ANALISIS PERBANDINGAN ALGORITMA *PLEDGE* DENGAN ALGORITMA *WALL FOLLOWER* PADA ROBOT *WALL MAZE*

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

oleh :

TRI MARYANI
NIM : 10751000243

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2013

ANALISIS PERBANDINGAN ALGORITMA *PLEDGE* DENGAN ALGORITMA *WALL FOLLOWER* PADA ROBOT *WALL MAZE*

TRI MARYANI
10751000243

Tanggal Sidang : 7 Januari 2013
Periode Wisuda : Februari 2013

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau
Jl. Soebrantas Km 15 No. 155 Pekanbaru

ABSTRAK

Kecerdasan buatan dalam bidang robotika adalah suatu algoritma cerdas yang diprogramkan ke dalam kontroler robot. Kecerdasan ini diperlukan robot untuk membantu manusia menjalankan suatu fungsi tertentu secara otomatis dan mandiri. Fungsi yang dilakukan oleh robot melalui kecerdasan buatan dalam tugas akhir ini adalah melakukan penelusuran *maze* dari titik *start* sampai menemukan titik *finish*. Kecerdasan buatan diimplementasikan menggunakan algoritma *pledge,* yaitu sebuah algoritma pencari jalur dengan mempertimbangkan arah utama dan prioritas penelusuran yaitu *right wall priority* dan *left wall priority*. Algoritma *wall follower* juga digunakan dalam tugas akhir ini, algoritma ini digunakan sebagai pembanding kinerja terhadap algoritma *pledge*. Algoritma *wall follower* memiliki dua konsep penelusuran, yaitu *right hand rule* dan *left hand rule.* Pengujian yang dilakukan meliputi pengujian algoritma, pengujian terhadap *Radiation cone* (radiasi sensor), pengujian terhadap *maze*, pengujian terhadap jarak sensor dan pengujian waktu tempuh ke titik *finish.* Dari pengujian yang dilakukan dapat ditarik kesimpulan bahwa tingkat keberhasilan dalam penelusuran *maze* sebesar 90%, karena pada pengujian dengan posisi *finish* diluar lintasan robot tidak dapat berhenti tepat pada titik *finish*. Perangkat lunak dalam tugas akhir ini dirancang dengan menggunakan bahasa pemrograman Basic dan menggunakan simulator Mobotsim V.10.

Kata kunci : kecerdasan buatan, *left hand rule, left wall priority, maze, pledge, radiation cone, right hand rule, right wall priority,* robot, *wall follower*

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR TABEL

Tabel

DAFTAR ISTILAH

Artificial Intelligence	:	Bidang ilmu komputer (*computer science*) yang khusus ditujukan untuk membuat perangkat lunak dan perangkat keras yang sepenuhnya bisa menirukan beberapa fungsi otak manusia.
Basic	:	Singkatan dari *Beginners All-purpose Symbolic Instruction Code* yang merupakan sebuah kelompok bahasa pemrograman tingkat tinggi.
Left hand rule	:	Aturan tangan kiri pada algoritma *wall follower,* dengan melakukan penelusuran mengikuti dinding kiri.
Left wall priority	:	Pemberian nama untuk prioritas kiri pada algoritma *pledge.*
Line maze	:	Penelusuran labirin berupa garis.
Mark	:	Tanda dalam Mobotsim yang merupakan pasangan koordinat X dan Y yang dapat digunakan sebagai acuan visual, misalnya sebagai titik sasaran yang harus dicapai oleh robot.
Maze	:	Tempat yang penuh dengan jalan dan lorong yang berliku-liku dan simpang siur atau sesuatu yang sangat rumit dan berbelit-belit.
Mobile robot	:	Robot yang dapat berpindah dari tempatnya menuju ke tempat lain.
Mobotsim V.10	:	Sebuah perangkat lunak untuk simulasi 2D robot roda (*mobile Robot* atau MOBOT)
Photodioda	:	Sebuah dioda (komponen elektronika semi konduktor) yang apabila dikenai cahaya akan memancarkan elektron sehingga akan mengalirkan arus listrik.
Pledge	:	Algoritma pencarian jalur yang didesain untuk rintangan melingkar dan memiliki arah awal untuk bergerak maju.
Radiation cone	:	Radiasi sensor dengan pendekatan kerucut (*cone*)

Right hand rule	:	Aturan tangan kanan pada algoritma *wall follower,* dengan melakukan penelusuran mengikuti dinding kanan.
Right wall priority	:	Pemberian nama untuk prioritas kanan pada algoritma *pledge.*
Robot	:	Alat mekanika yang dapat melakukan tugas fisik, baik lewat pantauan manusia, maupun bekerja secara komputerisasi yang menghasilkan kecerdasan dan perilaku yang individu.
Sensor	:	*Device* atau komponen elektronika yang digunakan untuk merubah besaran fisik menjadi besaran listrik sehingga bisa dianalisa dengan menggunakan rangkaian listrik.
Ultrasonik	:	Sebuah sensor yang mengubah besaran fisis (bunyi) menjadi besaran listrik.
Wall follower	:	Algoritma pencarian jalur yang akan mencari jalan sesuai dengan dinding labirin, baik itu ke kiri maupun ke kanan.
Wall maze	:	Penelusuran labirin berupa dinding.

DAFTAR ALGORITMA

Algoritma

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kecerdasan buatan merupakan bidang ilmu yang baru berkembang pada tahun 90-an dan aplikasinya banyak diterapkan dalam berbagai bidang, mulai dari *games* komputer, sistem kontrol cerdas, robotik, sistem pakar. Pada bidang robotik, kecerdasan buatan banyak diaplikasikan pada robot-robot tertentu dengan keistimewaan yang khusus dan berkaitan erat dengan kebutuhan dalam bidang industri modern yang menuntut adanya suatu instrumen dengan kemampuan yang tinggi yang dapat membantu menyelesaikan pekerjaan manusia atau untuk menyelesaikan pekerjaan yang tidak mampu diselesaikan oleh manusia.

Salah satu robot yang belakangan ini banyak menarik minat para ahli untuk dikembangkan adalah *mobile robot.* Menurut Hartanto (2005) *mobile robot* merupakan robot yang dapat berpindah dari tempatnya menuju tempat lain. *Mobile robot* menyerupai fungsi makhluk hidup yang dapat berpindah, jenis robot ini biasanya diciptakan untuk berbagai keperluan, seperti: mengangkut barang secara otomatis, melakukan perjalanan atau pemantauan ke tempat-tempat berbahaya, sebagai alat hiburan (*robotainment*) atau mainan.

Kemampuan dari *mobile robot* sangat beragam sesuai dengan tingkat dan jenis keperluan. *Mobile robot* terdiri dari robot *line maze* (penelusur garis) dan robot *wall maze* (penelusur dinding). Kedua robot ini merupakan jenis robot beroda yang memiliki sensor berupa garis untuk *line maze* dan suatu dinding untuk *wall maze* dengan sirkuit tertentu kemudian bergerak menelusuri garis atau dinding tersebut dengan modus tertentu. Kedua robot ini biasa digunakan dalam bidang industri besar yang dianggap berbahaya atau beresiko tinggi jika manusia yang melakukan hal tersebut atau bahkan justru manusia tidak mampu untuk melakukan hal tersebut dengan cara lain.

Dalam penerapannya robot *line maze* memiliki kemungkinan *error* yang lebih besar dibandingkan dengan robot *wall maze*. Robot *line maze* menggunakan sensor photodioda yang memiliki cara kerja dengan pantulan cahaya untuk melakukan penelusuran garis, sehingga jika pengaruh dari cahaya disekitar garis kurang, maka robot akan sulit mendeteksi keberadaan garis untuk bergerak maju. Kelemahan lain yang dimiliki oleh robot *line maze* adalah jika robot keluar dari lintasan garis yang dalam kondisi tertentu robot berada terlalu jauh dari garis maka kemungkinan *error* yang ditimbulkan sangat besar, sehingga diperlukan bantuan manual untuk mengembalikannya ke jalur sebelumnya. Sedangkan robot *wal maze* bekerja dengan menggunakan sensor *ultrasonik* yang bisa memantulkan sensor dari semua objek yang dibaca oleh sensor dan tidak terbatas hanya pada dinding. Robot *wall maze* pada umumnya digunakan untuk perjalanan atau pemantauan suatu tempat tertentu, dengan demikian robot *wall maze* harus dapat memahami dengan baik keberadaan lingkungan tempat robot tersebut berada dan harus memberikan respon terhadap keberadaan lingkungannya tersebut dengan baik. Lingkungan yang dimaksud dapat berupa labirin (*maze*).

Pemecahan *maze* dapat dilakukan dengan berbagai metode atau algoritma, diantaranya algoritma *Depth First Search, Maze Mapping, Flood Fill, Wall Follower* dan *Pledge.* Metode atau algoritma ini digunakan sebagai sistem kendali yang bisa membuat robot mampu melewati *maze* dengan baik dan dengan tingkat *error* seminimal mungkin. Dengan demikian, waktu yang ditempuh untuk mencapai tujuan menjadi lebih efektif sesuai dengan konsep dari masing-masing algoritma yang nantinya digunakan.

Terkait penelitian yang dilakukan sebelumnya mengenai penerapan algoritma *pledge* untuk menyelesaikan *maze* pada robot *line follower* atau *line maze* yang dilakukan oleh Arif Darmawan dan kawan-kawan (2010). Dalam penelitiannya, algoritma *pledge* dikembangkan dengan memberikan dua prioritas penelusuran yaitu *right wall priority* (prioritas dinding kanan) dan *left wall priority* (prioritas dinding kiri) hal ini dilakukan untuk memudahkan penelusuran ketika menemukan persimpangan berbentuk T pada *maze* yang berbentuk garis.

Berkaitan dengan hal tersebut di atas, tugas akhir ini mencoba untuk mengimplementasikan suatu jenis kecerdasan-buatan robot *wall maze* yang digunakan dalam pencarian jalur menuju ke tujuan yang diinginkan di suatu lingkungan, yaitu lingkungan labirin (*maze*) berbentuk dinding atau ruangan dengan *start* awal robot di salah satu sudut labirin dan tujuan berada di sudut yang lain dengan menggunakan algoritma pencarian jalur yaitu algoritma *pledge*. Untuk menguji kinerja algoritma *pledge* tugas akhir ini juga melakukan analisa terhadap algoritma *wall follower* yang nantinya dijadikan pembanding terhadap algoritma *pledge.*

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan sebelumnya, maka permasalahan yang akan dibahas dalam penelitian ini adalah, ”Bagaimana menganalisa perbandingan antara algoritma *pledge* dengan algoritma *wall follower* pada robot *wall maze*”.

1.3 Batasan Masalah

Batasan yang digunakan dalam penelitian ini agar pembahasan yang dilakukan tidak meluas yaitu:

1. Pembahasan hanya pada kecerdasan buatan yang diterapkan pada robot *wall maze*, tidak membahas tentang perancangan perangkat robot.
2. Pengujian dilakukan dalam bentuk simulasi dengan menggunakan simulator Mobotsim V.10.

1.4 Tujuan Penelitian

Tujuan penelitian tugas akhir ini adalah:

1. Melakukan analisa terhadap kecerdasan buatan yang dirancang pada robot *wall maze* untuk menyelesaikan sebuah *maze* dengan menganalisa perbandingan algoritma *pledge* dan algortima *wall follower,* agar sebuah robot *wall maze* dapat mencapai titik *finish* pada sebuah *maze* dan memiliki kemampuan untuk menentukan jalur yang lebih efektif dan efisien.

2. Dikatakan efektif karena keakuratan penelusuran berdasarkan masing-masing algoritma dalam menghindari rintangan dan lebih efisien karena waktu tempuh penelusuran dari masing-masing algoritma yang nantinya dapat ditentukan jalur terpendek menuju titik *finish*.

1.5 Sistematika Penulisan

Laporan Tugas Akhir ini terdiri dari enam bab yang disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi tentang Latar Belakang pelaksanaan penelitian, Rumusan Masalah yang dihadapi, Batasan yang digunakan, Tujuan Tugas Akhir yang hendak dicapai melalui penelitian ini serta Sistematika Penulisan.

BAB II. LANDASAN TEORI

Bab ini membahas teori-teori yang berhubungan dengan pembahasan penelitian yang diangkat, yang terdiri dari pembahasan mengenai teori kecerdasan buatan (*Artificial Intelligence)*, robotika, labirin (*maze),* algoritma pencarian jalur dan simulator Mobotsim.

BAB III. METODOLOGI PENELITIAN

Pada bab ini dijelaskan mengenai tahapan dalam pelaksanaan penelitian tugas akhir. Tahapan penelitian tugas akhir dimulai dari pengumpulan data, identifikasi dan perumusan masalah, analisa, perancangan, implementasi dan pengujian hingga diperoleh kesimpulan akhir dari penelitian

BAB IV. ANALISA DAN PERANCANGAN

Bab ini berisi tentang analisa algoritma yaitu algoritma *Pledge* dan algoritma *Wall Follower,* analisa simulasi dan setelah analisa selesai maka dilakukan perancangan.

BAB V. IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi penjelasan mengenai implementasi yang terdiri dari: tahapan implementasi yaitu batasan implementasi, lingkungan implementasi dan implementasi *interface* perangkat simulasi, kemudian dilanjutkan dengan tahapan pengujian dan kesimpulan pengujian.

BAB VI. PENUTUP

Bab ini berisi kesimpulan yang dihasilkan dari pembahasan tentang Analisis perbandingan antara algoritma *pledge* dengan algoritma *wall follower* pada robot *wall maze* dan beberapa saran sebagai hasil akhir dari penelitian yang telah dilakukan.

BAB II

LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan Buatan *(Artificial Intelligence)* merupakan salah satu bidang ilmu komputer (*computer science*) yang khusus ditujukan untuk membuat perangkat lunak dan perangkat keras yang sepenuhnya bisa menirukan beberapa fungsi otak manusia. Atau cabang ilmu komputer yang mempelajari otomatisasi tingkah laku cerdas (*intelligent*).

Kecerdasan harus didasarkan pada prinsip-prinsip teoritikal dan terapan yang menyangkut:

1. Struktur data yang digunakan dalam representasi pengetahuan (*knowledge representation*)
2. Algoritma yang diperlukan dalam penerapan pengetahuan itu
3. Teknik-teknik bahasa dan pemrograman yang dipakai dalam implementasinya.

2.1.1 Definisi Kecerdasan Buatan

Stuart Russel dan Peter Norvig mengelompokkan difinisi Kecerdasan buatan yang diperoleh dari beberapa *textbook* berbeda, kedalam empat kategori (Russel, Norvig, 1995), yaitu:

1. ***Thinking humanly****: the cognitive modeling approach*

   Pendekatan ini dilakukan dengan dua cara sebagai berikut:

   a. Melalui introspeksi: mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berpikir. Tetapi, seorang psikolog Barat mengatakan: *"how do you know that you understand?"* bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari

pemikiran anda, ternyata pemikiran tersebut sudah lewat dan digantikan kesadaran anda. Sehingga, defenisi ini terkesan mengada-ada dan tidak mungkin dilakukan.

b. Melalui eksperimen-eksperimen psikologi.

2. ***Acting humanly****: the Turing test approach*

Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berintelejensia untuk menguji apakah komputer tersebut mampu mengelabuhi seorang manusia yang menginterogasinya melalui *teletype* (komunikasi berbasis teks jarak jauh). Jika *interrogator* tidak membedakan yang diinterogasi adalah manusia atau komputer, maka komputer berintelejensia tersebut lolos dari *Turing test*. Komputer tersebut perlu memiliki kemampuan *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics. Turing test* sengaja menghindari interaksi fisik antara *interrogator* dan komputer karena simulasi fisik manusia tidak memerlukan intelejensia.

3. ***Thinking rationally****: the laws of thought approach*

Terdapat dua masalah dalam pendekatan ini, yaitu:

a. Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam *formal term* yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.

b. Terdapat perbedaan besar antara dapat memecahkan masalah “dalam prinsip” dan memecahkannya “dalam dunia nyata”.

4. ***Acting rationally****: the rational agent approach*

Membuat inferensi yang logis merupakan bagian dari suatu *rational agent.* Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka

bisa didapatkan kesimpulakan bahwa aksi yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka *agent* dapat melakukan aksi berdasarkan kesimpulan tersebut.

Thinking humanly dan *acting humanly* adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini, maka banyak produk komputasi cerdas saat ini yang tidak layak disebut produk kecerdasan buatan (Suyanto, 2007).

Definisi *thinking rationally* terasa lebih sempit daripada *acting rationally.* Oleh karena itu, definisi kecerdasan buatan yang paling tepat untuk saat ini adalah *acting rationally* dengan pendekatan *rational agent.* Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut (Suyanto, 2007).

Kecerdasan buatan menawarkan baik media maupun uji teori kecerdasan. Teori-teori semacam ini dapat dinyatakan dalam bahasa program komputer dan dibuktikan melalui eksekusinya pada komputer.

2.1.2 Bagian-Bagian Kecerdasan Buatan

Bagian utama aplikasi kecerdasan buatan adalah pengetahuan (*knowledge*), yaitu suatu pengertian tentang beberapa wilayah subyek yang diperoleh melalui pendidikan dan pengalaman. Pengetahuan merupakan informasi terorganisir dan teranalisa agar bisa lebih mudah dimengerti dan bisa diterapkan pada pemecahan masalah dan pengambilaan keputusan. Pengetahuan terdiri dari fakta, pemikiran, teori, prosedur, dan hubungannya satu sama lain.

Komputer tidak mungkin mendapatkan pengetahuannya sendiri dengan belajar, berpengalaman atau melakukan penelitian, akan tetapi ia memperolehnya melalui upaya yang diberikan oleh seorang pakar manusia.

Hampir semua basis pengetahuan (*knowledge base*) sangat terbatas, dalam arti terfokuskan kepada suatu masalah khusus. Pada saat basis pengetahuan itu sudah terbentuk, teknik Kecerdasan Buatan bisa digunakan untuk memberi kemampuan baru kepada komputer agar bisa berfikir, menalar, dan membuat *inferensi* (mengambil keputusan berdasarkan pengalaman) dan membuat pertimbangan-pertimbangan yang didasarkan kepada fakta dan hubungan-hubungannya yang terkandung dalam basis pengetahuan itu.

Dengan basis pengetahuan dan kemampuan untuk menarik kesimpulan melalui pengalaman (inferensi), komputer dapat disejajarkan sebagai alat bantu yang bisa digunakan secara praktis dalam memecahkan masalah dan pengambilan keputusan serta bisa mencapai satu atau lebih solusi alternatif pada masalah yang diberikan.

Bidang-bidang teknik kecerdasan buatan diantaranya adalah:

1. sistem pakar (*expert system*),
2. robot (*robotics*),
3. logika samar (*fuzzy logic*),
4. jaringan syaraf (*neural networks*) tiruan, dan
5. pengolahan bahasa alami (*natural language processing*)
6. pengolahan citra.

2.1.3 Faktor Pendorong Perkembangan Kecerdasan Buatan

Faktor pendorong bagi terlaksananya aplikasi Kecerdasan Buatan adalah:

1. Pesatnya perkembangan teknologi perangkat keras.

   Hampir semua aplikasi Kecerdasan Buatan memerlukan perangkat keras yang memiliki kecepatan daya tampung yang lebih tinggi, walaupun hanya menjalankan perangkat lunak Kecerdasan Buatan yang paling sederhana sekalipun. Disamping itu, harga perangkat keras yang dengan kemampuan lebih memiliki harga yang relatif semakin murah.

2. Pengembangan perangkat lunak Kecerdasan Buatan

   Dewasa ini bahasa dan alat pemrograman baru yang lebih canggih sudah banyak dikembangkan dan dipasarkaan secara luas, termasuk bahasa khusus untuk Kecerdasan Buatan.

3. Perkembangan khusus komputer pribadi (*personal computer*/PC).

   Sekarang sudah sangat banyak orang menggunakan komputer mikro (*microcomputer*) khususnya komputer pribadi baik di sekolah, perusahaan, atau bahkan di rumah yang menyebabkan permintaan mereka akan perangkat lunak yang lebih unggul untuk pekerjaan mereka.

4. Turut andilnya para investor dalam mendanai penelitian dan pengembangan teknologi Kecerdasan Buatan.

Hal ini mengakibatkan terjadinya semacam tekanan di kalangan masyarakat kecerdasan buatan untuk berlomba-lomba dalam mempercepat gerak dan langkah penelitiannya dan segera memproduksi kecerdasan buatan dalam waktu yang singkat. Masalah utama Kecerdasan Buatan adalah sulitnya merumuskan dan memvisualisasi inteligensia itu sendiri, karena mempunyai arti yang banyak.

Walaupun Kecerdasan Buatan telah banyak membuat komputer menjadi lebih pintar dan lebih canggih, tapi tampaknya impian manusia agar bisa membuat komputer yang betul-betul bisa membuat duplikasi otak manusia, atau bisa menjadi pengganti otak manusia yang sebenarnya masih jauh dari kenyataan. Mungkin belum bisa terlaksana pada zaman atau masa kini.

2.2 Robotika

Robotika adalah satu cabang teknologi yang berhubungan dengan desain, konstruksi, operasi, disposisi struktural, pembuatan, dan aplikasi dari robot. Robotika terkait dengan ilmu pengetahuan bidang elektronika, mesin, mekanika, dan perangkat lunak komputer.

2.2.1 Sejarah Robotika

Keunggulan dalam teknologi robotik tak dapat dipungkiri telah lama dijadikan *icon* kebanggan negara-negara maju di dunia. Kecanggihan teknologi yang dimiliki, gedung-gedung tinggi yang mencakar langit, tingkat kesejahteraan rakyatnya yang tinggi, kota-kotanya yang modern, belumlah terasa lengkap tanpa popularitas kepiawaian dalam dunia robotik.

Robot adalah alat menaknika yang dapat melakukan tugas fisik, baik lewat pantauan manusia, maupun bekerja secara komputerisasi yang menghasilkan kecerdasan dan perilaku yang individu.

Kata robot yang berasal dari bahasa Czech, *robota,* yang berarti pekerja, mulai menjadi populer ketika seorang penulis berbangsa Czech (Ceko), Karl Capek, membuat pertunjukan dari lakon komedi yang ditulisnya pada tahun 1921 yang berjudul RUR (Rossum's Universal Robot). Ia bercerita tentang mesin yang menyerupai manusia, tapi mampu bekerja terus-menerus tanpa lelah. (Pitowarno, 2006)

Gambar 2.1 Robot RUR (*Rossum's Universal Robot)*

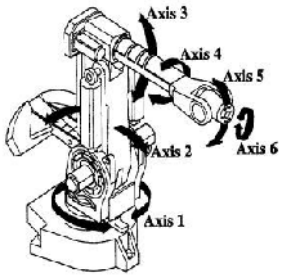
Penciptaan robot yang sesunguhnya (bukan robot dalam sandiwara Karel Capek), bermula dari keinginan manusia untuk membuat tiruan binatang atau manusia itu sendiri. Namun fasilitas pengetahuan pada masa itu nampaknya menjadi kendala terbesar dalam terwujudnya proyek tersebut. Akhirnya, dibuatlah

robot sederhana beroda yang digunakan untuk keperluan navigasi, pengamatan tingkah laku, sampai dengan perencanaan jalur.

Istilah robot ini kemudian memperoleh sambutan dengan diperkenalkannya robot jerman dalam film Metropolis tahun 1926 yang sempat dipamerkan dalam *New York World's Fair* 1939. Film ini mengisahkan tentang robot berjalan mirip manusia beserta hewan peliharaannya (Pitowarno, 2006).

