesor Basic Stamp, sensor infra merah, dan motor penggerak *continuous rotation servo*, 3) Dimensi maksimal robot (panjang x lebar) = 25 cm x 25 cm, 4) Labirin berbetuk

persegi empat terdiri atas sel *(cell)*, dengan ukuran *cell* adalah 18x18 cm, dan lebar pembatas jalur adalah 1,2 cm, 5) Checkpoint pada labirin minimal mempunyai satu buah dinding cell, 6) *Cell* tempat robot mulai start berada pada salah satu sudut labirin, dan *cell* tujuan adalah *cell* di tengah-tengah labirin, 7) Ketika mencapai *cell* tujuan, robot akan melakukan perjalanan balik ke *cell* awal, dan pada putaran berikutnya bisa menemukan jalur yang lebih optimal dari *cell* awal ke *cell* tujuan.

1.5 METODOLOGI PENELITIAN

Berdasarkan tujuan penelitian untuk membuat suatu prototipe, metode penelitian yang digunakan mengacu pada *System Development Life Cycle* (SDLC) yang dikembangkan oleh McLeod [5]. SDLC digunakan dengan pertimbangan agar sistem dapat dirancang dan diimplementasikan secara metodis, logis, dan melalui pendekatan tahap demi tahap.



Gambar 1.1. Salah Satu Varian Tahapan SDLC Untuk Pengembangan Sistem [5]

Secara singkat uraian tahapan-tahapan SDLC [5], yaitu sebagai berikut:

1. *Asses Needs*

   Pada tahap ini sistem yang ada dievaluasi dan kekurangan-kekurangannya diidentifikasi.

2. *Design Specification*

   Selanjutnya didefinisikan persyaratan-persyaratan sistem baru. Kekurangan-kekurangan yang ada pada sistem yang lama harus dapat diatasi oleh usulan perbaikan pada sistem yang baru.

3. *Design/Develop/Test Software for System Integration*

   Di sini, sistem dirancang. Perencanaan meliputi konstruksi logikal dan fisik, perangkat keras, sistem operasi, pemrograman, komunikasi, pelatihan, dan keamanan.

4. *Implement Systems*

   Pada tahap ini, sistem mulai digunakan. Sistem baru menggantikan sistem lama setahap demi setahap.

5. *Support Operations*

   Perfomansi sistem dimonitor pada tahap ini; *tuning* dan sinkronisasi dilakukan. Prosedur diubah dan pelatihan-pelatihan tambahan dilakukan sesuai dengan keperluan. Perubahan direkomendasikan melalui Otoritas Pengontrol Perubahan.

6. *Evaluate Perfomance*

   Pada tahap ini sistem baru yang operasional di tahap awal menjalani evaluasi secara intensif. Perawatan dilakukan secara intensif juga. Pemakai sistem diinformasikan tentang modifikasi-modifkasi dan prosedur-prosedur terbaru.

1.6 SISTEMATIKA PENULISAN

Sistematika penulisan berdasarkan metodologi penelitian, meliputi tahap-tahap: 1) Menilai permasalahan (pencarian jalur sekaligus pemetaan lingkungan labirin oleh robot), 2) Penentuan spesifikasi (meliputi spesifikasi robot dan lingkungan di mana robot akan beroperasi), 3) Merancang prototipe (meliputi perancangan perangkat keras dan perangkat lunak), 4) Implementasi prototipe (meliputi pemilihan *platform* perangkat keras, seperti sensor, prosesor, dan motor, serta bahasa pemrograman perangkat lunak), 5) Pengujian prototipe (integrasi operasional perangkat keras dan perangkat lunak robot di lingkungan operasional).

BAB II
KECERDASAN-BUATAN ROBOT PENCARI JALUR

2.1 KECERDASAN-BUATAN ROBOT

Kecerdasan-buatan (*Artificial Intelligence* atau AI) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu entitas buatan. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan-buatan antara lain sistem pakar, permainan komputer (*games*), logika *fuzzy*, jaringan syaraf tiruan dan robotika [6].

Aplikasi AI dalam kontrol robotik diilustrasikan oleh Gambar 2.1.

Gambar 2.1. Kontrol robot loop tertutup berbasis AI [7]

Penggunaan AI dalam kontroler dilakukan untuk mendapatkan sifat dinamik kontroler "secara cerdas". Secara klasik, kontrol P, I, D atau kombinasi, tidak dapat melakukan adaptasi terhadap perubahan dinamik sistem selama operasi karena parameter P, I dan D itu secara teoritis hanya mampu memberikan efek kontrol terbaik pada kondisi sistem yang sama ketika parameter tersebut di-*tune*. Di sinilah kemudian dikatakan bahwa kontrol klasik ini "belum cerdas" karena belum mampu mengakomodasi sifat-sifat nonlinieritas atau perubahan-perubahan dinamik, baik pada sistem robot itu sendiri maupun terhadap perubahan beban atau gangguan lingkungan [7].

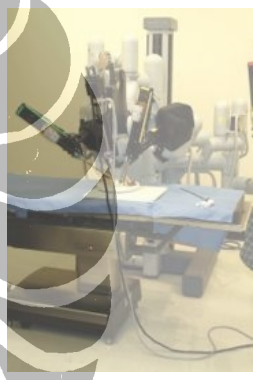
Gambar 2.1 mengilustrasikan tentang skema AI yang digunakan secara langsung sebagai kontroler sistem robot. Dalam aplikasi lain, AI juga dapat digunakan untuk membantu proses identifikasi model dari sistem robot, model lingkungan atau gangguan, model dari tugas robot (task) seperti membuat rencana

trajektori, dan sebagainya. Dalam hal ini konsep AI tidak digunakan secara langsung *(direct)* ke dalam kontroler, namun lebih bersifat tak langsung *(indirect)*.

2.2 PERKEMBANGAN ROBOT

Istilah ”robot” muncul pertama kali pada Czechoslovakian *satirical play*, *Rossum's Universal Robots*, oleh Karel Capek pada tahun 1920. Robot pada pementasan ini cenderung berperilaku seperti manusia *(human-like)*. Berangkat dari hal tersebut, terlihat beberapa cerita fiksi ilmiah melibatkan robot dengan emosi manusia dalam masyarakat. Hal tersebut berubah ketika General Motors memasang robot pertamanya di pabrik manufakturingnya pada tahun 1961. Mesin-mesin automatis ini merepresentasi image yang seluruhnya berbeda dari robot berbentuk manusia *(human form)* dari cerita fiksi ilmiah [8].

Gambar 2.2a. Robot Pencari Jalur



Gambar 2.2b. Robot Bedah [1]

Ketika para pencipta robot pertama kali mencoba meniru manusia dan hewan, mereka menemukan bahwa hal tersebut sangatlah sulit; membutuhkan tenaga penghitungan yang jauh lebih banyak dari yang tersedia pada masa itu. Jadi, penekanan perkembangan diubah ke bidang riset lainnya. Robot sederhana beroda digunakan untuk melakukan eksperimen dalam tingkah laku, navigasi, dan perencanaan jalur. Teknik navigasi tersebut telah berkembang menjadi sistem kontrol robot otonom yang tersedia secara komersial; contoh paling mutakhir dari sistem kontrol navigasi otonom yang tersedia sekarang ini termasuk sistem navigasi berdasarkan-laser dan VSLAM *(Visual Simultaneous Localization and Mapping)* dari ActivMedia Robotics dan Evolution Robotics [1].

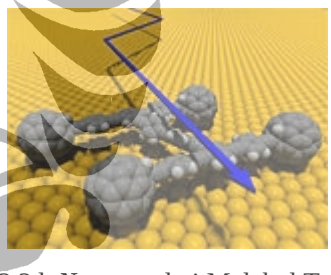
Ketika para teknisi siap untuk mencoba robot berjalan kembali, mereka mulai dengan heksapoda dan platform berkaki banyak lainnya. Robot-robot tersebut

meniru serangga dan arthropoda dalam bentuk dan fungsi. Tren menuju jenis badan tersebut menawarkan fleksibilitas yang besar dan terbukti dapat beradaptasi dengan berbagai macam lingkungan, tetapi biaya dari penambahan kerumitan mekanikal telah mencegah pengadopsian oleh para konsumer. Dengan lebih dari empat kaki, robot-robot ini stabil secara statis yang membuat mereka bekerja lebih mudah. Tujuan dari riset robot berkaki dua adalah mencapai gerakan berjalan menggunakan gerakan pasif-dinamik yang meniru gerakan manusia.

Perkembangan hebat telah dibuat dalam robot medis, dengan dua perusahaan khusus, Computer Motion dan Intuitive Surgical, yang menerima pengesahan pengaturan di Amerika Utara, Eropa, dan Asia atas robot-robotnya untuk digunakan dalam prosedur pembedahan minimal.

Tempat lain di mana robot disukai untuk menggantikan pekerjaan manusia adalah dalam eksplorasi laut dalam dan eksplorasi antariksa. Untuk tugas-tugas ini, bentuk tubuh artropoda umumnya disukai. Mark W. Tilden dahulunya spesialis Laboratorium Nasional Los Alamos membuat robot murah dengan kaki bengkok tetapi tidak menyambung, sementara orang lain mencoba membuat kaki kepiting yang dapat bergerak dan tersambung penuh.

Gambar 2.2c. Robot *Vacuum Cleaner* [1]



Gambar 2.2d. *Nanocar* dari Molekul Tunggal [1]

Robot bersayap eksperimental dan contoh lain mengeksploitasi biomimikri juga dalam tahap pengembangan dini. Yang disebut "nanomotor" dan "kawat cerdas" diperkirakan dapat menyederhanakan daya gerak secara drastis, sementara stabilisasi dalam penerbangan nampaknya cenderung diperbaiki melalui giroskop yang sangat kecil. Dukungan penting pekerjaan ini adalah untuk riset militer teknologi pemata-mataan.

2.3 MIKROKONTROLER BASIC STAMP

Mikrokontroler adalah sistem mikroprosesor lengkap yang terkandung di dalam sebuah *chip*. Mikrokontroler berbeda dari mikroprosesor serba-guna *(general-purpose microprocessor)* yang digunakan dalam sebuah PC, karena sebuah mikrokontroler umumnya telah berisi komponen pendukung sistem minimal mikroprosesor, yakni memori dan antarmuka I/O [9].

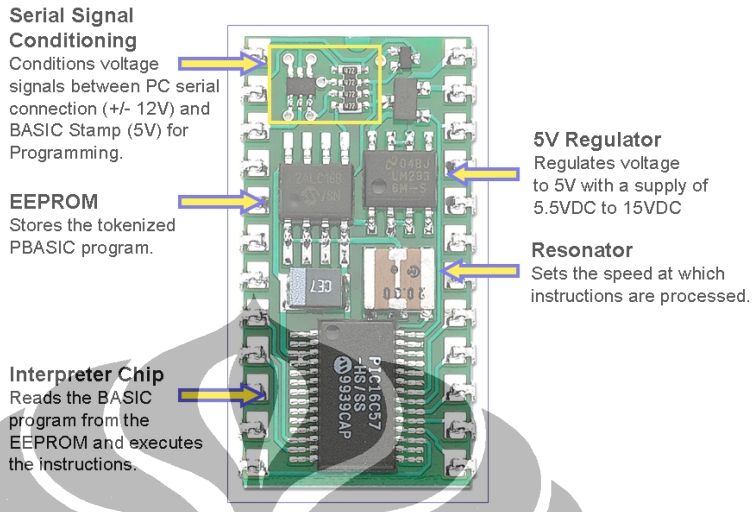
Berbeda dengan mikroprosesor serba-guna, mikrokontroler tidak selalu memerlukan memori eksternal, sehingga mikrokontroler dapat dibuat lebih murah dalam kemasan yang lebih kecil dengan jumlah *pin* yang lebih sedikit.

Sebuah *chip* mikrokontroler umumnya memiliki fitur:

1. *Central Processing Unit* (CPU) - mulai dari prosesor 4-*bit* yang sederhana hingga prosesor kinerja tinggi 64-*bit*.
2. Masukan/Keluaran (I/O) antarmuka jaringan seperti port serial (UART)
3. Antarmuka komunikasi serial lain seperti I²C, *Serial Peripheral Interface* and *Controller Area Network* untuk sambungan sistem
4. *Peripheral* seperti *timer* dan *watchdog*
5. RAM untuk penyimpanan data
6. ROM, EPROM, EEPROM atau memori *flash* untuk menyimpan program komputer
7. Pembangkit *clock* - biasanya berupa resonator rangkaian RC
8. Pengubah analog - ke - digital

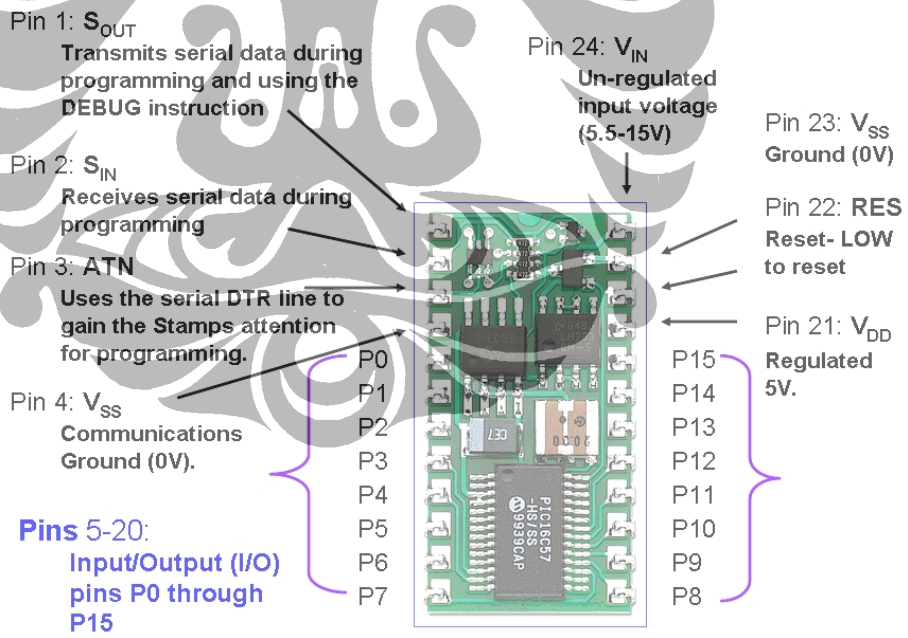
Modul BASIC Stamp dari Parallax merupakan mikrokontroler dengan *chip interpreter* BASIC, memori internal (RAM dan EEPROM), regulator 5 volt, beberapa *pin* I/O multi fungsi (TTL-level, 0-5 volt), dan set instruksi built-in untuk operasi matematika dan *pin* I/O. Modul BASIC Stamp mempunyai kemampuan untuk menjalankan beberapa ribu instruksi per detik dan dapat diprogram dengan sederhana menggunakan PBASIC, yaitu bahasa pemrograman sejenis BASIC yang sudah dimodifikasi untuk mikrokontroler ini [10].

Beberapa modul internal penyusun mikrokontroler BASIC Stamp bisa dilihat pada Gambar 2.3



Gambar 2.3. Modul Internal BASIC Stamp [11]

Gambar 2.4 memperlihatkan 24 I/O *pin* yang tersedia untuk peripheral.



Gambar 2.4. *Pinout* BASIC Stamp [11]

Daftar *Pinout* pada Tabel 2.1 memperlihatkan semua *resource* I/O secara detail. Pada kolom sebelah kiri notasi BS2x-24 merupakan semua modul BS2 24-*pin* berdasarkan gambar sebelumnya ("x" mengacu pada nomor model Basic Stamp pada seri BS2), sedangkan kolom kedua hanya untuk BS2p 40-*pin* [12].

Tabel 2.1. *Resource* I/O pada Basic Stamp

Pin BS2x-24[1]	*Pin BS2p-40*	*Name*	*Function*
1	1	SOUT	Serial output (to RxD of the PC COM Port)
2	2	SIN	Serial input (from TxD of the PC COM Port)
3	3	ATN	Attention (to DTR of the PC COM Port)
4	4	VSS	Ground (to GND of the PC COM Port)
5-20	5-20	P0-P15	Digital I/O pins
	21-36	X0-X15	Digital I/O pins
21	37	VDD	+ 5 V DC I/O (stabilized)
22	38	RES	Reset I/O
23	39	VSS	Ground
24	40	VIN	+ 5,5 - 12 V DC input (not stabilized)

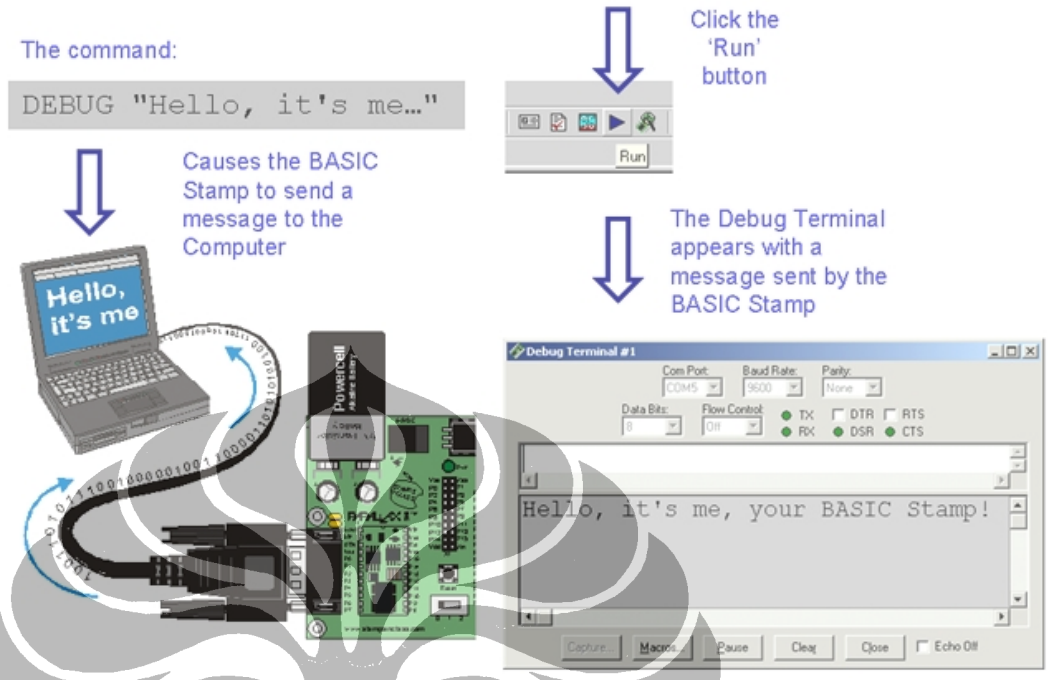
Sumber : Parallax, Inc., USA

Tegangan stabil +5 VDC pada V_{DD} dapat diperoleh dengan menghubungkan 5 sampai +12 VDC ke V_{IN} dan *internal voltage regulator*.

Jika diinginkan untuk memberikan tegangan stabil +5 VDC pada Basic Stamp maka tegangan tersebut dapat dihubungkan secara langsung pada V_{DD}. *Pin* V_{IN} dapat dibuat tidak terhubung dalam hal ini. Modul BASIC Stamp harus disuplai dengan hanya satu tegangan (kemungkinan baterai) dan akan menjalankan program segera setelah di-*download*.

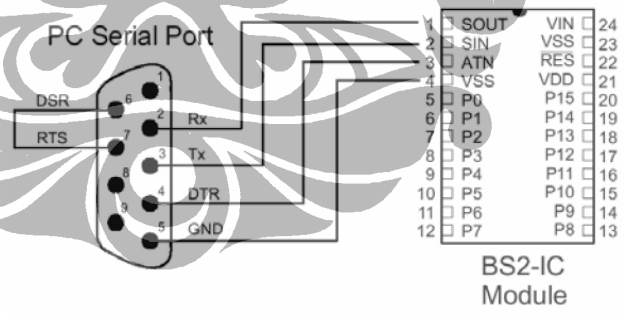
Reset internal ***(power-down reset)*** mengakibatkan *pin* RES bernilai *low* selama fasa *reset*.

Gambar 2.5 memperlihatkan infrastruktur pemrograman secara lengkap yang dibutuhkan.



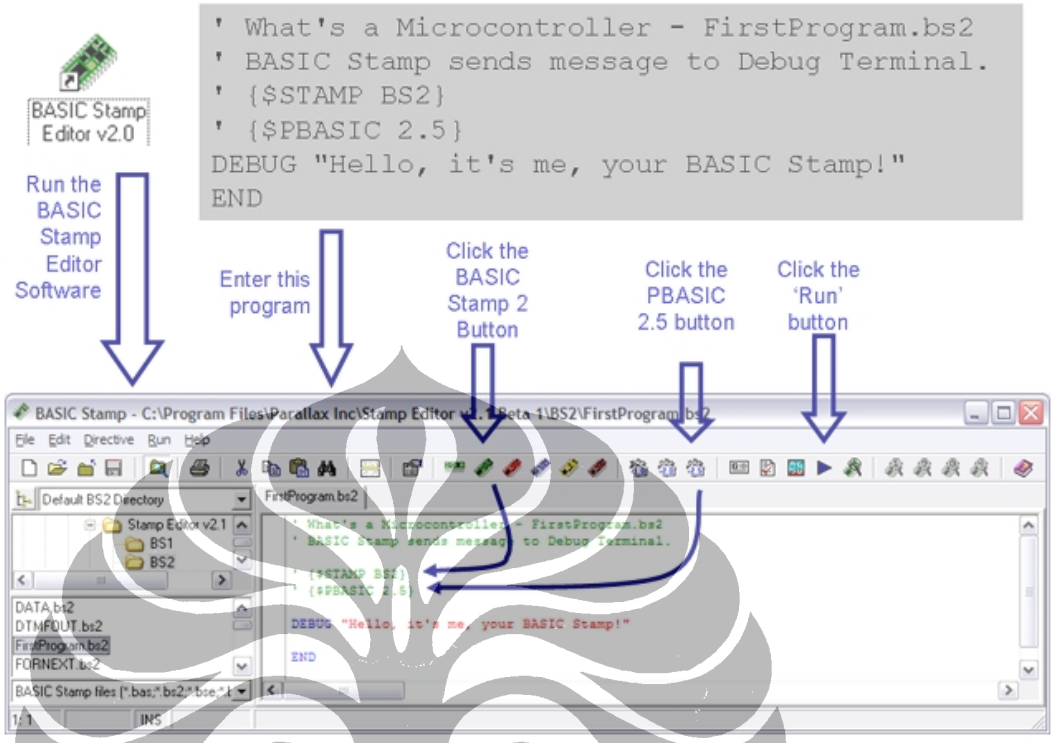
Gambar 2.5. Lingkungan Pengembangan BASIC Stamp [11]

Koneksi kabel *download* dari standard serial cable dengan modul BASIC Stamp bisa dilihat pada Gambar 2.6.



Gambar 2.6. Koneksi *Download* Program dari PC ke BASIC Stamp [12]

Pemrograman mikrokontroler BASIC Stamp dilakukan menggunakan *Editor* BASIC Stamp yang di-*instal* di PC.



Gambar 2.7. *Editor* Program BASIC Stamp [11]

Jika *source code* PBASIC bebas dari kesalahan, maka *source code* tersebut bisa di-*compile* dan di-*download* ke BASIC Stamp.

Token yang di-*download* akan disimpan di EEPROM eksternal. Mikrokontroler BASIC Stamp mempunyai *firmware* PBASIC yang disebut *"token interpreter"*. *Token interpreter* ini bertanggung jawab untuk menjalankan *token* yang di-*download* dan merepresentasikan *core intellectual property* (IP) dari Parallax. Prosedur *download* sama untuk semua jenis BS2.

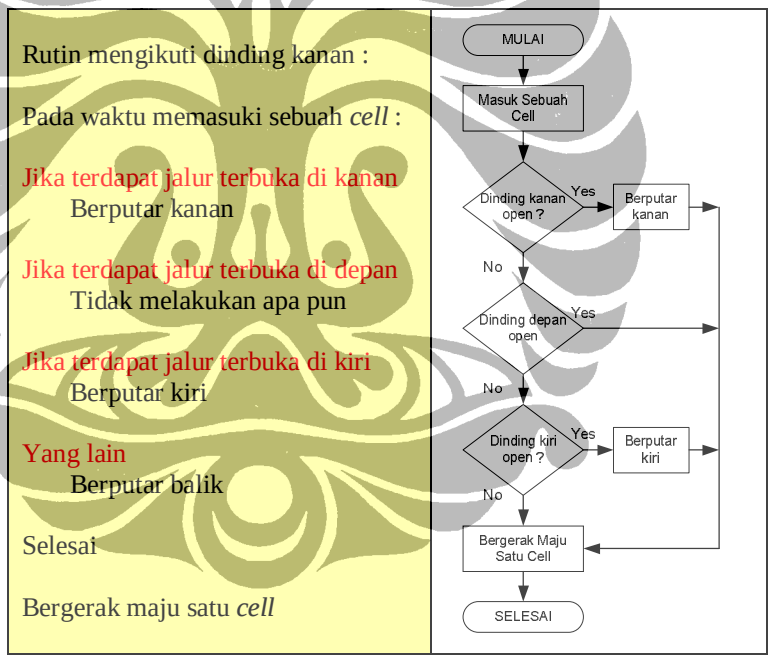
2.4 ALGORITMA PENCARI JALUR

Beberapa algoritma yang digunakan dalam pencarian jalur di lingkungan labirin, yaitu: *Wall Follower*, Depth First Search, *Flood-Fill*, dan *modified Flood-Fill*.

Penelitian ini mencoba mengimplementasi secara spesifik dua algoritma terakhir dalam kecerdasan-buatan robot pencari jalur. Uraian lebih detail mengenai kedua algoritma tersebut akan disampaikan di Bab 3, sedangkan untuk algoritma yang lain akan diuraikan secara singkat

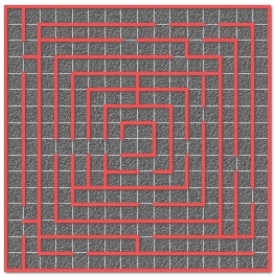
2.4.1 Algoritma *Wall Follower*

Algoritma ini merupakan teknik yang paling sederhana dalam pencarian jalur di lingkungan labirin. Pada dasarnya, robot dengan algoritma ini mengikuti dinding kiri atau kanan sebagai petunjuk di sekeliling labirin.



Gambar 2.8. Algoritma dan Diagram Alir *Wall Follower*

Pada kasus-kasus tertentu, algoritma ini tidak akan bekerja karena robot akan berputar-putar terus di pinggiran labirin (tidak akan bisa mencari jalur menuju ke tengah) seperti ditunjukkan Gambar 2.9.



Gambar 2.9. Lingkungan Labirin di mana Algoritma *Wall Following* Tidak Bekerja

2.4.2 Algoritma *Depth First Search*

Merupakan metode yang intuitif dalam pencarian jalur di lingkungan labirin. Pada dasarnya robot dengan algoritma ini bergerak dan ketika menemukan percabangan, secara acak memilih salah satu jalurnya. Jika jalur ini pada akhirnya buntu, robot ini kembali ke percabangan tadi dan memilih jalur yang lain. Algoritma ini mengakibatkan robot untuk mengeksplorasi setiap kemungkinan jalur di dalam labirin. Dengan mengeksplorasi setiap *cell* di dalam labirin, robot pada akhirnya sampai pada *cell* tujuan di tengah.

Jelas, mengeksplorasi keseluruhan labirin bukan merupakan cara yang efektif dan juga, walaupun menemukan jalur, itu bukan jalur tercepat atau terpendek menuju ke tengah.

2.4.3 Algoritma *Flood-Fill*

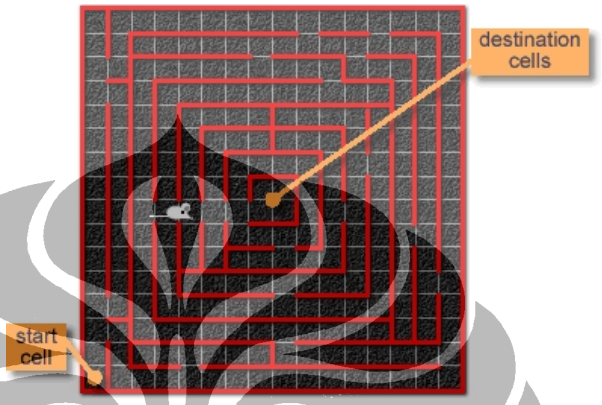
Algoritma ini melibatkan pemberian nilai pada masing-masing *cell* penyusun labirin di mana nilai ini merepresentasikan jarak dari sebarang *cell* ke *cell* tujuan. Selanjutnya *cell* tujuan diberikan nilai 0. Jika robot pencari jalur berada di sebuah *cell* dengan nilai 1, robot tersebut 1 *cell* jauhnya dari tujuan. Jika robot berada pada *cell* dengan nilai 3, robot tersebut 3 *cell* jauhnya dari tujuan.

2.4.4 Algoritma *Modified Flood-Fill*

Algoritma ini sama dengan algoritma *Flood-Fill* regular di mana kecerdasan buatam menggunakan nilai jarak untuk bergerak di dalam labirin. Nilai jarak, yang merepresentasi seberapa jauh robot dari *cell* tujuan, diikuti secara menurun (descending order) sampai robot mencapai tujuannya.

BAB III
PERANCANGAN
KECERDASAN-BUATAN ROBOT PENCARI JALUR

Kecerdasan-buatan yang dirancang untuk robot pencari jalur ini ditujukan pada lingkungan labirin (maze) dua dimensi seperti ditunjukkan oleh Gambar 3.1.



Gambar 3.1. Lingkungan Robot Pencari Jalur

Berdasarkan Gambar 3.1, lingkungan labirin tersusun atas matriks *n x n cell*, di mana *n* merupakan jumlah *cell* penyusunnya. Penelitian ini mencoba merancang dan mengimplementasikan kecerdasan-buatan untuk robot pencari jalur pada lingkungan labirin yang lebih kecil dengan ukuran matriks 5 x 5 (25 *cell* penyusun) seperti ditunjukkan pada Gambar 3.2

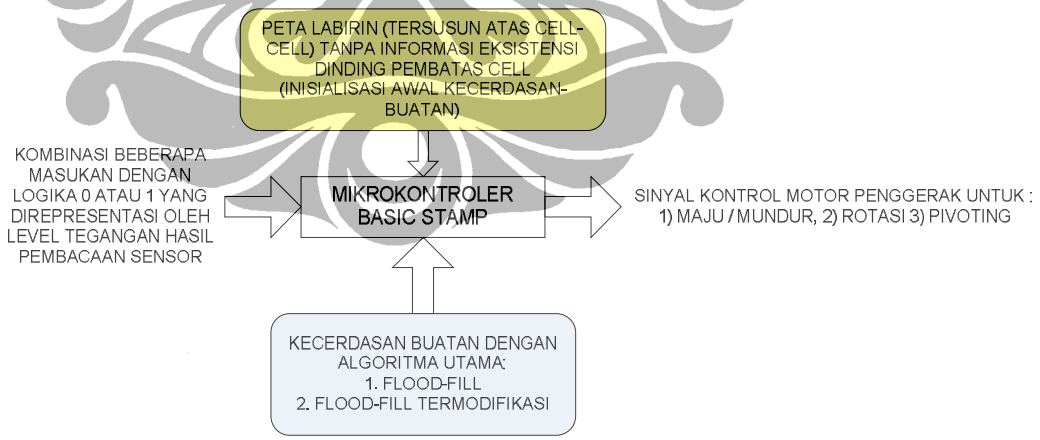
Gambar 3.2. Lingkungan Labirin 25 *Cell*

Perancangan meliputi aspek perangkat keras dan perangkat lunak di mana blok diagram secara keseluruhan dapat dilihat pada Gambar 3.3. Berdasarkan gambar tersebut, perancangan penelitian ini berorientasi pada tiga aspek besar, yaitu masukan, proses, dan keluaran.