Tahun 1941, barulah istilah *robotics* digunakan dalam teknologi robot oleh penulis fiksi ilmiah Isaac Asimov. Dia juga memprediksi akan munculnya robot-robot industri canggih dimasa datang. Jika kita lihat hari ini, maka apa yang dibayangkan olehnya terbukti dimana begitu pesatnya perkembangan robot-robot industri saat ini. Istilah revolusi robot, *robot age* atau era robot sudah menjadi hal biasa untuk menjelaskan perkembangan itu. *Robotics* diterima sebagai istilah atau kata untuk mendeskripsikan semua kemajuan teknologi yang berhubungan dengan robot.

Pada tahun 1956 Georde Devil dan Joseph Engelberger membentuk perusahaan robot pertama kali tahu 1956. Devil memprediksi robot akan menjadi bagian penting di industri sebagai operator pabrik dan membantu pekerja dalam menjalankan mesin-mesin pabrik. Beberapa tahun kemudian atau tepatnya 1961, General Motor pertama kali menggunakan robot untuk pabrik otomotifnya.



Gambar 2.2 Robot Industri

Dewasa ini mungkin definisi robot industri itu sudah tidak sesuai lagi karena teknologi mobile robot sudah dipakai secara meluas sejak tahun 80-an. Seiring itu pula kemudian muncul istilah robot *humanoid, animaloid*, dan sebagainya. Bahkan kini dalam industri spesifik seperti industri perfilman,

industri angkasa luar dan industri pertahanan atau mesin perang, robot arm atau manipulator bisa jadi hanya menjadi bagian saja dari sistem robot secara keseluruhan (Pitowarno, 2006).

Kembali atas jasa insan film, istilah robot ini makin popular dengan lahirnya robot R2D2 dan C3Po dalam film *Star Wars* pertama pada tahun 1977.

Gambar 2.3 Robot R2D2 dan C3Po dalam film starwars

Menurut Fu et al. (1987) penelitian dan pengembangan pertama yang berbuah produk robotik dapat dilacak mulai dari tahun 1940-an ketika Argone National Laboratories di Oak Ridge, Amerika, memperkenalkan sebuah mekanisme robotik yang dinamai master-slave manipulator. Robot ini digunakan untuk menangani material radioaktif. Kemudian produk pertama robot komersial diperkenalkan oleh *Unimation Incorporated*, Amerika pada tahun 1950-an. Hingga belasan tahun kemudian langkah komersial ini telah diikuti oleh perusahaan-perusahaan lain. Namun demikian, seperti ditulis dalam beberapa sumber, penelitian intensif dibidang teknologi robotika telah menjadikan robotik sebagai sebuah disiplin ilmu kala itu belum terpikirkan.

Di negara-negara yang telah mapan kala itu, seperti Amerika, Inggris, Jerman dan Perancis mulai bermunculan grup-grup riset yang menjadikan robotik sebagai temanya, kemudian diikuti oleh Jepang, yang dipelopori oleh ilmuwan-ilmuwan yang baru pulang dari menimba ilmu di Amerika. Bahkan, di kemudian

hari Jepang-lah yang tercatat sebagai negara yang paling produktif dalam mengembangkan teknologi robot. Hal ini tidak lain karena jepang gigih dalam melakukan penelitian teknologi infrastruktur seperti komponen dan piranti mikro (*microdevices*) yang akhirnya bidang ini terbukti sebagai inti dari pengembangan robot modern.

Gambar 2.4 Robot Asimo milik Honda

2.2.2 Sistem Robot dan Orientasi Fungsi

Sistem robot dan orientasi fungsi dalam dunia robotika meliputi (Pitowarno, 2006):

1. Sistem Kontroler

Adalah rangkaian elektronik yang setidak-tidaknya terdiri dari rangkaian prosesor (CPU, Memori, komponen *interface Input/output*), *signal conditioning* untuk sensor (analog dan atau digital), serta *driver* untuk aktuator. Bila diperlukan bisa dilengkapi dengan sistem monitor seperti *seven segment*, LCD (*liquid crstal display*) ataupun CRT (*cathode ray_tube*).

2. Mekanik Robot

Adalah sistem mekanik yang dapat terdiri dari setidak-tidaknya sebuah sistem gerak. Jumlah fungsi gerak disebut sebagai derajat kebebasan atau *degree of freedom* (DOF). Sebuah sendi yang diwakili oleh sebuah gerak *actuator* disebut

sebagai satu DOF. Sedangkan derajat kebebasan pada struktur roda dan kaki diukur berdasarkan fungsi *holonomic* atau *non-holonomic*.

3. Sensor

Adalah perangkat atau komponen yang bertugas mendeteksi (hasil) gerakan atau fenomena lingkungan yang diperlukan oleh sistem kontroller. Dapat dibuat dari sistem yang paling sederhana seperti sensor ON/OFF menggunakan limit *switch*, sistem analog, sistem bus parallel, sistem bus serial, hingga sistem mata kamera.

4. Aktuator

Adalah perangkat elektromekanik yang menghasilkan daya gerakan. Dapat dibuat dari sistem motor listrik (motor DC, permanent magnet, *brushless*, motor DC *servo*, motor DC *stepper, solenoid*, dsb.), sistem *pneumatic* (perangkat kompresi berbasis udara atau gas nitrogen), dan perangkat hidrolik (berbasis bahan cair seperti oli). Untuk meningkatkan tenaga mekanik aktuator atau torsi gerakan dapat dipasang sistem *gearbox*, baik sistem *direct-gear* (sistem lurus, sistem *ohmic/worm-gear*), *sprochet-chain* (gir-rantai, gir-belt, ataupun sistem *wire-roller*)

5. Sistem Roda

Adalah sistem mekanik yang dapat menggerakkan robot untuk berpindah posisi. Dapat terdiri dari sedikitnya sebuah roda penggerak (*drive* atau *steer*) dua roda deferensial (kiri kanan independent ataupun sistem *belt* seperti *tank*), tiga roda (*synchro drive* atau sistem *holonomic*), empat roda (*Ackermann model/car like mobile* robot atau sistem *mecanum wheels*) ataupun lebih.

6. Sistem Kaki

Pada dasarnya sistem kaki adalah gerakan ‘roda’ yang didesain sedemikian rupa hingga memiliki kemampuan gerak seperti makhluk hidup. Robot berjalan dengan sistem dua kaki atau *biped* robot memiliki struktur kaki seperti manusia setidak-tidaknya memiliki sendi-sendi yang mewakili pergelangan kaki, lutut dan pinggul. Dalam konfigurasi yang ideal pergerakan pada pinggul dapat terdiri dari

multi DOF dengan kemampuan gerakan memutar seperti orang menari jaipong. Demikian juga pada pergelangan kaki, idealnya adalah juga memiliki kemampuan gerakan polar. Untuk robot binatang, (*animaloid*) seperti serangga, jumlah kaki dapat didesain lebih dari empat. Bahkan robot ular yang memiliki DOF yang lebih dari 8 sesuai dengan panjang robot (ular) yang didefinisikan.

7. Sistem Tangan

Adalah bagian atau anggota badan robot selain sistem roda atau kaki. Dalam konteks mobile robot, bagian tangan ini dikenal sebagai manipulator yaitu sistem gerak yang berfungsi untuk memanipulasi (memegang, mengambil, mengangkat, memindah atau mengolah) obyek. Pada robot industri fungsi mengolah ini dapat berupa perputaran (memasang mur-baut, mengebor/ *drilling*, dll.), *tracking* (mengelas, membubut, dsb) ataupun mengaduk (control proses). Untuk robot tangan didesain sendi lengan diukur berdasarkan DOF. Lengan dapat dibuat kaku atau tegar (*rigid*) ataupun fleksibel (*flexible manipulator*).

8. Real World

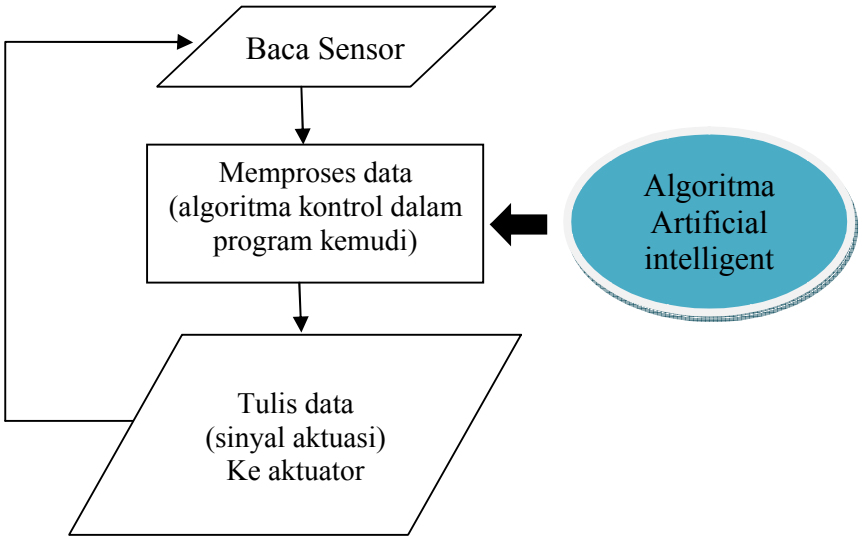
Real world atau dunia nyata didefinisikan sebagai daerah kerja (*workspace*) daripada robot. Robot yang tersusun dari tangan/manipulator saja memiliki workspace yang terbatas sesuai panjang jangkauan tangannya. Untuk robot beroda/berkaki, *workspace*-nya menjadi relatif tak terbatas tergantung kemampuan jelajahnya. Dengan menggabung robot tangan ke atas *mobile* robot maka daerah kerja untuk navigasi dasar dapat berupa mengikuti jalur di jalan (seperti *linefollower* atau *route-runner* robot, model labirin pada robot tikus, robot marka jalan berbasis vision), berjalan menuju ke obyek atau sasaran (menggunakan sensor radar, sonar, kamera, proximity), ataupun berjalan menuju sasaran dengan menghindari halangan (*obstacle*). Untuk bagian tangan, tugasnya dapat berupa tracking mengikkuti referensi trajektori, menuju atau menghindari obyek berupa *vision*, dan segala *terminology* manipulasi yang mungkin dilakukan sesuai dengan *tool* pada posisi TIP atau ujung/pergelangan tangan.

2.2.3 Kontrol Robotik

Kontrol adalah bagian yang amat penting dalam robotik, tanpa kontrol hanya akan menjadi benda mekatronik yang mati. Dalam sistem kontrol robotik, terdapat dua bagian, yaitu perangkat keras elektronik dan perangkat lunak yang berisi program kemudi serta algoritma *control* (Pitowarno, 2006).

Dalam hal ini, system control bertugas mengkolaborasikan system elektronik dan mekanik dengan baik agar mencapai fungsi seperti yang dikehendaki. Tanda dalam interseksi adalah posisi atau bagian dimana terjadi interaksi antara ketiga bagian itu.

Sistem kontroler sendiri memiliki mekanisme kerja seperti yang diilustrasikan berikut ini:



Gambar 2.5 Mekanisme Kerja (Program) Kontroler

Tiga prosedur utama, yaitu baca sensor, memproses data sensor, dan mengirim sinyal aktuasi ke aktuator adalah tugas utama kontroler. Ilustrasi ini mengisyaratkan bahwa sebenarnya tugas kontroler adalah sederhana. Dengan membaginya menjadi tiga bagian maka seorang *enginer* akan lebih mudah dalam melakukan analisa tentang bagaimana kontroler yang didesainnya bekerja. Meski dalam program kemudi robot secara kompleks namun sebenarnya tetap dapat

dibagi ke dalam tiga bagian besar itu yaitu: Baca sensor, Memproses data (algorithma kontrol dalam program kemudi), Tulis data (sinyal aktuasi ke aktuator).

Prosedur "Baca sensor" dapat terdiri dari berbagai teknik yang masing-masing membawa dampak kerumitan dalam pemrograman. Setidak-tidaknya ada dua macam teknik yang digunakan kontroler dalam menghubungi sensor, yaitu *polling* dan *interrupt*. Teknik *polling* adalah prosedur membaca data berdasarkan pengalamatan langsung yang dapat dilakukan kapan saja kontroler menghendaki. Sedang pada teknik *interrupt*, kontroler melakukan pembacaan jika sistem sensor melakukan interupsi, yaitu dengan memberikan sinyal interrupt ke kontroler (via perangkat keras) agar kontroler (CPU) melakukan proses pembacaan. Selama tidak ada interrupt maka kontroler tidak akan mengakses sensor tersebut.

Bagian yang berfungsi untuk memproses data sensor adalah bagian yang paling penting dalam program kontroler. Para peneliti dan enginer dapat dengan leluasa mengembangkan berbagai ide, teori dan teknik bagaimana membuat robot dapat bekerja sesuai harapan. Berbagai algoritma kontrol mulai dari teknik klasik dapat diterapkan. Jika dikehendaki kontrol yang lebih pintar dan dapat beradaptasi dapat memasukkan berbagai algoritma kontrol *adaptive* hingga teknik *artificial intelligent.*

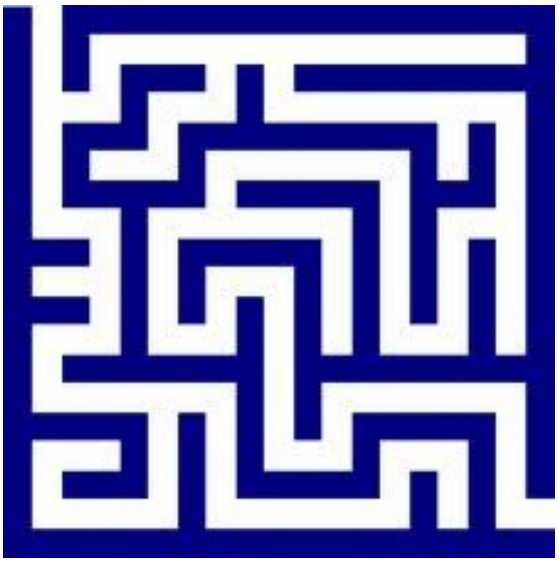
Bagian ketiga, yaitu prosedur "Tulis data" adalah bagian yang berisi pengalamatan ke aktuator untuk proses penulisan data. Dalam konteks rangkaian elektronik, data ini adalah sinyal aktuasi ke kontroler seperti berapa besar tegangan dan arus yang masuk ke motor, dan sebagainya.

2.3 Labirin (*Maze*)

Labirin (*Maze*) memiliki arti tempat yg penuh dengan jalan dan lorong yang berliku-liku dan simpang siur atau sesuatu yg sangat rumit dan berbelit-belit. *Maze* juga dapat juga diartikan sebagai sistem rongga atau saluran yg berhubungan. Labirin (*Maze*) biasanya dikenal sebagai sebuah taman yang

berliku-liku dan sangat rumit untuk menemukan jalan keluar, namun sebuah *maze* juga dapat diasumsikan pada bangunan atau gedung yang memiliki banyak ruangan dengan letak yang rumit.

Pada bidang robotika ada dua jenis *maze* yang umum digunakan, yaitu *wall maze* dan *line maze*. *Wall maze* pada umumnya dikenal dengan istilah labirin, yakni suatu jaringan jalan yang terbentuk atas lorong-lorong dengan dinding tanpa atap. Pada *line maze*, jaringan jalan yang terbentuk dibuat dengan menggunakan garis. Jika garis berwarna putih maka background berwarna hitam atau sebaliknya. Permasalahan yang timbul pada *maze* adalah cara untuk mendapatkan jalur terpendek, sehingga dibutuhkan metode untuk menyelesaikannya.



Gambar 2.6 Bentuk Labirin (*Maze*)

2.4 Algoritma Pencari Jalur

Ada beberapa algoritma yang sering digunakan dalam pencarian jalur dalam sebuah *maze* diantaranya *Depth first search*, *Flood Fill, Maze Mapping, Pledge dan Wall Follower* (Indrawan, 2008).

2.4.1 *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 itu pada akhirnya buntu, robot ini kembali ke percabangan tadi dan memilih jalur yang lain. Algoritma ini mengakibatkan robot untuk mengeksplorasi setiap *cell* di dalam labirin, robot pada akhirnya sampai pada *cell* tujuan.

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

2.4.2 *Flood Fill*

Algoritma ini melibatkan pemberian nilai pada masing-masing *cell* penyusun labirin, dimana nilai ini mempresentasikan jarak dari sembarang *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, begitu seterusnya.

Kelemahan dari algoritma ini ialah pada *flood fill* posisi *start* robot tidak bisa diubah-ubah dan bentuk dimensi *maze* harus diketahui terlebih dahulu sebelum robot dijalankan (Rahman, 2010).

2.4.3 *Maze Mapping*

Maze mapping merupakan algoritma yang digunakan untuk *mapping*, yakni mencari dan menggambarkan peta jalan keluar dari *maze*. *Maze mapping* pada umumnya di berbagai sumber menjelaskan dengan istilah *path mapping* yang konsep dasar dalam pencariannya mengikuti aturan *wall follower* (pada robot *wall follower*) atau *left/right hand rule* (pada robot *line tracer*) (Mishra, 2008).

2.4.4 *Pledge*

Algoritma *pledge* didesain untuk rintangan melingkar dan memiliki arah awal untuk bergerak maju. Ketika berhadapan dengan rintangan, maka robot akan menggunakan metode penulusaran *wall follower* yang akan menghindari rintangan dengan meprioritaskan sisi kanan atau kiri. Ketika robot kembali ke arah awal dan total hitungan belokan adalah "0", maka penyelesaian berlanjut sampai bertemu halangan kembali (Kamphans, 2004).

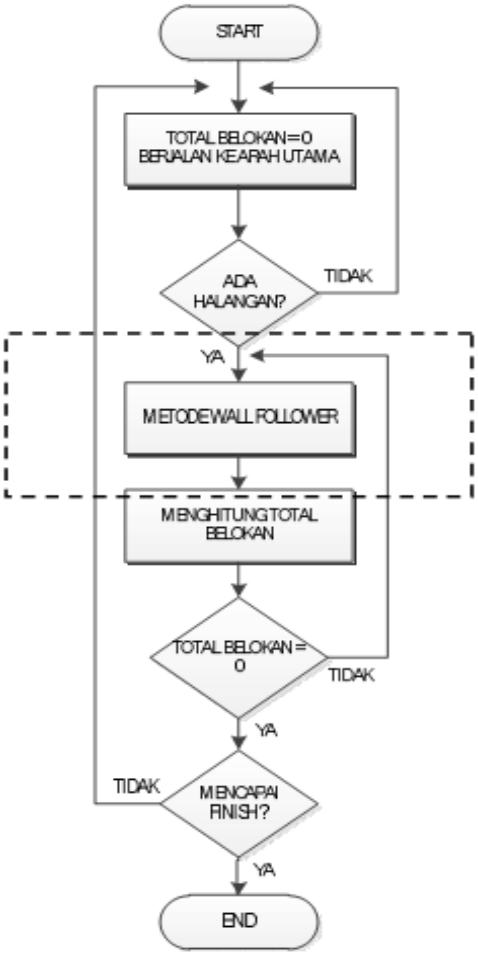
Berikut ini adalah *pseudocode* untuk mempermudah pemahaman tentang urutan algoritma *pledge*.

```
Total_belokan = 0
While (robot menyala)
{
        Berjalan ke arah utama
        If  (bertemu halangan)
        {
             Repeat
                 Berjalan dengan metode wall follower (mengikuti
                 lintasan kanan atau kiri)
                 Menghitung total belokan yang ditempuh
                 (belok kanan total belokan + 1, belok kiri
                 Total belokan – 1)
             Until
                 Total_belokan = 0
        }
}
```

Algoritma 2.1 Algoritma *Pledge* (Darmawan, 2010)

Algoritma diatas digunakan untuk perancangan robot dengan penelusuran *line maze* (penelusuran garis). Robot akan terus berjalan ke arah utama sampai menemukan halangan yang berbentuk garis. Ketika robot menemukan halangan, robot akan melakukan penelusuran dengan metode *wall follower* yaitu dengan mengikuti lintasan berupa garis sesuai prioritas yang diterapkan sambil menghitung total belokan.

Flowchart dari Algoritma *pledge* adalah sebagai berikut (Darmawan, 2010):



Gambar 2.7 *Flowchart* Algoritma *Pledge* dan Bagian yang Dimodifikasi

Flowchart yang dijelaskan pada Gambar 2.7 digunakan untuk robot *line maze.* Penjelasan dari *flowchart* pada Gambar 2.7 adalah sebagai berikut, robot berjalan menuju arah utama, misalnya arah utara sebagai arah utama robot. Arah utama adalah arah dimana robot pertama kali di *start*. Apabila belum terdapat halangan dan masih tersedia jalan utama maka robot akan terus berjalan kearah utama tersebut. Ketika robot mendapat halangan sehingga tidak mungkin berjalan ke arah utama maka robot akan melakukan teknik *wall follower* sambil melakukan perhitungan total belokan yang dilakukan. Apabila berbelok ke kanan ditambah 1 apabila berbelok ke kiri total belokan dikurangi satu. Robot akan kembali ke jalan utama apabila total belokan yang ditempuh = 1. Proses ini akan terus dilakukan

dengan robot akan melakukan *mapping* terhadap lintasan terus menerus sampai robot menemui garis *finish* (Kamphans, 2004).

2.4.5 *Wall Follower*

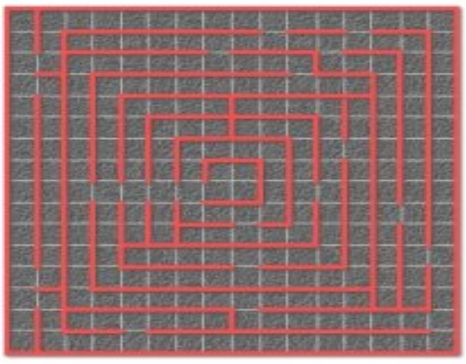
Algoritma ini merupakan teknik yang paling sederhana dalam pencarian jalur di lingkungan labirin. Pada dasarnya, robot dengan algoritma ini memiliki dua konsep penelusuran yaitu *right hand rule* atau konsep tangan kanan yaitu mengikuti dinding kanan dan *left hand rule* atau konsep tangan kiri dengan mengikuti dinding sebelah kiri sebagai petunjuk di sekeliling labirin untuk menemukan jalan keluar.

Algoritma ini cenderung berharap menemukan jalan keluar dibanding mencoba menyelesaikan labirin tersebut. Langkah-langkah yang dilakukan adalah sebagai berikut dalam *pseudo-code* (Harapan, 2009):

```
If ada jalan ke kanan
 Belok ke kanan
Else If ada jalan ke depan
 Do nothing
Else If ada jalan ke kiri
 Belok ke kiri
```

Algoritma 2.2 Algoritma *Wall Follower* (Harapan, 2009)

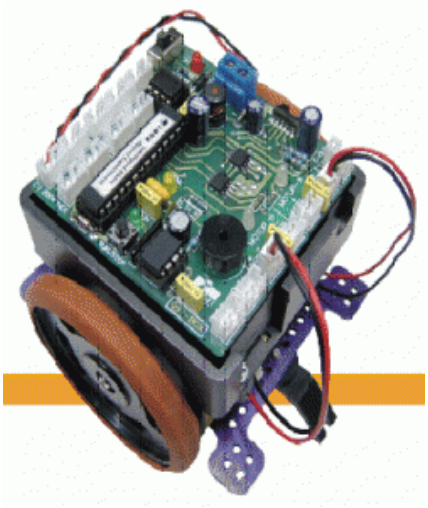
Pada kasus-kasus tertentu, algoritma ini tidak akan bekerja karena robot tidak bisa mengatasi *looping* dan akan berputar-putar terus di pinggir labirin sehingga tidak dapat mencari jalur menuju ke tengah dan menemukan titik *finish* (Indrawan, 2008).



Gambar 2.8 Lingkungan Labirin Dimana Algoritma *Wall Follower* Tidak Bekerja

2.5 Mobotsim

Mobotsim adalah sebuah perangkat lunak untuk simulasi 2D robot roda (*mobile robot* atau Mobot). Perangkat lunak Mobotsim ini menyediakan suatu antarmuka grafis yang mewakili suatu lingkungan dimana *user* dapat dengan mudah bisa membuat, mengatur dan menyunting mobot dan objek-objek rintangan (Putra, 2005).



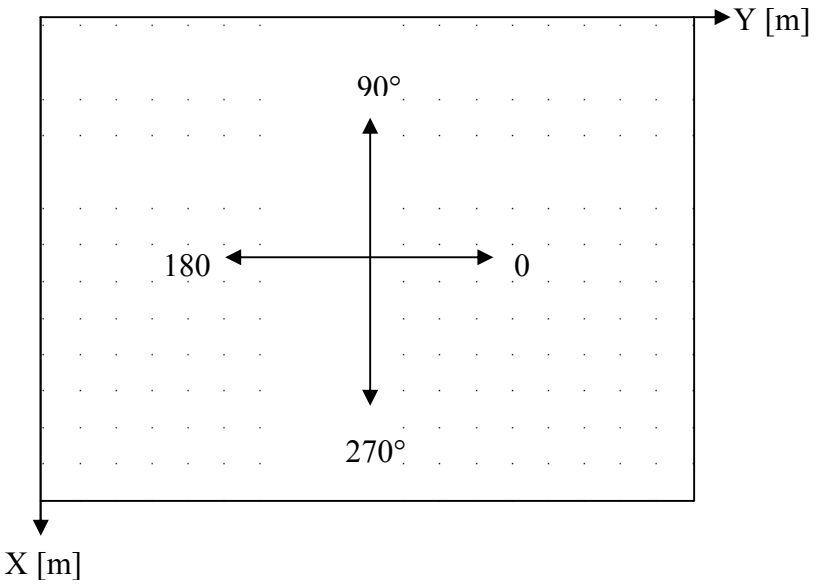
Gambar 2.9 *Mobile Robot*

Agar mampu mengatur gerakan robot roda tersebut, mobotsim memiliki editor bahasa Basic yang dapat digunakan untuk menuliskan makro-makro yang memanfaatkan fungsi-fungsi khusus robot untuk mendapatkan berbagai macam informasi, misalnya tentang koordinat robot dan data sensor, serta mangatur gerakan robot berdasarkan informasi-informasi yang diperoleh tersebut.

2.5.1 *World Frame* dan Sistem Koordinat

World frame merupakan area (*region)* dua-dimensi yang mewakili lingkungan *mobile robot* bekerja. Pertama kali *user* menjalankan Mobotsim, akan dibuat sebuah *world frame* secara *default* berukuran 20 x 20 meter, selanjutnya *user* bisa mengubah ukurannya sesuai dengan kebutuhan.

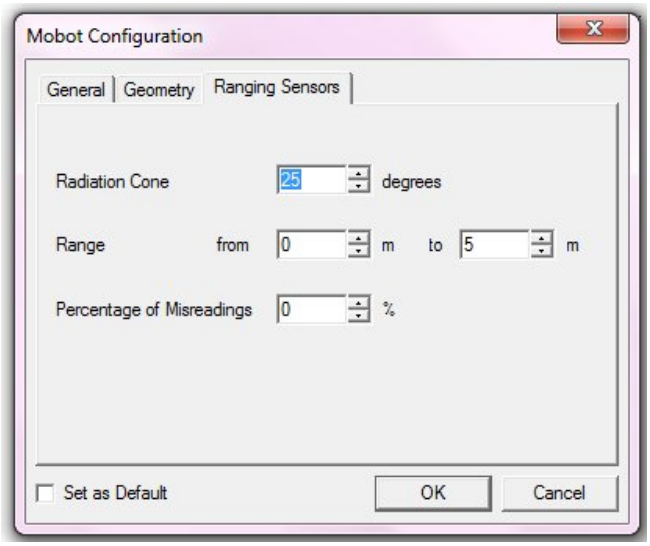
World frame menggunakan sistem koordinat kartesian standar dalam meter dan derajat, dengan titik asal di kiri atas, sebagaimana ditunjukkan pada gambar 2.10, sebuah *grid* dengan jarak 1 meter ditunjukkan sebagai acuan.



Gambar 2.10 *World Frame* dan Sistem Koordinatnya

2.5.2 Jarak atau *Ranging* Sensor

Untuk mengatur jarak atau *ranging* sensor dalam mobotsim dapat dilakukan pada *mobot configuration* lalu pilih tab *Ranging Sensor*. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini:



Gambar 2.11 Tab *Ranging Sensor*

Dari Gambar 2.11 dapat dijelaskan bahwa besar *radiation cone* yang digunakan adalah 25 derajat dengan *ranging sensor* antara 0-5 meter dan pada umumnya jumlah sensor yang digunakan adalah 12, namun user masih dapat mengganti jumlah sensor sesuai kebutuhan simulasi. Nomor sensor dihitung dari sisi paling kanan dari bagian robot dengan melawan arah jarum jam, semakin ke kiri maka semakin besar nomor sensor tersebut. Sensor dengan sisi paling kanan bernilai 0.

12 sensor **5 Sensor**

Gambar 2.12 Ilustrasi Sensor Yang Digunakan

Dalam penerapannya di dalam *coding*, sensor dapat diinisialisasikan berupa huruf seperti a,b,c atau abjad lainnya.

2.5.3 Pembacaan *Mark*

Mark atau titik *finish* dapat dideteksi oleh robot dengan cara melihat posisi robot pada koordinat X atau Y, jika robot membaca titik *finish* melalui koordinat sumbu X maka ketika robot berbanding lurus dengan koordinat X pada titik *finish* dalam arti nilai koordinat X pada titk *finish* sama dengan nilai koordinat X pada robot maka robot akan berhenti melakukan penelusuran, tetapi jika nilai koordinat X pada robot masih lebih besar dari koordinat X pada titik *finish* maka robot masih akan terus melakukan penelusuran sampai posisi robot pada koordinat X

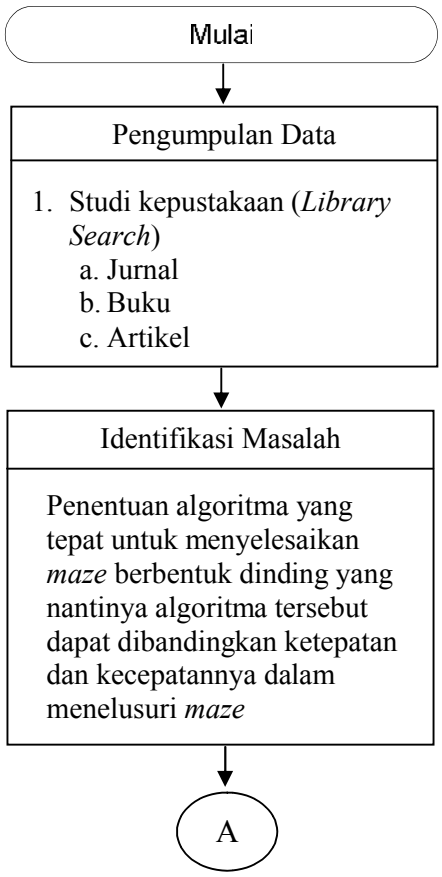
sama dengan koordinat X pada *mark* atau titik *finish*. Hal yang sama juga berlaku jika pembacaan *mark* atau titik *finish* dilakukan melalui koordinat sumbu Y.

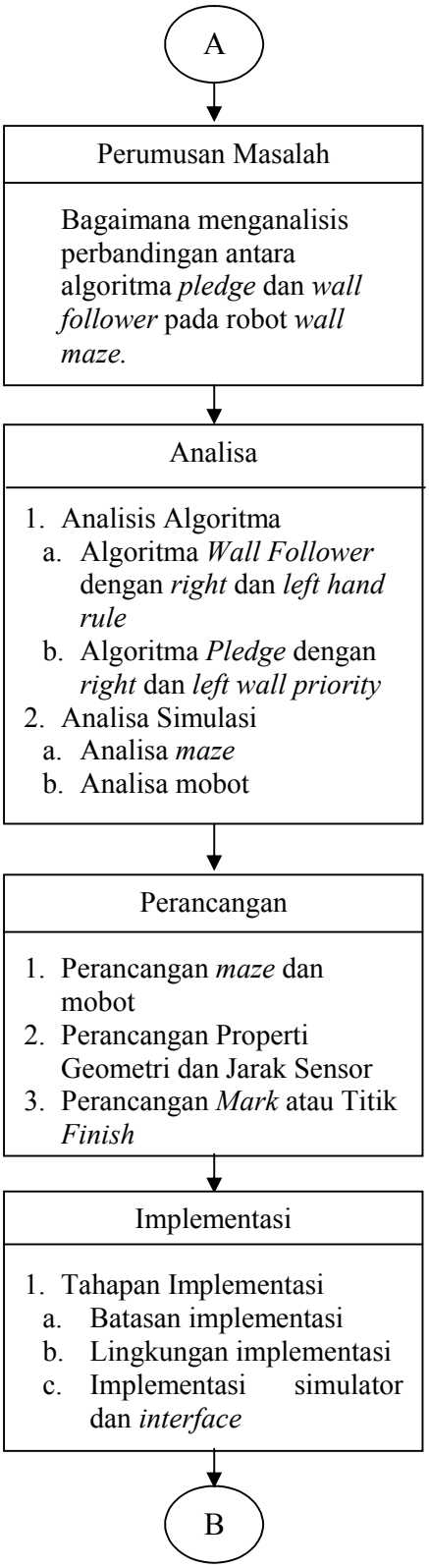
BAB III

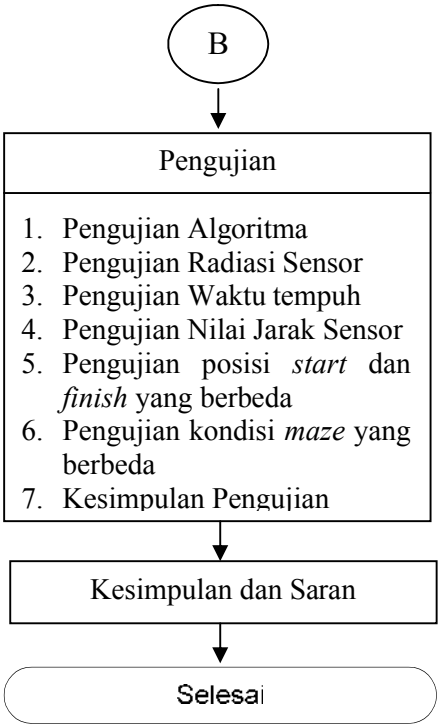
METODOLOGI PENELITIAN

Metodologi penelitian mempunyai peranan yang sangat penting dalam penelitian tugas akhir, karena pada metodologi penelitian ini menggambarkan langkah-langkah secara sistematis yang dilakukan dalam memecahkan permasalahan yang diangkat. Deskripsi dilengkapi dengan penyajian diagram alur pelaksanaan penelitian untuk memudahkan dalam memahami tahapan penelitian.

Dalam penulisan Tugas Akhir ini, studi literatur yang dilakukan yaitu dengan membaca berbagai literatur yang berkaitan dengan tulisan yang penulis kemukakan. Langkah-langkah yang akan ditempuh dalam penelitian ini dapat dilihat pada diagram alir dibawah ini:




A
Perumusan Masalah
Bagaimana menganalisis perbandingan antara algoritma *pledge* dan *wall follower* pada robot *wall maze.*
Analisa
1. Analisis Algoritma
a. Algoritma *Wall Follower* dengan *right* dan *left hand rule*
b. Algoritma *Pledge* dengan *right* dan *left wall priority*
2. Analisa Simulasi
a. Analisa *maze*
b. Analisa mobot
Perancangan
1. Perancangan *maze* dan mobot
2. Perancangan Properti Geometri dan Jarak Sensor
3. Perancangan *Mark* atau Titik *Finish*
Implementasi
1. Tahapan Implementasi
a. Batasan implementasi
b. Lingkungan implementasi
c. Implementasi simulator dan *interface*
B



Gambar 3.1 Metodologi Penelitian

Dalam metodologi penelitian dijabarkan tahapan-tahapan yang dilakukan dalam penelitian. Metodologi penelitian terdiri dari beberapa tahapan yang terkait secara sistematis. Tahapan ini diperlukan untuk memudahkan dalam melakukan penelitian. Tahapan yang dilakukan dalam penelitian adalah sebagai berikut:

3.1 Pengumpulan Data

Pada tahap ini dilakukan pengumpulan data yang berhubungan dengan penelitian dan pembuatan sistem, yaitu dengan:

1. Studi Kepustakaan (*Library Research*)

Studi kepustakaan dilakukan dengan cara mempelajari buku-buku, jurnal-jurnal dan artikel-artikel di internet yang berhubungan dengan permasalahan yang dibahas yaitu mengenai Pemanfaatan Kecerdasan Buatan (*Artificial Intelligence*) untuk Menyelesaikan *Maze* pada Robot *Wall Maze* dengan Algoritma *Pledge.*

Jurnal, buku dan artikel yang dipelajari, diantaranya: jurnal yang dibuat oleh Tom Kamphans dan Elmar Langetepe dengan judul "*The*

Pledge Algorithm Reconsidered Under Error in Sensor and Motion" dan jurnal Arif Darmawan, yang berjudul "*Penerapan Algoritma Pledge untuk Menyelesaikan Maze pada Line Follower Robot*", dari kedua jurnal ini penulis mempelajari tentang algoritma *pledge* yang digunakan sebagai acuan dalam tugas akhir ini. Sedangkan artikel yang penulis pelajari adalah artikel dengan judul "*Maze Solving Algorithm for Micro Mouse"* milik Swati Mishra, dari artikel ini penulis mempelajari tentang Algoritma-algoritma dalam penelusuran sebuah *maze.* Adapun buku yang penulis gunakan dalam penulisan tugas akhir ini adalah buku yang disusun oleh Suyanto, ST, M.Sc yang berjudul "*Artificial Intellegence: Searching, Reasoning, Planning, Learning"* dari penerbit Informatika Bandung, dari buku ini penulis mempelajari tentang dasar dan konsep tentang kecerdasan buatan (*Artificial Intellegence).*

3.2 Identifikasi Masalah

Dari pengamatan pendahuluan yang dilakukan, Permasalahan yang muncul adalah menentukan algoritma yang tepat untuk menyelesaikan *maze* berbentuk dinding yang nantinya algoritma tersebut dapat dibandingkan ketepatan dan kecepatannya dalam menelusuri *maze.*

3.3 Perumusan Masalah

Berdasarkan identifikasi masalah yang telah dijelaskan, maka perlu dirumuskan bagaimana menganalisis perbanding antara algoritma yang dipilih yaitu algoritma *pledge* dan *wall follower* sebagai pembanding kinerja algoritma *pledge* pada robot *wall maze.*

3.4 Analisa

Analisa permasalahan berkaitan dengan mengidentifikasi kebutuhan algoritma dalam suatu penelitian. Dan analisa yang dilakukan dalam tugas akhir ini terbagi atas dua tahapan, yaitu analisis algoritma yaitu algoritma *pledge* dan algoritma *wall follower* kemudian dilanjutkan dengan analisa simulasi dari masing-masing algoritma yang telah dianalisa.

3.5 Perancangan

Setelah melakukan analisa, maka kemudian dilanjutkan dengan perancangan berdasarkan analisa permasalahan yang telah dilakukan sebelumnya. Dalam tugas akhir ini perancangan dilakukan dalam tiga tahapa, yaitu: perancangan *maze* dan mobot (*mobile robot)*, perancangan geometri dan jarak sensor, kemudian perancangan *mark* atau titik *finish.*

3.6 Implementasi

Implementasi merupakan suatu konversi dari desain sistem yang telah dirancang kedalam sebuah program komputer dengan menggunakan bahasa pemrograman *Basic* berbasis *desktop*. Adapun fungsi-fungsi dari analisa perbandingan antara algoritma *pledge* dan algoritma *wall follower* adalah untuk melihat algoritma mana yang lebih unggul dan tepat dalam menyelesaikan *maze* yang digunakan dalam tugas akhir ini. Implementasi dalam tugas akhir ini terdiri dari 3 tahap yaitu: batasan implementasi, lingkungan implementasi dan implementasi simulator dan *interface.*

3.7 Pengujian

Pengujian ini berfokus pada perangkat lunak berupa kecerdasan buatan yang akan diterapkan untuk mencari jalan keluar dalam sebuah *maze* pada robot w*all maze* sehinggan robot dapat menemukan titik tujuan dengan algoritma *pledge* dan algoritma *wall follower,* yang nantinya dapat dilihat perbandingan antara kedua algoritma tersebut. Pengujian dalam tugas akhir ini terdiri dari: pengujian algoritma, pengujian radiasi sensor, pengujian waktu tempuh, pengujian nilai jarak sensor, pengujian posisi *start* dan *finish* yang berbeda dan pengujian dengan kondisi *maze* yang berbeda. Setelah pengujian dilakukan maka dalam tahap ini dilakukan analisa akhir terhadap hasil pengujian yang nantinya dapat dirumuskan kesimpulan dari semua pengujian.

3.8 Kesimpulan Dan Saran

Pada tahap ini ditarik kesimpulan-kesimpulan yang dihasilkan dari pembahasan tentang analisa perbandingan antara algoritma *pledge* dan algoritma *wall follower* pada robot *wall maze* serta saran untuk perbaikan pengembangan tugas akhir ini.

BAB IV

ANALISIS DAN PERANCANGAN

4.1. Analisis Algoritma

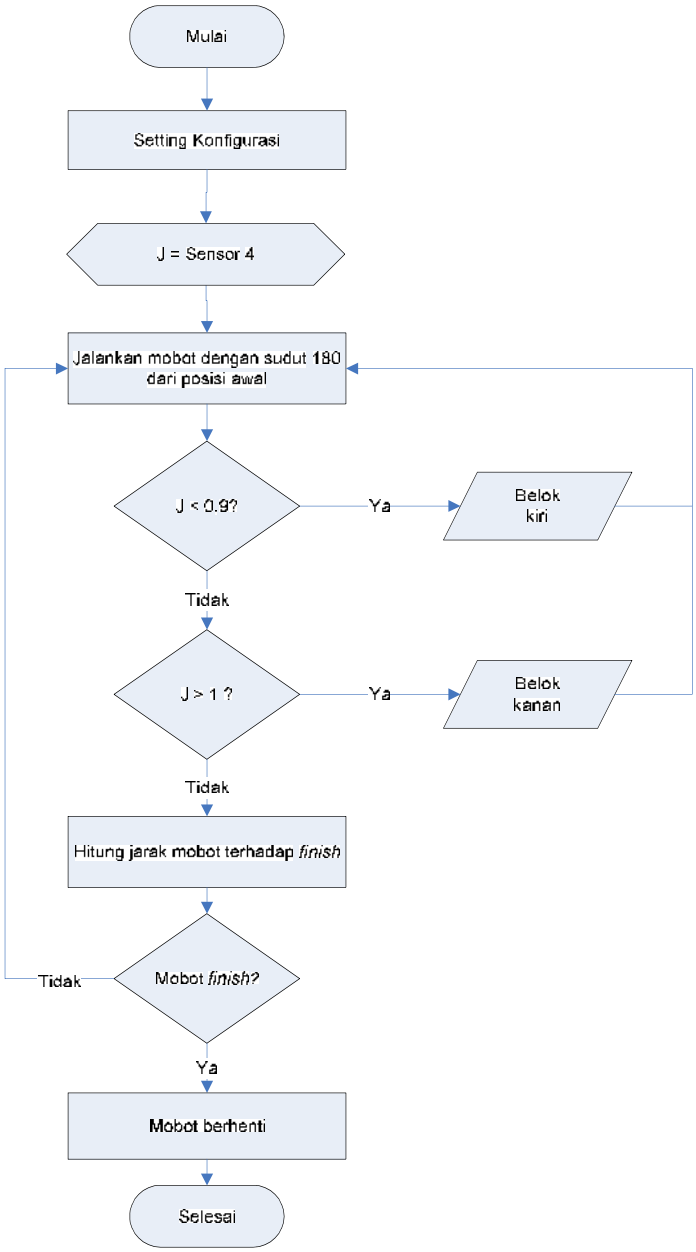
Analisis algoritma ini bertujuan untuk menjelaskan alur pergerakan robot. Algoritma yang digunakan adalah algoritma *pledge* dan algoritma *wall follower* yang digunakan sebagai pembanding terhadap algoritma *pledge.*

4.1.1 Analisis Pergerakan Robot Menggunakan Algoritma *Wall Follower*

Seperti yang telah dijelaskan di dalam bab II landasan teori, algoritma *wall follower* merupakan algoritma pengikut dinding kiri atau kanan sebagai petunjuk di sekeliling labirin. Dalam tugas akhir ini algoritma *wall follower* digunakan sebagai pembanding atas kinerja algoritma *pledge*. Penulis memilih algoritma *wall follower* karena algoritma ini juga digunakan dalam pengembangan algoritma *pledge*. Algoritma *Wall follower* terdiri dari dua teknik penelusuran yaitu *right hand rule* dan *left hand rule*

4.1.1.1 Algoritma *Wall Follower* Dengan *Right Hand Rule*

Algoritma ini dirancang untuk penelusuran *maze* dengan memprioritaskan dinding kanan sebagai petunjuk robot untuk menelusuri *maze*. *Flowchart* dari algoritma ini adalah sebagai berikut:



Gambar 4.1 *Flowchart* Algoritma *Wall Follower* dengan *Right Hand Rule*

Berdasarkan *flowchart* algoritma *wall follower* dengan *right hand rule* yang ditunjukkan pada Gambar 4.1 dapat dijelaskan bahwa:

1. Proses penelusuran *maze* dimulai dengan melakukan setting konfigurasi pada propertis robot, yang terdiri dari nama robot, index robot, posisi robot, propertis roda dan sensor robot.

2. Setelah setting konfigasi selesai, maka dilanjutkan dengan menginisialisasi sensor yang digunakan, J=sensor. Dalam simulasi *right hand rule* ini, sensor yang digunakan ialah sensor ke 4 dari 12 sensor atau J=4.
3. Selanjutnya, jalankan robot dengan sudut 180^o agar robot dapat berjalan ke arah utama atau berjalan lurus.
4. Ketika robot menemukan rintangan atau dinding dengan jarak sensor ke dinding kurang dari 0,9 meter maka robot akan berbelok ke kiri dan kembali berjalan lurus sampai menemukan rintangan atau dinding kembali.
5. Jika robot menemukan rintangan atau dinding dengan jarak sensor ke dinding lebih dari 1 meter, maka robot akan berbelok ke kanan.
6. Jika tidak terdapat halangan lagi maka robot akan mencari titik *finish* dengan menghitung jarak *finish* dengan keberadaan robot tersebut.
7. Jika robot menemukan titik *finish* maka robot akan berhenti dan proses penelusuran selesai, jika tidak maka robot akan mengulang perintah mulai dari berjalan dengan sudut 180 atau berjalan lurus dan seterusnya sampai robot menemukan titik *finish*.