Diagram blok secara keseluruhan menerima informasi eksistensi dinding pembatas *cell* di mana robot pencari jalur berada. Pendeteksian dilakukan oleh sensor infra merah di dalam blok masukan, menghasilkan keluaran berupa level tegangan yang merepresentasikan kondisi masukan. Level tegangan ini kemudian diproses oleh blok proses, di mana kecerdasan-buatan ini ditanamkan. Kecerdasan-buatan memiliki peta labirin, yang tersusun atas sejumlah *cell* tanpa informasi dinding pembatas *cell*. Kecerdasan-buatan dalam penelitian ini dibangun dengan dua algoritma utama (Gambar 3.3c), yaitu algoritma *Flood-Fill* algoritma dan *Modified Flood-Fill*. Hasil dari pemrosesan di blok proses bagi blok keluaran adalah berupa sinyal kontrol motor penggerak untuk manuver robot pencari jalur di dalam labirin. Sinyal kontrol ini akan memberikan kecepatan dan arah bagi perputaran roda penggerak robot pencari jalur.

Gambar 3.3a. Diagram Blok Sistem

Gambar 3.3b. Diagram Blok Masukan



Gambar 3.3c. Diagram Blok Proses

Gambar 3.3d. Diagram Blok Keluaran

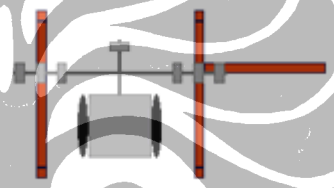
3.1 PERANGKAT KERAS

Kecerdasan-buatan akan digunakan pada rancangan perangkat keras, yang meliputi tiga aspek, yaitu perancangan subsistem masukan, proses, dan keluaran.

3.1.1 Perancangan Subsistem Masukan

Robot pencari jalur menggunakan sensor *near infrared*, yaitu sensor *proximity* untuk mendeteksi keberadaan dinding labirin. Dinding pada labirin dua dimensi ini direpresentasi menggunakan garis hitam.

Sensor *proximity* biasanya dipasang pada "sayap" robot dan menghadap ke bawah dari atas dinding labirin dua dimensi. Jika terdapat dinding di bawah, sensor akan mengembalikan nilai logika benar *(true)*. Jika tidak terdapat dinding di bawah, sensor akan mengembalikan nilai logika salah *(false)*.

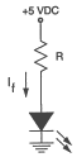


Gambar 3.4. Robot dengan Sensor ***Proximity***

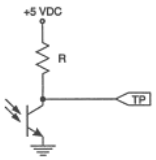
Paling tidak terdapat satu sensor di depan dan tiga sensor masing-masing di sisi kiri dan kanan robot. Sensor depan digunakan untuk mendeteksi keberadaan dinding di depan robot. Sensor sisi digunakan untuk mendeteksi keberadaan dinding samping serta untuk mengkoreksi posisi pergerakan robot. Berdasarkan gambar di atas sebagai contoh, jika sayap kiri mendapatkan pembacaan nilai 010 mengindikasikan robot berada dalam posisi di tengah-tengah *cell*. Jika terbaca 100, mengindikasikan robot terlalu jauh ke sebelah kanan dan perlu dikoreksi ke sebelah kiri. Jika terbaca 001, mengindikasikan robot terlalu jauh ke sebalah kiri dan perlu dikoreksi ke sebelah kanan.

Jenis sensor ini mempunyai keuntungan lain. Jika sayap kanan dibaca dari gambar di atas, akan memberikan nilai 011, mengindikasikan terdapat dinding yang tegak lurus terhadap dinding sebelah kanan. Pemetaan dinding pada *cell* yang bersebelahan memberikan waktu eksplorasi yang lebih singkat.

Implementasi sensor *proximity* bisa dilihat pada Gambar 3.5.



Gambar 3.5a. IR LED Sensor *Proximity*



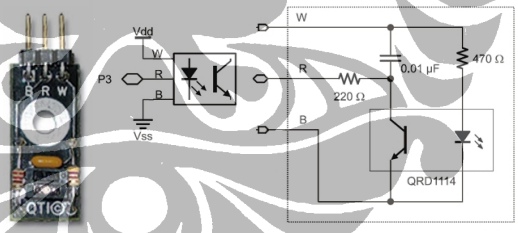
Gambar 3.5b. Phototransistor Sensor *Proximity*

Gambar 3.5c. Sensor *Proximity* (Satu Kemasan)

Resistor pada bagian *emitter* (Gambar 3.5a) dipilih untuk menghasilkan arus sehingga IR LED bisa memancarkan sinar infra merah. Jika tidak dihubungkan secara langsung ke tegangan konstan +5 V, cara lain dapat digunakan, yaitu dengan menghubungkannya melalui transistor yang dapat diaktifkan oleh mikrokontroler. Dengan cara tersebut, LED infra merah dapat dinyalakan oleh mikrokontroler hanya pada saat sensor perlu untuk dibaca. Phototransistor pada bagian sensor (Gambar 3.5b) bertindak sebagai saklar pada saat saturasi. LED infra merah dan sensor dapat berupa komponen yang terpisah, atau berada dalam satu kemasan plastik (Gambar 3.5c).

Penelitian ini menggunakan sensor *proximity* berbasis komponen sensor opto pantul *(reflective object sensor)* Fairchild Semiconductor QRD1114 [14].

Sensor *proximity* berbasis QRD1114 ditunjukkan oleh Gambar 3.6.



Gambar 3.6. Sensor *Proximity* Berbasis QRD1114 [8]

Kotak hitam kecil di atas label QTI merupakan sensor opto pantul. Sensor ini mempunyai sebuah diode infra merah di balik layar bening dan sebuah transistor infra merah di balik jendela hitam. Ketika infra merah dipancarkan oleh diode, memantul pada sebuah permukaan dan kembali ke jendela hitam, infra merah tersebut mengenai basis transistor, mengakibatkan transistor menghantarkan arus. Semakin banyak infra merah mengenai basis transistor, semakin banyak arus yang dapat dihantarkan [13].

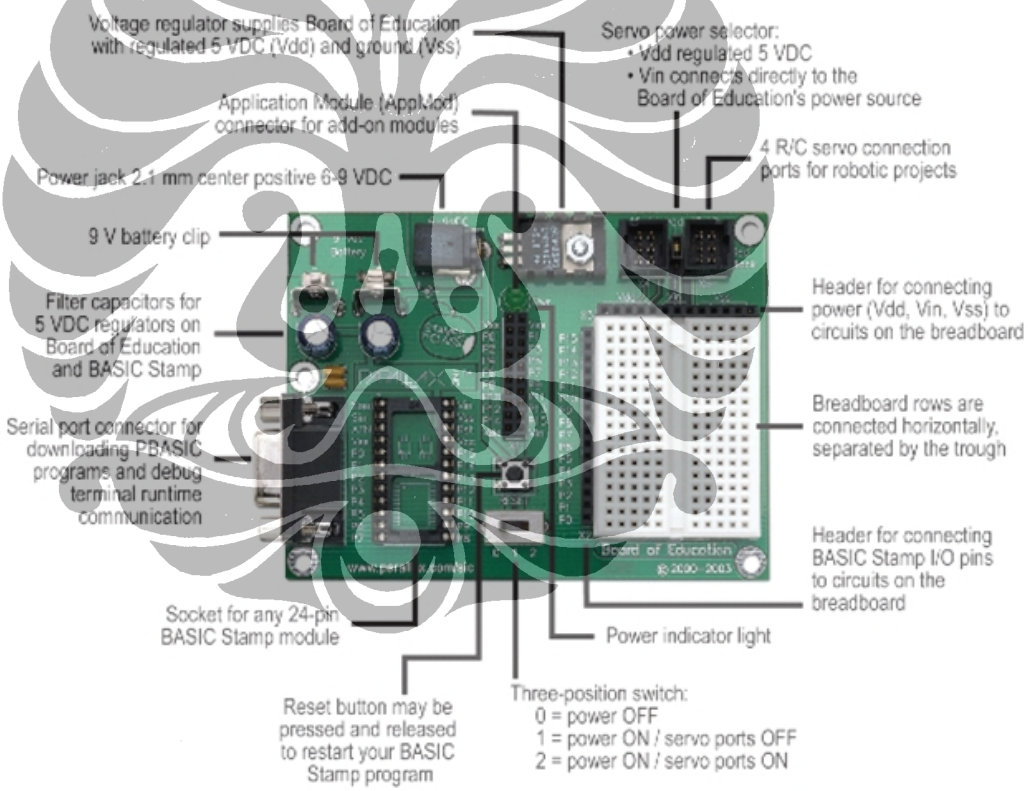
3.1.2 Perancangan Subsistem Proses

Pemrosesan dilakukan oleh mikrokontroler, yang melakukan pembacaan masukan (sensor), mengontrol keluaran (pergerakan motor), dan mencari jalur di lingkungan labirin.

Beberapa hal yang harus diperhatikan dalam pemilihan mikrokontroler, yaitu jumlah memori dan port I/O yang dimiliki.

Jumlah memori tertentu diperlukan dalam pemrosesan algoritma kecerdasan-buatan sedangkan jumlah port I/O diperlukan oleh masukan (sensor, saklar, tombol) dan keluaran (motor, LCD, LED, speaker).

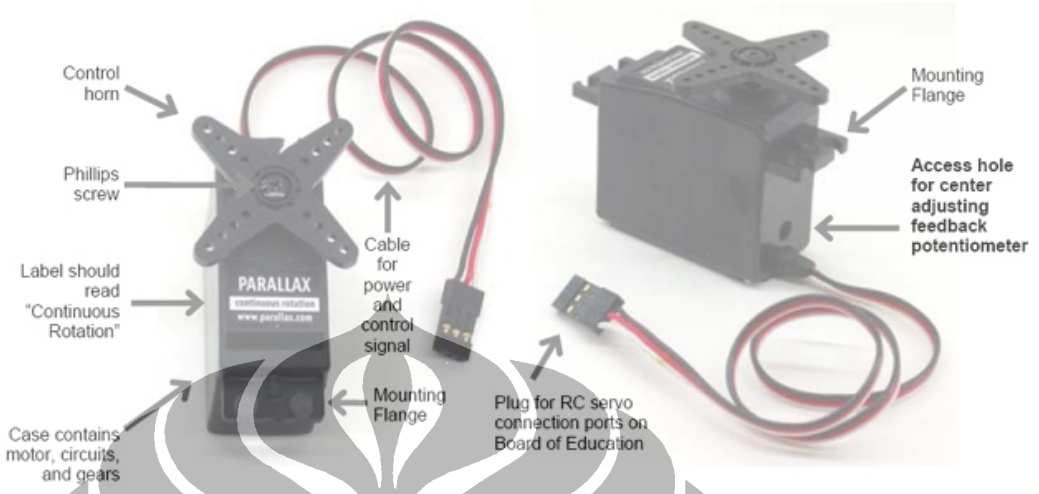
Penelitian ini direncanakan menggunakan *Board of Education* dari Parallax sebagai *carrier board* dari prosesor BASIC Stamp.



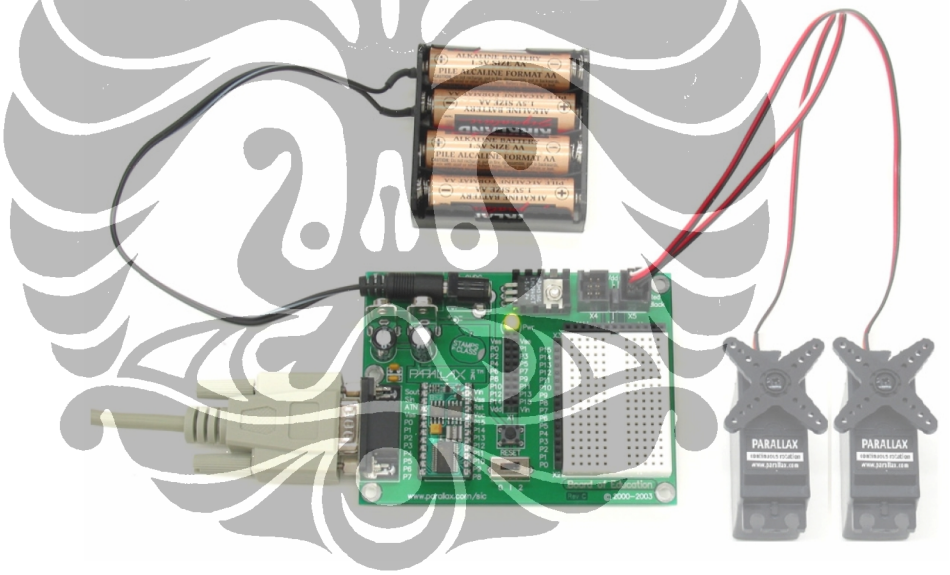
Gambar 3.7. *Board of Education* (BOE) Rev. C *Carrier Board* untuk Modul Mikrokontroler BASIC Stamp [11]

3.1.3 Perancangan Subsistem Keluaran

Motor penggerak robot dalam penelitian ini menggunakan jenis *continuous rotation servo*.

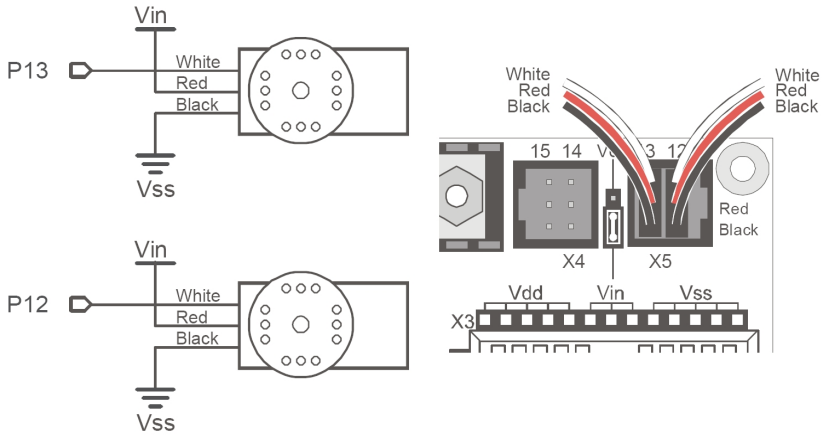


Gambar 3.8. *Continuous Rotation Servo* [8]



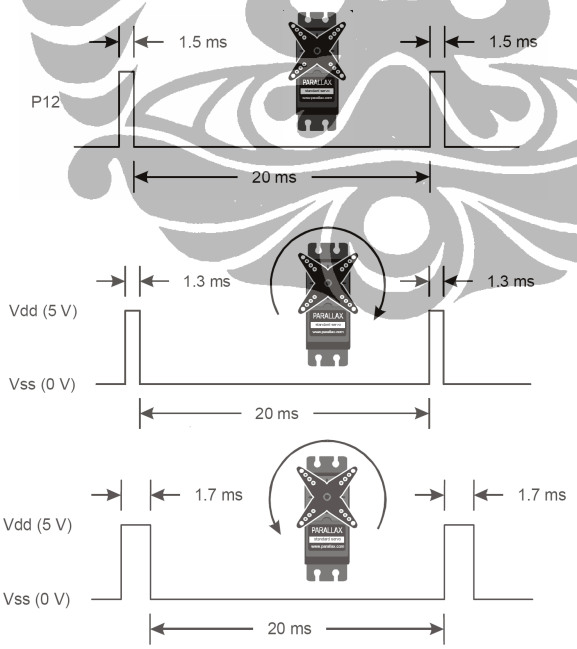
Gambar 3.9. Koneksi *Board of Education* dan *Servo* [8]

Servo mempunyai tiga koneksi, yaitu +5 V, *ground*, dan *signal*. *Signal* merupakan pulsa dengan panjang beberapa mili detik. Dengan mengontrol lebar pulsa ke *servo*, mikrokontroler akan dapat menentukan apakah perputaran bergerak maju *(forward)* atau mundur *(backward)*, juga dapat menentukan seberapa cepat perputaran tersebut.



Gambar 3.10. *Pinout Servo* (+5 V, ground, dan signal) [8]

Gambar 3.11 memperlihatkan kode program pengontrol *servo*. Pada intinya, untuk membuat *servo* tidak berputar, *pin* keluaran ke *servo* dari mikrokontroler BASIC Stamp diberi pulsa dengan lebar 1,5 mili detik (gambar bagian atas). Untuk membuat *servo* berputar searah jarum jam dengan kecepatan maksimum, *pin* keluaran diberi pulsa dengan lebar 1,3 mili detik. Untuk membuat *servo* berputar berlawanan arah jarum jam dengan kecepatan maksimum, *pin* keluaran diberi pulsa dengan lebar 1,7 mili detik.




```
DO
  PULSOUT 12, 750
  PAUSE 20
LOOP
```

```
DO
  PULSOUT 12, 650
  PAUSE 20
LOOP
```

```
DO
  PULSOUT 12, 850
  PAUSE 20
LOOP
```

Gambar 3.11. Kode Program Pengontrol *Servo* [8]

3.2 PERANGKAT LUNAK

Kecerdasan-buatan yang dirancang mengacu kepada dua algoritma utama, yaitu algoritma *Flood-Fill* dan algoritma *Modified Flood-Fill*.

3.2.1 Perancangan dengan Algoritma *Flood-Fill*

Algoritma ini melibatkan pemberian nilai pada masing-masing *cell* penyusun labirin di mana nilai ini merepresentasikan jarak dari sebarang *cell* ke *cell* tujuan. Selanjutnya *cell* tujuan diberikan nilai 0. Jika robot pencari jalur berada di sebuah *cell* dengan nilai 1, robot tersebut 1 *cell* jauhnya dari tujuan. Jika robot berada pada *cell* dengan nilai 3, robot tersebut 3 *cell* jauhnya dari tujuan.

Diasumsi robot tidak melakukan pergerakan secara diagonal, maka nilai untuk labirin dengan ukuran matriks 5 x 5 tanpa dinding diperlihatkan oleh Gambar 3.12.

Gambar 3.12. Pergerakan Awal Robot dalam Algoritma *Flood-Fill*

Untuk labirin dengan ukuran 16 baris kali 16 kolom sama dengan 256 nilai *cell*, diperlukan memori sebanyak 256 *byte* untuk menyimpan nilai jarak.

Ketika pergerakan dimulai, robot harus memeriksa semua *cell* bersebelahan yang tidak dipisahkan oleh dinding dan memilih satu dengan nilai jarak terendah. Pada contoh di atas, robot akan mengabaikan sisi kiri (Barat) sebab ada dinding, dan selanjutnya akan mengecek nilai jarak pada *cell* di bagian atas (Utara), *cell* di sisi kanan (Timur), dan *cell* di bagian bawah (Selatan) karena *cell-cell* tersebut tidak dipisahkan oleh dinding. *Cell* ke arah Utara bernilai 2, *cell* ke arah Timur bernilai 2, dan *cell* ke arah Selatan bernilai 4. Rutin pada kecerdasan-buatan akan

men-sorting nilai-nilai tersebut untuk menentukan *cell* mana yang mempunyai nilai terendah. *Cell* ke arah Utara dan Timur mempunyai nilai jarak 2. Itu berarti robot dapat pergi ke arah Utara atau Timur dan melewati jumlah *cell* yang sama dalam perjalanan ke *cell* tujuan. Karena berputar akan menghabiskan waktu lebih lama, robot akan memilih lurus ke arah *cell* Utara. Jadi proses pengambilan keputusan akan seperti Gambar 3.13.

Menentukan *cell* bersebelahan dengan nilai jarak terendah

Apakah *cell* Utara dibatasi dinding ?
Ya **à** Abaikan *cell* Utara
Tidak **à** *Push cell* Utara ke *stack* untuk diperiksa

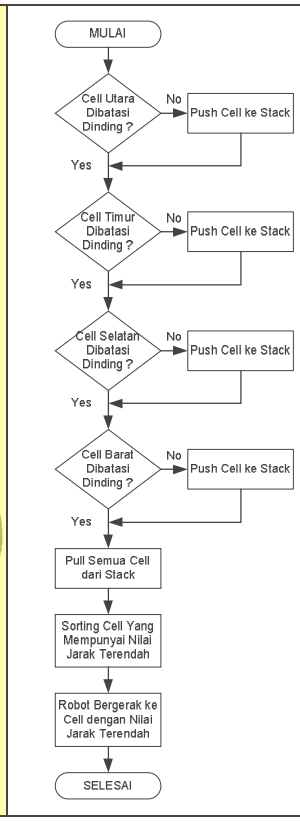
Apakah *cell* di Timur dibatasi dinding ?
Ya **à** Abaikan *cell* Timur
Tidak **à** *Push cell* Timur ke *stack* untuk diperiksa

Apakah *cell* Selatan dibatasi dinding ?
Ya **à** Abaikan *cell* Selatan
Tidak **à** *Push cell* Selatan ke *stack* untuk diperiksa

Apakah *cell* Barat dibatasi dinding ?
Ya **à** Abaikan *cell* Barat
Tidak **à** *Push cell* Barat ke *stack* untuk diperiksa

Pull semua *cell* dari *stack* (*stack* kosong sekarang)
Sorting mencari nilai jarak terendah *cell-cell* tersebut

Robot bergerak ke *cell* bersebelahan dengan nilai jarak terendah



Gambar 3.13. Algoritma dan Diagram Alir Menentukan *Cell* dengan Nilai Jarak Terendah

Dinding labirin mempengaruhi nilai jarak suatu *cell* sehingga informasi dinding *cell* tersebut perlu disimpan. Pada labirin yang tersusun atas 256 *cell* (matriks 16 x 16) perlu dialokasikan lagi memori sebanyak 256 *byte* untuk keperluan penyimpanan informasi tersebut. Terdapat 8 *bit* pada 1 *byte* untuk sebuah *cell*. 4 *bit* pertama dapat merepresentasi informasi keberadaan dinding *cell*, menyisakan 4 *bit* untuk keperluan lain.

Bit No.	7	6	5	4	3	2	1	0
Dinding					B	S	T	U

Gambar 3.14. *Byte* Merepresentasi Status Dinding *Cell*

di mana : B = dinding Barat, S = dinding Selatan, T = dinding Timur, dan U = dinding Utara.

Informasi dinding *cell* dapat dimanfaatkan oleh dua *cell* yang bersebelahan sehingga jika terdapat *updating* informasi dinding untuk satu *cell*, dapat juga dilakukan *updating* informasi dinding untuk *cell* yang bersebelahan. Instruksi untuk *updating* peta dinding dapat seperti Gambar 3.15

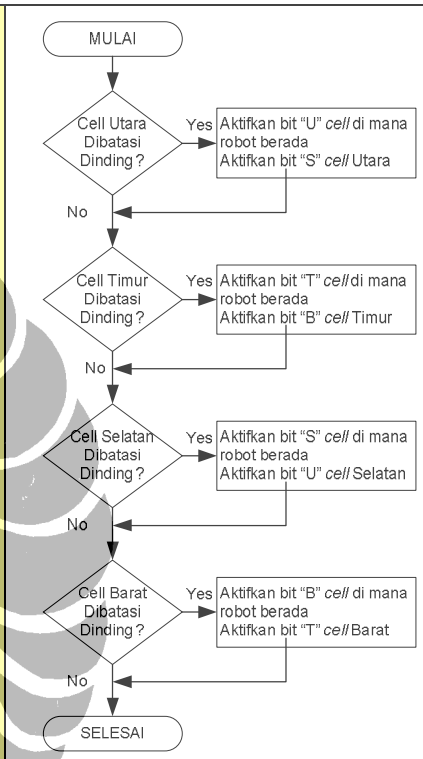
Updating peta dinding

Apakah *cell* Utara dibatasi dinding ?
Ya à Aktifkan *bit* "U" *cell* di mana robot berada
Aktifkan *bit* "S" *cell* Utara
Tidak à tidak ada *update*

Apakah *cell* Timur dibatasi dinding ?
Ya à Aktifkan *bit* "T" *cell* di mana robot berada
Aktifkan *bit* "B" *cell* Timur
Tidak à tidak ada *update*

Apakah *cell* Selatan dibatasi dinding ?
Ya à Aktifkan *bit* "S" *cell* di mana robot berada
Aktifkan *bit* "U" *cell* Selatan
Tidak à tidak ada *update*

Apakah *cell* Barat dibatasi dinding ?
Ya à Aktifkan *bit* "B" *cell* di mana robot berada
Aktifkan *bit* "T" *cell* Barat
Tidak à tidak ada *update*



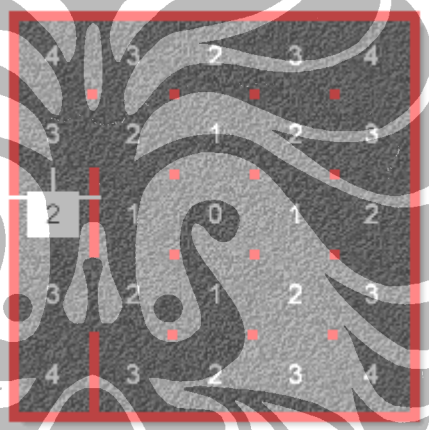
Gambar 3.15. Algoritma dan Diagram Alir *Updating* Peta Dinding *Cell*

Berdasarkan mekanisme pada Gambar 3.15, kecerdasan-buatan yang dibangun mempunyai cara untuk menyimpan informasi dinding *cell*. Jika dinding baru ditemukan, nilai jarak *cell* akan terpengaruh sehingga diperlukan suatu cara untuk melakukan *updating* nilai jarak tersebut.

Berdasarkan Gambar 3.16, seandainya robot menemukan dinding, robot tidak dapat menuju ke Barat (ke kiri) dan ke Timur (ke kanan), hanya dapat bergerak ke Utara (ke atas) atau ke Selatan (ke bawah).

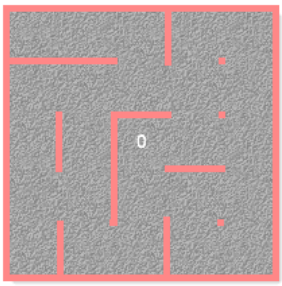
Jika bergerak ke Utara atau ke Selatan berarti menaikkan nilai jarak, sehingga perlu dilakukan *updating* pada nilai *cell* sebagai akibat ditemukannya dinding baru ini. Untuk melakukan ini, kecerdasan-buatan melakukan *"flood"* pada labirin dengan nilai baru.

"Flood" dalam hal ini berarti memberikan nilai jarak lebih besar pada *cell* bersebelahan yang tidak dibatasi dinding dari arah *cell* di mana robot berada *(open neighbour cell)*. Nilai yang diberikan pada *open neighbour cell* lebih besar satu daripada nilai *cell* di mana robot berada.

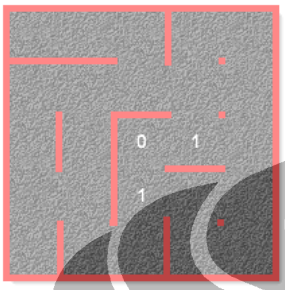


Gambar 3.16. Penemuan Dinding Labirin dalam Algoritma *Flood-Fill*

Sebagai contoh *"flooding"* pada labirin, asumsikan robot telah melakukan pergerakan dan menemukan beberapa dinding. Simulasi *flooding* diperlihatkan pada Gambar 3.17.

1. 

Rutin mulai menginisialisasi array yang menyimpan nilai jarak dan memberikan nilai 0 pada *cell* tujuan.

2. 

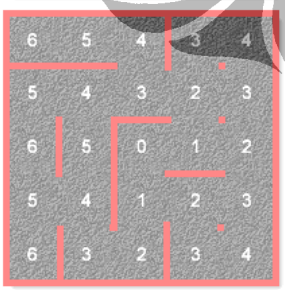
Rutin kemudian mengambil *cell* bersebelahan yang tidak dibatasi dinding dan memberi nilai jarak yang lebih tinggi, yaitu 1.

3. 

Rutin kembali menemukan *cell* bersebelahan yang tidak dibatasi dinding dan memberi nilai jarak yang lebih tinggi, yaitu 2.

4. 

Beberapa iterasi dilakukan

5. 

Hal ini diulang sebanyak yang diperlukan sampai *cell-cell* yang lain mempunyai nilai jarak

Gambar 3.17. Simulasi Algoritma *Flood-Fill*

Terlihat bagaimana nilai-nilai ini mengarahkan robot dari *cell* awal menuju *cell* tujuan melalui jalur terpendek.

Intruksi untuk “flooding” pada labirin dengan nilai jarak diperlihatkan oleh Gambar 3.18.

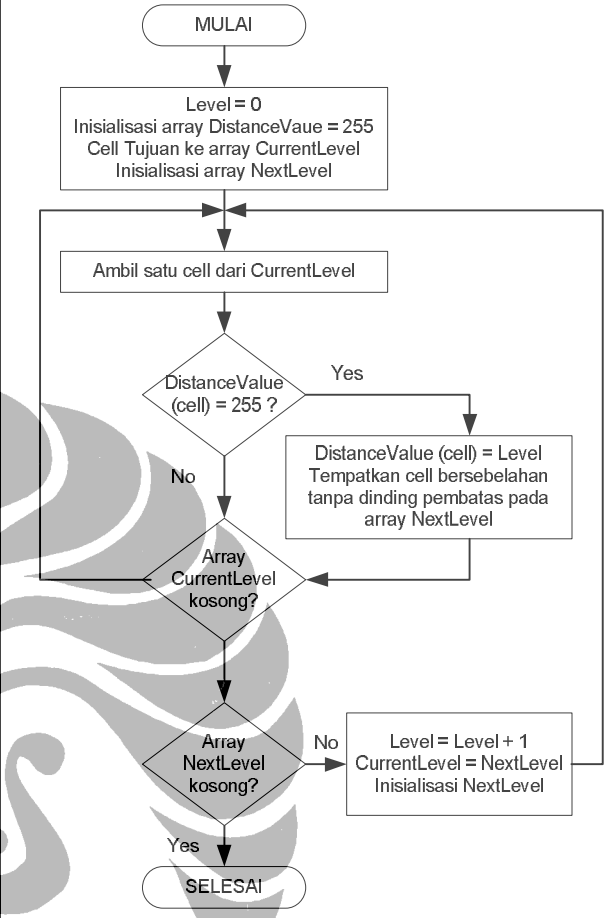
Flood labirin dengan nilai jarak yang baru
Variabel Level = 0

Inisialisasi array DistanceValue
sehingga semuanya bernilai = 255

Tempatkan *cell* tujuan dalam array CurrentLevel

Inisialisasi array kedua NextLevel

Mulai :
Ulangi instruksi berikut sampai
CurrentLevel kosong :
{
Ambil satu *cell* dari CurrentLevel
If DistanceValue(*cell*) = 255 then
DistanceValue(*cell*) = Level, dan
tempatkan *cell* bersebelahan tanpa
dinding pembatas pada NextLevel
End if
}
Array CurrentLevel sekarang kosong
Apakah array NextLevel kosong ?
Tidak **à**
{
Level = Level + 1
CurrentLevel = NextLevel
Inisialisasi NextLevel
Kembali ke ‘Mulai :’
}
Ya **à** Flooding labirin selesai



Gambar 3.18. Algoritma dan Diagram Alir Flood Labirin dengan Nilai Jarak Baru

Algoritma *Flood-Fill* memberikan cara yang baik untuk menemukan jalur terpendek dari *cell* awal menuju *cell* tujuan. Untuk labirin tersusun atas 256 *cell*, diperlukan 512 *byte* RAM untuk mengimplementasikan rutin kecerdasan buatan: satu array 256 *byte* untuk menyimpan informasi nilai jarak dan satu array 256 *byte* untuk menyimpan informasi peta dinding.