Untuk lebih memahami Algoritma dari *Wall follower* dengan *right hand rule*, berikut *pseudocode* dari algoritma tersebut:

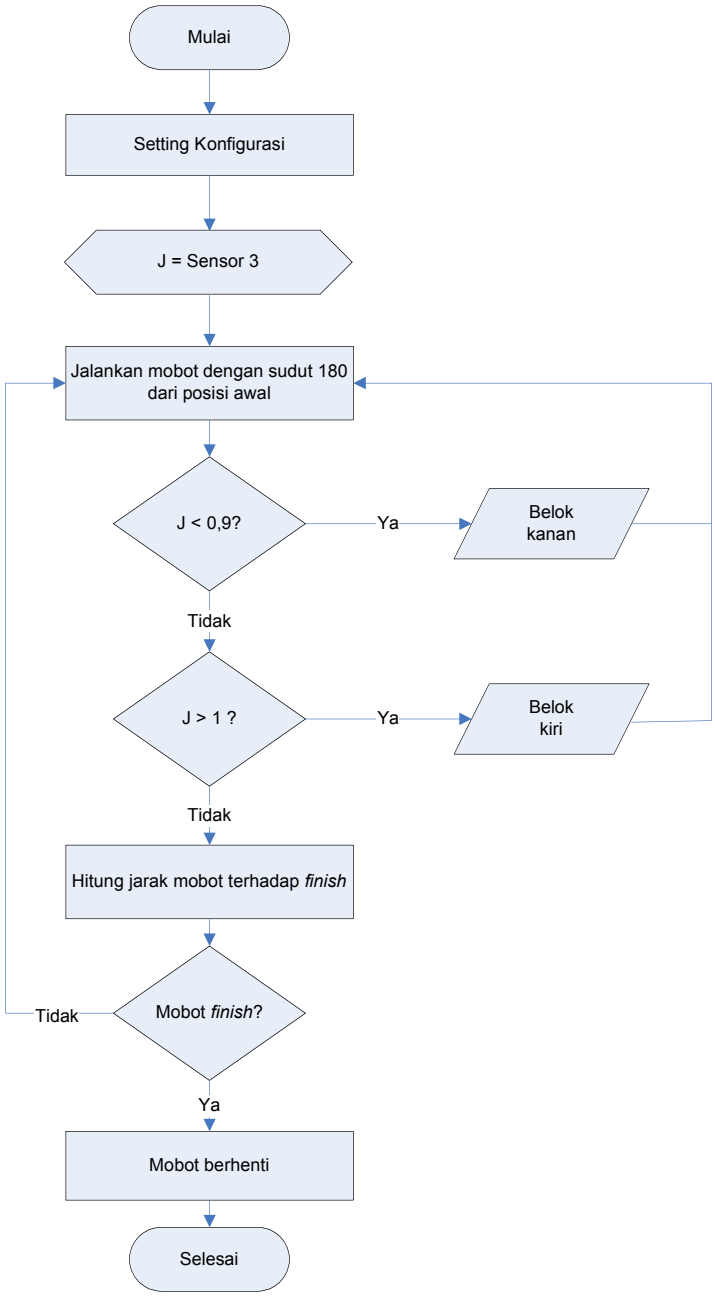
```
Start
      Berjalan ke arah utama
      If (Jarak Rintangan < 0,9) Then
          Belok kiri
      ElseIf (Jarak Rintangan > 1) Then
          Belok kanan
      EndIf

      Hitung jarak terhadap finish
      If (Bertemu titik Finish) Then
          Berhenti
      ElseIf (Tidak bertemu Finish) Then
          Kembali melakukan penelusuran
      EndIf
End
```

Algoritma 4.1 Algoritma *Wall Follower* dengan *Right Hand Rule*

4.1.1.2 Algoritma *Wall Follower* Dengan *Left Hand Rule*

Seperti algoritma *wall follower* dengan *right hand rule* yang telah dijelaskan sebelumnya. Algoritma ini juga dirancang untuk penelusuran *maze* dengan memprioritaskan dinding kiri terlebih dahulu sebagai petunjuk robot untuk menelusuri *maze*. *Flowchart* dari algoritma ini adalah sebagai berikut:



Gambar 4.2 *Flowchart* Algoritma *Wall Follower* dengan *Left Hand Rule*

Flowchart pada algoritma ini hampir sama *flowchart* pada algoritma *right hand rule*, perbedaannya hanya pada pengaturan sensor untuk pembacaan dinding. Berdasarkan *flowchart* algoritma *wall follower* dengan *left hand rule* yang ditunjukkan pada Gambar 4.2 dapat dijelaskan bahwa:

1. Proses penelusuran *maze* dimulai dengan melakukan setting konfigurasi pada propertis robot, yang terdiri dari nama robot, index robot, posisi robot, propertis roda dan sensor robot.
2. Setelah setting konfigasi selesai, maka dilanjutkan dengan menginisialisasi sensor yang digunakan, J=sensor. Dalam simulasi *left hand rule* ini, sensor yang digunakan ialah sensor ke 3 dari 5 buah sensor atau J=3.
3. Selanjutnya, jalankan robot dengan sudut 180^{o} agar robot dapat berjalan ke arah utama atau berjalan lurus.
4. Ketika robot menemukan rintangan atau dinding dengan jarak sensor ke dinding lebih dari 0,9 meter maka robot akan berbelok ke kanan dan kembali berjalan lurus sampai menemukan rintangan atau dinding kembali.
5. Jika robot menemukan rintangan atau dinding dengan jarak sensor ke dinding kurang dari 1 meter, maka robot akan berbelok ke kiri.
6. Jika tidak terdapat halangan lagi maka robot akan mencari titik *finish* dengan menghitung jarak *finish* dengan keberadaan robot tersebut.
7. Jika robot menemukan titik *finish* maka robot akan berhenti dan proses penelusuran pun selesai, jika tidak maka robot akan mengulang perintah mulai dari berjalan dengan sudut 180 atau berjalan lurus dan seterusnya sampai robot menemukan titik *finish*.

Untuk lebih memahami Algoritma dari *Wall follower* dengan *left hand rule*, berikut *pseudocode* dari algoritma tersebut:

```
Start
      Berjalan ke arah utama
      If (Jarak Rintangan < 0,9) Then
          Belok kiri
      ElseIf (Jarak Rintangan > 1) Then
          Belok kanan
      EndIf

      Hitung jarak terhadap finish
      If (Bertemu titik Finish) Then
          Berhenti
      ElseIf (Tidak bertemu Finish) Then
          Kembali melakukan penelusuran
      EndIf
End
```

Algoritma 4.2 Algoritma *Wall Follower* dengan *Left Hand Rule*

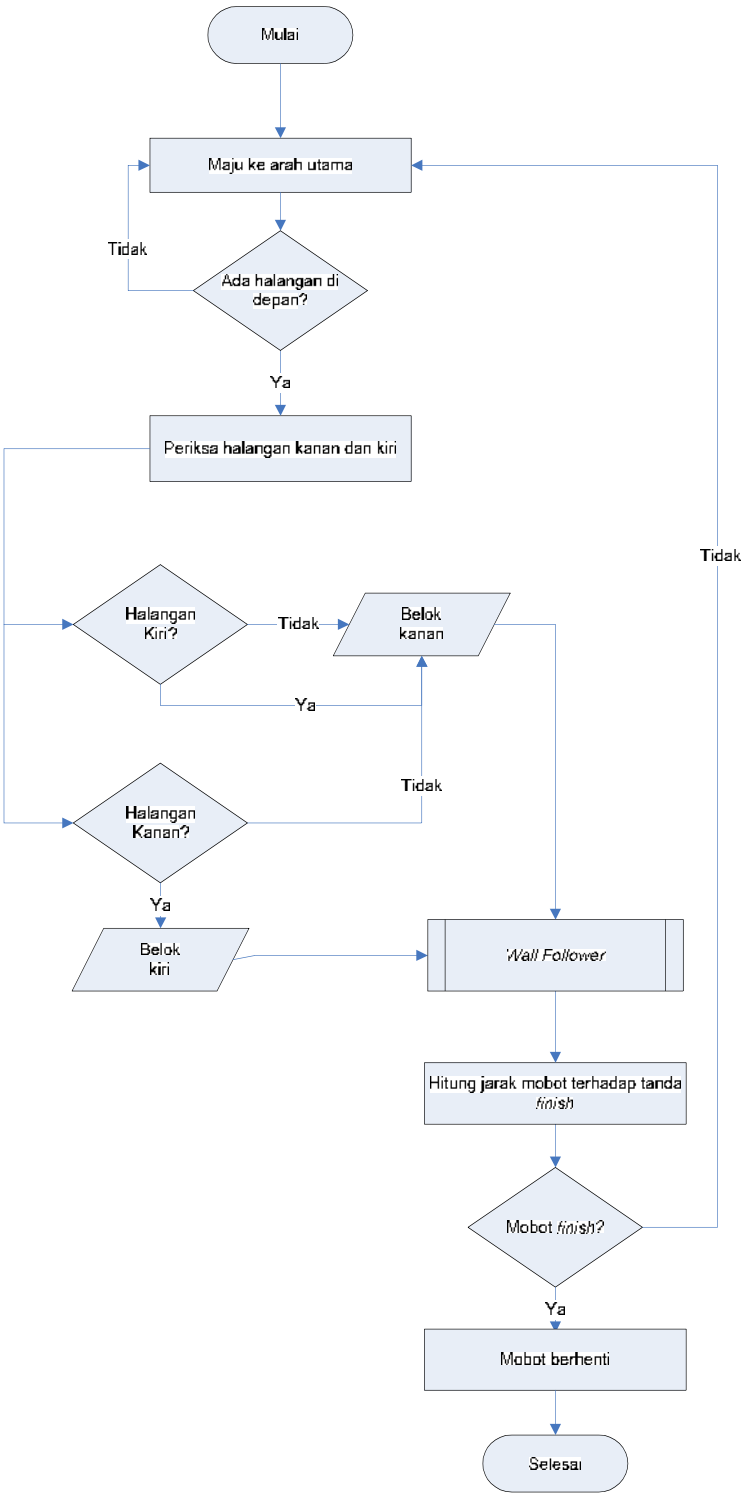
4.1.2 Analisis Pergerakan Robot Menggunakan Algoritma *Pledge*

Algoritma *pledge* merupakan pengembangan dari algoritma *wall follower*. Algoritma *pledge* didisain untuk rintangan melingkar dan memiliki arah utama untuk bergerak maju. Ketika robot masih berada pada arah utama maka robot akan terus berjalan sampai robot menemui rintangan dihadapannya. Dalam tugas akhir ini, arah utama sama dengan berjalan lurus dari titik *start* yang sejajar dengan arah pada titik *finish*. Algoritma *pledge* juga bertujuan untuk memberikan keputusan arah mana yang harus dilalui ketika robot bertemu halangan berupa simpang tiga yang berbentuk T. Dalam hal ini algoritma *pledge* akan bekerja dengan dua prioritas penelusuran, yaitu *right wall priority* dan *left wall priority*. Adapun yang membuat algoritma ini lebih baik dari algoritma *wall follower* adalah pemberian keputusan yang diberikan, jika pada algortima *wall follower* hanya ada prioritas kiri dan kanan tanpa memikirkan arah *finish*, maka pada algoritma *pledge* prioritas kiri dan kanan digunakan untuk penentuan jalur pada persimpangan berbentuk T dan penelusuran berpatokan pada arah utama robot, sehingga pada saat menemui

rintangan pada *maze* yang berbentuk melingkar, robot tidak akan berputar pada sisi yang sama seperti halnya robot dengan teknik *wall follower*.

4.1.2.1 Algoritma *Pledge* Dengan *Right Wall Priority*

Sama halnya dengan algoritma *wall follower* yang menggunakan dua prioritas *right* dan *left*, algoritma *pledge* juga menggunakan dua prioritas tersebut. Namun algoritma *pledge* menggunakan dua prioritas tersebut untuk melewati rintangan berupa simpang berbentuk T, setelah rintangan ini terlewati maka robot akan menggunakan teknik *wall follower* dengan menelusuri *maze* dengan prioritas yang digunakan. Jika robot didesain untuk penelusuran *right wall priority* maka robot akan mendahulukan dinding kanan walaupun dinding sebelah kiri tidak terdapat rintangan. *flowchart* penelusuran Algoritma *pledge* dengan *right wall priority* dalam dilihat pada Gambar 4.3.



Gambar 4.3 *Flowchart* Algoritma *Pledge* dengan *Right Wall Priority*

Berdasarkan *flowchart* algoritma *pledge* dengan *right wall priority* yang ditunjukkan pada Gambar 4.3 dapat dijelaskan bahwa:

1. Robot bergerak maju ke arah utama dari titik *start* dengan asumsi bahwa tidak ada halangan di dinding kiri atau pun kanan.
2. Ketika robot menemui halangan di depan atau menemui persimpangan berbentuk T dimana tidak terdapat halangan di dinding kiri ataupun kanan, maka robot akan tetap melakukan pemeriksaan untuk pengambilan keputusan. Di sinilah algoritma *pledge* mulai bekerja, robot akan melakukan pemeriksaan terhadap dinding kiri dan kanan, jika di dinding kiri terdapat halangan, maka robot akan melakukan aksi berbelok ke kanan. Karena algoritma ini merupakan algoritma dengan prioritas kanan, saat dinding kiri tidak terdapat halangan maka robot akan tetap berbelok ke kanan.
3. Robot tetap melakukan pemeriksaan terhadap sisi dinding sebelah kanan, jika terdapat halangan maka robot akan berbelok ke kiri, namun jika tidak terdapat halangan maka robot akan berbelok ke kanan.
4. Ketika robot tidak lagi menemui persimpangan berbentuk T, maka robot akan melakukan teknik *wall follower* dengan berjalan mengikuti dinding dengan memprioritaskan dinding sebelah kanan.
5. Setelah semua persimpangan dilalui dan robot mulai dapat menghitung jarak ke titik *finish* maka robot akan terus berjalan ke arah *finish*.
6. Jika robot menemukan titik *finish* maka robot akan berhenti dan proses penelusuran pun selesai, jika tidak maka robot akan terus bergerak maju dan melakukan penelusuran dengan algoritma *pledge* sampai robot menemukan titik *finish*.

Untuk lebih memahami Algoritma dari *Pledge* dengan *right wall priority*, berikut *pseudocode* dari algoritma tersebut:

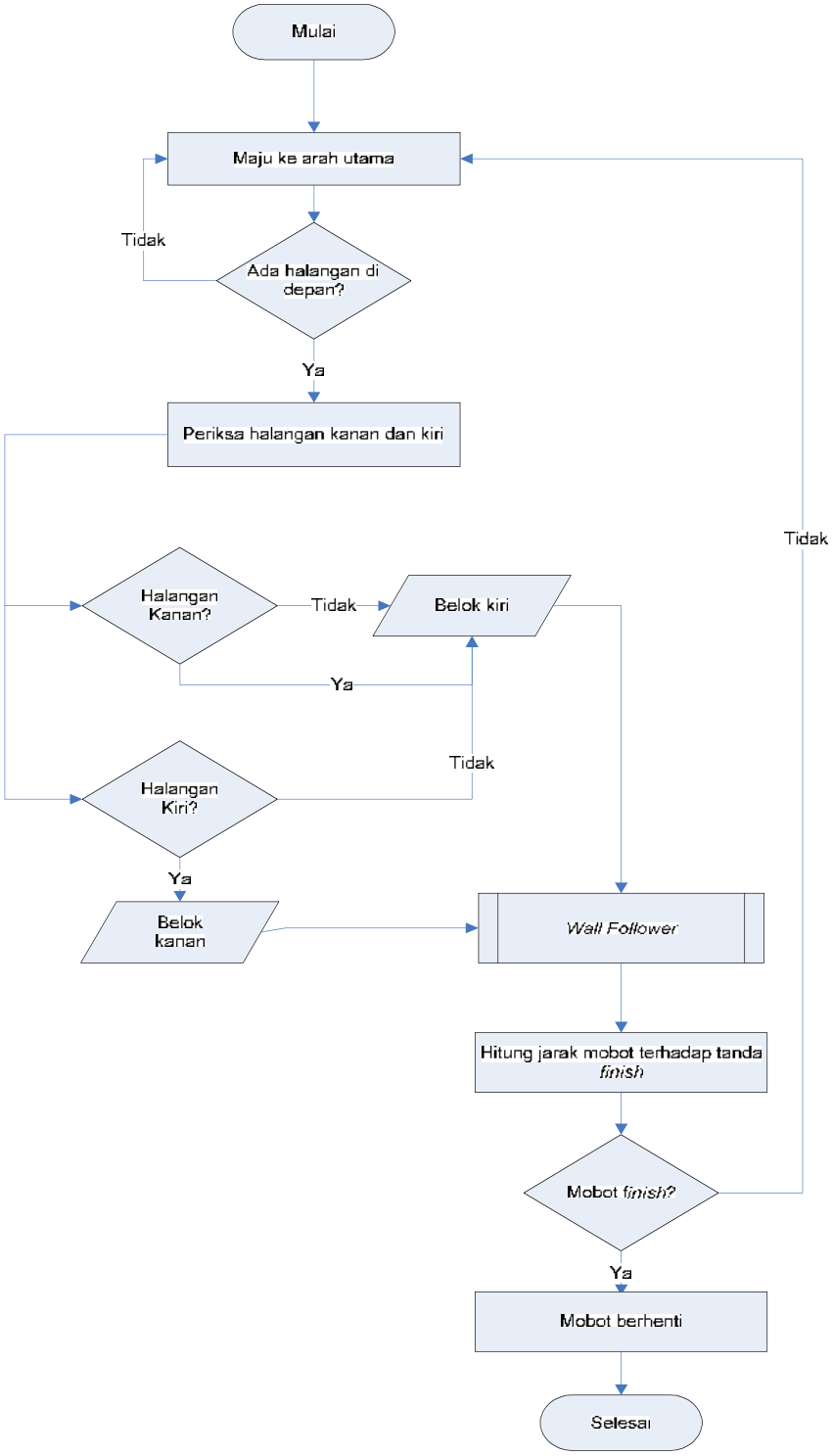
```
Start
      Berjalan ke arah utama
      If (Bertemu halangan) Then
          Periksa halangan kanan dan kiri
      ElseIf (Ada halangan kiri) Then
          Belok kanan
      ElseIf (Tidak ada halangan kiri) Then
          Belok kanan
      ElseIf (Ada halangan kanan) Then
          Belok kiri
      ElseIf (Tidak ada halangan kanan) Then
          Belok kanan
      Else(lakukan metode Wall follower)
      EndIf

      Hitung jarak terhadap finish
      If (Bertemu titik Finish) Then
          Berhenti
      ElseIf (Tidak bertemu Finish) Then
          Kembali melakukan penelusuran
      EndIf
 End
```

Algoritma 4.3 Algoritma *Pledge* dengan *Right Wall Priority*

4.1.2.2 Algoritma *Pledge* Dengan *Left Wall Priority*

Seperti algoritma *pledge* dengan *right wall priority* yang menelusuri *maze* dengan lebih mendahulukan dinding kanan, algoritma *pledge* dengan *left wall priority* didesain untuk penelusuran *maze* dengan lebih memprioritaskan dinding sebelah kiri. *Flowchart* penelusuran Algoritma *pledge* dengan *left wall priority* dalam dilihat pada Gambar 4.4 di bawah ini.



Gambar 4.4 *Flowchart* Algoritma *Pledge* dengan *Left Wall Priority*

Berdasarkan *flowchart* algoritma *pledge* dengan *left wall priority* yang ditunjukkan pada Gambar 4.4 dapat dijelaskan bahwa:

1. Robot bergerak maju ke arah utama dari titik *start* dengan asumsi bahwa tidak ada halangan di dinding kiri atau pun kanan.
2. Ketika robot menemui halangan di depan atau menemui persimpangan berbentuk T dimana tidak terdapat halangan di dinding kiri ataupun kanan, maka robot akan tetap melakukan pemeriksaan untuk pengambilan keputusan. Di sinilah algoritma *pledge* mulai bekerja, robot akan melakukan pemeriksaan terhadap dinding kiri dan kanan, jika di dinding kanan terdapat halangan, maka robot akan melakukan aksi berbelok ke kiri. Karena algoritma ini merupakan algoritma dengan prioritas kiri, saat dinding kanan tidak terdapat halangan maka robot akan tetap berbelok ke kiri.
3. Robot tetap melakukan pemeriksaan terhadap sisi dinding sebelah kiri, jika terdapat halangan maka robot akan berbelok ke kanan, namun jika tidak terdapat halangan maka robot akan berbelok ke kiri.
4. Ketika robot tidak lagi menemui persimpangan berbentuk T, maka robot akan melakukan teknik *wall follower* dengan berjalan mengikuti dinding dengan memprioritaskan dinding sebelah kiri.
5. Setelah semua persimpangan dilalui dan robot mulai dapat menghitung jarak ke titik *finish* maka robot akan terus berjalan ke arah *finish*.
6. Jika robot menemukan titik *finish* maka robot akan berhenti dan proses penelusuran pun selesai, jika tidak maka robot akan terus bergerak maju dan melakukan penelusuran dengan algoritma *pledge* sampai robot menemukan titik *finish*.

Untuk lebih memahami Algoritma dari *Pledge* dengan *Left wall priority*, berikut *pseudocode* dari algoritma tersebut:

```
Start
      Berjalan ke arah utama
      If (Bertemu halangan) Then
          Periksa halangan kanan dan kiri
      ElseIf (Ada halangan kanan) Then
          Belok kiri
      ElseIf (Tidak ada halangan kanan) Then
          Belok kiri
      ElseIf (Ada halangan kiri) Then
          Belok kanan
      ElseIf (Tidak ada halangan kiri) Then
          Belok kiri
      Else(lakukan metode Wall follower)
      EndIf

      Hitung jarak terhadap finish
      If (Bertemu titik Finish) Then
          Berhenti
      ElseIf (Tidak bertemu Finish) Then
          Kembali melakukan penelusuran
      EndIf
End
```

Algoritma 4.4 Algoritma *Pledge* dengan *Left Wall Priority*

4.2 Analisa Simulasi

Sebelum masuk ke tahap perancangan, maka perlu dilakukan analisa terhadap simulasi yang nantinya diterapkan sehingga dapat diketahui apa saja yang perlu dirancang untuk menerapkan kedua algoritma yang digunakan dan mengimplementasikannya ke dalam simulator. Simulasi dirancang berdasarkan pengujian yang dilakukan. Ada 6 simulasi yang nantinya akan diimplementasikan dalam simulator mobotsim, yaitu simulasi untuk pengujian algoritma, radiasi sensor, waktu tempuh, nilai jarak sensor, posisi *start* dan *finish* yang berbeda dan simulasi pengujian kondisi *maze* yang berbeda.

Simulasi dilakukan dengan menggunakan simulator mobotsim v.10 dan menggunakan bahasa pemrograman *basic.* Alasan penulis menggunakan simulator mobotsim karena mobotsim merupakan simulator yang lebih fokus terhadap pengujian algoritma yang nantinya akan diterapkan pada robot aslinya. Sehingga pada saat melakukan *setting* konfigurasi robot, tidak semua perancangan perangkat robot diperhatikan, hanya pada inti dari robot itu yang ditentukan untuk keperluan simulasi seperti diameter robot dan sensor yang digunakan. Berbeda dengan simulator yang lainnya seperti webots dengan tampilan 3D yang lebih fokus terhadap visualisasi perangkat robot dan rintangan yang digunakan.

Dalam simulasi dengan simulator mobotsim yang perlu diperhatikan adalah penentuan nilai-nilai pada *setting* konfigurasi mobot, seperti: *properties* robot, jumlah sensor, jarak sensor dan posisi *start* dan *finish.* Nilai-nilai yang dipakai harus sesuai dengan bentuk dan keadaan *maze* yang dirancang. Sehingga saat simulasi dijalankan robot dapat melakukan penelusuran sesuai dengan keinginan *user.*

4.3 Perancangan

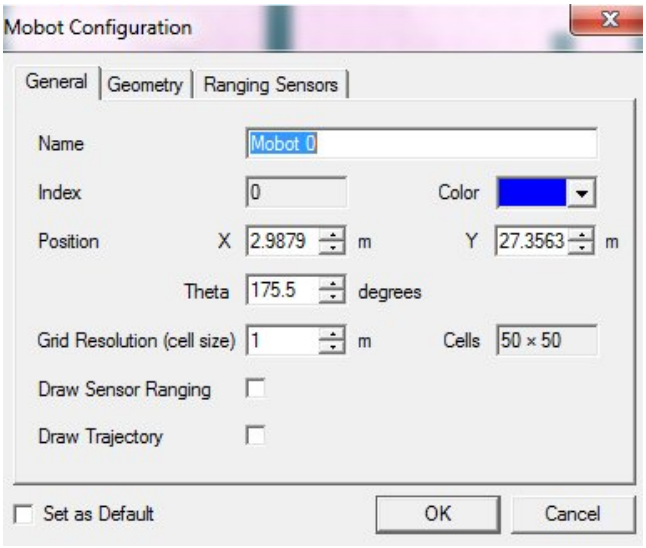
Setelah tahapan analisa kebutuhan telah selesai dilakukan maka tahapan selanjutnya yaitu perancangan sistem. Perancangan sistem ini terdiri dari perancangan *maze* dan robot, propertis Geometri dan jarak sensor serta perancangan mark untuk titik *finish* dengan menggunakan simulator Mobotsim v.10.

4.3.1 Perancangan *Maze* dan Robot

Maze biasanya dikenal sebagai sebuah taman yang berliku-liku dan sangat rumit untuk menemukan jalan keluar, namun sebuah *maze* juga dapat diasumsikan pada bangunan atau gedung yang memiliki banyak ruangan dengan letak yang rumit. Dalam tugas akhir ini *maze* dirancang dengan ukuran *cells* 50x50 meter dan *grid resolution* (*cell size*) berukuran 1 meter pada ukuran sebenarnya.

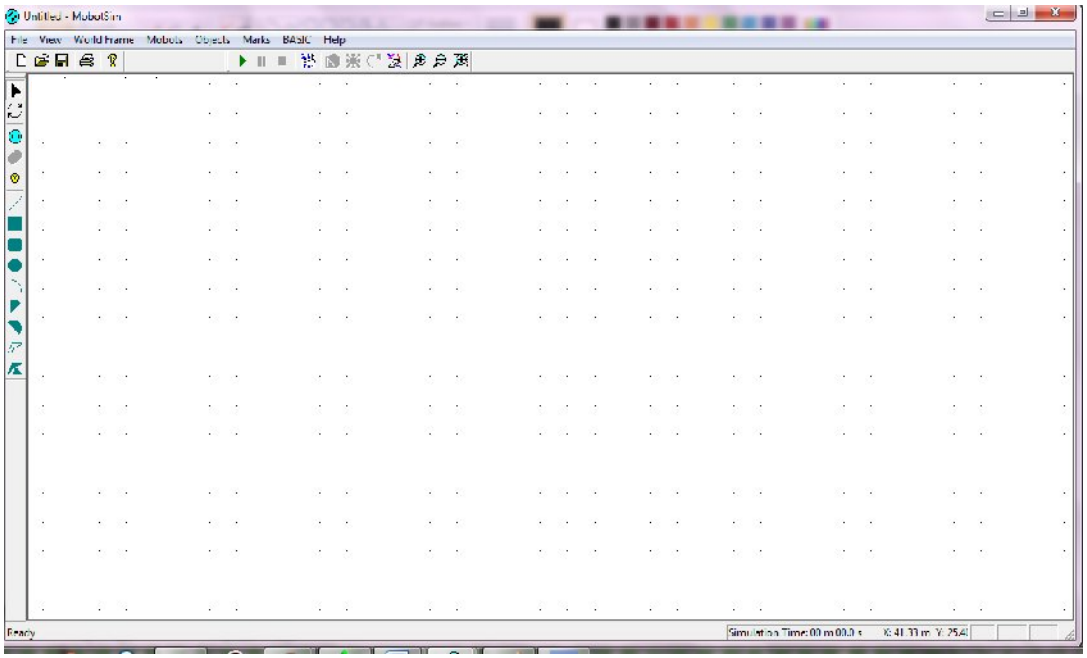
Robot merupakan robot *micromouse* yang digunakan dalam perangkat lunak mobotsim ini. Dalam perancangan *Maze* dan robot, terdapat Mobot

configuration yang mengatur Nama, Index, warna dan posisi robot serta ukuran *maze*. Untuk lebih jelasnya robot *configuration* dapat dilihat pada Gambar 4.5 di bawah ini:



Gambar 4.5 Robot *Configuration*

Setelah robot *configuration* telah diatur maka langkah selanjutnya adalah merancang *maze*.



Gambar 4.6 *Maze* dengan ukuran 50x50 meter

Dalam perancangan *maze* terdapat beberapa elemen dalam mobotsim yang dipergunakan, seperti Gambar 4.7 berikut:

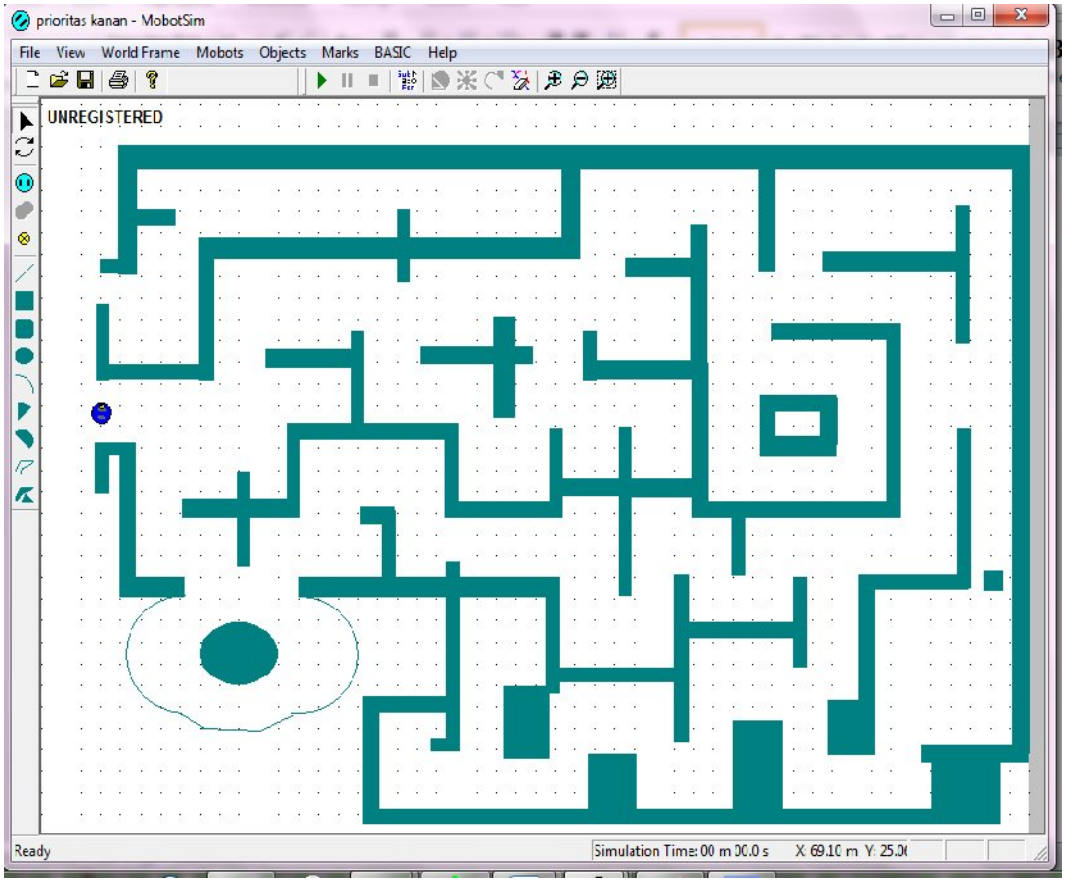
Gambar 4.7 Objek-Objek Dalam Mobotsim Untuk Perancangan *Maze*

Terdapat tiga macam elemen yang tersedia yang bisa ditambahkan atau ditempatkan di dalam *world frame,* yaitu:

1. **Mobot**: Suatu robot roda (*mobile robot*) penggerak diferensial yang konfigurasi geometri dan sensor-nya bisa diatur.
2. **Objek**: Terdapat beberapa pola (*shape*) yang tersedia yang mewakili objek-objek rintangan dalam *world frame*. Objek-objek ini dapat dideteksi oleh Mobot melalui sensor jaraknya, selain itu bisa juga ditabrak oleh mobot yang bersangkutan.
3. ***Mark*:** *Mark* atau tanda merupakan pasangan koordinat X dan Y yang dapat digunakan sebagai acuan visual, misalnya sebagai titik *finish* mobot yang harus dicapai.

Pola-pola yang ditampilkan pada Gambar 4.7 diatas digunakan untuk menambahkan berbagai macam objek ke dalam *world frame*, seperti: robot, mark, garis, persegi panjang, persegi panjang bundar, elips, busur, sektor kord, Gambar bebas maupun poligon.

Untuk menambahkan rintangan, harus dipilih salah satu pola yang ada pada elemen-elemen pada mobotsim dilanjutkan dengan mulai menggambar rintangan sesuai dengan pola yang dipilih. Berikut adalah gambar *maze* yang telah dibangun, *maze* ini digunakan untuk semua simulasi:

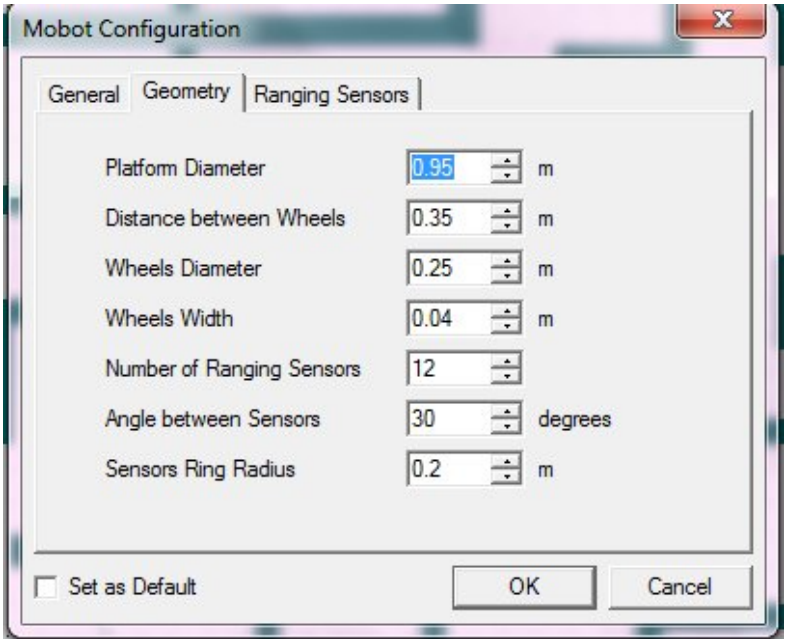


Gambar 4.8 *Maze* Dengan Rintangan

Maze yang dirancang bisa saja berbeda sesuai dengan keinginan dan kebutuhan simulasi. Bentuk rintangan dan lebar lintasan juga bisa berubah-ubah. Yang perlu diingat adalah kondisi *maze* yaitu posisi *start*, *finish* dan lebar lintasan, ketiga *point* itu akan sangat berpengaruh pada setting konfigurasi robot dan juga sensor.

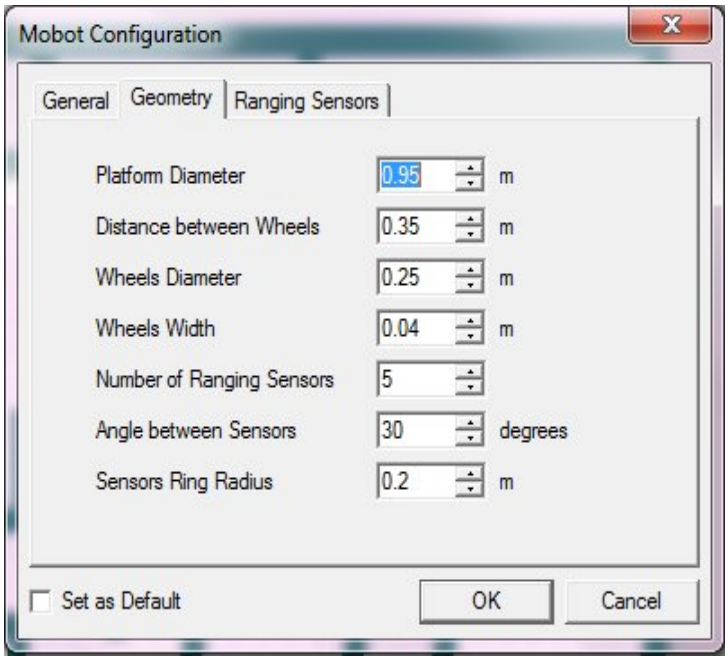
4.3.2 Perancangan Properti Geometri dan Jarak Sensor

Perancangan geometri robot berbeda berdasarkan prioritasnya, untuk algoritma *pledge* dengan *right wall priority* properti geometrinya sama dengan metode *wall follower* dengan *right hand rule*. Untuk lebih jelasnya properti geometri dari dua metode ini dapat dilihat pada Gambar 4.9:



Gambar 4.9 Properti Geometri untuk Prioritas Kanan

Properti geometri pada algoritma *pledge* dengan *rigth wall priority* dan *wall follower* dengan *right hand rule* juga sama. Properti geometri dari kedua algoritma ini dapat dilihat pada Gambar 4.10 dibawah ini:



Gambar 4.10 Properti Geometri untuk Prioritas Kiri

Berdasarkan dari dua properti geometri pada Gambar 4.9 dan Gambar 4.10 dapat dijelaskan bahwa:

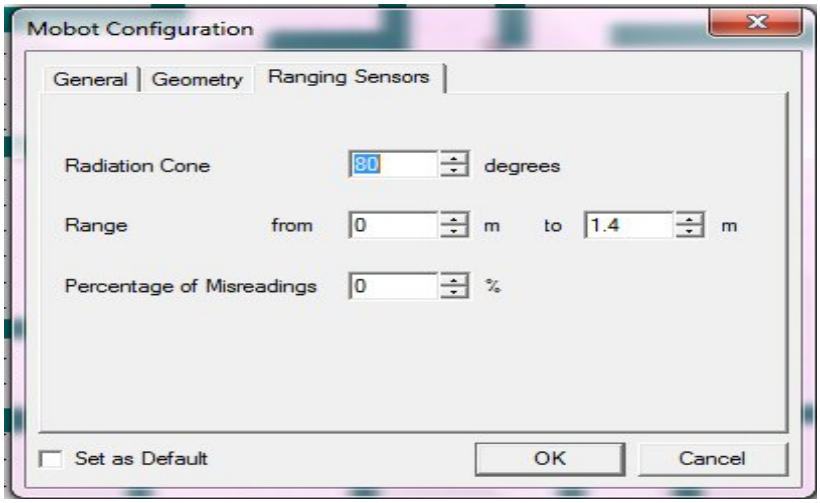
1. ***Platform diameter***: Diameter *platform* lingkaran robot dalam satuan meter, pada simulasi ini diameter robot adalah 0.95 meter.
2. ***Distance between wheels***: Jarak antara roda dari pusat-ke-pusat sumbu aksis dalam satuan meter, jarak yg ditentukan yaitu 0.35 meter.
3. ***Wheels diameter***: Diameter roda mobot sebesar 0.25 meter
4. ***Wheels width***: Luas roda 0.04 meter, sebenarnya *wheels width* ini hanya untuk pengGambaran saja, karena akan diabaikan dalam komputasi dinamik
5. ***Number of ranging sensors***: Jumlah sensor yang terintegrasi dengan mobot pada sisi-sisi platform (melingkar) dalam simulasi ini jumlah sensor yang digunakan pada prioritas kanan algoritma *pledge* dan *wall follower* adalah 12. Sedangkan sensor untuk prioritas kiri dari kedua algoritma tersebut berjumlah 5. Mengapa sensor yang digunakan berbeda? Karena penelusuran yang dilakukan oleh kedua prioritas arah ini lebih akurat pembacaannya dengan 12 sensor untuk prioritas kanan dan 5 sensor untuk prioritas kiri. Namun ketika menelusuri *maze* hanya 1 sensor yg digunakan, hal ini bertujuan untuk meminimalisir *error*. Sensor nomor 0 berada pada sisi terjauh dan indeks sensor naik seiring dengan arah berlawanan dengan jarum jam (*anti clockwise*). Jumlah sensor ini nantinya akan berkaitan dengan jarak sensor yang digunakan.
6. ***Angel between sensors***: Sudut antara dua sensor berurutan, yaitu 30°.
7. ***Sensor ring radius***: Jarak atau radius sensor sejauh 0.2 meter

Untuk ilustrasi mengenai jumlah sensor yang digunakan dapat dilihat dalam Bab II pada Gambar 2.12.

Setelah properti geometri selesai diatur, langkah selanjutnya ialah perancangan jarak sensor (*Ranging Sensor*). Jarak sensor pada tiap-tiap simulasi berbeda, hal ini bertujuan untuk keakuratan pembacaan dinding dan rintangan. Untuk lebih lengkapnya dapat dilihat pada gambar-gambar berikut:

1. Ranging sensor pada *Pledge* dengan *right wall priority*

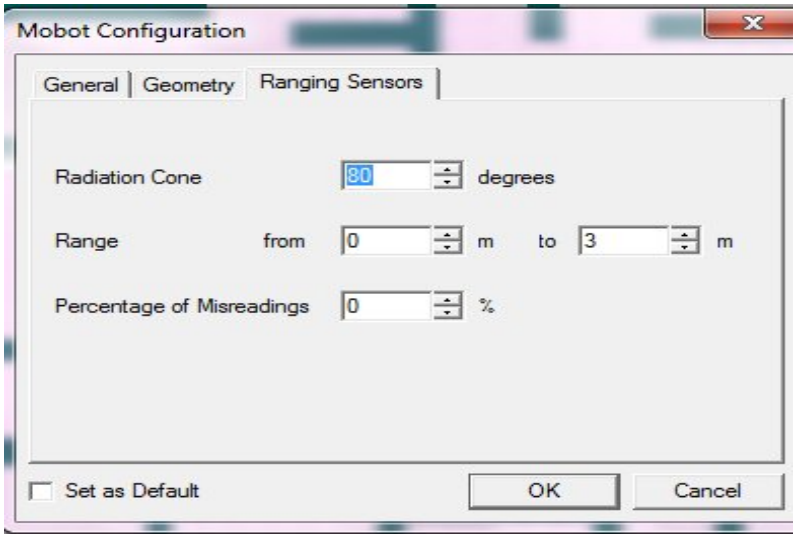
   Ranging sensor pada *Pledge* dengan *right wall priority* dapat dilihat pada gambar dibawah ini:



Gambar 4.11 Ranging Sensor Algoritma *Pledge* Dengan *Right Wall Priority*

2. Ranging sensor pada *Pledge* dengan *left wall priority*

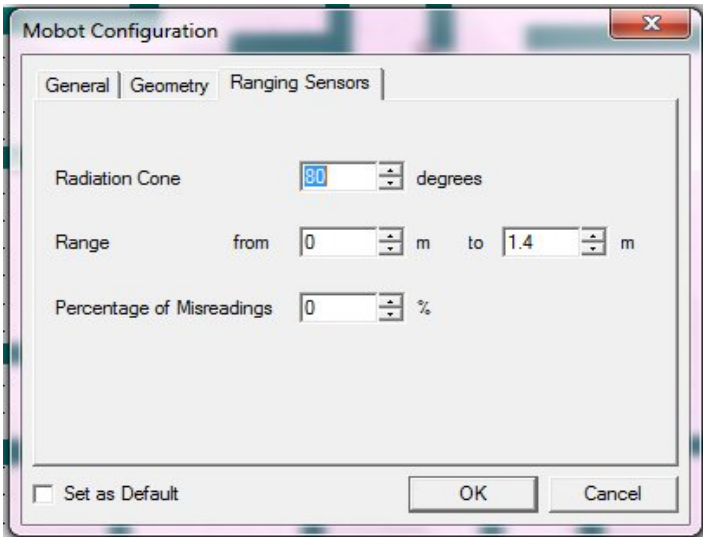
   Ranging sensor pada *Pledge* dengan *left wall priority* dapat dilihat pada Gambar dibawah ini:



Gambar 4.12 Ranging Sensor Algoritma *Pledge* Dengan *Left Wall Priority*

3. Ranging sensor pada *wall follower* dengan *right hand rule*

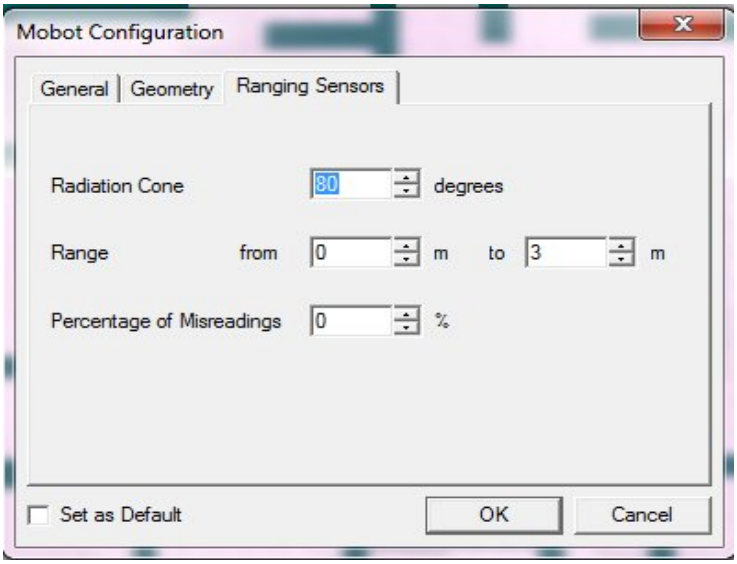
   Ranging sensor pada *wall follower* dengan *right hand rule* dapat dilihat pada Gambar dibawah ini:



Gambar 4.13 Ranging Sensor Algoritma *Wall Follower* Dengan *Right Hand Rule*

4. Ranging sensor pada *wall follower* dengan *left hand rule*

   Ranging sensor pada *wall follower* dengan *left hand rule* dapat dilihat pada Gambar dibawah ini:



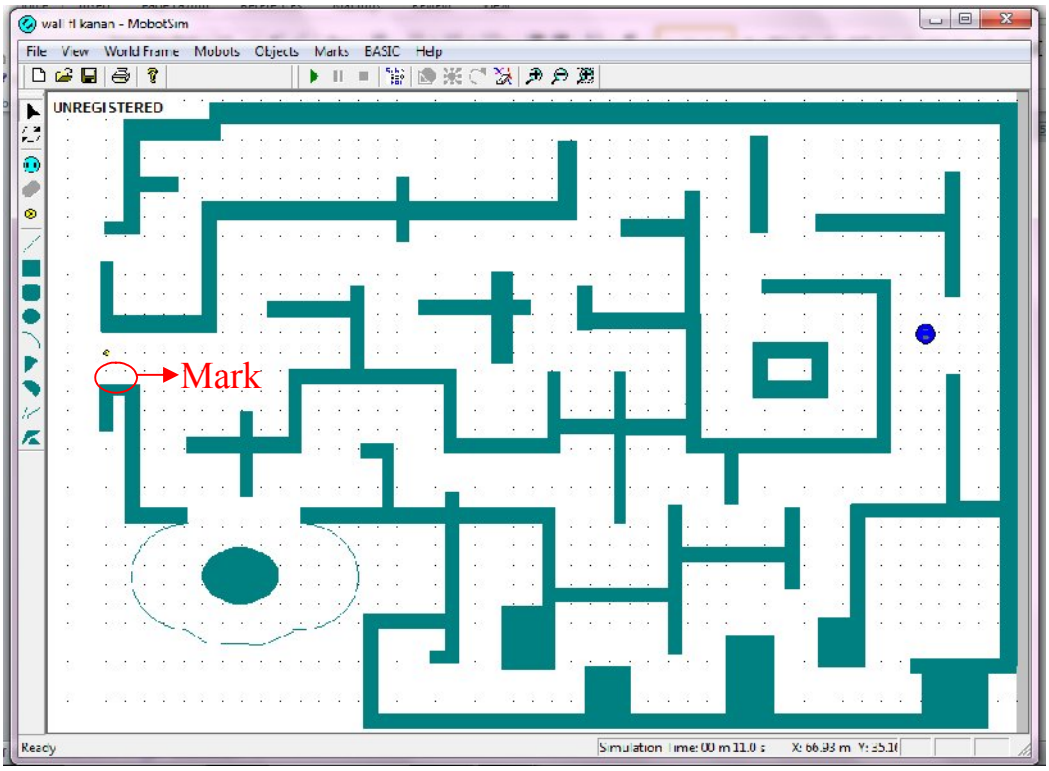
Gambar 4.14 Ranging Sensor Algoritma *Wall Follower* Dengan *Left Hand Rule*

Berdasarkan Gambar 4.11, 4.12, 4.13 dan 4.14 dapat dijelaskan mengenai jarak sensor yang digunakan pada masing-masing simulasi yaitu:

1. ***Radiation cone*:** Radiasi sensor dengan pendekatan kerucut (*cone)*. Besar *radiation cone* pada semua simulasi sama, yaitu 80 derajat. Nilai ini merupakan nilai sudut sektor yang merupakan penyajian bi-dimensional dari kerucut tersebut.
2. ***Range from-to*:** Jarak minimum dan maksimum yang dapat dideteksi sensor. *Range* ini berhubungan dengan jumlah sensor dan *radiation cone* yang digunakan. Semakin jauh atau dekat radiasi sensor maka akan berpengaruh terhadap akurasi pembacaan dinding atau rintangan. Hal ini yang membedakan range pada setiap simulasi.
3. ***Percentange of misreading*:** kita dapat dengan sengaja membuat suatu persentase kesalahan pembacaan sensor. Kesalahan pembacaan akan dilakukan secara acak dan nilai diatur sebagai suatu bilangan acak yang sama dengan probabilitas dari keseluruhan jarak. Jika terjadi kesalahan pembacaaan, maka radiasi kerucut sensor yang bersangkutan akan berwarna lain. Untuk meminimalisir *error* maka persentase kesalahan pembacaan sensor diberi nilai 0.