Setiap kali robot memasuki sebuah *cell*, kecerdasan buatannya akan melakukan langkah-langkah berikut

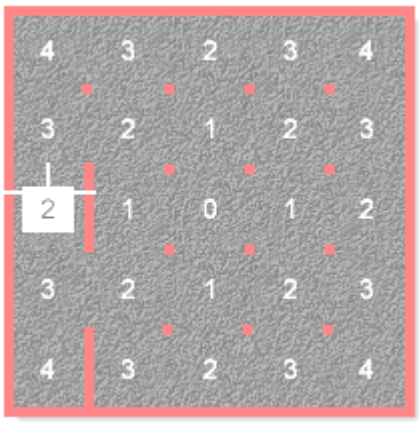
1. *Update* peta dinding
2. Flood labirin dengan nilai jarak yang baru
3. Menentukan *cell* bersebelahan yang mempunyai nilai jarak terendah
4. Bergerak ke *cell* bersebelahan yang mempunyai nilai jarak terendah

3.2.2 Perancangan dengan Algoritma *Modified Flood-Fill*

Algoritma ini sama dengan algoritma *Flood-Fill* regular di mana kecerdasan-buatan menggunakan nilai jarak untuk bergerak di dalam labirin. Nilai jarak, yang merepresentasi seberapa jauh robot dari *cell* tujuan, diikuti secara menurun (descending order) sampai robot mencapai tujuannya.

Gambar 3.19. Pergerakan Awal Robot dalam Algoritma *Modified Flood-Fill*

Ketika robot menemukan dinding baru, nilai jarak perlu di-*update*. Algoritma *Modified Flood-Fill* hanya mengubah nilai-nilai yang perlu diubah. Berbeda dengan algoritma food-fill regular, yang melakukan flooding seluruh labirin dengan suatu nilai. Sebagai contoh, berdasarkan Gambar 3.19, robot bergerak maju satu *cell* dan menemukan sebuah dinding.



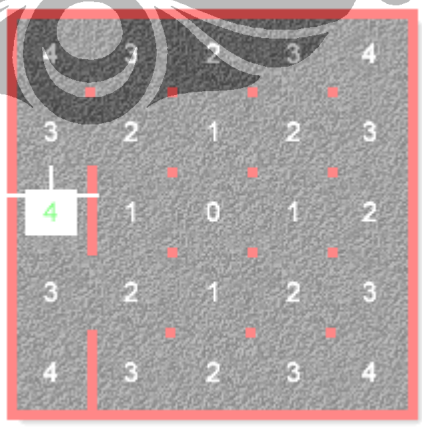
Gambar 3.20. Penemuan Dinding Labirin dalam Algoritma *Modified Flood-Fill*

Robot tidak dapat bergerak ke Barat dan ke Timur, hanya dapat bergerak ke Utara atau ke Selatan. Jika bergerak ke Utara atau ke Selatan berarti menaikkan nilai jarak, sehingga nilai *cell* perlu untuk di-*update*.

Ketika robot menghadapi kasus tersebut, kecerdasan-buatan akan melakukan hal berikut

Jika sebuah *cell* bukan merupakan *cell* tujuan, nilai jaraknya adalah satu plus nilai minimum *cell* bersebelahan tanpa dinding pembatas.

Berdasarkan contoh di atas, nilai minimum *cell* bersebelahan tanpa dinding pembatas adalah 3. Tambahkan 1 pada nilai ini menghasilkan nilai 3 + 1 = 4. *Update* nilai jarak *cell* dalam labirin diperlihatkan pada Gambar 3.21.



Gambar 3.21. *Updating* Nilai Jarak *Cell* dalam Algoritma *Modified Flood-Fill*

Ada waktunya ketika *updating* nilai jarak *cell* menyebabkan *cell* bersebelahan menyalahi aturan "1 + nilai minimum", sehingga hal tersebut harus dicek juga. Berdasarkan contoh di atas *cell* Utara dan Selatan mempunyai nilai minimum 2. Tambahkan 1 pada nilai ini menghasilkan 2 + 1 = 3 sehingga *cell* Utara dan Selatan tidak menyalahi aturan dan *updating* rutin selesai. Selanjutnya, jika nilai-nilai *cell* telah di-*update* semua, robot bisa bergerak mengikuti nilai jarak tersebut secara menurun *(descending order).* Prosedur algoritma *Modified Flood-Fill* dalam melakukan *updating* nilai jarak, diperlihatkan oleh Gambar 3.22 berikut.

Update nilai jarak (jika diperlukan)

Pastikan *stack* kosong

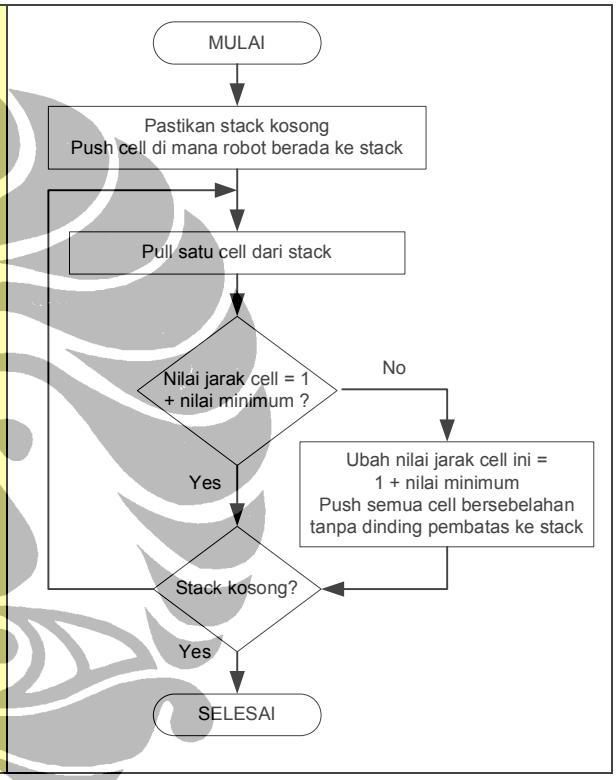
Push cell (di mana robot berada) ke *stack*

Ulangi set instruksi berikut sampai *stack* kosong :

{

Pull satu *cell* dari *stack*
Apakan nilai jarak *cell* ini = 1 + nilai minimum *cell* bersebelahan tanpa dinding pembatas ?
Tidak **à**
Ubah nilai jarak *cell* ini = 1 + nilai minimum *cell* bersebelahan tanpa dinding pembatas
dan
push semua *cell* bersebelahan tanpa dinding pembatas ke *stack* untuk diperiksa
Ya **à** Tidak ada *update*

}

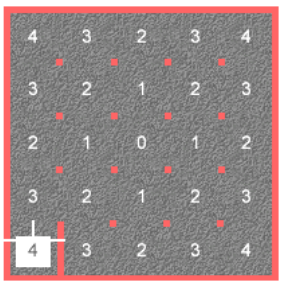


Gambar 3.22. Algoritma dan Diagram Alir *Update* Nilai Jarak (*Modified Flood-Fill*)

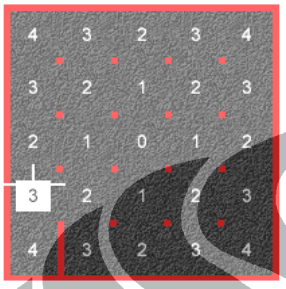
Dengan hanya melakukan *updating* pada nilai-nilai jarak yang perlu di-*update*, robot dapat melakukan pergerakan jauh lebih cepat. Setiap kali robot memasuki satu *cell*, kecerdasan-buatan melakukan langkah-langkah berikut

1. *Update* peta dinding
2. *Update* nilai jarak (hanya jika diperlukan)
3. Menentukan *cell* bersebelahan yang mempunyai nilai jarak terendah
4. Bergerak ke *cell* bersebelahan yang mempunyai nilai jarak terendah

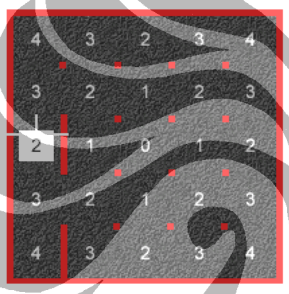
Simulasi Algoritma *Modified Flood-Fill*

1. 

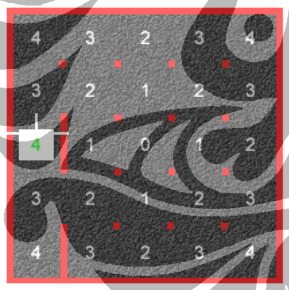
Robot berada di *cell* awal. Nomor mengindikasikan jarak sebarang *cell* ke *cell* tujuan di tengah labirin.

2. 

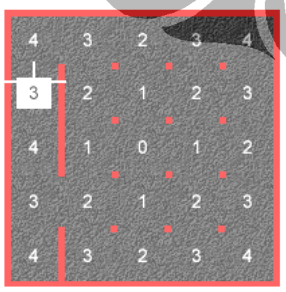
Dalam setiap pergerakan, kecerdasan-buatan mengecek *cell* bersebelahan dan pindah ke *cell* dengan nilai jarak terendah. Dalam kasus ini ada dua *cell* dengan nilai 2 namun berhubung berbelok memerlukan waktu lebih, kecerdasan-buatan menginstruksikan robot bergerak maju.

3. 

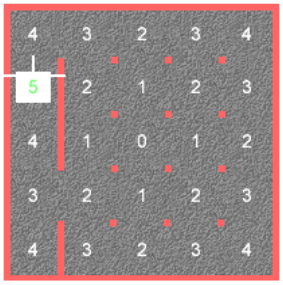
Di *cell* ini, robot menemukan dinding di sisi Timur sehingga robot tidak dapat bergerak ke sisi tersebut. Dua *cell* bersebelahan lain mempunyai nilai jarak lebih tinggi daripada *cell* di mana robot berada sehingga nilai jaraknya diubah menjadi 3 + 1 = 4.

4. 

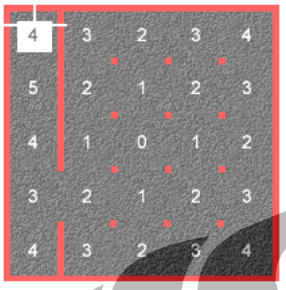
Selanjutnya, robot bisa melakukan pergerakan ke *cell* dengan nilai jarak terendah. Karena dua *cell* bersebelahan mempunyai nilai jarak 3, robot memilih melakukan pergerakan maju.

5. 

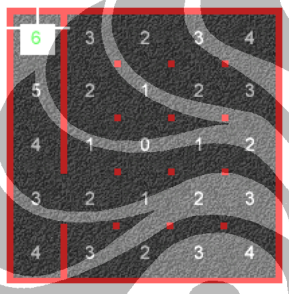
Menemukan dinding lain dan *update* dilakukan pada *cell* ini.

6. 

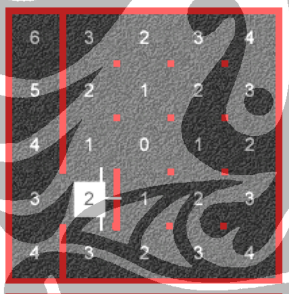
Update telah dilakukan.

7. 

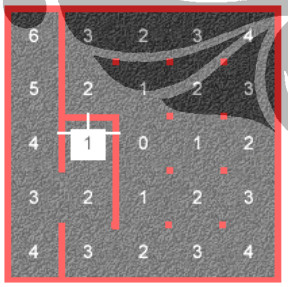
Jalan buntu. *Update* nilai jarak dilakukan dan robot melakukan putaran 180 derajat. Selanjutnya, pergerakan hanya masalah mengikuti nilai jarak *cell* secara menurun (descending order)

8. 

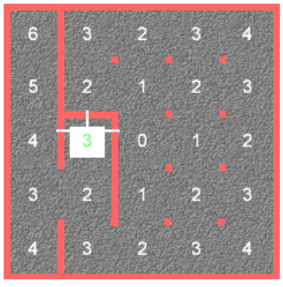
Update telah dilakukan

9. 

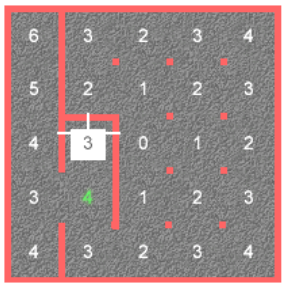
Menemukan dinding lain. Karena *cell* di sebelah Utara mempunyai nilai jarak terendah, robot melakukan putaran 90 derajat ke arah tersebut.

10. 

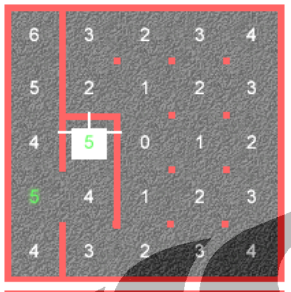
Jalan buntu lagi. *Update* nilai jarak *cell* ini.

11. 

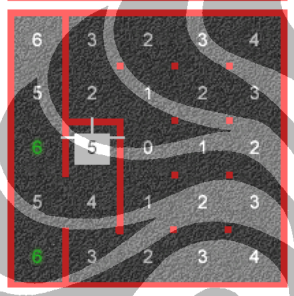
Tetapi ini menyebabkan nilai jarak *cell* di sebelah Selatan menjadi salah. *Update* akan dilakukan.

12. 

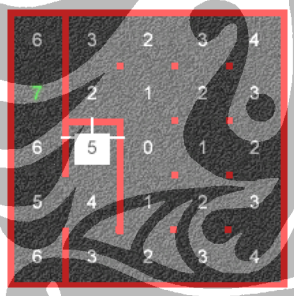
Beberapa *update* lagi untuk nilai jarak *cell*...

13. 

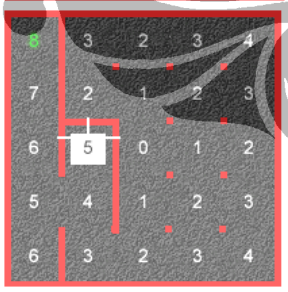
Beberapa *update* lagi untuk nilai jarak *cell*...

14. 

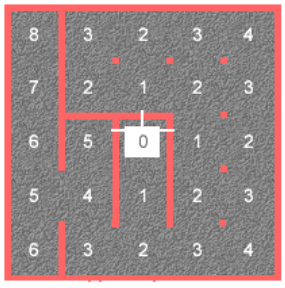
Beberapa *update* lagi untuk nilai jarak *cell*...

15. 

Beberapa *update* lagi untuk nilai jarak *cell*...

16. 

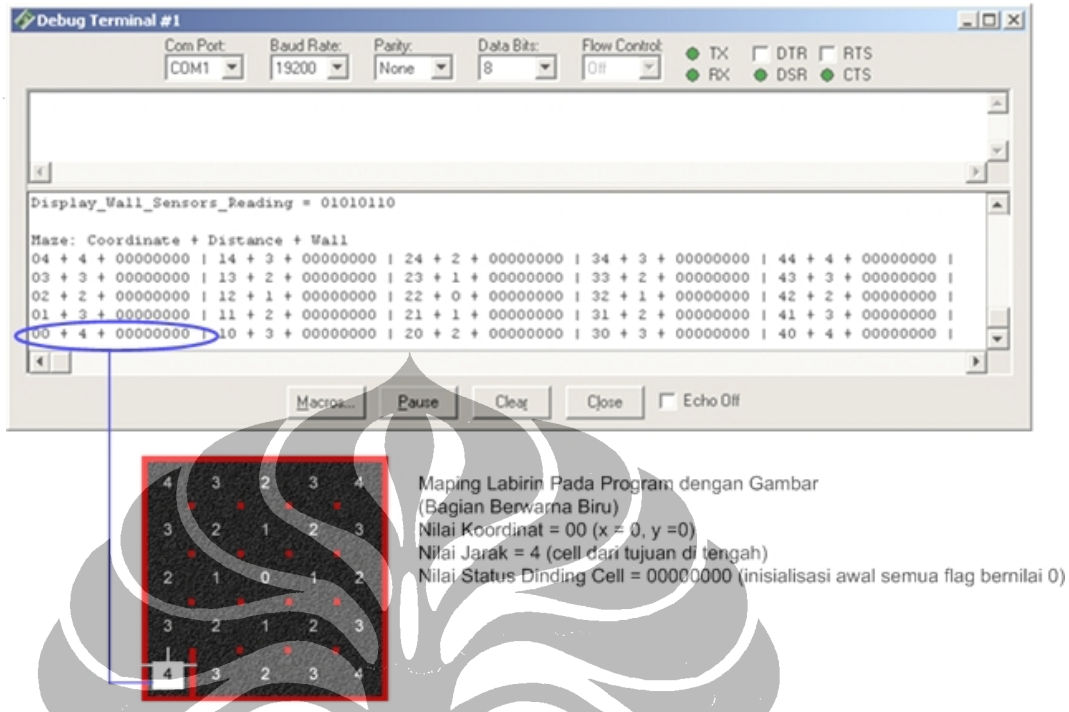
Update selesai. Robot melakukan putaran 180 derajat dan mengikuti nilai jarak secara menurun.

17. 

Terlihat bahwa bagian sebelah kanan labirin belum tereksplorasi. Hal tersebut tidak menjadi masalah karena algoritma *Modified Flood-Fill* telah memberikan jalur terpendek dari *cell* awal menuju *cell* tujuan.

Gambar 3.23. Simulasi Algoritma *Modified Flood-Fill*

Sebagian implementasi algoritma bisa dilihat pada Gambar 3.24.



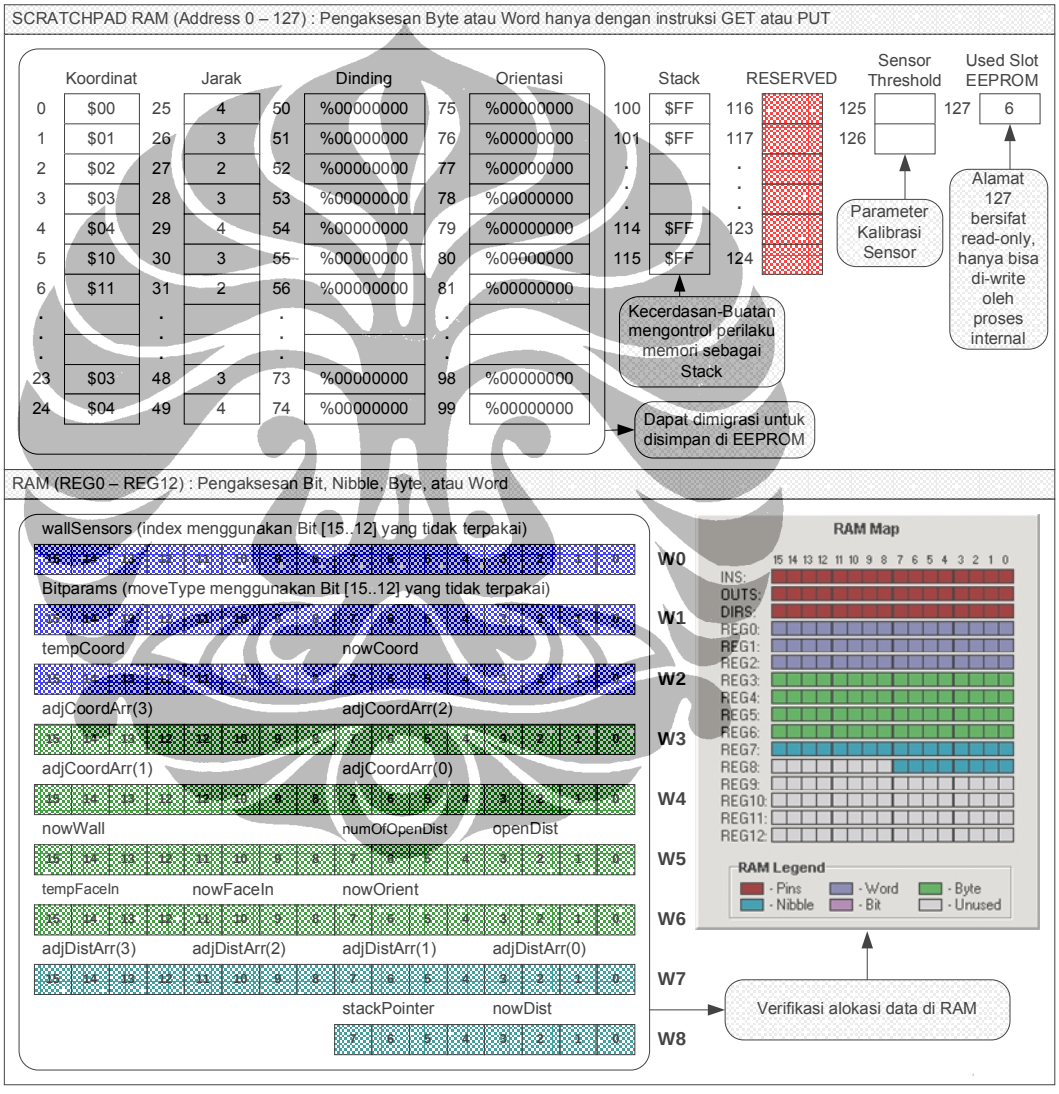
Gambar 3.24. Inisialisasi Awal Peta Labirin Pada Program PBASIC

Inisialisasi awal peta labirin telah dilakukan program yang kemudian di-debug memberikan visualisasi labirin dengan *cell-cell* penyusun yang mempunyai nilai koordinat, nilai jarak, dan nilai *byte* merepresentasi keberadaan dinding *cell*.

Sebagai contoh, *cell* kiri bawah dari peta labirin diberikan nilai koordinat = (0,0), nilai jarak dari *cell* tujuan di tengah = 4, dan nilai *byte* masih bernilai *reset* = 00000000.

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	isBrokenRule	status memenuhi rule algoritma dalam perhitungan jarak cell
1	isCheckpoint	status validitas pencapaian checkpoint (dinding depan & belakang)
2	isFinish	status pencapaian cell tujuan
3	isNoWall	status updating dinding cell
4	isSideWall	status validitas dinding samping
5	isSetPath	status pengesetan jalur ke cell tujuan
6	isVisitedCell	status kunjungan ke suatu cell
7	scanResult	status scan dinding cell setelah transisi antar cell
8	task	switching task di internal program
9	RESERVED	
10	RESERVED	
11	RESERVED	
12-15	moveType	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.

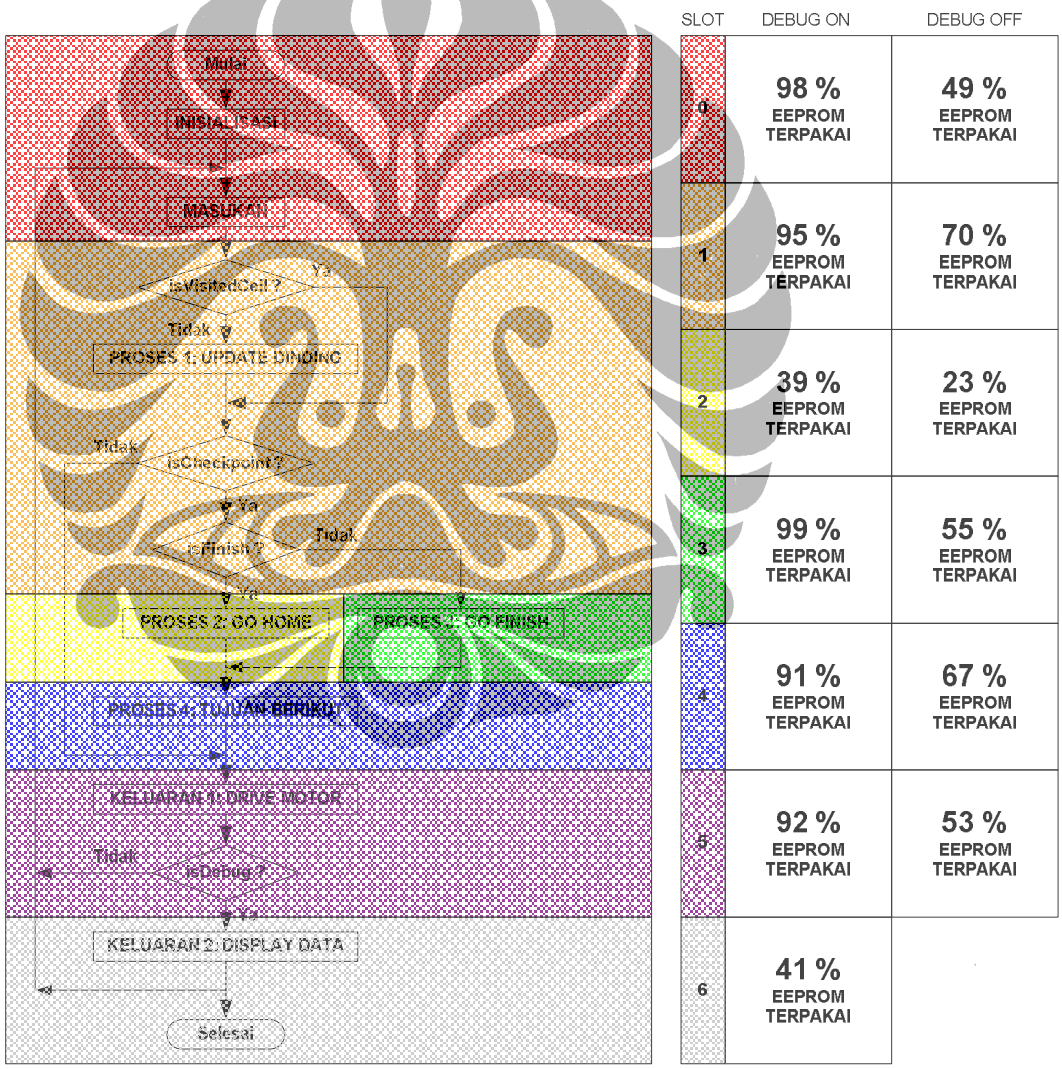
d. *nowCoord* (1 Byte), berfungsi sebagai buffer data koordinat dari Scratchpad RAM, untuk pemrosesan lebih lanjut.
e. *adjCoordArray* (array Byte (4)), berfungsi untuk menyimpan data koordinat cell yang bersebelahan tanpa dinding pembatas.
f. *nowWall* (1 Byte), berfungsi sebagai buffer data dinding 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.

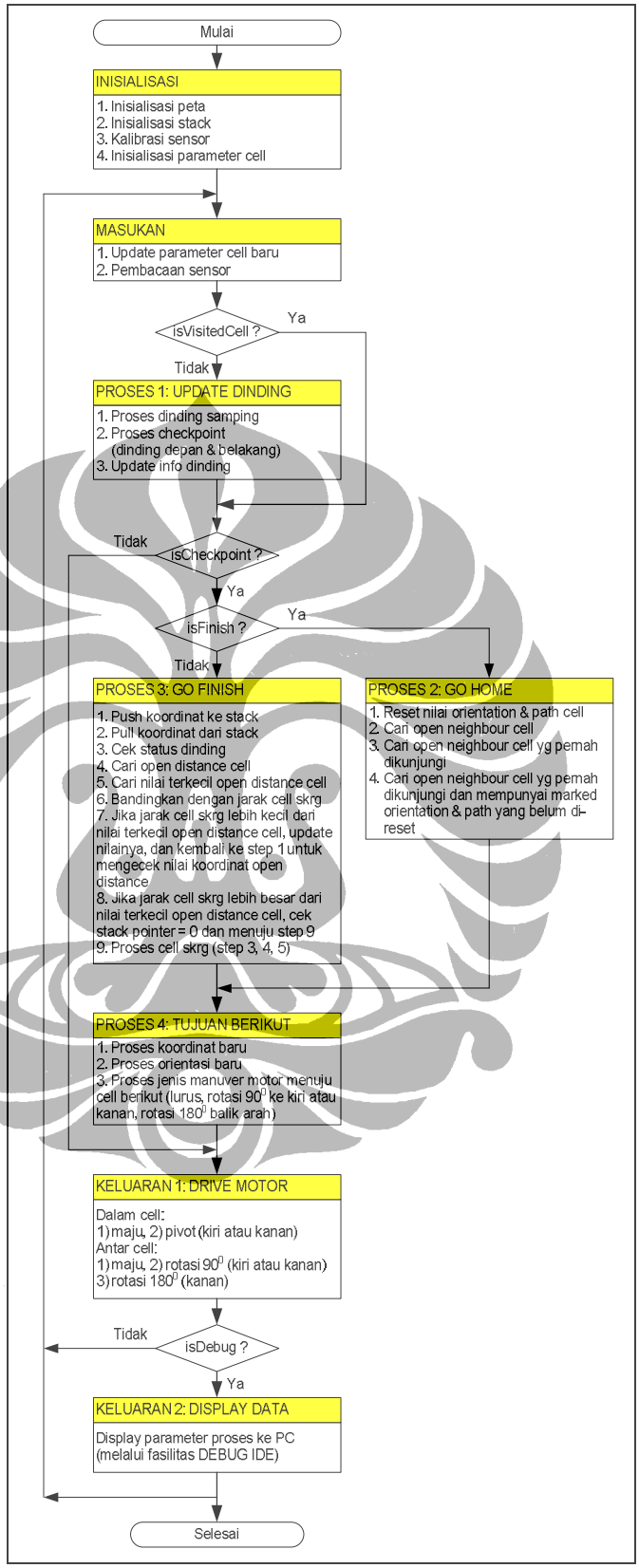
4.1 IMPLEMENTASI UMUM

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

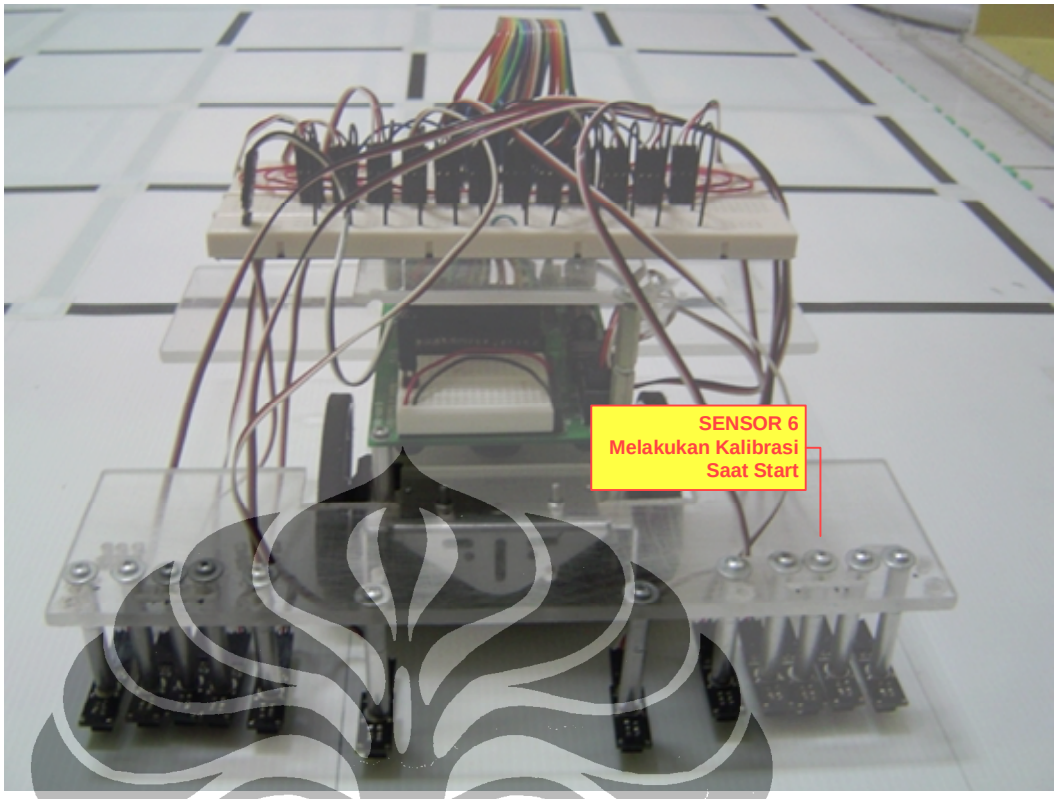
Write_Data = rutin menyimpan data ke Scratchpad RAM

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.



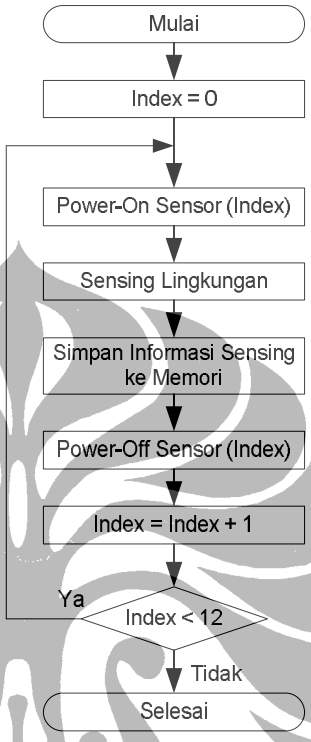

```
HIGH FarMiddleLeftSensor                    ' Turn SENSOR on
HIGH SensorInput                             ' Discharge capacitor
PAUSE 1
RCTIME SensorInput, 1, sensorThreshold      ' Measure charge time
LOW FarMiddleLeftSensor                      ' Turn SENSOR off

sensorThreshold = sensorThreshold / 4        ' Take 1/4 average

IF sensorThreshold > CodeOverhead THEN   ' Account for code overhead
  sensorThreshold = sensorThreshold - CodeOverhead
ELSE
  sensorThreshold = 0
ENDIF
```

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.

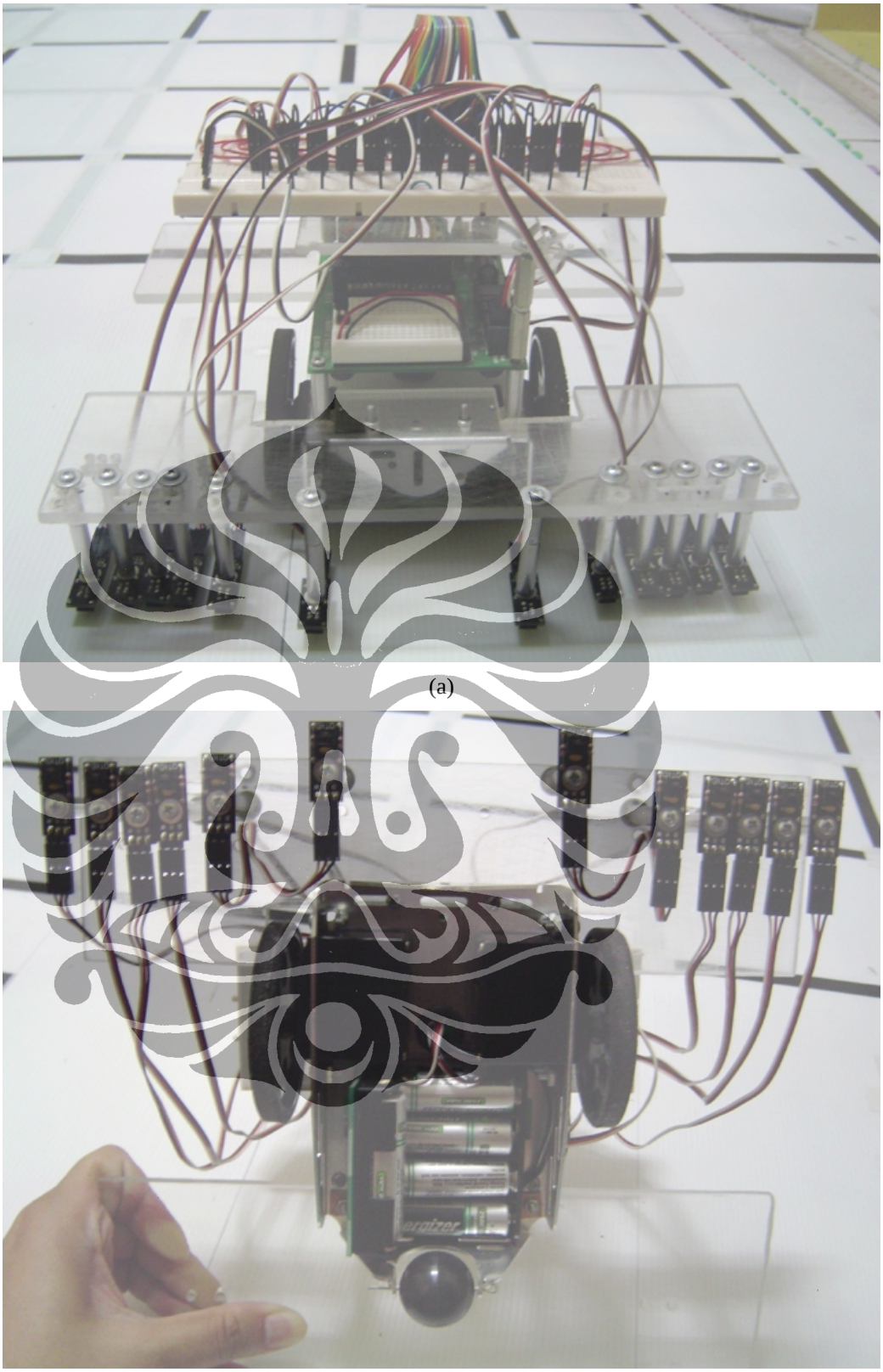
Gambar 4.7 memperlihatkan mekanisme kerja sensor yang hanya diaktifkan pada saat diperlukan saja. Ini menghemat pemakaian arus sehingga robot pencari jalur bisa bergerak lebih lama jika menggunakan sumber energi portabel, seperti baterai.



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 *(time-switching)*
15	Penggunaan internal untuk pewaktuan



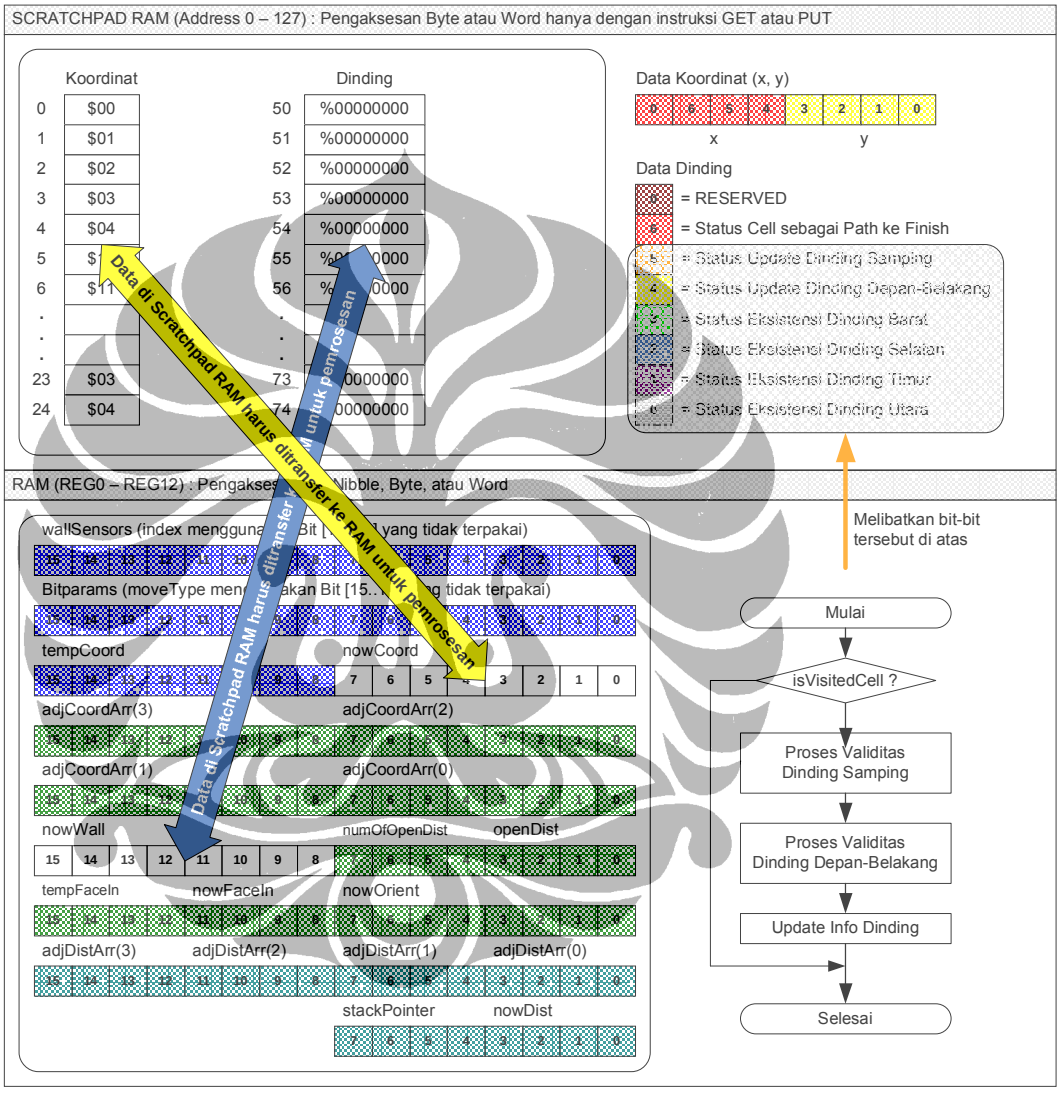
(a)

(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.

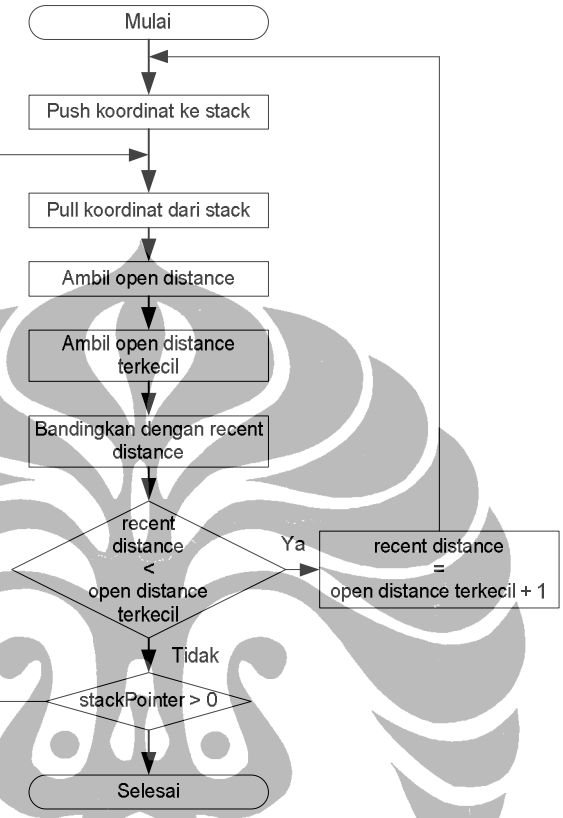


Gambar 4.9. Pemrosesan Informasi Dinding Cell

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

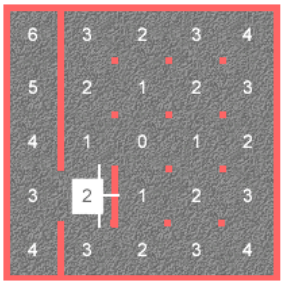
```
Push_Coordinate:

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
```

```
Pull_Coordinate:

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
```

Gambar 4.11. Kode Kecerdasan-Buatan untuk Push dan Pull

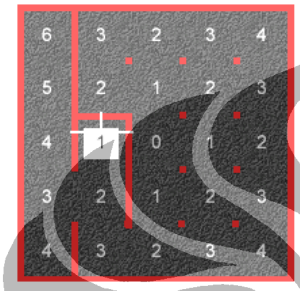
Simulasi kerja kode pemroses jarak cell ini diperlihatkan pada Gambar 4.12 yang mengambil sebagian simulasi deskriptif pada bagian sebelumnya.

9.



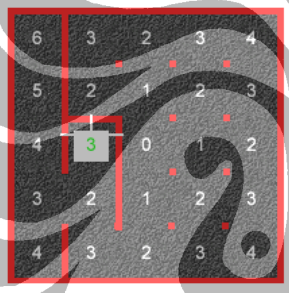
- **Push koordinat (1,1) ke stack**
- **Pull koordinat (1,1) dari stack**
- **Ambil open distance, yaitu 3 (sisi barat), 3 (sisi selatan), dan 1 (sisi utara)**
- **Ambil open distance terkecil, yaitu 1 (sisi utara)**
- **Apakah recent distance (2) < open distance terkecil (1) (tidak menyalahi aturan)**
- **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.

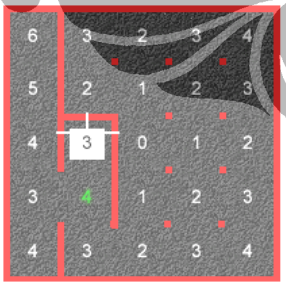


- **Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 2 + 1 = 3**
- **Karena menyalahi aturan, push koordinat open distance, yaitu (1,1), ke stack**
- **stackPointer = stackPointer + 1 = 0 + 1 = 1**

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.

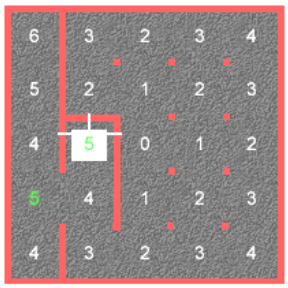


- **Ya, robot tetap di cell-nya dan meng-update recent distance = open distance terkecil + 1 = 3 + 1 = 4**
- **Karena menyalahi aturan, push koordinat open distance, yaitu (0,1), (1,0), dan (1,2), ke stack**
- **3 kali push maka stackPointer = 3**

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**

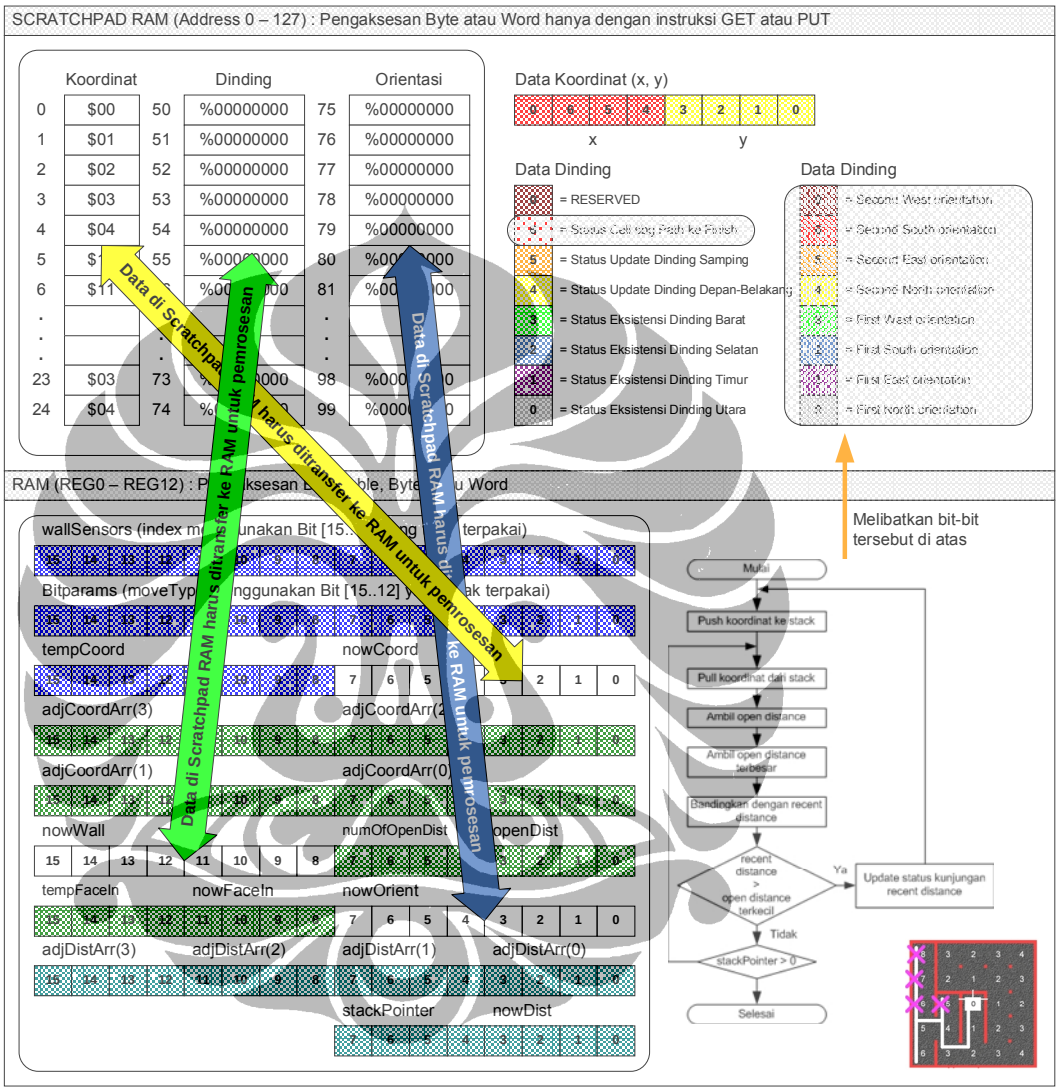
addr	Stack
75	$02
76	$11
77	$00

- **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.

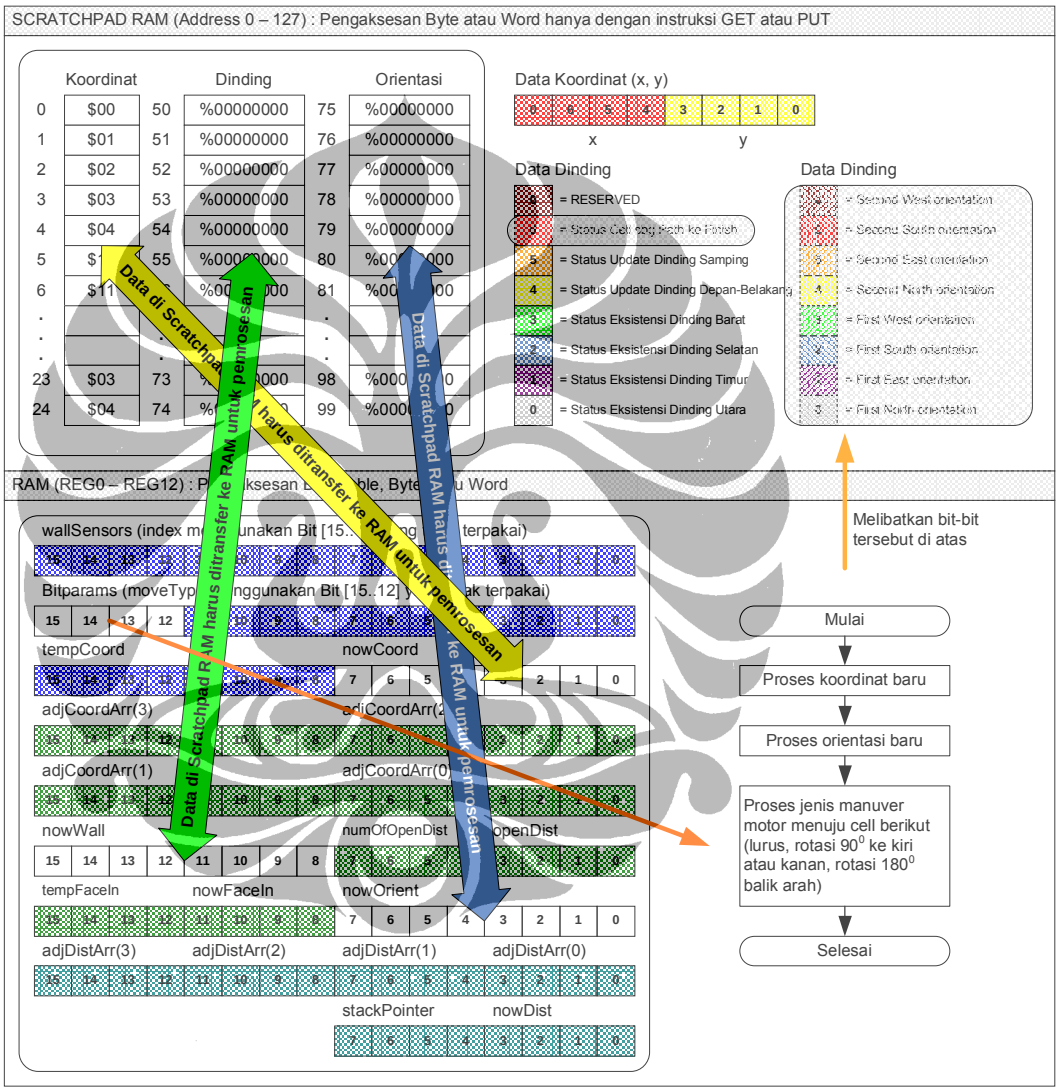


Gambar 4.13. Pemrosesan Status Kunjungan Cell Berbasis Stack

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.

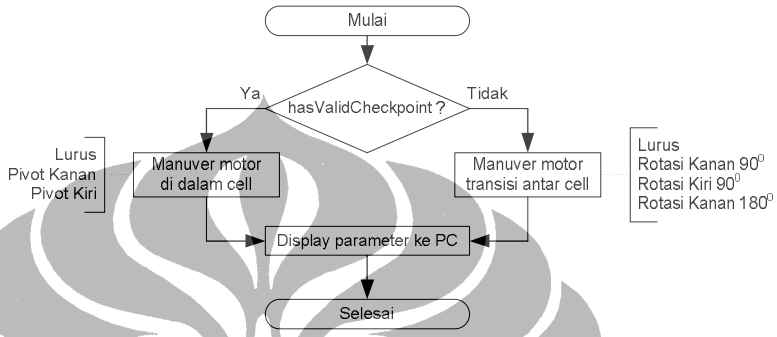


Gambar 4.14. Pemrosesan Tujuan

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

4.7 PEMROSESAN MANUVER MOTOR

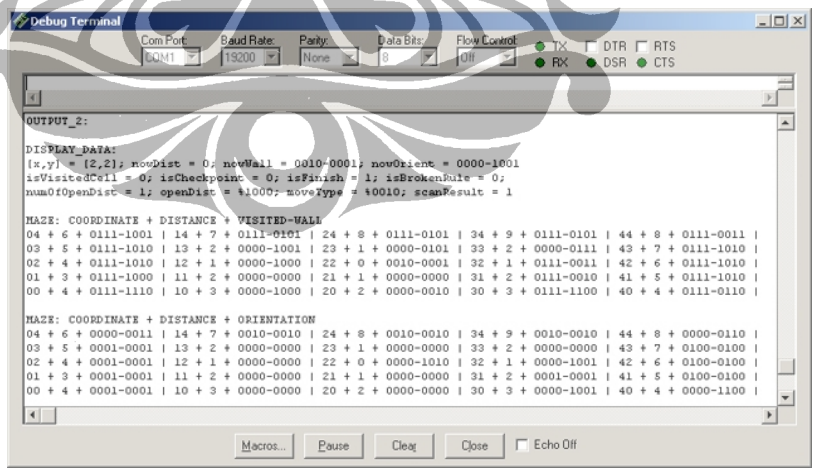
Bagian besar kelima sesuai dengan Gambar 4.2 adalah kode pemroses keluaran. Gambar 4.15 memperlihatkan bit kontrol *isCheckpoint* mengatur manuver robot, yaitu apakah melakukan manuver di dalam cell atau melakukan manuver transisi antar cell. Dua jenis manuver tersebut mempunyai jenis-jenis pergerakan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

BAB V
KESIMPULAN

1. Kecerdasan-buatan untuk robot pencari jalur telah berhasil dirancang dan diimplementasikan pada lingkungan terkontrol berupa labirin dua dimensi 5 x 5 (25 cell penyusun). Pergerakan yang dilakukan adalah mencari jalur dari cell awal di sudut labirin menuju ke cell tujuan di tengah labirin, di mana setelah sampai di tujuan, robot pencari jalur kembali menuju ke cell awal.
2. Kecerdasan-buatan pada robot pencari jalur yang diimplementasikan menggunakan bahasa pemrograman PBASIC dengan target mikrokontroler BASIC Stamp dirancang secara modular berbasis pemrograman multi slot (multi slot programming) dengan memperhitungkan faktor fleksibilitas dan skalabilitas untuk pengembangan fitur dan penanganan lingkungan labirin yang lebih kompleks
3. Implementasi kecerdasan-buatan memanfaatkan *resources* mikrokontroler, diantaranya EEPROM untuk menyimpan kode program dan data pemetaan sementara di Scratchpad RAM. Tidak tertutup kemungkinan untuk melakukan migrasi penyimpanan data pemetaan ke EEPROM untuk antisipasi *data loss* apabila terjadi *power loss*

DAFTAR ACUAN

[1] Wikipedia (2007). *Robot*. Diakses 31 Oktober 2007, dari Wikipedia Indonesia. http://id.wikipedia.org/wiki/Robot

[2] Parallax Inc., *"Parallax Line Follower Module,"* Parallax Product Document #29115, 2004.

[3] Pete Harrison (2007). *Micromouse Resources*. Diakses 18 April 2007, dari Micromouse Information Center. http://micromouse.cannock.ac.uk/index.htm

[4] Steve Benkovic (2007). *Hints, Ideas, Inspiration for Mice Builders*. Diakses 19 Juli 2007, dari MicroMouseInfo.com. http:// www.micromouseinfo.com/

[5] AINS Inc. (2005). *System Development Life Cycle (SDLC) Support*, http://www.ains-inc.com/sub/Services_SDLC.html.

[6] Wikipedia (2007). *Kecerdasan Buatan*. Diakses 31 Oktober 2007, dari Wikipedia Indonesia. http://id.wikipedia.org/wiki/Kecerdasan_buatan

[7] Endra Pitowarno (2007). *Serial Buku Robotik: Kecerdasan Buatan*. Diakses 29 Oktober 2007, dari Endro Pitowarno Homepage Online Portal. http://lecturer.eepis-its.edu/~epit

[8] Andy Lindsay, *Robotics with the Boe-Bot, Student Guide* (California: Parallax, Inc. Press, 2004)

[9] Wikipedia (2007). *Mikrokontroler*. Diakses 31 Oktober 2007, dari Wikipedia Indonesia. http://id.wikipedia.org/wiki/Mikrokontoler

[10] Andy Lindsay, *BASIC Stamp Syntax and Reference Manual, Version 2.2* (California: Parallax, Inc. Press, 2005)

[11] Andy Lindsay, *What's a Microcontroller - Student Guide, Version 2.2* (California: Parallax, Inc. Press, 2004)

[12] Claus Kühnel, Klaus Zahnert, *BASIC Stamp 2p - Commands, Features and Projects* (California: Parallax, Inc. Press, 2003)

[13] Parallax Inc., *"QTI Line Follower AppKit for the Boe-Bot,"* Parallax Product Document #28108, 2004.

[14] Fairchild Semiconductor, *"QRD1113/1114 Reflective Object Sensor,"* Fairchild Semiconductor Product *Datasheet*.

DAFTAR PUSTAKA

Altium Ltd., *DXP 2004 User Guide* (USA: Altium Ltd. Press, 2004)

Berg, Mike, etc., *"EE 382 - Junior Design: Final Report"* New Mexico Tech - EE Department, 2001.

EME Systems (2001). *BS2sx, BS2e, and BS2p application notes*. Diakses 19 Oktober 2007, dari BS2SX tech notes, http://www.emesystems.com/BS2index.htm/

Lindsay, Andy, *"IR Remote for the Boe-Bot, version 1.1"* (California: Parallax, Inc. Press, 2006)

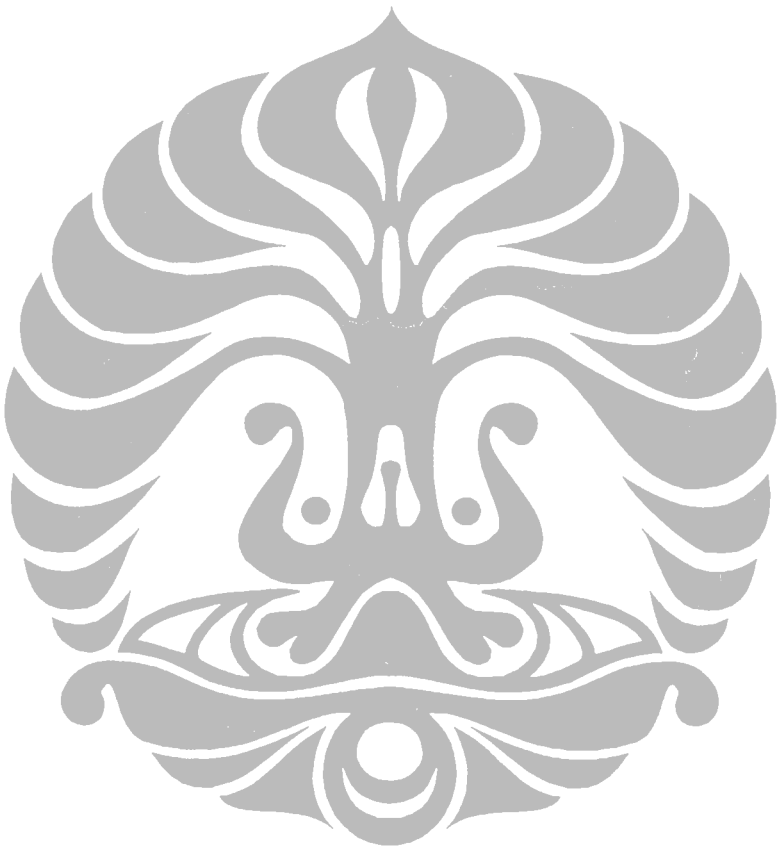
Lindsay, Andy, *"Applied Robotics with the SumoBot"* (California: Parallax, Inc. Press, 2006)

Pilgrin, Philip C., *"Applying the Boe-Bot Digital Encoder Kit,"* Parallax Product Document #28107, 2004.

Williams, Jon, *"Multi-Bank Programming," The* Nuts and Volts of BASIC Stamp, Column #87, 2004.

Williams, Jon, *"Ping … I See You,"* The Nuts and Volts of BASIC Stamp, 2005.