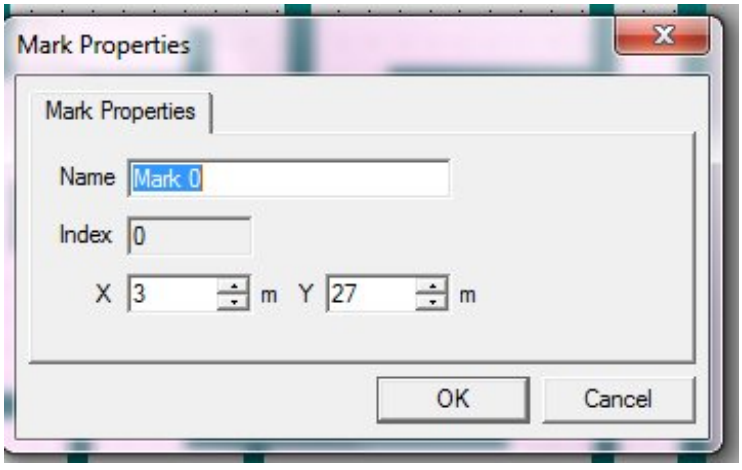
4.3.3 Perancangan *Mark* atau Titik *Finish*

Titik *finish* dibutuhkan untuk memberi tanda kepada robot sebagai penelusuran akhir dimana robot harus berhenti. Dalam *pledge* biasanya titik *start* dan *finish* berada dalam posisi arah mata angin yang sama. Ketika robot berjalan dari titik *start* dan masih mengarah kepada titik *finish* maka robot masih berada pada arah utama dan ketika robot menemui rintangan maka robot akan melakukan penelusuran sampai robot dapat membaca *mark* atau titik *finish* dan berhenti. Posisi titik *finish* pada *maze* yang telah dirancang dapat dilihat pada Gambar 4.15:



Gambar 4.15 Posisi *Mark* atau Titik *Finish*

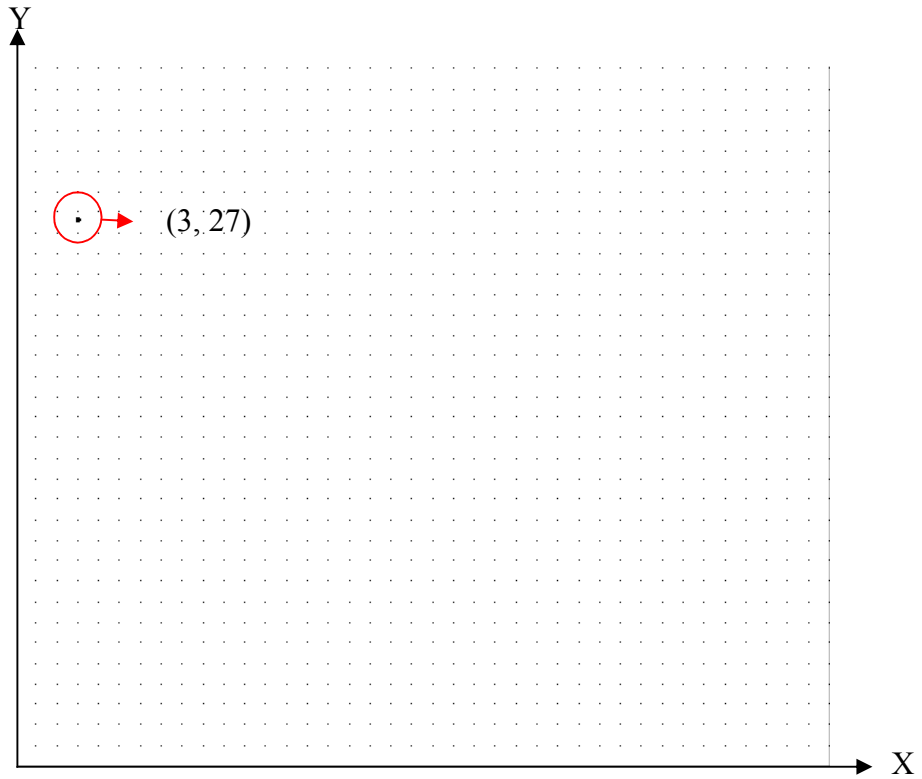
Untuk mengantur nama dan kordinar *mark* dapat dilkukan pada *Mark Properties* seperti Gambar 4.16 dibawah ini:



Gambar 4.16 *Mark Properties*

Berdasarkan *mark properties* yang ditunjukkan pada Gambar 4.16 dapat dijelaskan properti yang dapat diatur meliputi:

1. ***Name*:** Masing-masing tanda memiliki sebuah nama yang unik. Merupakan *string* dengan panjang maksimum 30 karakter. Jika tidak diubah, maka nama *default*-nya adalah *Mark* 0, *Mark* 1 dan seterusnya. Disini nama *mark* yang digunakan adalah *Mark* 0.
2. **Index:** Merupakan sebuah bilangan bulat unik untuk masing-masing tanda. Mobotsim akan memberikan bilangan ini secara dinamis seiring dengan penambahan maupun penghapusan tanda dan tidak bisa diubah oleh pengguna. Karena *mark* yang digunakan hanya 1 maka index yang diberikan adalah 0.
3. **X, Y:** Menentukan posisi tanda dalam meter di dalam *world frame*. Dalam *maze* yang telah dirancang, posisi *mark* berada pada posisi X = 3 dan Y = 27. Untuk lebih jelasnya posisi *mark* dapat dilihat pada Gambar 4.17 dibawah ini.



Gambar 4.17 Koordinat Posisi *Mark*

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1. Tahapan Implementasi

Tahapan implementasi merupakan suatu tahapan penting dalam membangun suatu sistem perangkat lunak. Tahapan ini adalah rangkaian akhir dari semua tahapan yang telah lakukan sebelumnya, tahapan ini nantinya akan memberikan gambaran tentang kelayakan aplikasi yang telah dibuat. Berikut ini akan dijelaskan tentang pengimplementasian dari analisis dan perancangan yang telah dilakukan terhadap analisa algoritma *pledge*.

5.1.1 Batasan Implementasi

Batasan implementasi dari Tugas Akhir ini adalah:

1. Menggunakan simulator MOBOTSIM v.10 dengan bahasa pemrograman Basic.
2. Perangkat lunak analisa algoritma *pledge* untuk menyelesaikan *maze* dengan robot *wall maze* ini hanya merupakan simulasi pergerakan robot dengan menggunakan algoritma *pledge.* Agar nantinya analisa yang telah dilakukan dapat diuji kebenarannya dengan mensimulasikan analisa tersebut dengan simulator mobotsim.
3. Ralat motor, roda yang selip, arah jatuh, kecepatan robot dan sifat permukaan objek diabaikan.
4. Robot dijalankan sesuai dengan kebutuhan algoritma dan *maze* yang dirancang.
5. Simulasi dengan algoritma *wall follower* dirancang hanya sebagai pembanding kinerja algoritma *pledge.*

5.1.2 Lingkungan Implementasi

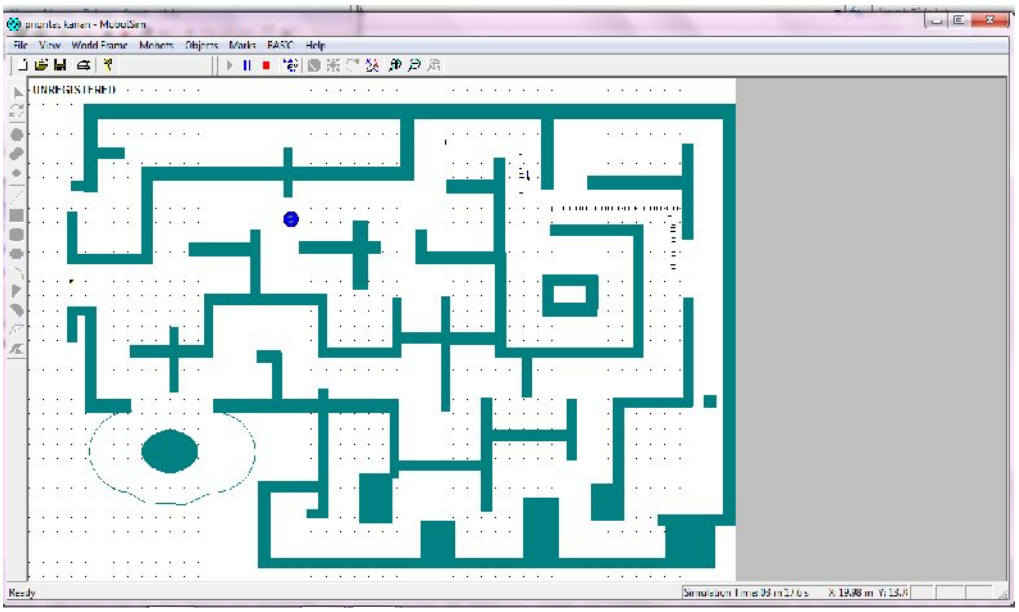
Lingkungan implementasi simulasi ini terdiri dari dua lingkungan yaitu, lingkungan perangkat keras dan lingkungan perangkat lunak.

Berikut adalah spesifikasi lingkungan implementasi perangkat keras dan perangkat lunak:

1. Perangkat Keras Komputer
   a. *Processor* : *Intel Pentium Core 2 Duo* 2.00 GHz
   b. *Memory* : 1 GB
   c. *Hard disk* : 250 GB
2. Perangkat Lunak Komputer
   a. Sistem Operasi : *Windows 7 Profesional 32-bit Operating System*
   b. Simulator : MOBOTSIM V.10
   c. Bahasa Pemrograman : Basic

5.1.3 Implementasi Simulator dan *Interface*

Setelah tahap analisa dan perancangan selesai dilakukan, maka dilanjutkan dengan tahap implementasi simulator dan i*nterface* dari hasil analisa yang telah diperoleh dan mengimplementasikan hasil perancangan simulasi yang telah dibuat. Berikut ini adalah gambaran dari implementasi *Interface* dari *maze* yang telah dirancang dalam simulator Mobotsim v.10:



Gambar 5.1 Hasil Implementasi *Interface*

Gambar 5.1 menjelaskan bahwa perangkat lunak dari analisis perbandingan algoritma *pledge* dengan algoritma *wall follower* pada robot *wall maze* yang telah dirancang berhasil diimplementasikan ke dalam simulator mobotsim. Tampilan pada Gambar 5.1 akan muncul ketika pertama kali *user* membuka aplikasi ini. *Maze* yang digunakan dalam setiap simulasi sama, namun penulusuran yang dilakukan robot ketika *user* menekan tombol *play* yang berada pada *toolbar world frame* berbeda sesuai dengan algoritma yang digunakan. Ketika robot sampai di titik *finish user* dapat menekan tombol *stop* untuk menghentikan waktu simulasi.

5.2. Tahapan Pengujian

Tahapan pengujian dilakukan untuk menguji kesiapan sistem atau aplikasi sebelum digunakan oleh *user*. Tahapan pengujian bertujuan untuk mendeteksi *error* dalam suatu aplikasi kemudian melakukan perbaikan.

Untuk kasus simulasi Robot dengan mobotsim pengujian dilakukan atas beberapa hal:

1. Pengujian Algoritma
2. Pengujian Terhadap Radiasi sensor (*Radiation cone)*

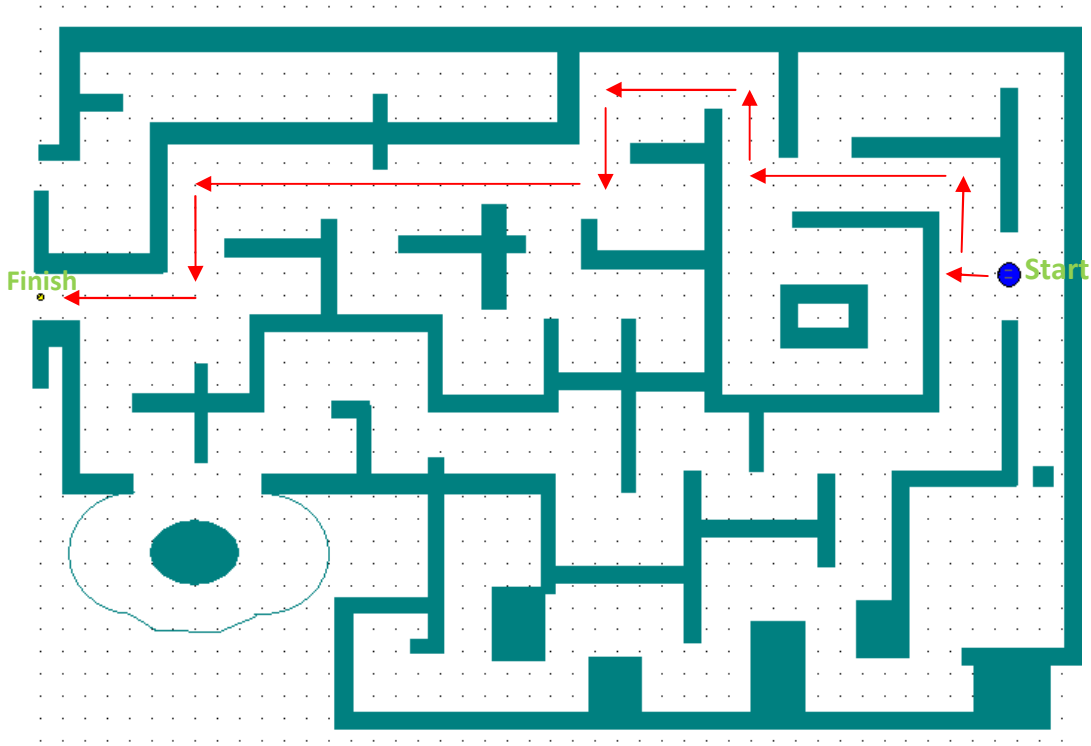
3. Pengujian Waktu Tempuh ke tititk *finish*
4. Pengujian terhadap nilai jarak sensor ke rintangan
5. Pengujian dengan posisi *start* dan *finish* yang berbeda
6. Pengujian dengan kondisi *maze* yang berbeda.

5.2.1 Pengujian Algoritma

Pengujian ini dimaksudkan untuk mengetahui keberhasilan robot menemukan jalan keluar. Jalan keluar ini adalah sebagai *finish* robot yang nantinya digunakan untuk menunjukkan bahwa robot sudah menemukan jalan keluar dan dapat berhenti melakukan penelusuran. Pengujian algoritma dilakukan berdasarkan algoritma dengan prioritas-prioritasnya masing-masing.

1. Pengujian Algoritma *pledge* dengan *right wall priority*

Pengujian Algoritma *pledge* dengan *right wall priority* dapat dilihat pada gambar dibawah ini:

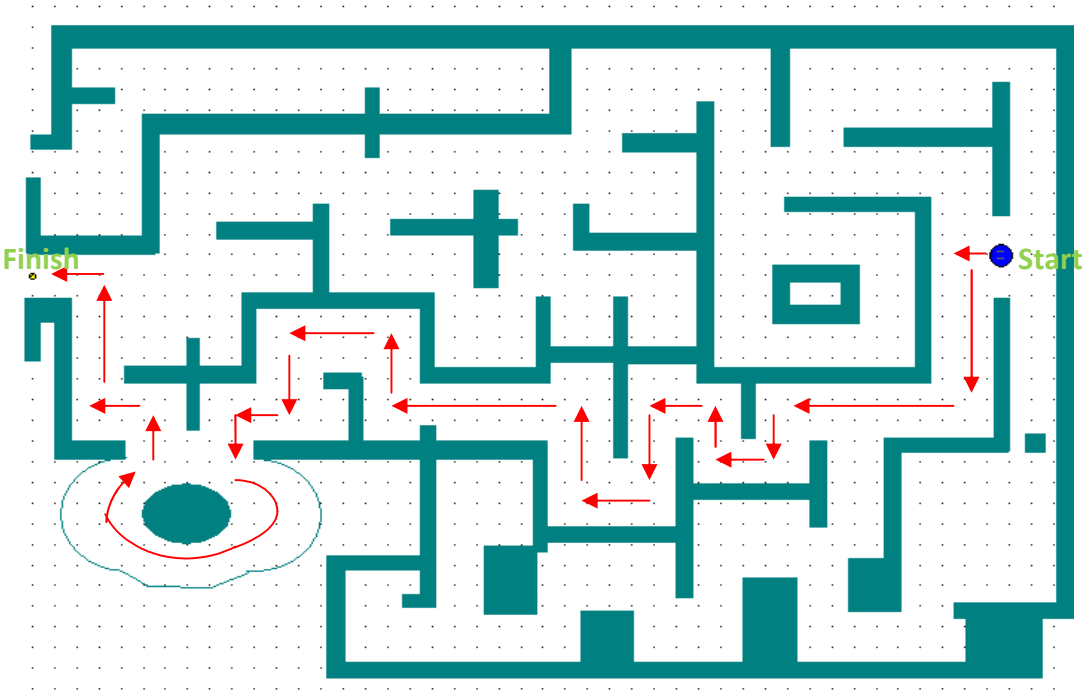


Gambar 5.2 Penelusuran Algoritma *Pledge* Dengan *Right Wall Priority*

Berdasarkan Gambar 5.2 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *pledge* dengan *right wall priority*. Arah utama robot adalah arah dimana robot pertama kali dijalankan. Halangan pertama dijumpai pada simpangan pertama yang memiliki dua simpang yaitu kiri dan kanan, karena menggunakan *right wall priority* maka robot harus berbelok ke kanan. Pada simpangan kedua robot menemui halangan di dinding sebelah kanan, maka robot harus berbelok ke kiri dan saat itu robot berjalan sesuai dengan arah utama, karena itu robot akan tetap berjalan lurus walaupun disebelah kanan terdapat persimpangan. Ketika robot menemui halangan di depan, maka robot akan kembali kepada konsep prioritasnya yaitu meprioritaskan dinding sebelah kanan dan melakukan teknik *wall follower* sampai menemui titik *finish*. Ketika robot sampai di titik *finish* maka robot akan otomatis berhenti melakukan penelusuran.

2. Pengujian Algoritma *pledge* dengan *left wall priority*

Pengujian Algoritma *pledge* dengan *left wall priority* dapat dilihat pada gambar dibawah ini:

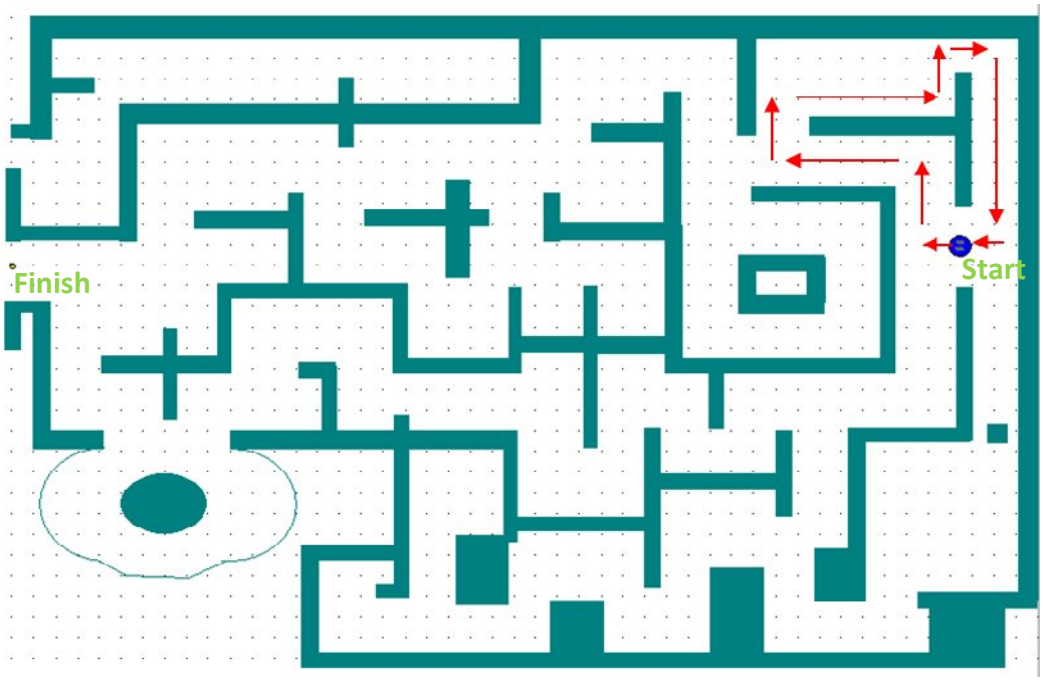


Gambar 5.3 Penelusuran Algoritma *Pledge* dengan *Left Wall Priority*

Berdasarkan Gambar 5.3 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *pledge* dengan *left wall priority*. Sama seperti *pledge* dengan *right wall priority*, arah utama robot adalah arah dimana robot pertama kali dijalankan. Karena menggunakan *left wall priority*, maka ketika robot menjumpai halangan berupa simpang tiga, robot harus berbelok ke kiri. Ketika robot menemui halangan di depan dan dinding kiri, maka robot harus berbelok ke kanan. Setelah berbelok ke kanan, robot berjalan lurus searah dengan arah utama, maka robot akan tetap berjalan lurus sampai menemui halangan di depannya. Saat menjumpai halangan maka robot kembali menelusuri *maze* dengan teknik *wall follower* yang tentunya menggunakan *left wall priority* sampai robot dapat menghitung jarah ke titik *finish* dan berhenti.