LAMPIRAN



Lampiran 1. Keluarga Mikrokontroler Basic Stamp

Released Products	Rev.Dx / BS1-IC	BS2-IC	BS2e-IC	BS2sx-IC
Package	PCB w/Proto / 14-pin SIP	24-pin DIP	24-pin DIP	24-pin DIP
Package Size (L x W x H)	2.5" x 1.5" x .5" / 1.4" x .6" x .1"	1.2" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"
Environment *	0° - 70° C (32° - 158° F) **	0° - 70° C (32° - 158° F) **	0° - 70° C (32° - 158° F)	0° - 70° C (32° - 158° F)
Microcontroller	Microchip PIC16C56	Microchip PIC16x57	Parallax SX28	Parallax SX28
Processor Speed	4 MHz	20 MHz	20 MHz	50 MHz
Program Execution Speed	~2,000 instructions/sec.	~4,000 instructions/sec.	~4,000 instructions/sec.	~10,000 instructions/sec.
RAM Size	16 Bytes (2 I/O, 14 Variable)	32 Bytes (6 I/O, 26 Variable)	32 Bytes (6 I/O, 26 Variable)	32 Bytes (6 I/O, 26 Variable)
Scratch Pad RAM	N/A	N/A	64 Bytes	64 Bytes
EEPROM (Program) Size	256 Bytes, ~80 instructions	2K Bytes, ~500 instructions	8 x 2K Bytes, ~4,000 inst.	8 x 2K Bytes, ~4,000 inst.
Number of I/O pins	8	16 + 2 Dedicated Serial	16 + 2 Dedicated Serial	16 + 2 Dedicated Serial
Voltage Requirements	5 - 15 vdc	5 - 15 vdc	5 - 12 vdc	5 - 12 vdc
Current Draw @ 5V	1 mA Run / 25 µA Sleep	3 mA Run / 50 µA Sleep	25 mA Run / 200 µA Sleep	60 mA Run / 500 µA Sleep
Source / Sink Current per I/O	20 mA / 25 mA	20 mA / 25 mA	30 mA / 30 mA	30 mA / 30 mA
Source / Sink Current per unit	40 mA / 50 mA	40 mA / 50 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins
PBASIC Commands***	32	42	45	45
PC Programming Interface	Serial (w/BS1 Serial Adapter)	Serial (9600 baud)	Serial (9600 baud)	Serial (9600 baud)
Windows Text Editor	Stampw.exe (v2.1 and up)	Stampw.exe (v1.04 and up)	Stampw.exe (v1.096 and up)	Stampw.exe (v1.091 and up)

Released Products	BS2p24-IC	BS2p40-IC	BS2pe-IC	BS2px-IC
Package	24-pin DIP	40-pin DIP	24-pin DIP	24-pin DIP
Package Size (L x W x H)	1.2" x 0.6" x 0.4"	2.1" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"	1.2" x 0.6" x 0.4"
Environment *	0° - 70° C (32° - 158° F)	0° - 70° C (32° - 158° F)	0° - 70° C (32° - 158° F)	0° - 70° C (32° - 158° F)
Microcontroller	Parallax SX48	Parallax SX48	Parallax SX48	Parallax SX48
Processor Speed	20 MHz Turbo	20 MHz Turbo	8 MHz Turbo	32 MHz Turbo
Program Execution Speed	~12,000 instructions/sec.	~12,000 instructions/sec.	~6000/sec.	~19,000 instructions/sec.
RAM Size	38 Bytes (12 I/O, 26 Variable)	38 Bytes (12 I/O, 26 Variable)	38 Bytes (12 I/O, 26 Variable)	38 Bytes (12 I/O, 26 Variable)
Scratch Pad RAM	128 Bytes	128 Bytes	128 Bytes	128 Bytes
EEPROM (Program) Size	8 x 2K Bytes, ~4,000 inst.	8 x 2K Bytes, ~4,000 inst.	16 x 2K Bytes (16 K for source)	8 x 2K Bytes, ~4,000 inst.
Number of I/O pins	16 + 2 Dedicated Serial	32 + 2 Dedicated Serial	16 + 2 Dedicated Serial	16 + 2 Dedicated Serial
Voltage Requirements	5 - 12 vdc	5 - 12 vdc	5 - 12 vdc	5 - 12 vdc
Current Draw @ 5V	40 mA Run / 350 µA Sleep	40 mA Run / 350 µA Sleep	15 mA Run / 36 µA Sleep	55 mA Run / 450 µA Sleep
Source / Sink Current per I/O	30 mA / 30 mA	30 mA / 30 mA	30 mA / 30 mA	30 mA / 30 mA
Source / Sink Current per unit	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins	60 mA / 60 mA per 8 I/O pins
PBASIC Commands***	61	61	61	63
PC Programming Interface	Serial (9600 baud)	Serial (9600 baud)	Serial (9600 baud)	Serial (19200 baud)
Windows Text Editor	Stampw.exe (v1.1 and up)	Stampw.exe (v1.1 and up)	Stampw.exe (v1.33 and up)	Stampw.exe (v2.2 and up)

* 70% Non-Condensing Humidity
** Industrial Models Available, -40° - 85° C (-40° - 185° F).
*** Using PBASIC 2.5 for BS2-type models.

Sumber : Parallax, Inc., USA

Lampiran 2. Foto-Foto Robot Pencari Jalur

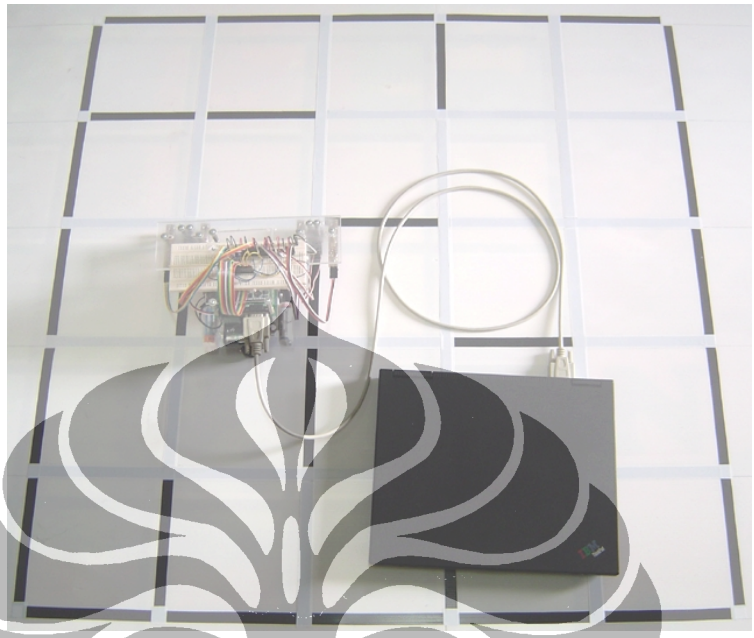


Foto 1 Development Environment:
Download Kecerdasan Buatan pada Robot Pencari Jalur Menggunakan Komunikasi Serial

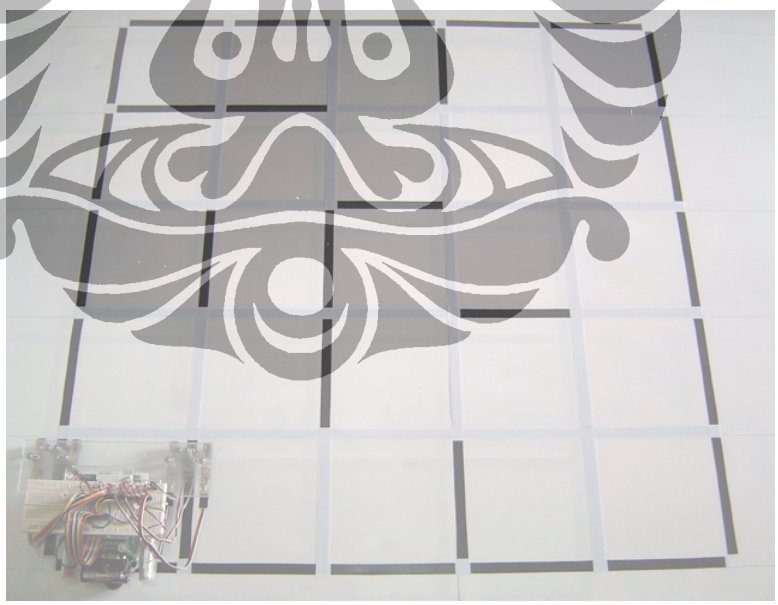


Foto 2. *Cell* Awal Robot Pencari Jalur di Lingkungan Labirin Dua Dimensi

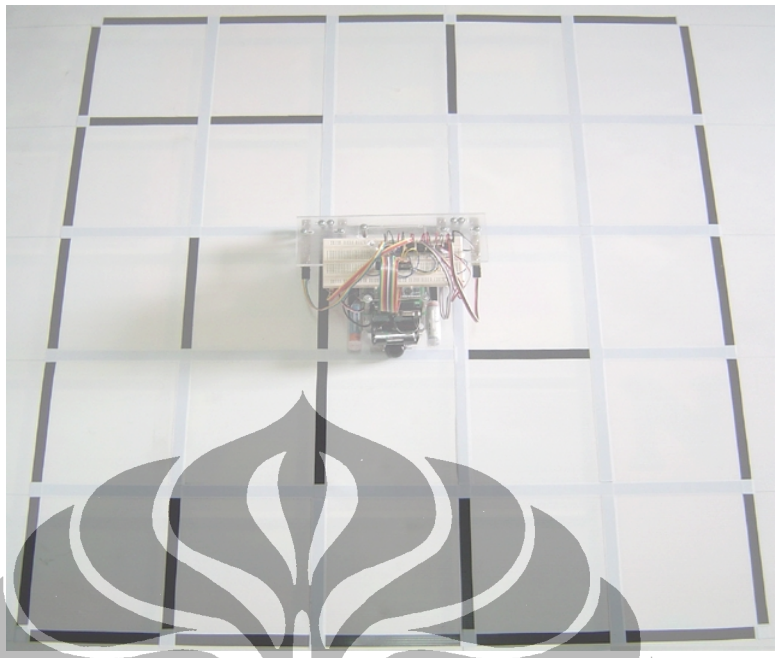
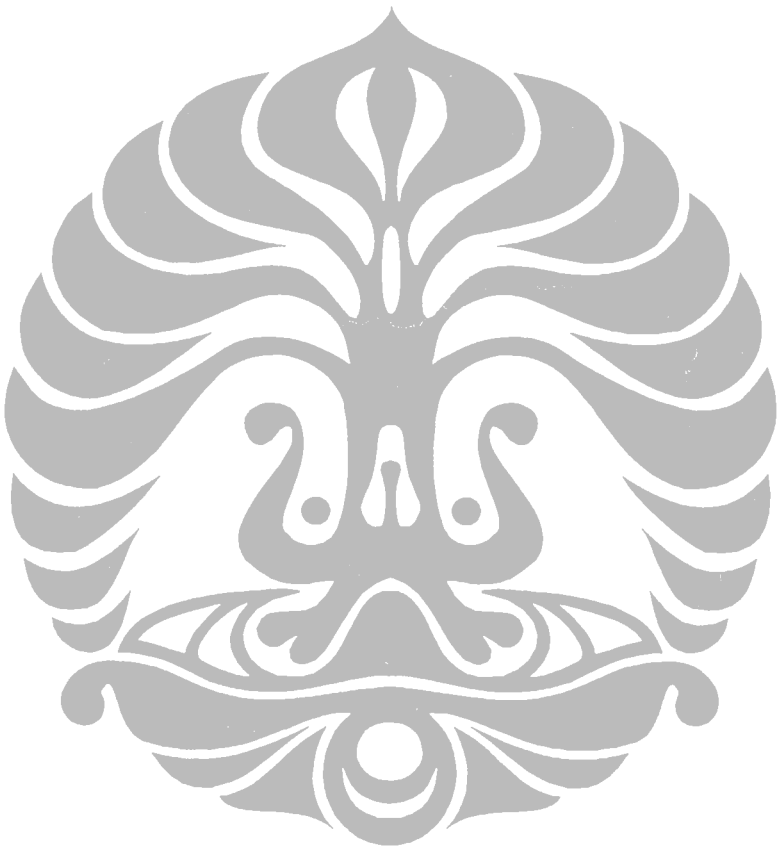


Foto 3. *Cell* Tujuan Robot Pencari Jalur di Lingkungan Labirin Dua Dimensi

Lampiran 2.

Paper pada "The 10th Quality In Research (QIR) International Conference"

University of Indonesia, December 4-6, 2007



Design and Implementation of Artificial Intelligence of Path Searching Robot Based on Microcontroller BASIC Stamp

Harry Sudibyo Soetjokro*, and Gede Indrawan†

* Fac. of Engineering, Universitas Indonesia (UI), Depok, 16424
Tel. 62-021-7270077, fax. 62-021-7270077, email: harisudi@ee.ui.ac.id
†Fac. of Engineering, Universitas Pendidikan Ganesha (Undiksha), Singaraja - Bali
Tel. 62-0362-22570, fax. 62-0362-25735, email: gede.indrawan@gmail.com

Abstract– **Artificial intelligence in robotics, as a smart algorithm programmed into the robot, is needed by the robot to help human to do some work automatically and in autonomous way. In this research, the implanted artificial intelligence is designed for path searching, included in mapping activity, from starting point at corner to destination point at center, in controlled environment like maze. General speaking, this research want to contribute in knowledge development under robotics domain of autonomous position-sensing and navigation.**

Robot design for this artificial intelligence consists of two aspects, i.e. robot prototype itself, and maze environment where path-searching robot will run. Implementation of robot involves three aspects, i.e. input (sensor to capture information from the environment), process (processor and its supporting system as robot brain for data processing), and output (as result of data processing, it can be signal for controlling the motor, etc).

The artificial intelligence in this research is implemented using PBASIC programming language, with BASIC Stamp BS2px from Parallax as targeted microcontroller. Multi bank programming style is used to utilize 16 KB internal EEPROM resource, comprise of eight memory bank with 2 KB capacity respectively, to save program code. This code supports robot function for path searching in maze, by using Flood-Fill algorithm and modified Flood-Fill algorithm as main algorithms. Flexibility and scalability are two concepts of this artificial intelligence to accommodate features addition and to anticipate more complex maze environment.

Keywords– Artificial Intelligence, robot, autonomous position-sensing and navigation, microcontroller, PBASIC, Basic Stamp, multi bank programming, Flood-Fill algorithm¸modified Flood-Fill algorithm

I. INTRODUCTION

Robot, to associate behaviors with a place (localization) requires to know where it is and to be able to navigate point-to-point. Such navigation began with wire-guidance in the 1970s and progressed in the early 2000s to beacon-based triangulation.

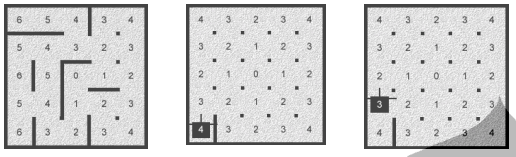
For indoor application, current commercial robots autonomously navigate based on sensing natural features. The first commercial robots to achieve this were Pyxus' HelpMate hospital robot and the CyberMotion guard robot, both designed in the 1980s. These robots originally used manually created CAD floor plans, sonar sensing and wall-following variations to navigate buildings. The next generation, such as MobileRobots' PatrolBot and autonomous wheelchair both introduced in 2004, have the ability to create their own laser-based maps of a building and to navigate open areas as well as corridors. Their control system changes its path on-the-fly if something blocks the way [1].

At the other side, outdoor autonomy is most easily achieved in the air, since obstacles are rare. Cruise missiles are rather dangerous highly autonomous robots. Some of unmanned aerial vehicles (UAVs) are capable of flying their entire mission without any human interaction at all except possibly for the landing where a person intervenes using radio remote control. But some drone aircraft are capable of a safe, automatic landing also.

Outdoor autonomy is the most difficult for ground vehicles, due to: a) 3-dimensional terrain, b) great disparities in surface density, c) weather exigencies, and d) instability of the sensed environment. The Mars rovers MER-A and MER-B can find the position of the sun and navigate their own routes to destinations on-the-fly by: a) mapping the surface with 3-D vision, b) computing safe and unsafe areas on the surface within that field of vision, c) computing optimal paths across the safe area towards the desired destination, d) driving along the calculated route, e) repeating this cycle until either the destination is reached, or there is no known path to the destination

II. BASIC THEORY

The objective of this research is to create artificial intelligence for path searching robot from starting cell at corner to destination cell at center of the maze without breaking 2-dimension walls, represented by black line that exist in different side on each cell (Fig.1a). Fig.1b illustrates how at starting cell, robot has no information about walls except at the current cell where robot is standing on and detect wall using its sensors.



(a) (b) (c)

Fig.1. (a) Maze with wall (b) Maze view by robot at start (c) Robot move to other cell

To solve this path searching problem, Flood-Fill algorithm and modified Flood-Fill algorithm are used as main algorithms for this artificial intelligence.

The Flood-Fill algorithm involves assigning values to each of the cells in the maze where these values represent the distance from any cell on the maze to the destination cell without breaking the wall. The destination cell, therefore, is assigned a value of 0. If the path searching robot is standing in a cell with a value of 1, it is 1 cell away from the goal. If the robot is standing in a cell with a value of 3, it is 3 cells away from the goal. Fig.1b represents initial value for every cell known by robot at start with initial assumption there is no wall at all.

For the maze at Fig.1, we would have 5 rows by 5 columns = 25 cell values. Therefore we would need 25 bytes to store the distance values for a complete maze. Actually, for this research, we have designed other data needed by a cell, as shown by Table 1 below.

Table 1. Cell data design (put in Scratchpad RAM)

CoordinateValue (Address: 0 - 24)	
7 6 5 4 3 2 1 0	x-axis = nibble1 (bit 7 … 4) y-axis = nibble0 (bit 3 … 0)
DistanceValue (Address: 25 - 49)	
7 6 5 4 3 2 1 0	Minimum = 0 (00000000b) Maximum = 255 (11111111b)
WallValue (Address: 50 - 74)	
7 6 5 4 3 2 1 0	Bit 6 = Active path to finish Bit 5 = Side wall check status Bit 4 = Checkpoint status Bit 3 = West (W) wall Bit 2 = South (S) wall Bit 1 = East (E) wall Bit 0 = North (N) wall
OrientationValue (Address: 75 - 99)	
7 6 5 4 3 2 1 0	nibble1 for second W, S, E, N nibble0 for first W, S, E, N

When it comes time to make a move, the robot must examine all adjacent cells which are not separated by walls and choose the one with the lowest distance value. In Fig.1c above, the robot would ignore any cell to the West (left side) because there is a wall, and it would look at the distance values of the cells to the North, East, and South since those are not separated by walls. The cell to the North has a value of 2, the cell to the East has a value of 2 and the cell to the South has a value of 4. The routine sorts the values to determine which cell has the lowest distance value. It turns out that both the North and East cells have a distance value of 2. That means that the robot can go North or East and traverse the same number of cells on its way to the destination cell. Since turning would take time, the robot will choose to go forward to the North cell. Fig.2a gives generic algorithm for that case.

Furthermore, every interior wall is shared by two cells so when robot update the wall value for one cell, robot can update the wall value for its neighbor as well (Fig.2b).

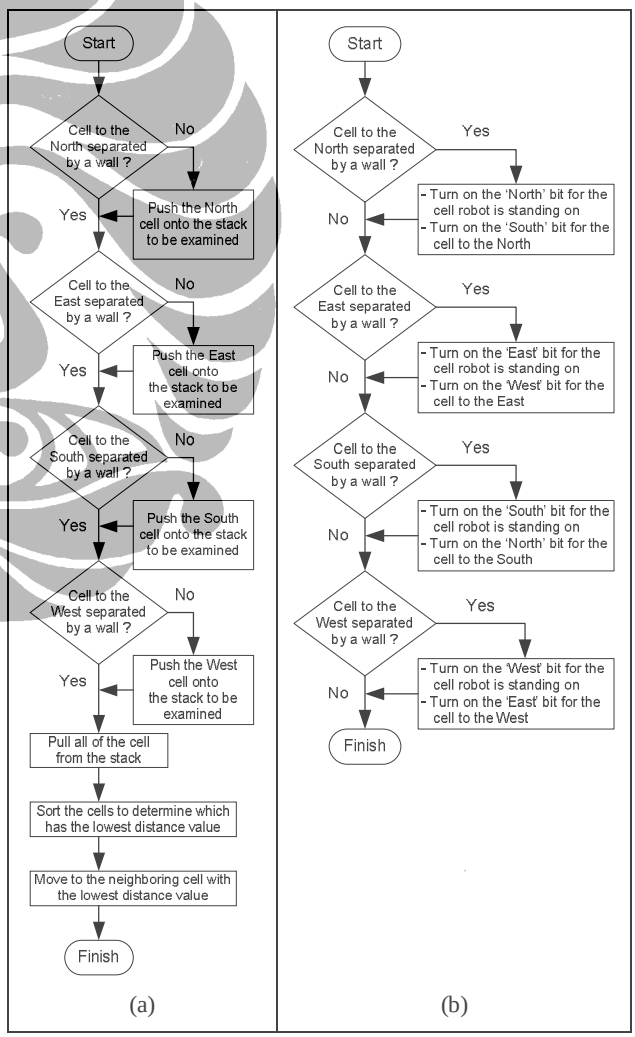


Fig.2. (a) Deciding lowest distance value (b) Updating wall map

The instructions for flooding the maze with distance values could be like algorithm at Fig.3 below.

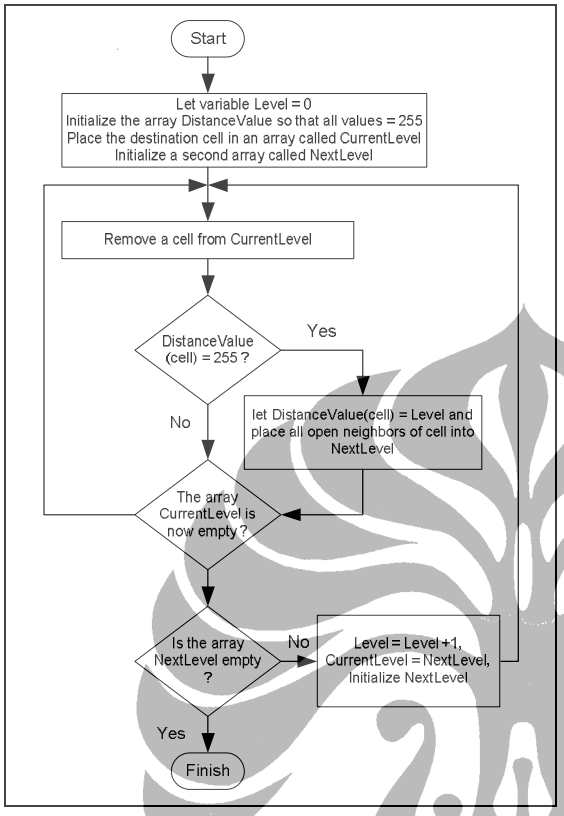


Fig.3. Flooding with distance values for Flood-Fill algorithm

The modified Flood-Fill algorithm at the other side is similar to the regular Flood-Fill algorithm in that the robot uses distance values to move about the maze. The distance values which represent how far the robot is from the destination cell, are followed in descending order until the robot reaches its goal. As the robot finds new walls during its exploration, the distance values need to be updated. Instead of flooding the entire maze with values, as is the case with the regular Flood-Fill, the modified Flood-Fill only changes those values which need to be changed.

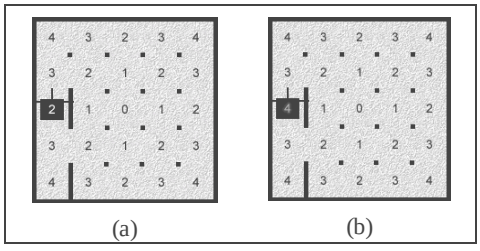


Fig.4. (a) Old distance value (b) New distance value

Fig.4 shows how our robot moves forward one cell and discovers a wall. The robot cannot go West and it cannot go East, it can only travel North or South. But going North or South means going up in distance values. So the cell values need to be updated. When the robot encounters this, it follows this rule: *"If a cell is not the destination cell, its value should be one plus the minimum value of its open neighbors"*. In the case above, the minimum value of its open neighbors is 3. Adding 1 to this value, results in 3 + 1 = 4. The maze now looks like Fig.4b.

There are times when updating a cell's value will cause its neighbors to violate the *"1 + minimum value"* rule and so they must be checked as well. We can see in our example above that the cells to the North and to the South have neighbors whose minimum value is 2. Adding a 1 to this value results in 2 + 1 = 3 therefore the cells to the North and to the South do not violate the rule and the updating routine is done. Now that the cell values have been updated, the robot can once again follow the distance values in descending order.

So our modified Flood-Fill procedure for updating the distance values is looked like Fig.5 below.

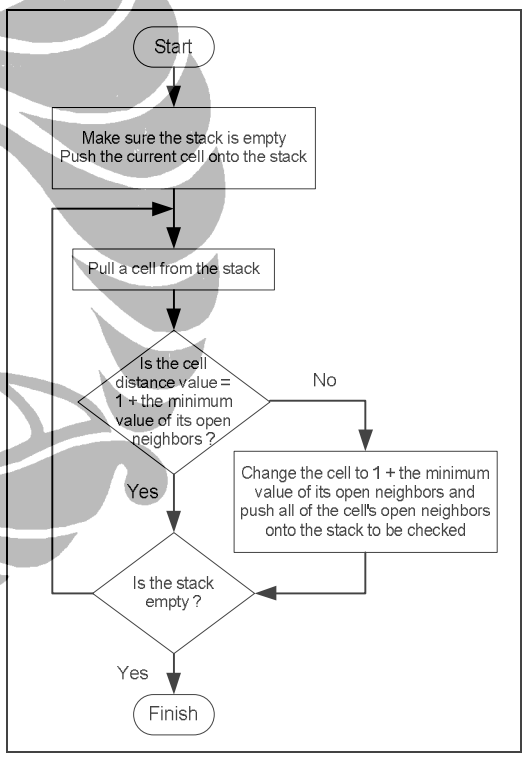


Fig.5. Flooding with distance values (if necessary) for modified Flood-Fill algorithm

As summary for both of algorithm, every time robot arrives in a cell, the Flood-Fill algorithm will perform the following steps [2], i.e.: 1) Update the wall map, 2) Flood the maze with new distance values (if necessary for modified Flood-Fill algorithm), 3) Decide which neighboring cell has the lowest distance value, 4) Move to the neighboring cell with the lowest distance value.

III. EXPERIMENTAL RESULTS

In this research, experiment is conducted to confirm the functionality of our artificial intelligence for path searching robot from starting cell to destination cell in the maze.

The artificial intelligence itself will control the system with simple diagram block (Fig.6a) that always consist of input, process, and output section.

Input section receives wall information from the maze using infra red reflective object sensor Fairchild Semiconductor QRD1114.

Output section for robot maneuver -- in order to avoid breaking the wall, is handled by continuous rotation servo that received signal from Process section to control motor speed and direction.

Process section is handled by microcontroller BASIC Stamp BS2px and its supporting system. Here, the artificial intelligence is implanted with two main algorithms, as shown by Fig.6a. Information of maze map size, with initial distance value for every cell, also initialized at this section. Of course, at initialization, robot does not know about wall map information.

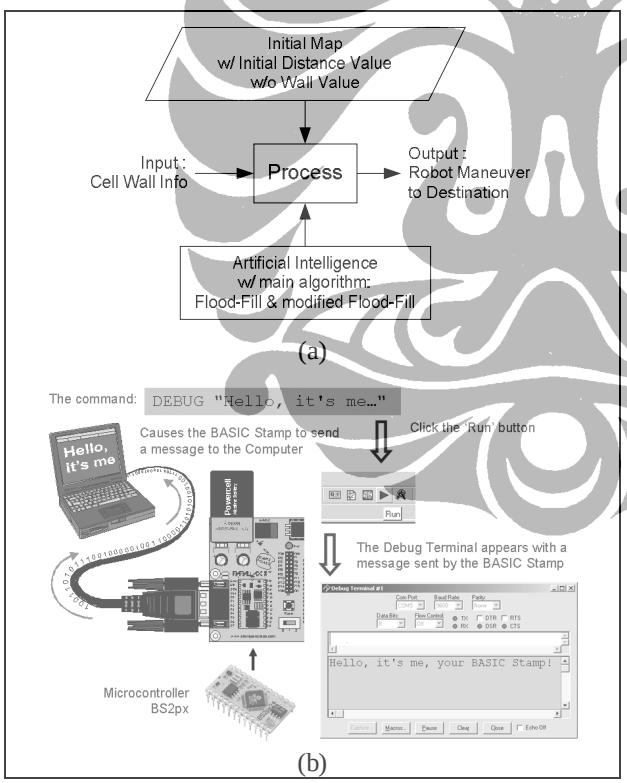


Fig.6. (a) System diagram block (b) Implementation of block process in BS2px development environment [3]

Fig.6b illustrates how the artificial intelligence was developed and implanted to the microcontroller BS2px with specification provided by Table 2.

Table 2. BS2px Specification [4]

Package	24-pin DIP
Package Size (L x W x H)	1.2" x 0.6" x 0.4"
Environment*	0º - 70º C (32º - 158º F)
Microcontroller	Parallax SX48
Processor Speed	32 MHz Turbo
Program Execution Speed	~19,000 instructions/sec.
RAM Size	38 Bytes (12 I/O, 26 Variable)
Scratch Pad RAM	128 Bytes
EEPROM (Program) Size	8 x 2K Bytes, ~4,000 inst.
Number of I/O pins	16 + 2 Dedicated Serial
Voltage Requirements	5 - 12 vdc
Current Draw @ 5V	55 mA Run / 450 µA Sleep
Source / Sink Current per I/O	30 mA / 30 mA
Source / Sink Current per unit	60 mA / 60 mA per 8 I/O pins
PBASIC Commands***	63
PC Programming Interface	Serial (19200 baud)
Windows Text Editor	Stampw.exe (v2.2 and up)

* 70% Non-Condensing Humidity
*** Using PBASIC 2.5 for BS2-type models

Experiment is conducted using some maneuvers as illustrated by Fig.7 and internal data processing is monitored via PC using serial communication, as shown by Fig.8a. Monitoring system itself involves debugging code that was embedded to the artificial intelligence, so it will make life easier to detect and repair any existing bug. Fig.10 shows the monitoring system in detail.