3. Pengujian Algoritma *wall follower* dengan *right hand rule*

Karena algoritma *wall follower* digunakan dalam tugas akhir ini sebagai pembanding kinerja algoritma *pledge* maka perlu dilakukan pengujian juga terhadap algoritma *wall follower* ini. Pengujian Algoritma *wall follower* dengan *right hand rule* dapat dilihat pada gambar dibawah ini:

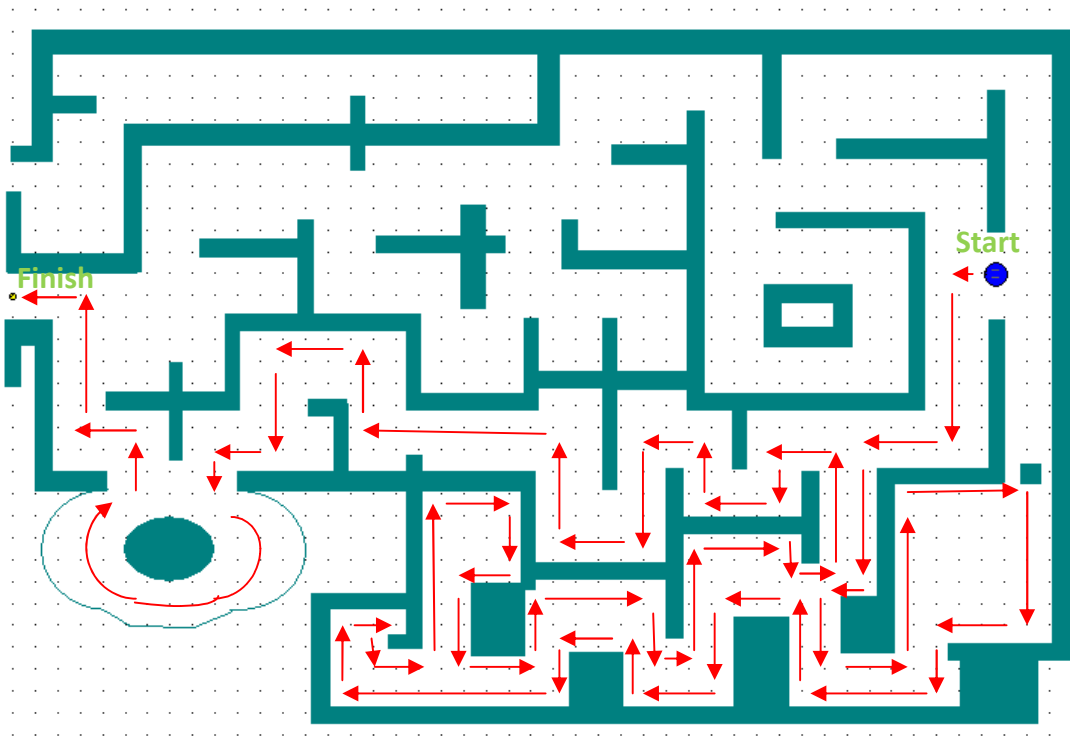


Gambar 5.4 Penelusuran Algoritma *Wall Follower* dengan *Right Hand Rule*

Berdasarkan Gambar 5.4 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *wall follower* dengan *right hand rule.* Pada algoritma ini robot hanya diperintahkan untuk melakukan penelusuran dengan memprioritaskan dinding sebelah kanan. Sehingga pada persimpangan pertama robot langsung berbelok ke kanan. Ketika menemui halangan di simpang kedua maka robot berbelok ke kiri dan kembali melakukan penelusuran dengan tetap memprioritaskan dinding kanan, sehingga pada *maze* dengan bentuk melingkar seperti pada Gambar 5.4 robot akan terus melakukan penelusuran dan kembali ke titik *start.* Sehingga penelusuran dengan menggunakan algoritma *wall follower* dengan *right hand rule* ini tidak berhasil mencapai titik *finish*.

4. Pengujian Algoritma *wall follower* dengan *left hand rule*

Berbeda dengan algoritma *wall follower* dengan *right hand rule,* algoritma *wall follower* dengan *left hand rule* lebih mendahulukan dinding sebelah kiri dari pada kanan. Pengujian Algoritma *wall follower* dengan *left hand rule* dapat dilihat pada gambar dibawah ini:



Gambar 5.5 Penelusuran Algoritma *Wall Follower* dengan *Left Hand Rule*

Berdasarkan Gambar 5.5 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *wall follower* dengan *left hand rule.* Pada algoritma ini robot hanya diperintahkan untuk melakukan penelusuran dengan memprioritaskan dinding sebelah kiri. Sehingga pada persimpangan pertama robot langsung berbelok ke kiri. Ketika sampai pada persimpangan kedua robot menemui halangan di depan maka robot langsung berbelok ke kanan. Saat robot berjalan lurus, robot kembali mengikuti dinding kiri dan menelusuri *maze*. Ketika robot berada pada ruangan *maze* yang tidak memiliki jalan keluar, robot tetap menelusurinya dengan prioritas dinding kiri dan akhirnya robot kembali ke arah awal robot bergerak. Kemudian robot kembali melakukan teknik *wall follower* dengan *left hand rule* sampai robot dapat sampai pada titik *finish* dan berhenti melakukan penelusuran.

Setelah melakukan pengujian terhadap semua algoritma yang digunakan maka kesimpulan yang didapat adalah:

Tabel 5.1 Kesimpulan pengujian Algoritma

No	Algoritma	Pencapaian ke titik *finish*	
		Berhasil	Tidak berhasil
1	*Pledge* dengan *Right wall priority*	√	
2	*Pledge* dengan *Left wall priority*	√	
3	*Wall Follower* dengan *Right hand rule*		√
4	*Wall Follower* dengan *Left hand rule*	√	

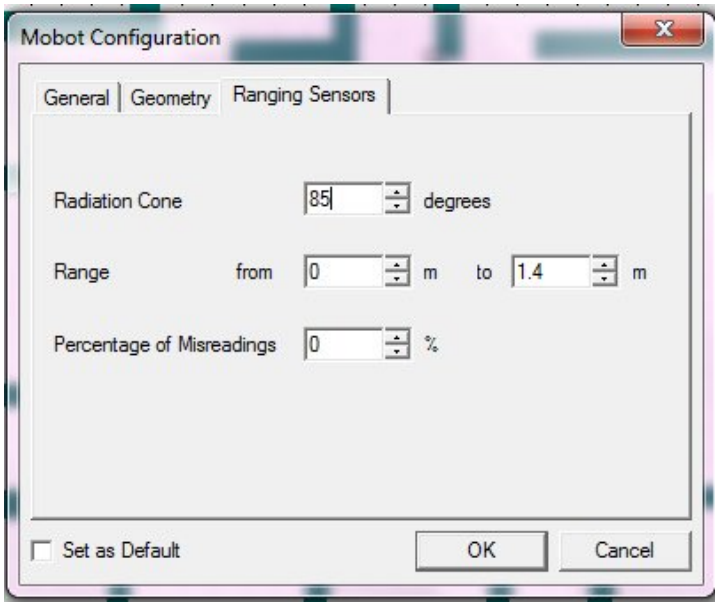
5.2.2 Pengujian Terhadap Radiasi Sensor (Radiation Cone)

Radiation cone mempunyai pengaruh yang sangat besar terhadap proses penelusuran *maze*, khususnya pembacaan rintangan atau halangan di dalam *maze.* Semakin besar *radiation cone* yang digunakan maka akan semakin akurat pembacaan sensor terhadap lingkungan *maze* ataupun halangan. Sebaliknya, semakin kecil *radiation cone* yang digunakan, maka akan semakin besar

kemungkinan *error* yang akan terjadi dalam pembacaan sensor terhadap halangan. *Radiation cone* juga berhubungan dengan luas area lintasan atau jarak antara dinding atau halangan dalam sebuah *maze.* Dalam tugas akhir ini, simulasi yang dirancang memiliki *setting* konfigurasi yang sama pada penggunaan *radiation cone* untuk semua simulasi dengan algoritma yang berbeda yaitu 80 derajat dan jarak antara dinding *maze* berbeda-beda, yaitu diantara 2-4 meter. Pengujian yang dilakukan dimulai dari nilai *radiation cone* > 80° yaitu 85°, hingga nilai *radiation cone* tidak dapat melakukan pembacaan lagi terhadap dinding hingga menyebabkan *error.* Berikut ini akan dijelaskan hasil pengujian menggunakan *radiation cone:*

1. Pengujian *radiation cone* pada algoritma *pledge* dengan *right wall priority*:
   a. *Radiation cone* dengan parameter nilai 85° - 65°

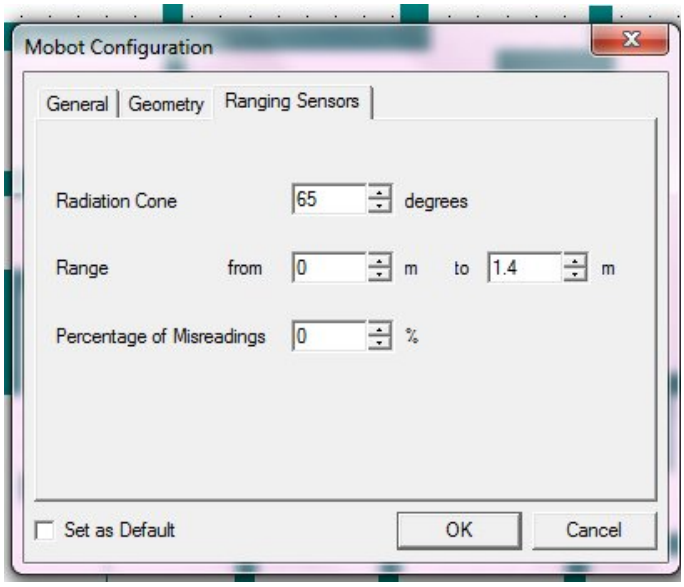
      Radiation cone dengan parameter nilai 85° dapat dilihat pada gambar di bawah ini:



Gambar 5.6 Parameter nilai 85°

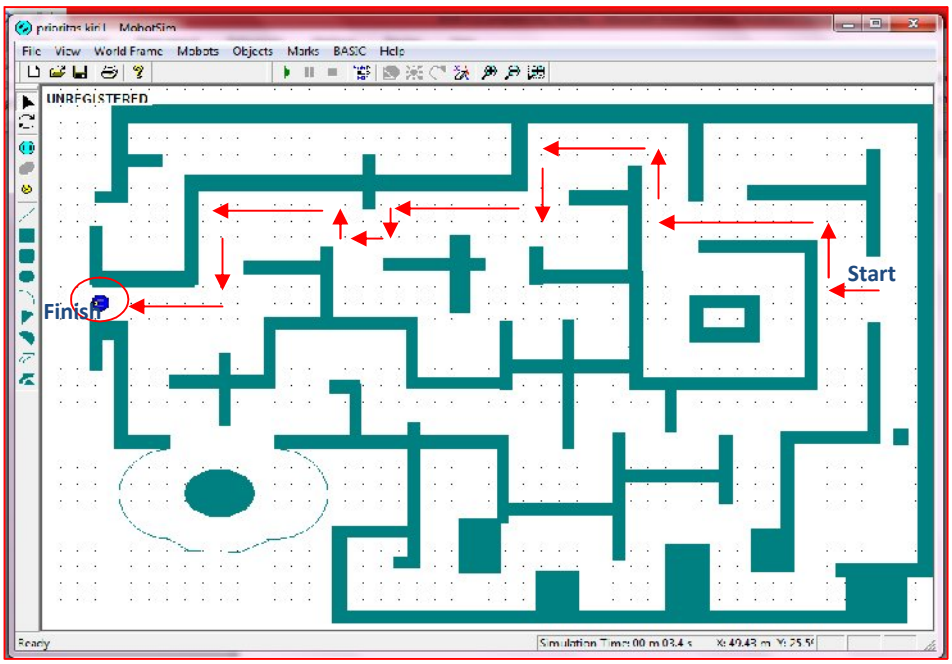
b. *Radiation cone* dengan parameter nilai 65°

Radiation cone dengan parameter nilai 65° dapat dilihat pada gambar di bawah ini:



Gambar 5.7 Parameter nilai 65°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 85° - 65°

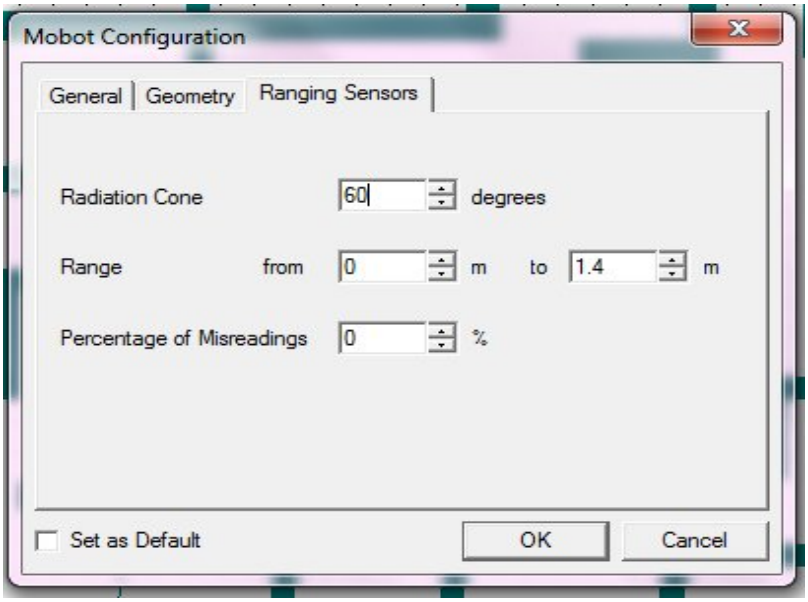


Gambar 5.8 Hasil Pengujian dengan nilai radiation cone 85° - 65°

Pada pengujian nilai radiation cone dengan parameter nilai 85° - 65° robot berhasil melakukan penelusuran *maze* sesuai rute yang benar dan sampai pada titik *finish.*

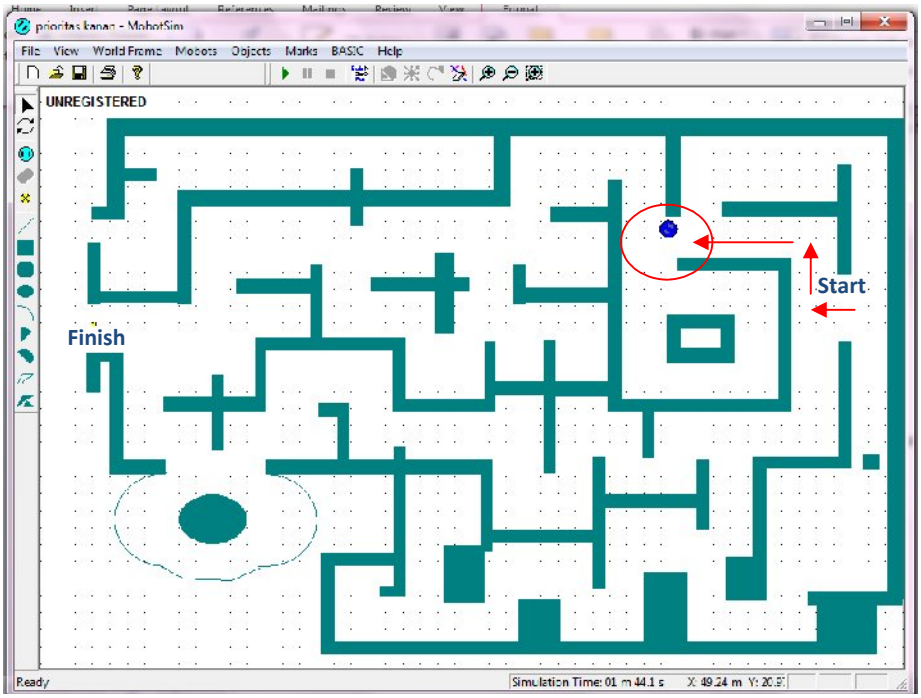
c. *Radiation cone* dengan parameter nilai 60°

Radiation cone dengan parameter nilai 60° dapat dilihat pada gambar di bawah ini:



Gambar 5.9 Parameter nilai 60°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 60°

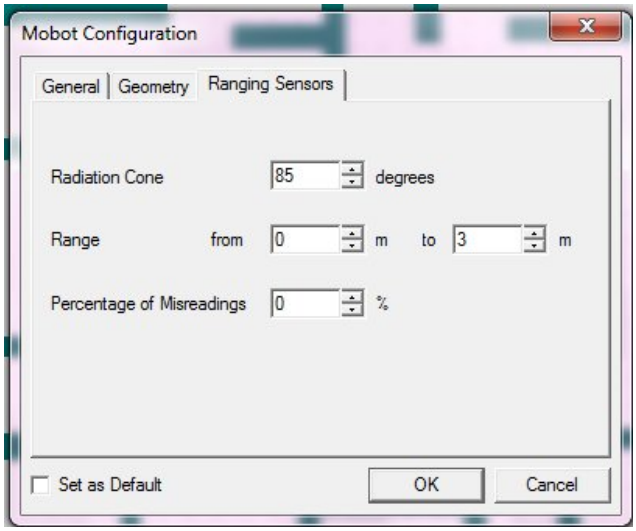


Gambar 5.10 Hasil Pengujian

Pada persimpangan ke tiga, robot tidak berhasil melakukan penelusuran ke rute yang benar lalu robot berputar-putar dan tidak sampai pada titik *finish.*

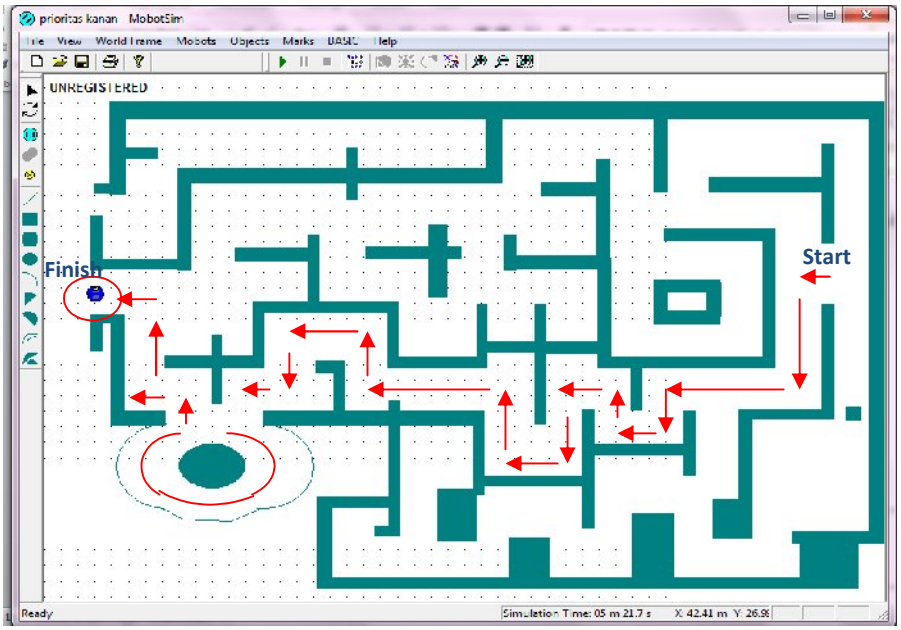
2. Pengujian *radiation cone* pada algoritma *pledge* dengan *left wall priority*:
   a. *Radiation cone* dengan parameter nilai 85°

      Radiation cone dengan parameter nilai 85° dapat dilihat pada Gambar 5.11:



Gambar 5.11 Parameter nilai 85°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 85°

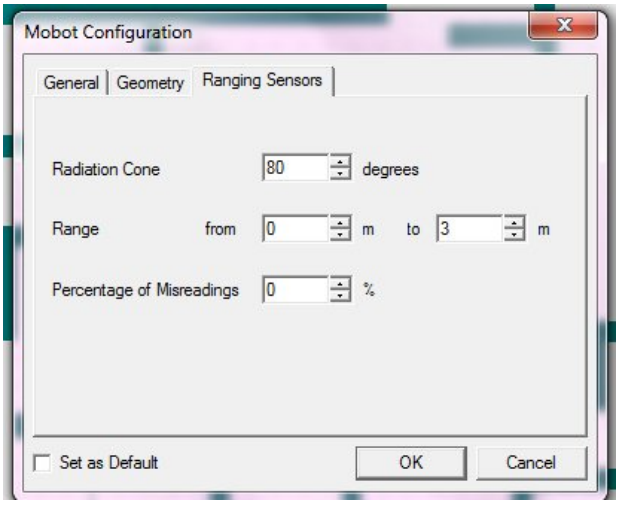


Gambar 5.12 Hasil Pengujian

Robot berhasil melakukan penelusuran *maze* sesuai rute yang benar dan sampai pada titik *finish*.