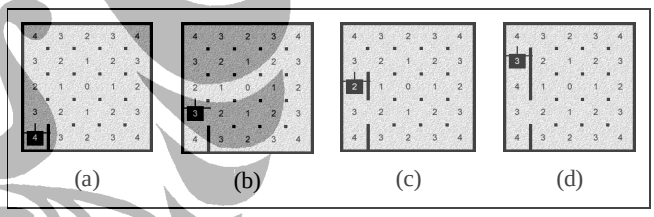


Fig.7. Prototype maneuver

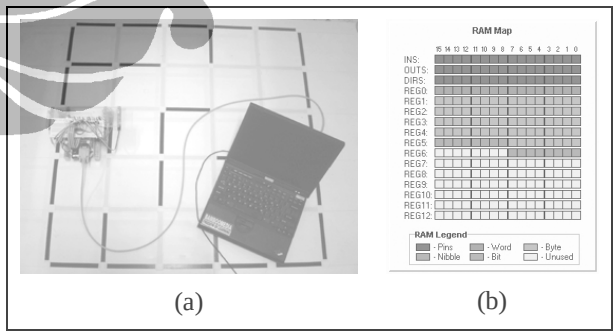
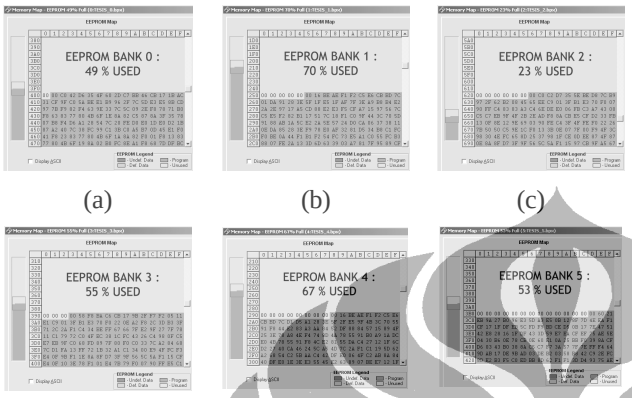


Fig.8. (a) Prototype testing (b) Usage of RAM

Microcontroller BS2px has three types of internal memory resource, i.e. 1) EEPROM for storing program code, 2) RAM for assigned and temporary variables, and 3) Scratchpad RAM for storing cell's data (see Table 1). Fig.8b shows internal RAM resource used by assigned variables that construct this artificial intelligence. RAM itself has total capacity of 13 Words (REG0 - REG12).

Using multi bank programming style, program code for artificial intelligence is stored from EEPROM bank 0 to bank 5 with its own usage percentage shown by Fig.9. Internal EEPROM itself comprised of eight memory banks (bank 0 to bank 7) with capacity 2 KB in each, together form total capacity of 16 KB.



(a) (b) (c)

(d) (e) (f)

Fig.9. Usage of EEPROM Bank

Fig.10 shows the monitoring process for some maneuvers of path searching robot, as illustrated by Fig.7. At robot position like Fig.7c, monitoring result is shown by Fig.10a. At robot position like Fig.7d, monitoring result is shown by Fig.10b.

What monitoring process want to tell us is that modified Flood-Fill algorithm has already work. As summary for that algorithm, every time robot arrives in a cell, the Flood-Fill algorithm will perform the following steps:

1. Update the wall map
Fig.10a gives Wall value = 00011010b at cell with Coordinate value = (0, 2). From Table 1, it means that robot facing North because bit (7-6) = 00b. The cell has already been checked for side wall and front-rear wall (has reach checkpoint) because bit (4) = 1 and bit (5) = 1. Based on that, it is no need again for gathering wall info that has already got, i.e. there are walls at West and East side because bit (3) = 1 and bit (1) = 1.

2. Flood the maze with new distance values (if necessary)
The cell at (0, 2) need to update its distance value because its recent distance value = 2, violates rule: *"If a cell is not the destination cell, its value should be one plus the minimum value of its open neighbors"*. Artificial intelligence updates this value. Fig.10b show updated distance value = 4.

3. Decide which neighboring cell has the lowest distance
Fig.10a shows there are two open neighboring cells with lowest distance = 3, i.e. cell at (0,1) and (0,3). The artificial intelligence choose cell (0,3) to move on because robot just go forward rather than turning that take more time.

4. Move to the neighboring cell with the lowest distance
Fig.10b shows robot move from cell at (0,2) to cell at (0,3). It proves this step.

Fig.10. Monitoring for prototype maneuver

IV. CONCLUSIONS

Path searching from starting cell to destination cell has been accomplished by the artificial intelligence using Flood-Fill algorithm and modified Flood-Fill algorithm.

REFERENCES

[1] Wikipedia, "Autonomous robot", 2007, http://id.wikipedia.org/wiki/Robot

[2] Steve Benkovic, "*Hints, Ideas, Inspiration for Mice Builders*", 2007, http:// micromouseinfo.com/.

[3] Andy Lindsay, "*Robotics with the Boe-Bot, Student Guide*", Parallax, Inc. Press, California, 2004.

[4] Parallax Inc., "*Stamp Specifications*", 2007, http://parallax.com/Portals/0/Downloads/docs/prod/stamps/stampscomparison.pdf.

Lampiran 4.

Kode Program Kecerdasan-Buatan Robot Pencari Jalur

Header (Ada di setiap slot EEPROM yang digunakan)

Kode Program Bank 0 EEPROM

Kode Program Bank 1 EEPROM

Kode Program Bank 2 EEPROM

Kode Program Bank 3 EEPROM

Kode Program Bank 4 EEPROM

Kode Program Bank 5 EEPROM

Kode Program Bank 6 EEPROM

Header (Ada di setiap slot EEPROM yang digunakan)

```
' -----[ Title ]-----------------------------------------------------------------
'
' File..... TESIS.BPX
' Purpose... Micromouse Style Path Searching Robot
' Author.... Gede Indrawan
' E-mail.... gede.indrawan@ui.edu

' {$STAMP BS2px, TESIS_1.bpx, TESIS_2.bpx, TESIS_3.bpx, TESIS_4.bpx, TESIS_5.bpx}
' {$PBASIC 2.5}


' -----[ Program Description ]---------------------------------------------------
'
' This program is used to search the path at maze environment.

' -----[ Revision History ]------------------------------------------------------
'
' DEC. 25TH, 2007 - Version 1.0

' -----[ Conditional Compilation Directive ]--------------------------------------
'
'#DEFINE DebugLow     = 1  'ADD ", TESIS_6.bpx" AT $STAMP DIRECTIVE AT SLOT 0
'#DEFINE DebugNormal  = 1
#DEFINE DebugLow     = 0
#DEFINE DebugNormal  = 0

' -----[ I/O Definitions ]--------------------------------------------------------
'
' PROXIMITY SENSOR ARRAY
' -------- FRONT --------
' 07 06 05 04 03 02 01 00
' L7 L6 L5 L4 R3 R2 R1 R0
' -------- REAR! --------

OuterRightSensor      PIN  0   ' PIN 0-11: ON/OFF SENSOR
FarMiddleRightSensor  PIN  1
NearMiddleRightSensor PIN  2
InnerRightSensor      PIN  3

InnerLeftSensor       PIN  4
NearMiddleLeftSensor  PIN  5
FarMiddleLeftSensor   PIN  6
OuterLeftSensor       PIN  7

OuterRearRightSensor  PIN  8
InnerRearRightSensor  PIN  9

InnerRearLefttSensor  PIN  10
OuterRearLeftSensor   PIN  11

RightMotor            PIN  12  ' Servo motor connections
LeftMotor             PIN  13

SensorInput           PIN  14  ' INPUT FOR ALL SENSORS

UnusedPin             PIN  15  ' Unused I/O pin

' -----[ Constants ]-------------------------------------------------------------
'
Pause4BigLoop         CON  100

EqualTo15             CON  15   ' %1111
EqualTo255            CON  255  ' %11111111

IsLow                 CON  0
IsHigh                CON  1

MatrixSize            CON  5  ' Maximum for Matrix Size = 16
MatrixSizeSq          CON  MatrixSize*MatrixSize
MatrixSizeMinSatu     CON  MatrixSize-1
MatrixSizeSqMinSatu   CON  MatrixSizeSq-1

' DATA DESIGN AT SCRATCHPAD RAM
' HIGHEST LOCATION AT 127 IS READ ONLY (CONTAIN NUMBER OF RUNNING SLOT)
CoordinateValue       CON  MatrixSizeSq*0  ' SPRAM Address =   0 -  24
DistanceValue         CON  MatrixSizeSq*1  ' SPRAM Address =  25 -  49
WallValue             CON  MatrixSizeSq*2  ' SPRAM Address =  50 -  74
OrientValue           CON  MatrixSizeSq*3  ' SPRAM Address =  75 -  99
StackValue            CON  MatrixSizeSq*4  ' SPRAM Address = 100 - 115
StackMaxAddress       CON  115
SensorThres           CON  125  ' Stores black/white threshold = 125 - 126
```

```
GoStraight          CON  0
RotateRight90        CON  1
RotateLeft90        CON  2
RotateRight180       CON  3

GoStraightFactor     CON  30
Loop4GoStraight      CON  60
Pause4GoStraight     CON  20

RotateFactor        CON  30
Loop4Rotate90        CON  90  ' DEBUG: ON = 100, OFF = 90
Pause4Rotate90       CON  20
Loop4Rotate180       CON  180  ' DEBUG: ON = 200, OFF = 180
Pause4Rotate180      CON  20

Loop4PivotLeftScan    CON  70
Loop4PivotRightScan   CON  2*Loop4PivotLeftScan
Pause4Scan           CON  20

GoForwardFactor      CON  30
Loop4GoForward       CON  4
Pause4GoForward      CON  20

PivotFactor          CON  30
Loop4Pivot           CON  2
Pause4Pivot          CON  20
StopMotor            CON  1850  ' stop motor for BS2px (750 for BS2)

TskDoInput           CON  1  ' Program Task at Memory Bank 0

Bank_0               CON  0  ' Memory Bank 0
Bank_1               CON  1  ' Memory Bank 1
Bank_2               CON  2  ' Memory Bank 2
Bank_3               CON  3  ' Memory Bank 3
Bank_4               CON  4  ' Memory Bank 4
Bank_5               CON  5  ' Memory Bank 5
Bank_6               CON  6  ' Memory Bank 6
Bank_7               CON  7  ' Memory Bank 7

North               CON  0
East                CON  1
South               CON  2
West                CON  3

'CodeOverhead          CON  587  ' BS2   -> 220 * 2us = 440us
                       ' BS2px -> 440us / 0.75us = 587
OneOpenNeighbour      CON  1
TwoOpenNeighbours     CON  2
ThreeOpenNeighbours   CON  3

' -----[ Variables ]----------------------------------------------------------
'
wallSensors          VAR  Word
sensor0              VAR  wallSensors.BIT0
sensor1              VAR  wallSensors.BIT1
sensor2              VAR  wallSensors.BIT2
sensor3              VAR  wallSensors.BIT3
sensor4              VAR  wallSensors.BIT4
sensor5              VAR  wallSensors.BIT5
sensor6              VAR  wallSensors.BIT6
sensor7              VAR  wallSensors.BIT7
sensor8              VAR  wallSensors.BIT8
sensor9              VAR  wallSensors.BIT9
sensor10             VAR  wallSensors.BIT10
sensor11             VAR  wallSensors.BIT11
index               VAR  wallSensors.NIB3

'sensorThreshold       VAR  Word  ' Stores black/white threshold

bitparams            VAR  Word
isBrokenRule          VAR  bitparams.BIT0
isCheckpoint          VAR  bitparams.BIT1
isFinish             VAR  bitparams.BIT2
isNoWall             VAR  bitparams.BIT3
isSideWall           VAR  bitparams.BIT4
isSetPath            VAR  bitparams.BIT5
isVisitedCell        VAR  bitparams.BIT6
scanResult           VAR  bitparams.BIT7
task                VAR  bitparams.BIT8
moveType             VAR  bitparams.NIB3

coordinate           VAR  Word
tempCoord            VAR  coordinate.BYTE1
nowCoord             VAR  coordinate.BYTE0
```

```
tempX           VAR  tempCoord.NIB1
tempY           VAR  tempCoord.NIB0
nowX            VAR  nowCoord.NIB1
nowY            VAR  nowCoord.NIB0

adjCoordArr      VAR  Byte(4)

openDistance     VAR  Byte
numOfOpenDist      VAR  openDistance.NIB1
openDist         VAR  openDistance.NIB0

nowWall          VAR  Byte

nowOrient        VAR  Byte

faceIn           VAR  Byte
tempFaceIn         VAR  faceIn.NIB1
nowFaceIn          VAR  faceIn.NIB0

adjDistArr       VAR  Nib(4)

nowDist          VAR  Nib

stackPointer     VAR  Nib

' -----[ EEPROM Data ]----------------------------------------------------------
'
' WallValue
' Bit : 7 6       -> FACING TO
'       0 0       -> NORTH
'       0 1       -> EAST
'       1 0       -> SOUTH
'       1 1       -> WEST
' Bit : 5 4       -> WALL STATUS
'       0 0       -> NO WALL HAVE BEEN CHECKED
'       0 1       -> WEST & EAST WALLS HAVE BEEN CHECKED
'       1 0       -> NORTH & SOUTH WALLS HAVE BEEN CHECKED
'       1 1       -> WEST, EAST, NORTH, SOUTH WALLS HAVE BEEN CHECKED
' Bit : 3 2 1 0    -> WALL EXISTENCE FLAG
'       W S E N     (W = WEST, S = SOUTH, E = EAST, N = NORTH)
'
' DistanceValue    -> NUMBER OF CELL TO BE TRAVELLED FROM CURRENT TO TARGET CELL
'
' CoordinateValue  -> COORDINATE OF CURRENT CELL
' Nibble1 : 7 6 5 4 -> X-AXIS
' Nibble0 : 3 2 1 0 -> Y-AXIS

'               DATA     @CoordinateValue, (MatrixSize*MatrixSize)
'               DATA     @DistanceValue,   (MatrixSize*MatrixSize)
'               DATA     @WallValue,       (MatrixSize*MatrixSize)
'ResetValue     DATA     $FF
```

Kode Program Bank 0 EEPROM

```
' -----[ Initialization ]----------------------------------------------------
'
'Check_Run:
'  READ ResetValue, B25                  ' get reset value
'  B25 = ~B25                           ' invert bits
'  WRITE ResetValue, B25                 ' write inverted bits back
'  IF (B25) THEN Initialize              ' run if inverted > 0

'No_Run:
'  END                                  ' low power mode

' -----[ Main Code ]----------------------------------------------------------
'
  'PAUSE Pause4BigLoop  ' pause for big looping of system

  BRANCH task, [Do_Init, Do_Input]

' ---------------------------------------
Do_Init:

  #IF DebugLow #THEN
    DEBUG CR, "INIT:", CR
  #ENDIF

  GOSUB Init_Map
  GOSUB Init_Stack
  GOSUB Calibrate_Sensors
  GOSUB Init_Start

' ---------------------------------------
Do_Input:

  #IF DebugLow #THEN
    DEBUG CR, "INPUT:", CR
  #ENDIF

  GOSUB Prepare_Cell_Paramaters
  GOSUB Read_Wall_Sensors

' ---------------------------------------
Do_Process_1:
  RUN Bank_1

' -----[ Subroutines ]--------------------------------------------------------
'
' ---------------------------------------
Init_Map:

  #IF DebugLow #THEN
    DEBUG CR, "INIT_MAP:", CR
  #ENDIF

  B24 = MatrixSize
  B23 = 0  ' PLAY SAVE, MAKE IT 0 BEFORE

  ' GENERATE INITIAL MAP FOR MATRIX SIZE WITH ODD VALUE (5, 7, 9, ...)
  IF (B24.BIT0) THEN
    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
```

```
' ' GENERATE INITIAL MAP FOR MATRIX SIZE WITH EVEN VALUE (6, 8, 10, ...)
' ELSE
'   FOR nowX = 0 TO (MatrixSize - 2)
'     FOR nowY = 0 TO (MatrixSize - 2)
'       IF (nowX < ((MatrixSize - 2) / 2)) THEN
'         IF (nowY < ((MatrixSize - 2) / 2)) THEN
'           nowDist = ((MatrixSize - 2) - nowX) - nowY
'         ELSE
'           IF (nowY >= (((MatrixSize - 2) / 2) + 1)) THEN
'             nowDist = (nowY - 1) - nowX
'           ELSE
'             nowDist = nowY - nowX
'           ENDIF
'         ENDIF
'       ELSE
'         IF (nowX >= (((MatrixSize - 2) / 2) + 1)) THEN
'           IF (nowY < ((MatrixSize - 2)/ 2)) THEN
'             nowDist = (nowX - 1) - nowY
'           ELSE
'             IF (nowY >= (((MatrixSize - 2) / 2) + 1)) THEN
'               nowDist = (nowY - 1) - ((MatrixSize - 2) - (nowX - 1))
'             ELSE
'               nowDist = nowY - ((MatrixSize - 2)-(nowX - 1))
'             ENDIF
'           ENDIF
'         ELSE
'           IF (nowY < ((MatrixSize - 2) / 2)) THEN
'             nowDist = nowX - nowY
'           ELSE
'             IF (nowY >= (((MatrixSize - 2) / 2) + 1)) THEN
'               nowDist = (nowY - 1) - nowX
'             ELSE
'               nowDist = nowY - nowX
'             ENDIF
'           ENDIF
'         ENDIF
'       ENDIF
'       B24 = CoordinateValue + B23
'       B25 = nowCoord
'       GOSUB Put_Byte
'       B24 = DistanceValue + B23
'       B25 = nowDist
'       GOSUB Put_Byte
'       B23 = B23 + 1
'     NEXT
'   NEXT

  ENDIF

  '#IF DebugNormal #THEN
  '  GOSUB Maze_Reading
  '#ENDIF

  RETURN

' ----------------------------------------
Init_Stack:

  #IF DebugLow #THEN
    DEBUG CR, "INIT_STACK:", CR
  #ENDIF

  FOR B24 = StackValue TO StackMaxAddress

    B25 = EqualTo255
    GOSUB Put_Byte

    #IF DebugNormal #THEN
      GOSUB Get_Byte
      IF (B24 = 107 OR B24 = 115) THEN
        DEBUG "[", DEC B24, "] = ", IHEX2 B25, CR
      ELSE
        DEBUG "[", DEC B24, "] = ", IHEX2 B25, "; "
      ENDIF
    #ENDIF

  NEXT

  RETURN
```

```
' ----------------------------------------
Calibrate_Sensors:

  #IF DebugLow #THEN
    DEBUG CR, "CALIBRATE_SENSORS:", CR
  #ENDIF

  HIGH FarMiddleLeftSensor              ' Turn SENSOR 6 on
  HIGH SensorInput                    ' Discharge capacitor
  PAUSE 1
  'RCTIME SensorInput, 1, sensorThreshold  ' Measure charge time
  RCTIME SensorInput, 1, W9  ' Measure charge time
  LOW FarMiddleLeftSensor               ' Turn SENSOR 6 off

  'sensorThreshold = sensorThreshold / 3   ' Take 1/4 average
  W9 = W9 / 3   ' Take 1/4 average

  'IF sensorThreshold > CodeOverhead THEN  ' Account for code overhead
  '  sensorThreshold = sensorThreshold - CodeOverhead
  'ELSE
  '  sensorThreshold = 0
  'ENDIF

  B20 = SensorThres
  GOSUB Put_Word:
  'GOSUB Get_Word:

  #IF DebugNormal #THEN
    'DEBUG ? sensorThreshold
    DEBUG "SensorThreshold = ", DEC W9, CR
  #ENDIF

  RETURN

' ----------------------------------------
Init_Start:

  #IF DebugLow #THEN
    DEBUG CR, "INIT_START", CR
  #ENDIF

  nowCoord = 0
  tempCoord = nowCoord
  nowFaceIn = 0
  GOSUB Update_Orientation:

  #IF DebugNormal #THEN
    DEBUG "nowCoord = ", IHEX2 nowCoord, "; ","tempCoord = ", IHEX2 tempCoord, CR
    DEBUG "nowFaceIn = ", IBIN4 nowFaceIn, "; ", "nowOrient = ", BIN4 nowOrient.NIB1, "-", BIN4 nowOrient.NIB0, CR
  #ENDIF

  RETURN

' ----------------------------------------
' PREPARE PARAMETERS FROM PREVIOUS PROCESS BEFORE ENTERING BIG LOOPING PROCESS AGAIN
Prepare_Cell_Paramaters:

  #IF DebugLow #THEN
    DEBUG CR, "PREP_CELL_PARAMS:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG ? isCheckpoint
  #ENDIF

  IF (isCheckpoint) THEN

    isCheckpoint = isLow

    ' TAKE CARE OF NEW VALUE : COORDINATE VALUE ----------
    nowCoord = tempCoord

    ' CHECK HAS REACHED START CELL AT GO BACK HOME
    GOSUB Get_Start_Status

    ' TAKE CARE OF NEW VALUE : DISTANCE VALUE ------------
    ' CHECK HAS REACHED DESTINATION CELL
    GOSUB Get_Finish_Status

    ' TAKE CARE OF NEW VALUE : WALL VALUE ----------------
    ' CHECK VISIT STATUS
    GOSUB Get_Visit_Status

    ' SET PATH STATUS
    GOSUB Set_Path_Status
```

```
    ' TAKE CARE OF NEW VALUE : ORIENTATION VALUE ---------
    nowFaceIn = tempFaceIn
    GOSUB Update_Orientation

    #IF DebugNormal #THEN
      DEBUG "nowCoord = ", IHEX2 nowCoord, "; ","tempCoord = ", IHEX2 tempCoord, CR
      DEBUG ? nowDist
      DEBUG "nowWall = ", BIN4 nowWall.NIB1, "-", BIN4 nowWall.NIB0, CR
      DEBUG "nowOrient = ", BIN4 nowOrient.NIB1, "-", BIN4 nowOrient.NIB0, CR
      DEBUG "nowFaceIn = ", IBIN4 nowFaceIn, "; ","tempFaceIn = ", IBIN4 tempFaceIn, CR
      DEBUG "isFinish = ", BIN isFinish, "; ", "isSetPath = ", BIN isSetPath, "; ", "isVisitedCell = ", BIN isVisitedCell, CR
    #ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Get_Start_Status:

  IF (nowCoord = 0 AND isFinish = IsHigh) THEN
    isFinish = IsLow
  ENDIF

  RETURN

' ----------------------------------------
Get_Finish_Status:

  B24 = DistanceValue + ((nowX * MatrixSize) + nowY)
  GOSUB Get_Byte
  nowDist = B25

  IF (nowDist = 0) THEN
    isFinish = IsHigh
    isSetPath = IsHigh
  ENDIF

  RETURN

' ----------------------------------------
Get_Visit_Status:

  isVisitedCell = IsLow

  B24 = WallValue + ((nowX * MatrixSize) + nowY)
  GOSUB Get_Byte
  nowWall = B25

  IF (nowWall.BIT5 = IsHigh) AND (nowWall.BIT4 = IsHigh) THEN
    isVisitedCell = IsHigh
  ENDIF

  RETURN

' ----------------------------------------
Set_Path_Status:

  #IF DebugLow #THEN
    DEBUG CR, "SET_PATH:", CR
  #ENDIF

  IF (isSetPath) THEN

    #IF DebugLow #THEN
      DEBUG "Yes", CR
    #ENDIF

    isSetPath = IsLow

    FOR B23 = 0 TO MatrixSizeSqMinSatu

      B24 = OrientValue + B23
      GOSUB Get_Byte
      B22 = B25

      IF (B22 <> 0) THEN

        B24 = WallValue + B23
        GOSUB Get_Byte
        B25.BIT6 = IsHigh
        GOSUB Put_Byte
```

```
      #IF DebugNormal #THEN
        DEBUG "[", DEC B24, "] = ", BIN4 B22.NIB1, "-", BIN4 B22.NIB0, "; ", "[", DEC B24, "] = ", BIN4 B25.NIB1, "-",
BIN4 B25.NIB0, CR
      #ENDIF

    ENDIF

   NEXT

  ENDIF

  RETURN

' ----------------------------------------
Update_Orientation:

  #IF DebugLow #THEN
   DEBUG CR, "UPDATE_ORIENT:", CR
  #ENDIF

  IF (isFinish = IsLow) THEN

   #IF DebugLow #THEN
    DEBUG "Yes", CR
   #ENDIF

   B24 = OrientValue + ((nowX * MatrixSize) + nowY)
   GOSUB Get_Byte
   nowOrient = B25

   SELECT tempFaceIn
    CASE North

     IF (nowOrient.BIT0 = IsLow) THEN
      IF (nowOrient.BIT6) THEN
       nowOrient.BIT6 = IsLow
      ELSEIF (nowOrient.BIT2) THEN
       nowOrient.BIT2 = IsLow
      ELSE
       nowOrient.BIT0 = IsHigh
      ENDIF
     ELSEIF (nowOrient.BIT4 = IsLow) THEN
      IF (nowOrient.BIT6) THEN
       nowOrient.BIT6 = IsLow
      ELSEIF (nowOrient.BIT2) THEN
       nowOrient.BIT2 = IsLow
      ELSE
       nowOrient.BIT4 = IsHigh
      ENDIF
     ENDIF

    CASE East

     IF (nowOrient.BIT1 = IsLow) THEN
      IF (nowOrient.BIT7) THEN
       nowOrient.BIT7 = IsLow
      ELSEIF (nowOrient.BIT3) THEN
       nowOrient.BIT3 = IsLow
      ELSE
       nowOrient.BIT1 = IsHigh
      ENDIF
     ELSEIF (nowOrient.BIT5 = IsLow) THEN
      IF (nowOrient.BIT7) THEN
       nowOrient.BIT7 = IsLow
      ELSEIF (nowOrient.BIT3) THEN
       nowOrient.BIT3 = IsLow
      ELSE
       nowOrient.BIT5 = IsHigh
      ENDIF
     ENDIF

    CASE South

     IF (nowOrient.BIT2 = IsLow) THEN
      IF (nowOrient.BIT4) THEN
       nowOrient.BIT4 = IsLow
      ELSEIF (nowOrient.BIT0) THEN
       nowOrient.BIT0 = IsLow
      ELSE
       nowOrient.BIT2 = IsHigh
      ENDIF
     ELSEIF (nowOrient.BIT6 = IsLow) THEN
      IF (nowOrient.BIT4) THEN
       nowOrient.BIT4 = IsLow
      ELSEIF (nowOrient.BIT0) THEN
```

```
          nowOrient.BIT0 = IsLow
        ELSE
          nowOrient.BIT6 = IsHigh
        ENDIF
      ENDIF

    CASE West

      IF (nowOrient.BIT3 = IsLow) THEN
        IF (nowOrient.BIT5) THEN
          nowOrient.BIT5 = IsLow
        ELSEIF (nowOrient.BIT1) THEN
          nowOrient.BIT1 = IsLow
        ELSE
          nowOrient.BIT3 = IsHigh
        ENDIF
      ELSEIF (nowOrient.BIT7 = IsLow) THEN
        IF (nowOrient.BIT5) THEN
          nowOrient.BIT5 = IsLow
        ELSEIF (nowOrient.BIT1) THEN
          nowOrient.BIT1 = IsLow
        ELSE
          nowOrient.BIT7 = IsHigh
        ENDIF
      ENDIF

  ENDSELECT

  B25 = nowOrient
  GOSUB Put_Byte

ENDIF

#IF DebugNormal #THEN
  DEBUG "isFinish = ", BIN isFinish, CR, CR
#ENDIF

RETURN

' -------------------------------------
Read_Wall_Sensors:

#IF DebugLow #THEN
  DEBUG CR, "READ_WALL_SENSORS:", CR
#ENDIF

B20 = SensorThres
GOSUB Get_Word

'HIGH OuterRightSensor     ' Turn on sensor
'GOSUB Change_IO_Direction
'sensor0 = SensorInput     ' Snapshot of sensor signal states
'LOW OuterRightSensor      ' Turn off sensor

HIGH FarMiddleRightSensor
GOSUB Change_IO_Direction
sensor1 = SensorInput
LOW FarMiddleRightSensor

HIGH NearMiddleRightSensor
GOSUB Change_IO_Direction
sensor2 = SensorInput
LOW NearMiddleRightSensor

HIGH InnerRightSensor
GOSUB Change_IO_Direction
sensor3 = SensorInput
LOW InnerRightSensor

HIGH InnerLeftSensor
GOSUB Change_IO_Direction
sensor4 = SensorInput
LOW InnerLeftSensor

HIGH NearMiddleLeftSensor
GOSUB Change_IO_Direction
sensor5 = SensorInput
LOW NearMiddleLeftSensor

HIGH FarMiddleLeftSensor
GOSUB Change_IO_Direction
sensor6 = SensorInput
LOW FarMiddleLeftSensor
```

```
  'HIGH OuterLeftSensor
  'GOSUB Change_IO_Direction
  'sensor7 = SensorInput
  'LOW OuterLeftSensor

  'HIGH OuterRearRightSensor
  'GOSUB Change_IO_Direction
  'sensor8 = SensorInput
  'LOW OuterRearRightSensor

  'HIGH InnerRearRightSensor
  'GOSUB Change_IO_Direction
  'sensor9 = SensorInput
  'LOW InnerRearRightSensor

  'HIGH InnerRearLefttSensor
  'GOSUB Change_IO_Direction
  'sensor10 = SensorInput
  'LOW InnerRearLefttSensor

  'HIGH OuterRearLeftSensor
  'GOSUB Change_IO_Direction
  'sensor11 = SensorInput
  'LOW OuterRearLeftSensor

  #IF DebugNormal #THEN
    DEBUG "wallSensors = ", BIN4 wallSensors.NIB1, "-", BIN4 wallSensors.NIB0, CR
  #ENDIF

  RETURN

' -----------------------------------------
Change_IO_Direction:

  HIGH SensorInput                ' Push signal voltages to 5 V
  PAUSE 0                         ' Wait 230 us for capacitors
  INPUT SensorInput               ' Start the decays
  'PULSOUT UnusedPin, sensorThreshold  ' Wait for threshold time
  PULSOUT UnusedPin, W9           ' Wait for threshold time

  RETURN

' -----------------------------------------
Put_Byte:
  PUT B24, B25
  RETURN

' -----------------------------------------
Get_Byte:
  GET B24, B25
  RETURN

' -----------------------------------------
Put_Word:
  PUT B20, Word W9
  RETURN

' -----------------------------------------
Get_Word:
  GET B20, Word W9
  RETURN
```

Kode Program Bank 1 EEPROM

```
' -----[ Initialization ]------------------------------------------------------
'
' -----[ Main Code ]-------------------------------------------------------------
'
' ----------------------------------------
Do_Process_1:

  #IF DebugLow #THEN
    DEBUG CR, "PROCESS_1:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG "nowCoord = ", IHEX2 nowCoord, "; ", "isVisitedCell = ", BIN isVisitedCell, "; ", "isCheckpoint = ", BIN
isCheckpoint, "; ", "isFinish = ", BIN isFinish, CR
  #ENDIF

  IF (isVisitedCell) THEN
    GOSUB Get_Checkpoint
  ELSE
    GOSUB Get_Wall
  ENDIF

  IF (isCheckpoint) THEN
    IF (isFinish) THEN
      GOTO Do_Process_2
    ELSE
      GOTO Do_Process_3
    ENDIF
  ELSE
    GOTO Do_Output_1
  ENDIF

' ----------------------------------------
Do_Process_2:  ' GO BACK HOME AFTER FINISH
  RUN Bank_2

' ----------------------------------------
Do_Process_3:  ' PROCESS DISTANCE VALUE
  RUN Bank_3

' ----------------------------------------
Do_Output_1:  ' MAKE MOVE
  RUN Bank_5

' -----[ Subroutines ]-----------------------------------------------------------
'
' ----------------------------------------
Get_Checkpoint:

  #IF DebugLow #THEN
    DEBUG CR, "GET_CHECKPOINT:", CR
  #ENDIF

  IF (sensor6 = IsLow AND sensor5 = IsLow) AND (sensor2 = IsLow AND sensor1 = IsLow) THEN
    isCheckpoint = IsHigh
  ENDIF

  #IF DebugNormal #THEN
    DEBUG ? isCheckpoint
  #ENDIF

  RETURN

' ----------------------------------------
Get_Wall:

  #IF DebugLow #THEN
    DEBUG CR, "GET_WALL:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG "nowFaceIn = ", IBIN4 nowFaceIn, "; ", "nowWall[54] = %", BIN nowWall.BIT5, BIN nowWall.BIT4, CR
  #ENDIF

  GOSUB Process_Side_Wall
  GOSUB Process_Checkpoint
  GOSUB Process_Wall

  RETURN
```

```
' ----------------------------------------
Process_Side_Wall:

  '#IF DebugLow #THEN
  '  DEBUG CR, "PROC_SIDE_WALL", CR
  '#ENDIF

  IF (nowFaceIn = North OR nowFaceIn = South) THEN
    IF (nowWall.BIT4 = IsLow) THEN
      GOSUB Get_Side_Wall
    ENDIF

  ELSEIF (nowFaceIn = East OR nowFaceIn = West) THEN
    IF (nowWall.BIT5 = IsLow) THEN
      GOSUB Get_Side_Wall
    ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Get_Side_Wall:

  #IF DebugLow #THEN
    DEBUG CR, "GET_SIDE_WALL:", CR
  #ENDIF

  IF (sensor6 = IsHigh AND sensor5 = IsHigh) OR (sensor2 = IsHigh AND sensor1 = IsHigh) THEN
    isSideWall = IsHigh
  ENDIF

  #IF DebugNormal #THEN
    DEBUG ? isSideWall
  #ENDIF

  RETURN
' ----------------------------------------
Process_Checkpoint:

  '#IF DebugLow #THEN
  '  DEBUG CR, "PROC_CHECKPOINT", CR
  '#ENDIF

  IF (nowFaceIn = North OR nowFaceIn = South) THEN
    IF (nowWall.BIT4 = IsHigh AND nowWall.BIT5 = IsLow) THEN
      GOSUB Get_Checkpoint
    ENDIF

  ELSEIF (nowFaceIn = East OR nowFaceIn = West) THEN
    IF (nowWall.BIT5 = IsHigh AND nowWall.BIT4 = IsLow) THEN
      GOSUB Get_Checkpoint
    ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Process_Wall:

  #IF DebugLow #THEN
    DEBUG CR, "PROC_WALL:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG "isSideWall = ", BIN isSideWall, "; ", "isCheckpoint = ", BIN isCheckpoint, CR
  #ENDIF

  IF (isSideWall OR isCheckpoint) THEN

    ' ---------- ROBOT FACING NORTH ----------------------------

    IF (nowFaceIn = North) THEN

      ' ---------- PROCESS SIDE WALL ----------------------------

      IF (nowWall.BIT4 = IsLow) THEN

        ' ----------------------------------------
        ' Set WEST WALL  = %00001000
        IF (nowWall.BIT3 = IsLow) THEN

          IF (nowX = 0) THEN
```

```
      nowWall.BIT3 = IsHigh
    ELSEIF (sensor6 OR sensor5) THEN
      nowWall.BIT3 = IsHigh
      ' HERE TO SET ADJACENT CELL'S WALL !!!
      tempX = nowX - 1
      tempY = nowY
      GOSUB Read_Adj_Wall
      IF (B25.BIT4 = IsLow) THEN
        B25.BIT1 = IsHigh  ' NEIGHBORHOOD CELL : EAST WALL  = %00000010
        GOSUB Put_Byte
      ENDIF
    ENDIF

  ENDIF

  ' ----------------------------------------
  ' Set EAST WALL  = %00000010
  IF (nowWall.BIT1 = IsLow) THEN

    IF (nowX = MatrixSizeMinSatu) THEN
      nowWall.BIT1 = IsHigh
    ELSEIF (sensor2 OR sensor1) THEN
      nowWall.BIT1 = IsHigh
      ' HERE TO SET ADJACENT CELL'S WALL !!!
      tempX = nowX + 1
      tempY = nowY
      GOSUB Read_Adj_Wall
      IF (B25.BIT4 = IsLow) THEN
        B25.BIT3 = IsHigh  ' NEIGHBORHOOD CELL : WEST WALL = %00001000
        GOSUB Put_Byte
      ENDIF
    ENDIF

  ENDIF

  ' ----------------------------------------
  nowWall.BIT4 = IsHigh  ' VISITED CELL
  isSideWall = IsLow

  ' ---------- PROCESS FRONT-REAR WALL ----------------------

  ELSEIF (nowWall.BIT4 = IsHigh AND nowWall.BIT5 = IsLow) THEN

    ' ----------------------------------------
    ' Set SOUTH WALL = %00000100
    IF (nowWall.BIT2 = IsLow) THEN

      IF (nowY = 0) THEN
        nowWall.BIT2 = IsHigh
      ENDIF

    ENDIF

    ' ----------------------------------------
    ' Set NORTH WALL = %00000001
    IF (nowWall.BIT0 = IsLow) THEN

      IF (nowY = MatrixSizeMinSatu) THEN
        nowWall.BIT0 = IsHigh
      ELSEIF (sensor4 OR sensor3) THEN
        nowWall.BIT0 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX
        tempY = nowY + 1
        GOSUB Read_Adj_Wall
        IF (B25.BIT5 = IsLow) THEN
          B25.BIT2 = IsHigh  ' NEIGHBORHOOD CELL : SOUTH WALL = %00000100
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ----------------------------------------
    nowWall.BIT5 = IsHigh  ' VISITED CELL
    ' isCheckpoint = IsHigh  ' STAY HIGH BECAUSE WILL BE USED FOR CROSS 2 ANOTHER CELL !

  ENDIF

' ---------- ROBOT FACING EAST -----------------------------

ELSEIF (nowFaceIn = East) THEN

  ' ---------- PROCESS SIDE WALL -----------------------------
```

```
IF (nowWall.BIT5 = IsLow) THEN

  ' ----------------------------------------
  ' SET SOUTH WALL  = %00000100
  IF (nowWall.BIT2 = IsLow) THEN

    IF (nowY = 0) THEN
      nowWall.BIT2 = IsHigh
    ELSEIF (sensor2 OR sensor1) THEN
      nowWall.BIT2 = IsHigh
      ' HERE TO SET ADJACENT CELL'S WALL !!!
      tempX = nowX
      tempY = nowY - 1
      GOSUB Read_Adj_Wall
      IF (B25.BIT5 = IsLow) THEN
        B25.BIT0 = IsHigh  ' NEIGHBORHOOD CELL : NORTH WALL = %00000001
        GOSUB Put_Byte
      ENDIF
    ENDIF

  ENDIF

  ' ----------------------------------------
  ' SET NORTH WALL  = %00000001
  IF (nowWall.BIT0 = IsLow) THEN

    IF (nowY = MatrixSizeMinSatu) THEN
      nowWall.BIT0 = IsHigh
    ELSEIF (sensor6 OR sensor5) THEN
      nowWall.BIT0 = IsHigh
      ' HERE TO SET ADJACENT CELL'S WALL !!!
      tempX = nowX
      tempY = nowY + 1
      GOSUB Read_Adj_Wall
      IF (B25.BIT5 = IsLow) THEN
        B25.BIT2 = IsHigh  ' NEIGHBORHOOD CELL : SOUTH WALL = %00000100
        GOSUB Put_Byte
      ENDIF
    ENDIF

  ENDIF

  ' ----------------------------------------
  nowWall.BIT5 = IsHigh  ' VISITED CELL
  isSideWall = IsLow

' ---------- PROCESS FRONT-REAR WALL ----------------------

ELSEIF (nowWall.BIT5 = IsHigh AND nowWall.BIT4 = IsLow) THEN

  ' ----------------------------------------
  ' Set WEST WALL = %00001000
  IF (nowWall.BIT3 = IsLow) THEN

    IF (nowX = 0) THEN
      nowWall.BIT3 = IsHigh
    ENDIF

  ENDIF

  ' ----------------------------------------
  ' Set EAST WALL = %00000010
  IF (nowWall.BIT1 = IsLow) THEN

    IF (nowX = MatrixSizeMinSatu) THEN
      nowWall.BIT1 = IsHigh
    ELSEIF (sensor4 OR sensor3) THEN
      nowWall.BIT1 = IsHigh
      ' HERE TO SET ADJACENT CELL'S WALL !!!
      tempX = nowX + 1
      tempY = nowY
      GOSUB Read_Adj_Wall
      IF (B25.BIT4 = IsLow) THEN
        B25.BIT3 = IsHigh   ' NEIGHBORHOOD CELL : WEST WALL = %00001000
        GOSUB Put_Byte
      ENDIF
    ENDIF

  ENDIF

  ' ----------------------------------------
  nowWall.BIT4 = IsHigh  ' VISITED CELL
  ' isCheckpoint = IsHigh  ' STAY HIGH BECAUSE WILL BE USED FOR CROSS 2 ANOTHER CELL !

ENDIF
```

```
' ---------- ROBOT FACING SOUTH -----------------------------

ELSEIF (nowFaceIn = South) THEN

  ' ---------- PROCESS SIDE WALL -----------------------------

  IF (nowWall.BIT4 = IsLow) THEN

    ' ---------------------------------------
    ' Set WEST WALL  = %00001000
    IF (nowWall.BIT3 = IsLow) THEN

      IF (nowX = 0) THEN
        nowWall.BIT3 = IsHigh
      ELSEIF (sensor2 OR sensor1) THEN
        nowWall.BIT3 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX - 1
        tempY = nowY
        GOSUB Read_Adj_Wall
        IF (B25.BIT4 = IsLow) THEN
          B25.BIT1 = IsHigh  ' NEIGHBORHOOD CELL : EAST WALL  = %00000010
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ---------------------------------------
    ' Set EAST WALL  = %00000010
    IF (nowWall.BIT1 = IsLow) THEN

      IF (nowX = MatrixSizeMinSatu) THEN
        nowWall.BIT1 = IsHigh
      ELSEIF (sensor6 OR sensor5) THEN
        nowWall.BIT1 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX + 1
        tempY = nowY
        GOSUB Read_Adj_Wall
        IF (B25.BIT4 = IsLow) THEN
          B25.BIT3 = IsHigh  ' NEIGHBORHOOD CELL : WEST WALL = %00001000
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ---------------------------------------
    nowWall.BIT4 = IsHigh  ' VISITED CELL
    isSideWall = IsLow

  ' ---------- PROCESS FRONT-REAR WALL -----------------------

  ELSEIF (nowWall.BIT4 = IsHigh AND nowWall.BIT5 = IsLow) THEN

    ' ---------------------------------------
    ' Set SOUTH WALL = %00000100
    IF (nowWall.BIT2 = IsLow) THEN

      IF (nowY = 0) THEN
        nowWall.BIT2 = IsHigh
      ELSEIF (sensor4 OR sensor3) THEN
        nowWall.BIT2 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX
        tempY = nowY - 1
        GOSUB Read_Adj_Wall
        IF (B25.BIT5 = IsLow) THEN
          B25.BIT0 = IsHigh  ' NEIGHBORHOOD CELL : NORTH WALL = %00000001
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ---------------------------------------
    ' Set NORTH WALL = %00000001
    IF (nowWall.BIT0 = IsLow) THEN

      IF (nowY = MatrixSizeMinSatu) THEN
        nowWall.BIT0 = IsHigh
      ENDIF
```

```
    ENDIF

    ' ----------------------------------------
    nowWall.BIT5 = IsHigh  ' VISITED CELL
    ' isCheckpoint = IsHigh  ' STAY HIGH BECAUSE WILL BE USED FOR CROSS 2 ANOTHER CELL !

  ENDIF

' ---------- ROBOT FACING WEST ----------------------------

ELSEIF (nowFaceIn = West) THEN

  ' ---------- PROCESS SIDE WALL ----------------------------

  IF (nowWall.BIT5 = IsLow) THEN

    ' ----------------------------------------
    ' SET SOUTH WALL  = %00000100
    IF (nowWall.BIT2 = IsLow) THEN

      IF (nowY = 0) THEN
        nowWall.BIT2 = IsHigh
      ELSEIF (sensor6 OR sensor5) THEN
        nowWall.BIT2 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX
        tempY = nowY - 1
        GOSUB Read_Adj_Wall
        IF (B25.BIT5 = IsLow) THEN
          B25.BIT0 = IsHigh  ' NEIGHBORHOOD CELL : NORTH WALL = %00000001
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ----------------------------------------
    ' SET NORTH WALL  = %00000001
    IF (nowWall.BIT0 = IsLow) THEN

      IF (nowY = MatrixSizeMinSatu) THEN
        nowWall.BIT0 = IsHigh
      ELSEIF (sensor2 OR sensor1) THEN
        nowWall.BIT0 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX
        tempY = nowY + 1
        GOSUB Read_Adj_Wall
        IF (B25.BIT5 = IsLow) THEN
          B25.BIT2 = IsHigh  ' NEIGHBORHOOD CELL : SOUTH WALL = %00000100
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ----------------------------------------
    nowWall.BIT5 = IsHigh  ' VISITED CELL
    isSideWall = IsLow

  ' ---------- PROCESS FRONT-REAR WALL ----------------------

  ELSEIF (nowWall.BIT5 = IsHigh AND nowWall.BIT4 = IsLow) THEN

    ' ----------------------------------------
    ' Set WEST WALL = %00001000
    IF (nowWall.BIT3 = IsLow) THEN

      IF (nowX = 0) THEN
        nowWall.BIT3 = IsHigh
      ELSEIF (sensor4 OR sensor3) THEN
        nowWall.BIT3 = IsHigh
        ' HERE TO SET ADJACENT CELL'S WALL !!!
        tempX = nowX - 1
        tempY = nowY
        GOSUB Read_Adj_Wall
        IF (B25.BIT4 = IsLow) THEN
          B25.BIT1 = IsHigh   ' NEIGHBORHOOD CELL : EAST WALL = %00000010
          GOSUB Put_Byte
        ENDIF
      ENDIF

    ENDIF

    ' ----------------------------------------
```

```
      ' Set EAST WALL = %00000010
      IF (nowWall.BIT1 = IsLow) THEN

        IF (nowX = MatrixSizeMinSatu) THEN
          nowWall.BIT1 = IsHigh
        ENDIF

      ENDIF

      ' ----------------------------------------
      nowWall.BIT4 = IsHigh  ' VISITED CELL
      ' isCheckpoint = IsHigh  ' STAY HIGH BECAUSE WILL BE USED FOR CROSS 2 ANOTHER CELL !

    ENDIF

  ENDIF

 ' ----------------------------------------
  GOSUB Write_Wall

 ' ----------------------------------------
  #IF DebugNormal #THEN
    DEBUG "nowWall = ", BIN4 nowWall.NIB1, "-", BIN4 nowWall.NIB0, CR
  #ENDIF

 ENDIF

 RETURN

' ----------------------------------------
Read_Adj_Wall:

 B24 = WallValue + ((tempX * MatrixSize) + tempY)
 GOSUB Get_Byte

 RETURN

' ----------------------------------------
Write_Wall:

 B24 = WallValue + ((nowX * MatrixSize) + nowY)
 B25 = nowWall
 GOSUB Put_Byte

 RETURN

' ----------------------------------------
Put_Byte:
 PUT B24, B25
 RETURN

' ----------------------------------------
Get_Byte:
 GET B24, B25
 RETURN
```

Kode Program Bank 2 EEPROM

```
' -----[ Initialization ]----------------------------------------------------
'
' -----[ Main Code ]-----------------------------------------------------------
'
' ----------------------------------------
Do_Process_2:

  #IF DebugLow #THEN
    DEBUG CR, "PROCESS_2:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG IHEX2 ? nowCoord
  #ENDIF

  GOSUB Reset_Path_Status
  GOSUB Get_Open_Distance
  GOSUB Get_Visited_Open_Distance
  GOSUB Get_Open_Marked_Path

' ----------------------------------------
Do_Process_4:  ' PREPARE FOR NEXT MOVE
  RUN Bank_4

' -----[ Subroutines ]---------------------------------------------------------
'
' ----------------------------------------
Reset_Path_Status:

  #IF DebugLow #THEN
    DEBUG CR, "RESET_PATH:", CR
  #ENDIF

  IF (nowCoord <> 0) THEN

    B24 = OrientValue + ((nowX * MatrixSize) + nowY)
    B25 = 0
    GOSUB Put_Byte

  ENDIF

  B24 = WallValue + ((nowX * MatrixSize) + nowY)
  GOSUB Get_Byte
  B25.BIT6 = IsLow
  GOSUB Put_Byte

  RETURN

' ----------------------------------------
Get_Open_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "GET_OPEN_DIST:", CR
  #ENDIF

  ' INITIALIZING
  FOR index = West TO North
    adjCoordArr(index) = EqualTo255
    adjDistArr(index) = EqualTo15
  NEXT
  numOfOpenDist = 0
  openDist = 0

  ' PROCESS FOR OPEN NEIGHBOUR AT WEST
  IF (nowX > 0) THEN
    IF (nowWall.BIT3 = IsLow) THEN
      index = West
      tempX = nowX - 1
      tempY = nowY
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' PROCESS FOR OPEN NEIGHBOUR AT SOUTH
  IF (nowY > 0) THEN
    IF (nowWall.BIT2 = IsLow) THEN
      index = South
      tempX = nowX
      tempY = nowY - 1
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF
```

```
  ' PROCESS FOR OPEN NEIGHBOUR AT EAST
  IF (nowX < MatrixSizeMinSatu) THEN
    IF (nowWall.BIT1 = IsLow) THEN
      index = East
      tempX = nowX + 1
      tempY = nowY
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' PROCESS FOR OPEN NEIGHBOUR AT NORTH
  IF (nowY < MatrixSizeMinSatu) THEN
    IF (nowWall.BIT0 = IsLow) THEN
      index = North
      tempX = nowX
      tempY = nowY + 1
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' DEBUG CODE INCLUDED
  GOSUB Get_Number_Of_Open_Distance

  RETURN

' ----------------------------------------
Save_Open_Distance:

  adjCoordArr(index) = tempCoord

  B24 = DistanceValue + ((tempX * MatrixSize) + tempY)
  GOSUB Get_Byte
  adjDistArr(index) = B25

  openDist.LOWBIT(index) = IsHigh

  RETURN

' ----------------------------------------
Get_Number_Of_Open_Distance:

  'INITIALIZING
  numOfOpenDist = 0

  FOR index = West TO North
    IF (openDist.LOWBIT(index)) THEN

      #IF DebugNormal #THEN
        DEBUG "adjCoordArr", "[", DEC index, "] = ", IHEX2 adjCoordArr(index), "; ", "adjDistArr", "[", DEC index, "] = ",
DEC adjDistArr(index), CR
      #ENDIF

      numOfOpenDist = numOfOpenDist + 1

    ENDIF
  NEXT

  #IF DebugNormal #THEN
    DEBUG "numOfOpenDist = ", DEC numOfOpenDist, "; ", "openDist = ", IBIN4 openDist, CR
  #ENDIF

  RETURN

' ----------------------------------------
Get_Visited_Open_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "GET_VIS_OPEN_DIST:", CR
  #ENDIF

  FOR index = West TO North
    IF (openDist.LOWBIT(index)) THEN

      tempCoord = adjCoordArr(index)
      B24 = WallValue + ((tempX * MatrixSize) + tempY)
      GOSUB Get_Byte

      IF (B25.BIT5 = IsLow) OR (B25.BIT4 = IsLow) THEN
        openDist.LOWBIT(index) = IsLow
      ENDIF

    ENDIF
  NEXT
```

```
  ' DEBUG CODE INCLUDED
  GOSUB Get_Number_Of_Open_Distance

  RETURN

' ----------------------------------------
Get_Open_Marked_Path:

  #IF DebugLow #THEN
    DEBUG CR, "GET_OPEN_MARKED_PATH:", CR
  #ENDIF

  FOR index = West TO North
    IF (openDist.LOWBIT(index)) THEN

      tempCoord = adjCoordArr(index)

      B24 = WallValue + ((tempX * MatrixSize) + tempY)
      GOSUB Get_Byte
      IF (B25.BIT6 = IsLow) THEN
        openDist.LOWBIT(index) = IsLow
      ENDIF

    ENDIF
  NEXT

  ' DEBUG CODE INCLUDED
  GOSUB Get_Number_Of_Open_Distance

  RETURN

' ----------------------------------------
Put_Byte:
  PUT B24, B25
  RETURN

' ----------------------------------------
Get_Byte:
  GET B24, B25
  RETURN
```

Kode Program Bank 3 EEPROM

```
' -----[ Initialization ]------------------------------------------------------
'
' -----[ Main Code ]-------------------------------------------------------------
'
' ----------------------------------------
Do_Process_3:

  #IF DebugLow #THEN
    DEBUG CR, "PROCESS_3:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG IHEX2 ? nowCoord
  #ENDIF

  stackPointer = 0
  tempCoord = nowCoord
  GOSUB Push_Coordinate

' ----------------------------------------
Process_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "PROC_DIST:", CR
  #ENDIF

  GOSUB Pull_Coordinate
  GOSUB Read_Wall

  IF (isNoWall) THEN
    GOTO Check_Stack_Pointer
  ENDIF

  GOSUB Get_Open_Distance
  GOSUB Get_Lowest_Open_Distance
  GOSUB Check_Current_Distance

  IF (isBrokenRule) THEN
    GOSUB Push_Open_Distance

  #IF DebugNormal #THEN
  ELSE
    DEBUG "No push", CR
  #ENDIF

  ENDIF
' ----------------------------------------
Check_Stack_Pointer:

  #IF DebugLow #THEN
    DEBUG CR, "CHECK_SP:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG ? stackPointer
  #ENDIF

  IF (stackPointer > 0) THEN
    GOTO Process_Distance
  ENDIF

' ----------------------------------------
Current_Coordinate:
  GOSUB Process_Current_Coordinate

' ----------------------------------------
Do_Process_4:  ' PREPARE FOR NEXT MOVE
  RUN Bank_4

' -----[ Subroutines ]-----------------------------------------------------------
'
' ----------------------------------------
Process_Current_Coordinate:

  #IF DebugLow #THEN
    DEBUG CR, "PROC_CUR_COORD:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG IHEX2 ? nowCoord
  #ENDIF
```

```
  tempCoord = nowCoord
  GOSUB Read_Wall
  GOSUB Get_Open_Distance
  GOSUB Get_Lowest_Open_Distance

  RETURN

' ----------------------------------------
Push_Coordinate:

  #IF DebugLow #THEN
    DEBUG CR, "PUSH_COORD:", CR
  #ENDIF

  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

  #IF DebugNormal #THEN
    DEBUG "Push = ", IHEX2 tempCoord, CR
    GOSUB Show_Stack
  #ENDIF

  RETURN

' ----------------------------------------
Pull_Coordinate:

  #IF DebugLow #THEN
    DEBUG CR, "PULL_COORD:", CR
  #ENDIF

  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

  #IF DebugNormal #THEN
    DEBUG "Pull = ", IHEX2 tempCoord, CR
    GOSUB Show_Stack
  #ENDIF

  RETURN

' ----------------------------------------
Show_Stack:

  #IF DebugNormal #THEN
    DEBUG ? stackPointer
    IF (stackPointer > 0) THEN
      FOR B23 = 0 TO stackPointer-1
        B24 = StackValue + B23
        GOSUB Get_Byte
        DEBUG "[", DEC B23, "] = ", IHEX2 B25, CR
      NEXT
    ENDIF
  #ENDIF

  RETURN
```

```
' --------------------------------------
Read_Wall:

  #IF DebugLow #THEN
    DEBUG CR, "READ_WALL:", CR
  #ENDIF

  ' INITIALIZING
  isNoWall = 0

  B24 = WallValue + ((tempX * MatrixSize) + tempY)
  GOSUB Get_Byte
  B23 = B25

  IF (B23.BIT5 = IsLow) OR (B23.BIT4 = IsLow) THEN
    isNoWall = 1
  ENDIF

  #IF DebugNormal #THEN
    DEBUG "Wall = ", BIN4 B23.NIB1, "-", BIN4 B23.NIB0, CR
    DEBUG ? isNoWall
  #ENDIF

  RETURN

' --------------------------------------
Get_Open_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "GET_OPEN_DIST:", CR
  #ENDIF

  ' INITIALIZING
  FOR index = West TO North
    adjCoordArr(index) = EqualTo255
    adjDistArr(index) = EqualTo15
  NEXT
  numOfOpenDist = 0
  openDist = 0

  ' PROCESS FOR OPEN NEIGHBOUR AT WEST
  IF (tempX > 0) THEN
    IF (B23.BIT3 = IsLow) THEN
      index = West
      B22.NIB1 = tempX - 1
      B22.NIB0 = tempY
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' PROCESS FOR OPEN NEIGHBOUR AT SOUTH
  IF (tempY > 0) THEN
    IF (B23.BIT2 = IsLow) THEN
      index = South
      B22.NIB1 = tempX
      B22.NIB0 = tempY - 1
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' PROCESS FOR OPEN NEIGHBOUR AT EAST
  IF (tempX < MatrixSizeMinSatu) THEN
    IF (B23.BIT1 = IsLow) THEN
      index = East
      B22.NIB1 = tempX + 1
      B22.NIB0 = tempY
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' PROCESS FOR OPEN NEIGHBOUR AT NORTH
  IF (tempY < MatrixSizeMinSatu) THEN
    IF (B23.BIT0 = IsLow) THEN
      index = North
      B22.NIB1 = tempX
      B22.NIB0 = tempY + 1
      GOSUB Save_Open_Distance
    ENDIF
  ENDIF

  ' DEBUG CODE INCLUDED
  GOSUB Get_Number_Of_Open_Distance

  RETURN
```

```
' ---------------------------------------
Save_Open_Distance:

  adjCoordArr(index) = B22

  B24 = DistanceValue + ((B22.NIB1 * MatrixSize) + B22.NIB0)
  GOSUB Get_Byte
  adjDistArr(index) = B25

  openDist.LOWBIT(index) = IsHigh

  RETURN

' ---------------------------------------
Get_Number_Of_Open_Distance:

  'INITIALIZING
  numOfOpenDist = 0

  FOR index = West TO North
    IF (openDist.LOWBIT(index)) THEN

      #IF DebugNormal #THEN
        DEBUG "adjCoord", "[", DEC index, "] = ", IHEX2 adjCoordArr(index), "; ", "adjDist", "[", DEC index, "] = ", DEC
adjDistArr(index), CR
      #ENDIF

      numOfOpenDist = numOfOpenDist + 1

    ENDIF
  NEXT

  #IF DebugNormal #THEN
    DEBUG "numOfOpenDist = ", DEC numOfOpenDist, "; ", "openDist = ", IBIN4 openDist, CR
  #ENDIF

  RETURN

' ---------------------------------------
Get_Lowest_Open_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "GET_LOWEST:", CR
  #ENDIF

  IF (numOfOpenDist > OneOpenNeighbour) THEN
    IF (numOfOpenDist = TwoOpenNeighbours) THEN
      GOSUB Sort_2_Distances
    ELSEIF (numOfOpenDist = ThreeOpenNeighbours) THEN
      GOSUB Sort_3_Distances
    ENDIF

    ' DEBUG CODE INCLUDED
    GOSUB Get_Number_Of_Open_Distance

  #IF DebugNormal #THEN
  ELSE
      DEBUG "Only 1", CR
  #ENDIF

  ENDIF

  RETURN

' ---------------------------------------
Sort_2_Distances:

  '#IF DebugLow #THEN
  '  DEBUG CR, "SORT_2:", CR
  '#ENDIF

  SELECT openDist
    CASE %0011
      B25 = East
      B24 = North
    CASE %0101
      B25 = South
      B24 = North
    CASE %1001
      B25 = West
      B24 = North
    CASE %0110
      B25 = South
      B24 = East
```

```
    CASE %1010
      B25 = West
      B24 = East
    CASE %1100
      B25 = West
      B24 = South
  ENDSELECT

  GOSUB Set_Lowest_Of_2_Distances

  RETURN

' ---------------------------------------
Set_Lowest_Of_2_Distances:

  IF adjDistArr(B25) < adjDistArr(B24) THEN
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B25) = IsHigh
    openDist.LOWBIT(B24) = IsLow

  ELSEIF adjDistArr(B25) > adjDistArr(B24) THEN
    openDist.LOWBIT(B25) = IsLow
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B24) = IsHigh

  'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
  'ELSE
  '  openDist.LOWBIT(B25) = IsHigh
  '  openDist.LOWBIT(B24) = IsHigh

  ENDIF

  RETURN

' ---------------------------------------
Sort_3_Distances:

  '#IF DebugLow #THEN
  '  DEBUG CR, "SORT_3:", CR
  '#ENDIF

  SELECT openDist
    CASE %0111
      B25 = South
      B24 = East
      B23 = North
    CASE %1011
      B25 = West
      B24 = East
      B23 = North
    CASE %1101
      B25 = West
      B24 = South
      B23 = North
    CASE %1110
      B25 = West
      B24 = South
      B23 = East
  ENDSELECT

  GOSUB Set_Lowest_Of_3_Distances

  RETURN

' ---------------------------------------
Set_Lowest_Of_3_Distances:

  IF adjDistArr(B25) < adjDistArr(B24) THEN

    IF adjDistArr(B25) < adjDistArr(B23) THEN
      'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
      'openDist.LOWBIT(B25) = IsHigh
      openDist.LOWBIT(B24) = IsLow
      openDist.LOWBIT(B23) = IsLow

    ELSEIF adjDistArr(B25) > adjDistArr(B23) THEN
      openDist.LOWBIT(B25) = IsLow
      openDist.LOWBIT(B24) = IsLow
      'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
      'openDist.LOWBIT(B23) = IsHigh

    ELSE
      'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
      'openDist.LOWBIT(B25) = IsHigh
      openDist.LOWBIT(B24) = IsLow
```

```
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B23) = IsHigh

   ENDIF

  ELSEIF adjDistArr(B25) > adjDistArr(B24) THEN

   IF adjDistArr(B24) < adjDistArr(B23) THEN
    openDist.LOWBIT(B25) = IsLow
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B24) = IsHigh
    openDist.LOWBIT(B23) = IsLow

   ELSEIF adjDistArr(B24) > adjDistArr(B23) THEN
    openDist.LOWBIT(B25) = IsLow
    openDist.LOWBIT(B24) = IsLow
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B23) = IsHigh

   ELSE
    openDist.LOWBIT(B25) = IsLow
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B24) = IsHigh
    'openDist.LOWBIT(B23) = IsHigh

   ENDIF

  ELSE

   IF adjDistArr(B25) < adjDistArr(B23) THEN
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B25) = IsHigh
    'openDist.LOWBIT(B24) = IsHigh
    openDist.LOWBIT(B23) = IsLow

   ELSEIF adjDistArr(B25) > adjDistArr(B23) THEN
    openDist.LOWBIT(B25) = IsLow
    openDist.LOWBIT(B24) = IsLow
    'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
    'openDist.LOWBIT(B23) = IsHigh

   'NO NEED BECAUSE ALREADY HIGH FROM PREVIOUS SETTING
   'ELSE
   '  openDist.LOWBIT(B25) = IsHigh
   '  openDist.LOWBIT(B24) = IsHigh
   '  openDist.LOWBIT(B23) = IsHigh

   ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Check_Current_Distance:

  #IF DebugLow #THEN
   DEBUG CR, "CHECK_CUR_DIST:", CR
  #ENDIF

  ' INITIALIZING
  isBrokenRule = IsLow

  B24 = DistanceValue + ((tempX * MatrixSize) + tempY)
  GOSUB Get_Byte

  #IF DebugNormal #THEN
   DEBUG "tempDist = ", DEC B25, CR
  #ENDIF

  FOR index = West TO North
   IF (openDist.LOWBIT(index)) THEN
    IF (B25 < adjDistArr(index)) THEN

     isBrokenRule = IsHigh
     B25 = adjDistArr(index) + 1
     'B24 = DistanceValue + ((tempX * MatrixSize) + tempY)
     GOSUB Put_Byte

    ENDIF
    EXIT
   ENDIF
  NEXT

  #IF DebugNormal #THEN
```

```
    DEBUG "isBrokenRule = ", BIN isBrokenRule, "; ", "tempDist = ", DEC B25, CR
  #ENDIF

  RETURN

' ----------------------------------------
Push_Open_Distance:

  #IF DebugLow #THEN
    DEBUG CR, "PUSH_OPEN_DIST:", CR
  #ENDIF

  'IF (isBrokenRule) THEN

    FOR index = West TO North
      IF (adjCoordArr(index) < EqualTo255) THEN

        #IF DebugNormal #THEN
          DEBUG CR, "adjCoord", "[", DEC index, "] = ", IHEX2 adjCoordArr(index), CR
        #ENDIF

        tempCoord = adjCoordArr(index)
        ' THAT'S WHY Push_Coordinate DO NOT USE index (REPLACED BY B23)
        GOSUB Push_Coordinate

      ENDIF
    NEXT

  '#IF DebugNormal #THEN
  'ELSE
  '    DEBUG "No push", CR
  '#ENDIF
  '
  'ENDIF

  RETURN

' ----------------------------------------
Put_Byte:
  PUT B24, B25
  RETURN

' ----------------------------------------
Get_Byte:
  GET B24, B25
  RETURN
```

Kode Program Bank 4 EEPROM

```
' -----[ Initialization ]------------------------------------------------------
'
' -----[ Main Code ]-------------------------------------------------------------
'
' ----------------------------------------
Do_Process_4:

  #IF DebugLow #THEN
    DEBUG CR, "PROCESS_4:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG IHEX2 ? nowCoord
  #ENDIF

  GOSUB Process_Destination

' ----------------------------------------
Do_Output_1:
  RUN Bank_5

' -----[ Subroutines ]-----------------------------------------------------------
'
' ----------------------------------------
Process_Destination:

  #IF DebugLow #THEN
    DEBUG CR, "PROC_DEST:", CR
  #ENDIF

  IF (nowFaceIn = North) THEN

    SELECT openDist

      CASE %0001, %0011, %0101, %1001, %0111, %1011, %1101
        index = North
        GOSUB Process_New_Coordinate
        GOSUB Process_New_Orientation
        moveType = GoStraight

      CASE %0010, %0110, %1010, %1110
        index = East
        GOSUB Process_New_Coordinate
        GOSUB Process_New_Orientation
        moveType = RotateRight90

      CASE %0100
        index = South
        GOSUB Process_New_Coordinate
        GOSUB Process_New_Orientation
        moveType = RotateRight180

      CASE %1000, %1100
        index = West
        GOSUB Process_New_Coordinate
        GOSUB Process_New_Orientation
        moveType = RotateLeft90

      'CASE %0011 ' GO EAST OR NORTH
      'CASE %0101 ' GO SOUTH OR NORTH
      'CASE %1001 ' GO WEST OR NORTH
      'CASE %0110 ' GO SOUTH OR EAST
      'CASE %1010 ' GO WEST OR EAST *
      'CASE %1100 ' GO WEST OR SOUTH
      'CASE %0111 ' GO SOUTH, EAST, OR NORTH
      'CASE %1011 ' GO WEST, EAST, OR NORTH
      'CASE %1101 ' GO WEST, SOUTH, OR NORTH
      'CASE %1110 ' GO WEST, SOUTH, OR EAST * (ABSOLUTELY NOT EAST)

    ENDSELECT

  ELSEIF (nowFaceIn = East) THEN

    SELECT openDist

      CASE %0001, %1001
        index = North
        GOSUB Process_New_Coordinate
        GOSUB Process_New_Orientation
        moveType = RotateLeft90
```

```
    CASE %0010, %0011, %0110, %1010, %0111, %1011, %1110
      index = East
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = GoStraight

    CASE %0100, %0101, %1100, %1101
      index = South
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateRight90

    CASE %1000
      index = West
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateRight180

    'CASE %0011 ' GO EAST OR NORTH
    'CASE %0101 ' GO SOUTH OR NORTH *
    'CASE %1001 ' GO WEST OR NORTH
    'CASE %0110 ' GO SOUTH OR EAST
    'CASE %1010 ' GO WEST OR EAST
    'CASE %1100 ' GO WEST OR SOUTH
    'CASE %0111 ' GO SOUTH, EAST, OR NORTH
    'CASE %1011 ' GO WEST, EAST, OR NORTH
    'CASE %1101 ' GO WEST, SOUTH, OR NORTH *
    'CASE %1110 ' GO WEST, SOUTH, OR EAST

  ENDSELECT

ELSEIF (nowFaceIn = South) THEN

  SELECT openDist

    CASE %0001
      index = North
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateRight180

    CASE %0010, %0011
      index = East
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateLeft90

    CASE %0100, %0101, %0110, %1100, %0111, %1101, %1110
      index = South
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = GoStraight

    CASE %1000, %1001, %1010, %1011
      index = West
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateRight90

    'CASE %0011 ' GO EAST OR NORTH
    'CASE %0101 ' GO SOUTH OR NORTH
    'CASE %1001 ' GO WEST OR NORTH
    'CASE %0110 ' GO SOUTH OR EAST
    'CASE %1010 ' GO WEST OR EAST *
    'CASE %1100 ' GO WEST OR SOUTH
    'CASE %0111 ' GO SOUTH, EAST, OR NORTH
    'CASE %1011 ' GO WEST, EAST, OR NORTH *
    'CASE %1101 ' GO WEST, SOUTH, OR NORTH
    'CASE %1110 ' GO WEST, SOUTH, OR EAST

  ENDSELECT

ELSEIF (nowFaceIn = West) THEN

  SELECT openDist

    CASE %0001, %0011, %0101, %0111
      index = North
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateRight90

    CASE %0010
      index = East
      GOSUB Process_New_Coordinate
```

```
      GOSUB Process_New_Orientation
      moveType = RotateRight180

    CASE %0100, %0110
      index = South
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = RotateLeft90

    CASE %1000, %1001, %1010, %1100, %1011, %1101, %1110
      index = West
      GOSUB Process_New_Coordinate
      GOSUB Process_New_Orientation
      moveType = GoStraight

    'CASE %0011 ' GO EAST OR NORTH
    'CASE %0101 ' GO SOUTH OR NORTH *
    'CASE %1001 ' GO WEST OR NORTH
    'CASE %0110 ' GO SOUTH OR EAST
    'CASE %1010 ' GO WEST OR EAST
    'CASE %1100 ' GO WEST OR SOUTH
    'CASE %0111 ' GO SOUTH, EAST, OR NORTH *
    'CASE %1011 ' GO WEST, EAST, OR NORTH
    'CASE %1101 ' GO WEST, SOUTH, OR NORTH
    'CASE %1110 ' GO WEST, SOUTH, OR EAST

  ENDSELECT

 ENDIF

 #IF DebugNormal #THEN
  DEBUG CR, "openDist = ", IBIN4 openDist, "; ","index = ", IBIN4 index, "; ","moveType = ", IBIN4 moveType, CR
 #ENDIF

 RETURN

' ---------------------------------------
Process_New_Coordinate:

 #IF DebugLow #THEN
  DEBUG CR, "PROC_NEW_COORD:", CR
 #ENDIF

 SELECT index
  CASE North
   tempX = nowX
   tempY = nowY + 1
  CASE East
   tempX = nowX + 1
   tempY = nowY
  CASE South
   tempX = nowX
   tempY = nowY - 1
  CASE West
   tempX = nowX - 1
   tempY = nowY
 ENDSELECT

 #IF DebugNormal #THEN
  DEBUG "nowCoord = ", IHEX2 nowCoord, "; ","tempCoord = ", IHEX2 tempCoord, CR
 #ENDIF

 RETURN

' ---------------------------------------
Process_New_Orientation:

 #IF DebugLow #THEN
  DEBUG CR, "PROC_NEW_ORIENT:", CR
 #ENDIF

 SELECT index
  CASE North
   tempFaceIn = North
  CASE East
   tempFaceIn = East
  CASE South
   tempFaceIn = South
  CASE West
   tempFaceIn = West
 ENDSELECT

 GOSUB Process_Destination_Orientation

 GOSUB Update_Orientation
```

```
  #IF DebugNormal #THEN
    DEBUG "nowFaceIn = ", IBIN4 nowFaceIn, "; ","tempFaceIn = ", IBIN4 tempFaceIn, CR
  #ENDIF

  RETURN

' ----------------------------------------
Process_Destination_Orientation:

  #IF DebugLow #THEN
    DEBUG CR, "PROC_DES_ORIENT:", CR
  #ENDIF

  IF (isFinish = IsLow) THEN

    B24 = DistanceValue + ((tempX * MatrixSize) + tempY)
    GOSUB Get_Byte

    #IF DebugNormal #THEN
      DEBUG "tempDist = ", DEC B25, CR
    #ENDIF

    IF (B25 = 0) THEN

      #IF DebugLow #THEN
        DEBUG "Yes", CR
      #ENDIF

      ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
      'GOSUB Update_Destination_Orientation

      B23 = 0

      B24 = OrientValue + ((tempX * MatrixSize) + tempY)
      GOSUB Get_Byte
      B23 = B25

      SELECT tempFaceIn
        CASE North, South
          B23.BIT0 = IsHigh
          B23.BIT2 = IsHigh
        CASE East, West
          B23.BIT1 = IsHigh
          B23.BIT3 = IsHigh
      ENDSELECT

      B25 = B23
      GOSUB Put_Byte

    ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Update_Orientation:

  #IF DebugLow #THEN
    DEBUG CR, "UPDATE_ORIENT:", CR
  #ENDIF

  IF (isFinish = IsLow) THEN

    #IF DebugLow #THEN
      DEBUG "Yes", CR
    #ENDIF

    B24 = OrientValue + ((nowX * MatrixSize) + nowY)
    GOSUB Get_Byte
    nowOrient = B25

    SELECT tempFaceIn
      CASE North

        IF (nowOrient.BIT0 = IsLow) THEN
          IF (nowOrient.BIT6) THEN
            nowOrient.BIT6 = IsLow
          ELSEIF (nowOrient.BIT2) THEN
            nowOrient.BIT2 = IsLow
          ELSE
            nowOrient.BIT0 = IsHigh
          ENDIF
        ELSEIF (nowOrient.BIT4 = IsLow) THEN
```

```
      IF (nowOrient.BIT6) THEN
        nowOrient.BIT6 = IsLow
      ELSEIF (nowOrient.BIT2) THEN
        nowOrient.BIT2 = IsLow
      ELSE
        nowOrient.BIT4 = IsHigh
      ENDIF
    ENDIF

  CASE East

    IF (nowOrient.BIT1 = IsLow) THEN
      IF (nowOrient.BIT7) THEN
        nowOrient.BIT7 = IsLow
      ELSEIF (nowOrient.BIT3) THEN
        nowOrient.BIT3 = IsLow
      ELSE
        nowOrient.BIT1 = IsHigh
      ENDIF
    ELSEIF (nowOrient.BIT5 = IsLow) THEN
      IF (nowOrient.BIT7) THEN
        nowOrient.BIT7 = IsLow
      ELSEIF (nowOrient.BIT3) THEN
        nowOrient.BIT3 = IsLow
      ELSE
        nowOrient.BIT5 = IsHigh
      ENDIF
    ENDIF

  CASE South

    IF (nowOrient.BIT2 = IsLow) THEN
      IF (nowOrient.BIT4) THEN
        nowOrient.BIT4 = IsLow
      ELSEIF (nowOrient.BIT0) THEN
        nowOrient.BIT0 = IsLow
      ELSE
        nowOrient.BIT2 = IsHigh
      ENDIF
    ELSEIF (nowOrient.BIT6 = IsLow) THEN
      IF (nowOrient.BIT4) THEN
        nowOrient.BIT4 = IsLow
      ELSEIF (nowOrient.BIT0) THEN
        nowOrient.BIT0 = IsLow
      ELSE
        nowOrient.BIT6 = IsHigh
      ENDIF
    ENDIF

  CASE West

    IF (nowOrient.BIT3 = IsLow) THEN
      IF (nowOrient.BIT5) THEN
        nowOrient.BIT5 = IsLow
      ELSEIF (nowOrient.BIT1) THEN
        nowOrient.BIT1 = IsLow
      ELSE
        nowOrient.BIT3 = IsHigh
      ENDIF
    ELSEIF (nowOrient.BIT7 = IsLow) THEN
      IF (nowOrient.BIT5) THEN
        nowOrient.BIT5 = IsLow
      ELSEIF (nowOrient.BIT1) THEN
        nowOrient.BIT1 = IsLow
      ELSE
        nowOrient.BIT7 = IsHigh
      ENDIF
    ENDIF

  ENDSELECT

  B25 = nowOrient
  GOSUB Put_Byte

ENDIF

#IF DebugNormal #THEN
  DEBUG "isFinish = ", BIN isFinish, CR, CR
#ENDIF

RETURN

' ----------------------------------------
Put_Byte:
  PUT B24, B25
```

```
  RETURN

' ----------------------------------------
Get_Byte:
  GET B24, B25
  RETURN

'' ----------------------------------------
'Put_Word:
'  PUT B20, Word W9
'  RETURN

'' ----------------------------------------
'Get_Word:
'  GET B20, Word W9
'  RETURN

'' ----------------------------------------
'Update_Destination_Orientation:
'
'    ' INITIALIZING
'    B23 = 0
'
'    B24 = OrientValue + ((tempX * MatrixSize) + tempY)
'    GOSUB Get_Byte
'    B23 = B25
'
'    SELECT tempFaceIn
'      CASE North, South
'        B23.BIT0 = IsHigh
'        B23.BIT2 = IsHigh
'      CASE East, West
'        B23.BIT1 = IsHigh
'        B23.BIT3 = IsHigh
'    ENDSELECT
'
'    B25 = B23
'    GOSUB Put_Byte
'
'  RETURN
```

Kode Program Bank 5 EEPROM

```
' -----[ Initialization ]----------------------------------------------------
'
' -----[ Main Code ]-----------------------------------------------------------
'
' ----------------------------------------
Do_Output_1:

  #IF DebugLow #THEN
    DEBUG CR, "OUTPUT_1:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG IHEX2 ? nowCoord
  #ENDIF

  GOSUB Drive_Motor

' ----------------------------------------
#IF DebugNormal #THEN
Do_Output_2:
  RUN Bank_6
#ENDIF

' ----------------------------------------
Do_Input:
  task = TskDoInput  ' TskDoInput = 1
  RUN Bank_0

' -----[ Subroutines ]---------------------------------------------------------
'
' ----------------------------------------
Drive_Motor:

  #IF DebugLow #THEN
    DEBUG CR, "DRIVE_MOTOR:", CR
  #ENDIF

  #IF DebugNormal #THEN
    DEBUG "isCheckpoint = ", BIN isCheckpoint, "; "
  #ENDIF

  IF (isCheckpoint) THEN

    #IF DebugNormal #THEN
      DEBUG IBIN4 ? moveType
    #ENDIF

    SELECT moveType
      CASE GoStraight
        GOSUB Go_Straight
      CASE RotateRight90
        GOSUB Go_Rotate_Right_90
        GOSUB Go_Straight
      CASE RotateLeft90
        GOSUB Go_Rotate_Left_90
        GOSUB Go_Straight
      CASE RotateRight180
        GOSUB Go_Rotate_Right_180
        GOSUB Go_Straight
    ENDSELECT

    GOSUB Scan_Side_Wall

  ELSE

    #IF DebugNormal #THEN
      DEBUG "wallSensors[65-21] = ", BIN sensor6, BIN sensor5, "-", BIN sensor2, BIN sensor1, CR
    #ENDIF

    IF (sensor6 = isHigh AND sensor5 = isHigh) OR (sensor2 = isHigh AND sensor1 = isHigh) THEN
      GOSUB Go_Forward
    ELSEIF (sensor6 = isLow AND sensor5 = isHigh) OR (sensor2 = isLow AND sensor1 = isHigh) THEN
      GOSUB Pivot_Right
    ELSEIF (sensor6 = isHigh AND sensor5 = isLow) OR (sensor2 = isHigh AND sensor1 = isLow) THEN
      GOSUB Pivot_Left
    ELSE
      GOSUB Stop_Motor
    ENDIF

  ENDIF

  RETURN
```

```
' ----------------------------------------
Go_Straight:
  GOSUB Set_Go_Straight_Duration
  FOR B21 = 1 TO Loop4GoStraight
    GOSUB Pulsout_Motor
    PAUSE Pause4GoStraight
  NEXT
  RETURN

' ----------------------------------------
Set_Go_Straight_Duration:
  W12 = StopMotor + GoStraightFactor
  W11 = StopMotor - GoStraightFactor
  RETURN

' ----------------------------------------
Go_Rotate_Right_90:
  GOSUB Set_Go_Rotate_Right_90_Duration
  FOR B21 = 1 TO Loop4Rotate90
    GOSUB Pulsout_Motor
    PAUSE Pause4Rotate90
  NEXT
  RETURN

' ----------------------------------------
Set_Go_Rotate_Right_90_Duration:
  W12 = StopMotor + RotateFactor
  W11 = StopMotor + RotateFactor
  RETURN

' ----------------------------------------
Go_Rotate_Left_90:
  GOSUB Set_Go_Rotate_Left_90_Duration
  FOR B21 = 1 TO Loop4Rotate90
    GOSUB Pulsout_Motor
    PAUSE Pause4Rotate90
  NEXT
  RETURN

' ----------------------------------------
Set_Go_Rotate_Left_90_Duration:
  W12 = StopMotor - RotateFactor
  W11 = StopMotor - RotateFactor
  RETURN

' ----------------------------------------
Go_Rotate_Right_180:
  GOSUB Set_Go_Rotate_Right_90_Duration
  FOR B21 = 1 TO Loop4Rotate180
    GOSUB Pulsout_Motor
    PAUSE Pause4Rotate180
  NEXT
  RETURN

' ----------------------------------------
Scan_Side_Wall:

  #IF DebugLow #THEN
    DEBUG CR, "SCAN_SIDE_WALL:", CR
  #ENDIF

  ' INITIALIZING
  scanResult = IsLow

  GOSUB Read_Wall_Sensors

  IF NOT (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN

    IF NOT (sensor2 OR sensor1) THEN

      #IF DebugNormal #THEN
        DEBUG "Right wing", CR
        DEBUG ? sensor2
        DEBUG ? sensor1
      #ENDIF

      IF (sensor0 OR sensor8) THEN

        #IF DebugNormal #THEN
          DEBUG "Scan right", CR
          DEBUG ? sensor8
          DEBUG ? sensor0
        #ENDIF

        ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
```

```
    'GOSUB Pivot_Right_Scanning

    GOSUB Set_Pivot_Right_Duration
    FOR B21 = 1 TO Loop4PivotRightScan
      GOSUB Pulsout_Motor
      PAUSE Pause4Scan
      GOSUB Read_Wall_Sensors
      IF (sensor2 OR sensor1) THEN
        scanResult = IsHigh
        EXIT
      ENDIF
    NEXT

  ELSEIF (sensor9) THEN

    #IF DebugNormal #THEN
      DEBUG "Scan left", CR
      DEBUG ? sensor9
    #ENDIF

    ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
    'GOSUB Pivot_Left_Scanning

    GOSUB Set_Pivot_Left_Duration
    FOR B21 = 1 TO Loop4PivotLeftScan
      GOSUB Pulsout_Motor
      PAUSE Pause4Scan
      GOSUB Read_Wall_Sensors
      IF (sensor2 OR sensor1) THEN
        scanResult = IsHigh
        EXIT
      ENDIF
    NEXT

  ENDIF

ENDIF

IF (scanResult = IsLow) AND (NOT (sensor6 OR sensor5)) THEN

  #IF DebugNormal #THEN
    DEBUG "Left wing", CR
    DEBUG ? scanResult
    DEBUG ? sensor6
    DEBUG ? sensor5
  #ENDIF

  IF (sensor7 OR sensor11) THEN

    #IF DebugNormal #THEN
      DEBUG "Scan left", CR
      DEBUG ? sensor7
      DEBUG ? sensor11
    #ENDIF

    ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
    'GOSUB Pivot_Left_Scanning

    GOSUB Set_Pivot_Left_Duration
    FOR B21 = 1 TO Loop4PivotLeftScan
      GOSUB Pulsout_Motor
      PAUSE Pause4Scan
      GOSUB Read_Wall_Sensors
      IF (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN
        scanResult = IsHigh
        EXIT
      ENDIF
    NEXT

  ELSEIF (sensor10) THEN

    #IF DebugNormal #THEN
      DEBUG "Scan right", CR
      DEBUG ? sensor10
    #ENDIF

    ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
    'GOSUB Pivot_Right_Scanning

    GOSUB Set_Pivot_Right_Duration
    FOR B21 = 1 TO Loop4PivotRightScan
      GOSUB Pulsout_Motor
      PAUSE Pause4Scan
      GOSUB Read_Wall_Sensors
      IF (sensor6 OR sensor5) THEN
```

```
          scanResult = IsHigh
          EXIT
        ENDIF
      NEXT

    ENDIF

  ENDIF

  IF (scanResult = IsLow) THEN

    ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
    'GOSUB Pivot_Left_Scanning

    GOSUB Set_Pivot_Left_Duration
    FOR B21 = 1 TO Loop4PivotLeftScan
      GOSUB Pulsout_Motor
      PAUSE Pause4Scan
      GOSUB Read_Wall_Sensors
      IF (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN
        scanResult = IsHigh
        EXIT
      ENDIF
    NEXT

    IF (scanResult = IsLow) THEN

      ' CAN NOT USE ROUTINE BECAUSE OF STRANGE BEHAVIOUR
      'GOSUB Pivot_Right_Scanning

      GOSUB Set_Pivot_Right_Duration
      FOR B21 = 1 TO Loop4PivotRightScan
        GOSUB Pulsout_Motor
        PAUSE Pause4Scan
        GOSUB Read_Wall_Sensors
        IF (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN
          scanResult = IsHigh
          EXIT
        ENDIF
      NEXT

    ENDIF

    ENDIF

  #IF DebugNormal #THEN
  ELSE
    DEBUG "No need", CR
  #ENDIF

  ENDIF

  RETURN

' ----------------------------------------
Read_Wall_Sensors:

  #IF DebugLow #THEN
    DEBUG CR, "READ_WALL_SENSORS:", CR
  #ENDIF

  B20 = SensorThres
  GOSUB Get_Word

  HIGH OuterRightSensor      ' Turn on sensor
  GOSUB Change_IO_Direction
  sensor0 = SensorInput      ' Snapshot of sensor signal states
  LOW OuterRightSensor       ' Turn off sensor

  HIGH FarMiddleRightSensor
  GOSUB Change_IO_Direction
  sensor1 = SensorInput
  LOW FarMiddleRightSensor

  HIGH NearMiddleRightSensor
  GOSUB Change_IO_Direction
  sensor2 = SensorInput
  LOW NearMiddleRightSensor

  'HIGH InnerRightSensor
  'GOSUB Change_IO_Direction
  'sensor3 = SensorInput
  'LOW InnerRightSensor
```

```
  'HIGH InnerLeftSensor
  'GOSUB Change_IO_Direction
  'sensor4 = SensorInput
  'LOW InnerLeftSensor

  HIGH NearMiddleLeftSensor
  GOSUB Change_IO_Direction
  sensor5 = SensorInput
  LOW NearMiddleLeftSensor

  HIGH FarMiddleLeftSensor
  GOSUB Change_IO_Direction
  sensor6 = SensorInput
  LOW FarMiddleLeftSensor

  HIGH OuterLeftSensor
  GOSUB Change_IO_Direction
  sensor7 = SensorInput
  LOW OuterLeftSensor

  HIGH OuterRearRightSensor
  GOSUB Change_IO_Direction
  sensor8 = SensorInput
  LOW OuterRearRightSensor

  HIGH InnerRearRightSensor
  GOSUB Change_IO_Direction
  sensor9 = SensorInput
  LOW InnerRearRightSensor

  HIGH InnerRearLefttSensor
  GOSUB Change_IO_Direction
  sensor10 = SensorInput
  LOW InnerRearLefttSensor

  HIGH OuterRearLeftSensor
  GOSUB Change_IO_Direction
  sensor11 = SensorInput
  LOW OuterRearLeftSensor

  #IF DebugNormal #THEN
   DEBUG "wallSensors = ", BIN4 wallSensors.NIB2, "-", BIN sensor7, BIN sensor6, BIN sensor5, "x", "-", "x", BIN
    sensor2, BIN sensor1, BIN sensor0, CR
  #ENDIF

  RETURN

' ----------------------------------------
Change_IO_Direction:
  HIGH SensorInput              ' Push signal voltages to 5 V
  PAUSE 0                     ' Wait 230 us for capacitors
  INPUT SensorInput             ' Start the decays
  'PULSOUT UnusedPin, sensorThreshold  ' Wait for threshold time
  PULSOUT UnusedPin, W9             ' Wait for threshold time
  RETURN

' ----------------------------------------
Go_Forward:
  GOSUB Set_Go_Forward_Duration
  FOR B21 = 1 TO Loop4GoForward
    GOSUB Pulsout_Motor
    PAUSE Pause4GoForward
  NEXT
  RETURN

' ----------------------------------------
Set_Go_Forward_Duration:
  W12 = StopMotor + GoForwardFactor
  W11 = StopMotor - GoForwardFactor
  RETURN

' ----------------------------------------
Pivot_Right:
  GOSUB Set_Pivot_Right_Duration
  FOR B21 = 1 TO Loop4Pivot
    GOSUB Pulsout_Motor
    PAUSE Pause4Pivot
  NEXT
  RETURN

' ----------------------------------------
Set_Pivot_Right_Duration:
  W12 = StopMotor + PivotFactor
  W11 = StopMotor
  RETURN
```

```
' ----------------------------------------
Pivot_Left:
  GOSUB Set_Pivot_Left_Duration
  FOR B21 = 1 TO Loop4Pivot
    GOSUB Pulsout_Motor
    PAUSE Pause4Pivot
  NEXT
  RETURN

' ----------------------------------------
Set_Pivot_Left_Duration:
  W12 = StopMotor
  W11 = StopMotor - PivotFactor
  RETURN

' ----------------------------------------
Stop_Motor:
  GOSUB Set_Stop_Motor_Duration
  GOSUB Pulsout_Motor
  RETURN

' ----------------------------------------
Set_Stop_Motor_Duration:
  W12 = StopMotor
  W11 = StopMotor
  RETURN

' ----------------------------------------
Pulsout_Motor:
  PULSOUT LeftMotor, W12
  PULSOUT RightMotor, W11
  RETURN

' ----------------------------------------
Get_Word:
  GET B20, Word W9
  RETURN

'' ----------------------------------------
'Pivot_Left_Scanning:
'
'  GOSUB Set_Pivot_Left_Duration
'
'  FOR B20 = 1 TO Loop4PivotLeftScan
'    GOSUB Pulsout_Motor
'    PAUSE Pause4Scan
'    GOSUB Read_Wall_Sensors
'    IF (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN
'      scanResult = IsHigh
'      EXIT
'    ENDIF
'  NEXT
'
'  RETURN

'' ----------------------------------------
'Pivot_Right_Scanning:
'
'  GOSUB Set_Pivot_Right_Duration
'
'  FOR B20 = 1 TO Loop4PivotRightScan
'    GOSUB Pulsout_Motor
'    PAUSE Pause4Scan
'    GOSUB Read_Wall_Sensors
'    IF (sensor6 OR sensor5 OR sensor2 OR sensor1) THEN
'      scanResult = IsHigh
'      EXIT
'    ENDIF
'  NEXT
'
'  RETURN
```

Kode Program Bank 6 EEPROM

```
' -----[ Initialization ]------------------------------------------------------
'
' -----[ Main Code ]-------------------------------------------------------------
'
' ----------------------------------------
Do_Output_2:

  #IF DebugLow #THEN
    DEBUG CR, "OUTPUT_2:", CR
  #ENDIF

  #IF DebugNormal #THEN
    GOSUB Display_Data
  #ENDIF

' ----------------------------------------
Do_Input:
  task = TskDoInput  ' TskDoInput = 1
  RUN Bank_0

' -----[ Subroutines ]-----------------------------------------------------------
'
' ----------------------------------------
Display_Data:
  #IF DebugLow #THEN
    DEBUG CR, "DISPLAY_DATA:", CR
  #ENDIF

  #IF DebugNormal #THEN
    'DEBUG CLS
    GOSUB Parameter_Reading
    GOSUB Maze_Reading
  #ENDIF

  RETURN

' ----------------------------------------
Parameter_Reading:

  #IF DebugNormal #THEN

    DEBUG "[x,y] = [", DEC nowX, ",", DEC nowY, "]", "; ", "nowDist = ", DEC nowDist, "; ", "nowWall = ", BIN4
      nowWall.NIB1, "-", BIN4 nowWall.NIB0, "; ", "nowOrient = ", BIN4 nowOrient.NIB1, "-", BIN4 nowOrient.NIB0, CR
    DEBUG "isVisitedCell = ", BIN isVisitedCell, "; ", "isCheckpoint = ", BIN isCheckpoint, "; ", "isFinish = ", BIN isFinish,
      "; ", "isBrokenRule = ", BIN isBrokenRule, "; ", CR
    'DEBUG "sensorThreshold = ", DEC sensorThreshold, "; ", "numOfOpenDist = ", DEC numOfOpenDist, "; ", "openDist
      = ", IBIN4 openDist, "; ", "moveType = ", IBIN4 moveType, "; ", "scanResult = ", BIN scanResult, CR
    DEBUG "numOfOpenDist = ", DEC numOfOpenDist, "; ", "openDist = ", IBIN4 openDist, "; ", "moveType = ", IBIN4
      moveType, "; ", "scanResult = ", BIN scanResult, CR

  #ENDIF

  RETURN

' ----------------------------------------
Maze_Reading:

  #IF DebugNormal #THEN

    DEBUG CR, "MAZE: COORDINATE + DISTANCE + VISITED-WALL", CR

    FOR B23 = MatrixSizeMinSatu TO 0
      FOR B22 = 0 TO MatrixSizeMinSatu

        B21 = (B22 * MatrixSize) + B23
        B24 = B21 + CoordinateValue
        GOSUB Get_Byte
        DEBUG HEX2 B25, " + "
        B24 = B21 + DistanceValue
        GOSUB Get_Byte
        DEBUG DEC B25, " + "
        B24 = B21 + WallValue
        GOSUB Get_Byte
        DEBUG BIN4 B25.NIB1, "-", BIN4 B25.NIB0, " | "

      NEXT
      DEBUG CR
    NEXT

    DEBUG CR, "MAZE: COORDINATE + DISTANCE + ORIENTATION", CR
```

```
  FOR B23 = MatrixSizeMinSatu TO 0
    FOR B22 = 0 TO MatrixSizeMinSatu

      B21 = (B22 * MatrixSize) + B23
      B24 = B21 + CoordinateValue
      GOSUB Get_Byte
      DEBUG HEX2 B25, " + "
      B24 = B21 + DistanceValue
      GOSUB Get_Byte
      DEBUG DEC B25, " + "
      B24 = B21 + OrientValue
      GOSUB Get_Byte
      DEBUG BIN4 B25.NIB1, "-", BIN4 B25.NIB0, " | "

    NEXT
    DEBUG CR
  NEXT

 #ENDIF

 RETURN

' --------------------------------------
Get_Byte:
 GET B24, B25
 RETURN
```