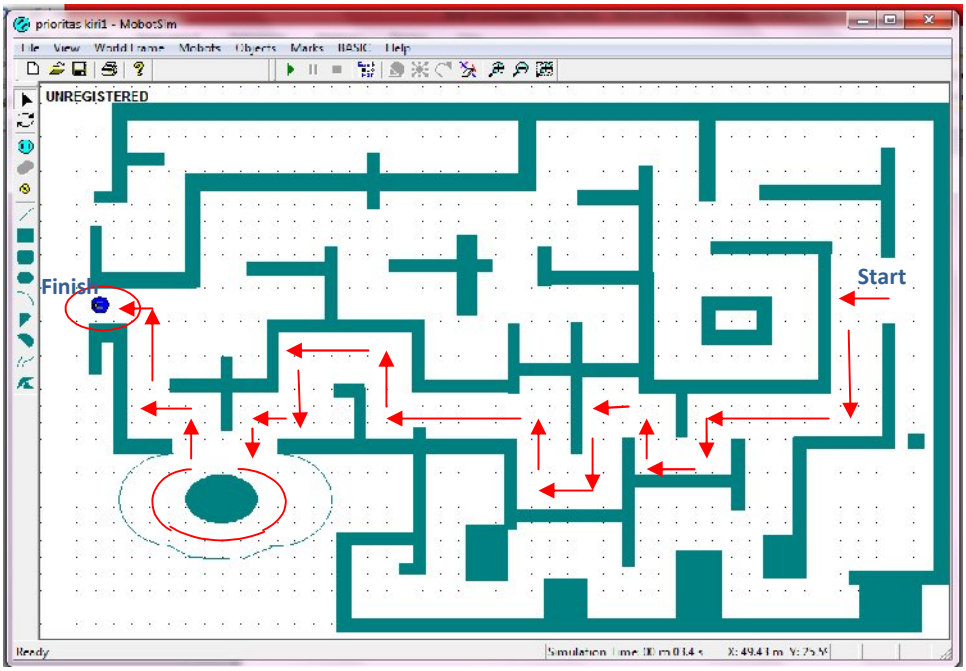
b. *Radiation cone* dengan parameter nilai 80°

Radiation cone dengan parameter nilai 80° dapat dilihat pada gambar di bawah ini:



Gambar 5.13 Parameter nilai 80°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 80°

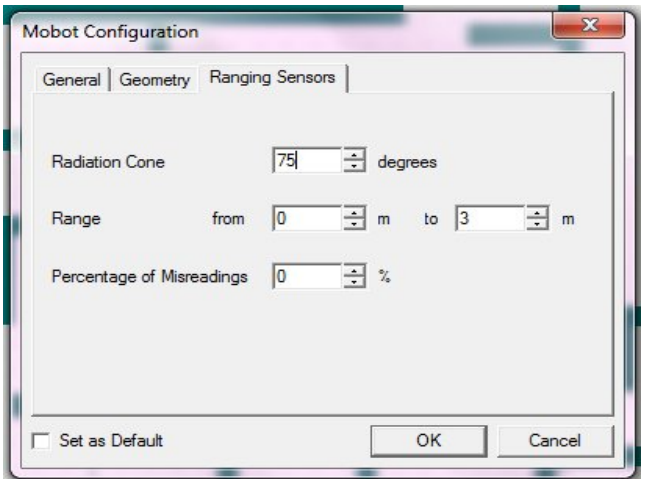


Gambar 5.14 Hasil Pengujian

Robot berhasil melakukan penelusuran *maze* sesuai rute yang benar dan sampai pada titik *finish.*

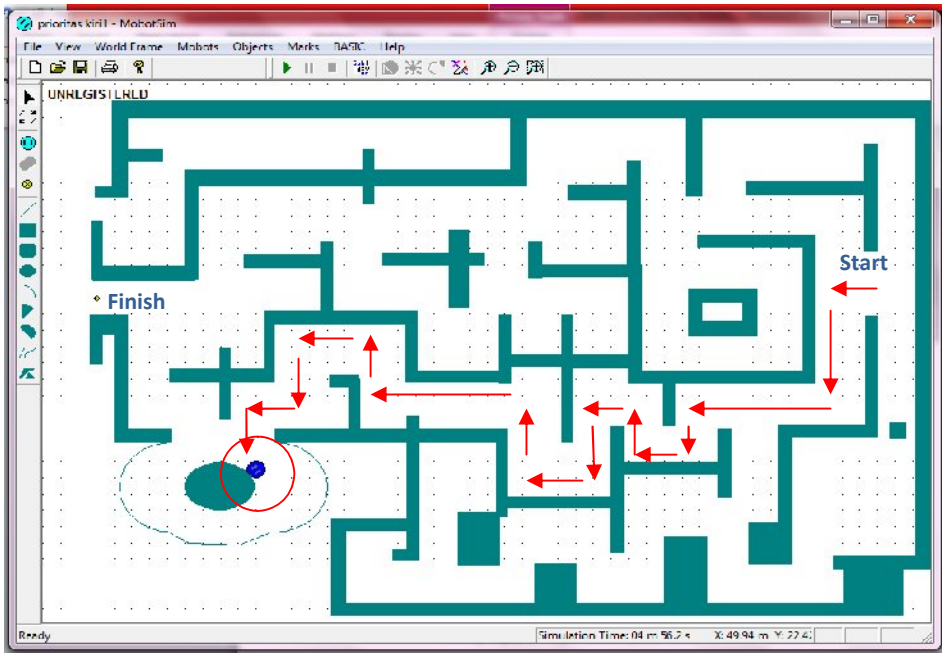
c. *Radiation cone* dengan parameter nilai 75°

Radiation cone dengan parameter nilai 75° dapat dilihat pada gambar di bawah ini:



Gambar 5.15 Parameter nilai 75°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 75°

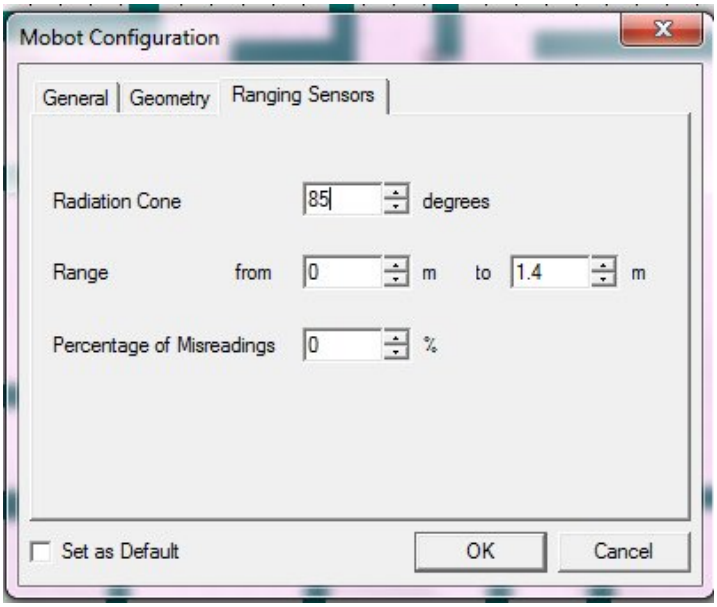


Gambar 5.16 Hasil Pengujian

Robot tidak berhasil melakukan penelusuran *maze* sesuai rute yang benar dan tidak sampai pada titik *finish.* Pada Gambar 5.16 terlihat bahwa robot tidak dapat menghindari rintangan yang ada di depannya. Dapat disimpulkan bahwa robot dengan algoritma *pledge* dengan *left wall priority* hanya dapat menelusuri *maze* dengan rute yang benar dan sampai pada titik *finish* dengan nilai *radiation cone* lebih atau sama dengan 80°

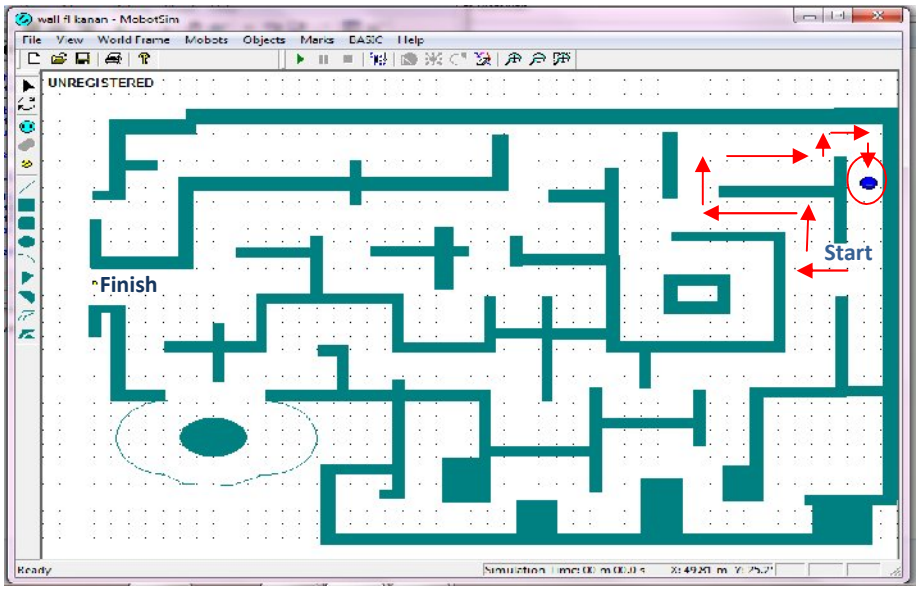
3. Pengujian *radiation cone* pada algoritma *wall follower* dengan *right hand rule*:
   a. *Radiation cone* dengan parameter nilai 85°

      Radiation cone dengan parameter nilai 85° dapat dilihat pada gambar di bawah ini:



Gambar 5.17 Parameter nilai 85°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 85°

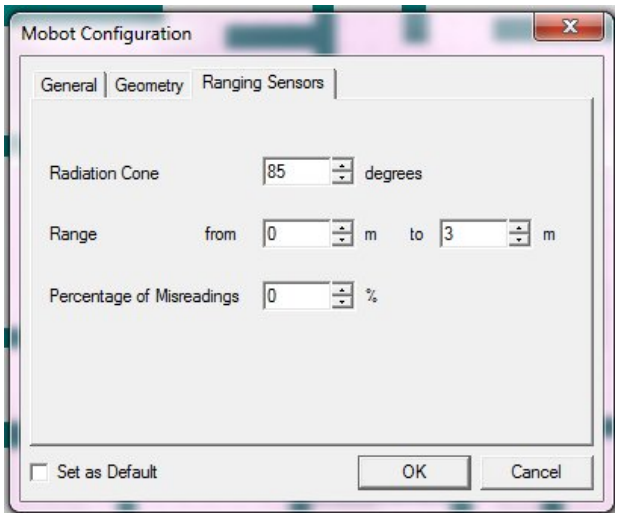


Gambar 5.18 Hasil Pengujian

Robot berhasil melakukan penelusuran *maze* sesuai rute yang benar tetapi tidak sampai pada titik *finish* dikarnakan *maze* yang dilalui memiliki rute melingkar sehingga robot hanya berjalan memutar dan tidak sampai di *finish.*

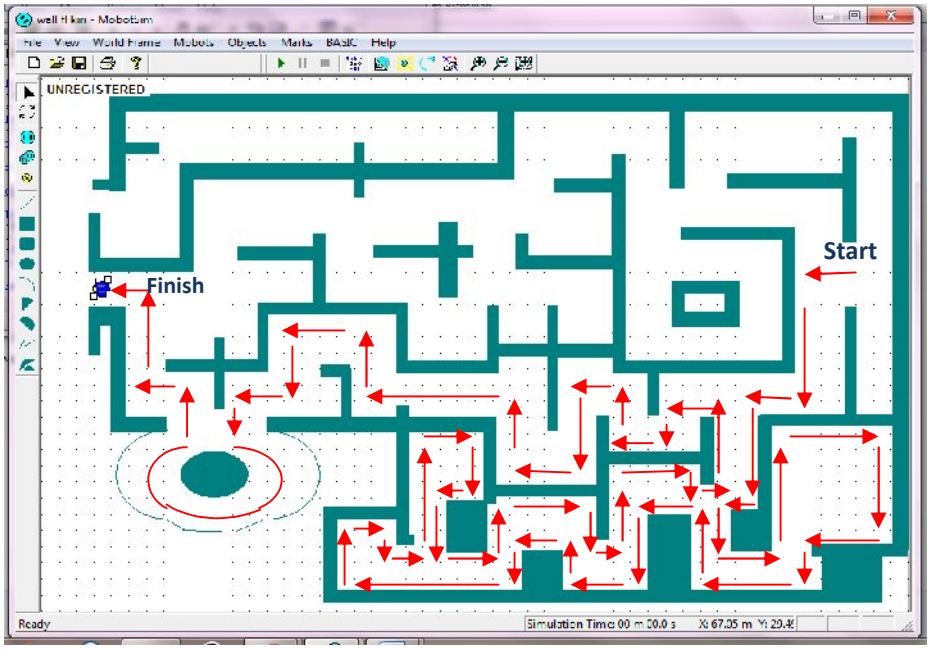
4. Pengujian *radiation cone* pada algoritma *wall follower* dengan *left hand rule*:
   a. *Radiation cone* dengan parameter nilai 85°

      Radiation cone dengan parameter nilai 85° dapat dilihat pada Gambar 5.19:



Gambar 5.19 Parameter nilai 85°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 85°

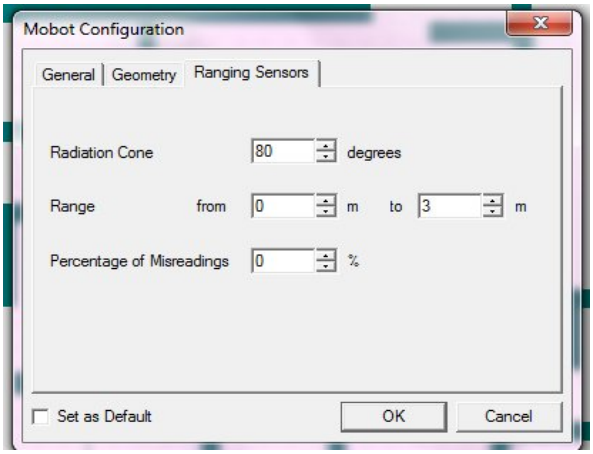


Gambar 5.20 Hasil Pengujian

Robot berhasil melakukan penelusuran *maze* sesuai rute yang benar dan sampai pada titik *finish.*

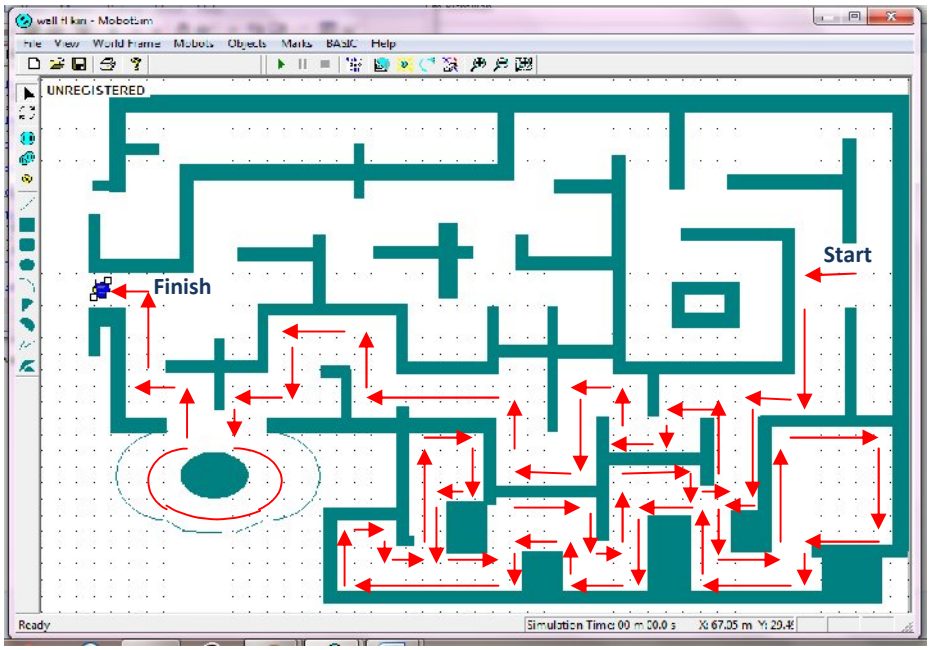
b. *Radiation cone* dengan parameter nilai 80°

Radiation cone dengan parameter nilai 80° dapat dilihat pada gambar dibawah ini:



Gambar 5.21 Parameter nilai 80°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 80°

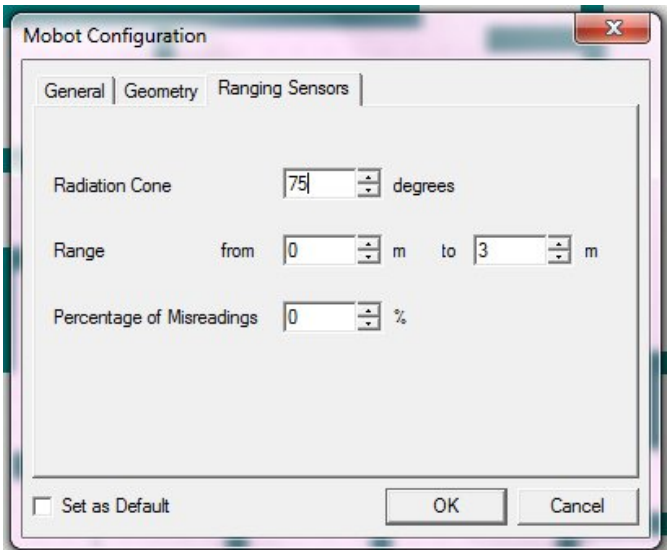


Gambar 5.22 Hasil Pengujian

Robot berhasil melakukan penelusuran *maze* sesuai rute yang benar dan sampai pada titik *finish.*

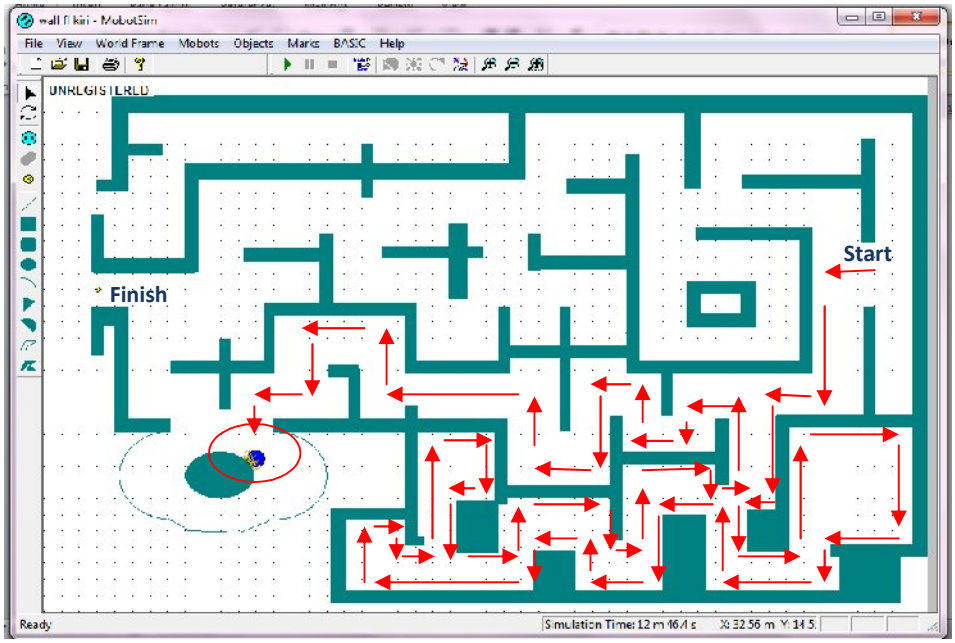
c. *Radiation cone* dengan parameter nilai 75°

Radiation cone dengan parameter nilai 75° dapat dilihat pada gambar di bawah ini:



Gambar 5.23 Parameter nilai 75°

Berikut adalah tampilan dari hasil penelusuran dengan nilai *radiation cone* 80°



Gambar 5.24 Hasil Pengujian

Robot tidak berhasil melakukan penelusuran *maze* sesuai rute yang benar dan tidak sampai pada titik *finish.* Pada Gambar 5.24 terlihat bahwa robot tidak dapat menghindari rintangan yang ada di depannya. Dapat disimpulkan bahwa robot dengan algoritma *pledge* dengan *left wall priority* hanya dapat menelusuri *maze* dengan rute yang benar dan sampai pada titik *finish* dengan nilai *radiation cone* lebih atau sama dengan 80°

Setelah melakukan pengujian terhadap semua algoritma yang digunakan maka kesimpulan yang didapat adalah:

Tabel 5.2 Kesimpulan Hasil pengujian *Radiation Cone*

Radiation Cone	Algoritma				*Keterangan*
	Pledge		*Wall Follower*		
	Right	*Left*	*Right*	*Left*	
85°	√	√	√	√	Semua Berhasil
80°	√	√	√	√	Semua Berhasil
75°	√	X	√	X	Prioritas *Right* berhasil, Prioritas *left* tidak berhasil
70°	√	X	√	X	Prioritas *Right* berhasil, Prioritas *left* tidak berhasil
65°	√	X	√	X	Prioritas *Right* berhasil, Prioritas *left* tidak berhasil
60°	X	X	X	X	Semua tidak berhasil

Keterangan:

X : Tidak berhasil melakukan penelusuran

√ : Berhasil melakukan penelusuran

Berdasarkan Tabel 5.2 dapat dilihat kesimpulan dari pengujian nilai *Radiation Cone.* Pengujian dengan nilai *radiation cone* 85° dan 80° berhasil diterapkan untuk semua prioritas algoritma, hal ini menandakan bahwa semakin

besar nilai *radiation cone* yang digunakan maka semakin akurat pula pembacaan rintangan pada saat melakukan penelusuran. Sedangkan untuk nilai pengujian 75°, 70° dan 65° hanya berhasil diterapkan untuk penelusuran pada prioritas kanan untuk setiap algoritma, hal ini karena kondisi *maze* yang berbeda antara prioirtas kanan dan kiri. Nilai pengujian yang terakir adalah 60°, pengujian dengan nilai ini menimbulkan *error* untuk semua prioritas algoritma, karena nilai 60° terlalu kecil untuk *maze* yang digunakan dalam tugas akhir ini. Sehingga pada saat penelusuran dilakukan, robot tidak dapat berjalan dan menabrak dinding, dan nilai yang paling tepat untuk semua prioritas adalah 80°.

5.2.3 Pengujian Terhadap Waktu Simulasi

Dalam pengujian yang perlu diperhatikan bukan hanya pergerkan robot dalam menelusuri *maze* dan menemukan titik *finish*, tetapi waktu simulasi juga perlu diperhatikan. Algoritma yang digunakan dikatakan efisien, apabila robot mampu sampai pada titik tujuan *(finish*) dengan waktu paling sedikit. Hasil pengamatan pergerakan robot dan waktu simulasi dapat dilihat pada Tabel 5.3 dibawah ini:

Tabel 5.3 Hasil pengujian waktu simulasi robot

No	Algoritma	Pencapaian ke titik *finish*		Waktu Tempuh
		Berhasil	Tidak berhasil	
1	*Pledge* dengan *Right wall priority*	√		0:05:59
2	*Pledge* dengan *Left wall priority*	√		0:06:23
3	*Wall Follower* dengan *Right hand rule*		√	
4	*Wall Follower* dengan *Left hand rule*	√		0:14:01

Berdasarkan Tabel 5.3 dapat disimpulkan bahwa algoritma yang paling cepat waktu tempuhnya sampai pada titik *finish* adalah algortima *pledge* dengan

right wall priority, sebesar 5 menit 59 detik. Hal ini menunjukkan bahwa rute terpendek yang dapat dilalui sampai pada titik *finish* dalam *maze* yang digunakan dalam tugas akhir ini adalah algoritma *pledge* dengan *right wall priority.*

5.2.4 Pengujian Terhadap Nilai Jarak Sensor Ke Rintangan

Nilai jarak sensor sangatlah penting dalam melakukan penelusuran, karena nilai yang ditetapkan akan menjadi acuan robot dalam membaca rintangan melalui jarak yang ditetapkan.

Dalam tugas akhir ini, nilai jarak sensor yang ditetapkan adalah 0,9 meter sampai dengan 1 meter. Jika robot berjarak < 0.9 meter dari rintangan maka robot akan berbelok ke kiri pada algoritma *wall follower* dengan *right hand rule*. Dan jika robot berjarak > 1 meter dari rintangan maka robot akan berbelok ke kanan untuk algortima *wall follower* dengan *left hand rule.* Kenapa menggunakan nilai 0,9 dan 1 meter? Nilai ini digunakan sesuai dengan kondisi *maze* yang dilalui. Nilai ini juga bisa berubah jika kondisi *maze* yang digunakan berbeda ukuran jarak antara dindingnya dengan *maze* yang digunakan dalam tugas akhir ini.

Berikut tabel hasil pengujian dari nilai jarak sensor yang berbeda dengan menggunakan *maze* yang dipakai dalam tugas akhir ini:

Tabel 5.4 Kesimpulan Hasil Pengujian Jarak Sensor

Jarak Sensor	Wall Follower	
	Right hand rule	Left hand rule
0.5 – 1 meter	X	X
0.9 – 1 meter	√	√
1 – 1.5 meter	√	X
1.5 – 2 meter	X	X
1.9 – 2 meter	X	X

Keterangan:

X : Tidak berhasil melakukan penelusuran

√ : Berhasil melakukan penelusuran

Tabel 5.4 menjelaskan tentang hasil pengujian dari nilai jarak sensor yang berbeda. Pemberian nilai dimulai dari 0.5 – 1 meter sampai 1.9 – 2 meter. Nilai ini dapat berubah dan bukan merupakan *default* pengujian. Pada pengujian dengan menggunakan jarak sensor 0.5 – 1 meter robot sama sekali tidak dapat melakukan penelusuran, hal ini karena nilai 0.5 meter terlalu dekat dengan dinding, sehingga pada saat penelusuran dilakukan robot akan menabrak dinding dan tidak dapat berjalan. Dalam tugas akhir ini jarak sensor yang digunakan adalah 0.9 – 1 meter, nilai ini dipakai karena pada saat penelusuran robot dapat melakukan penelusuran dengan baik tanpa menabrak dinding atau keluar lintasan yang ditentukan. Sedangkan untuk pengujian dengan nilai jarak sensor 1 - 1.5 meter robot hanya dapat melakukan penelusuran pada prioritas kanan saja, hal ini karena kondisi *maze* yang berbeda antara prioritas kiri dan kanan. Yang terakhir adalah jarak sensor dengan nilai 1.5 – 2 meter dan 1.9 dan 2 meter, dengan dua nilai jarak sensor ini robot tidak dapat melakukan penelusuran karena nilai yang diberikan terlalu besar dibandingkan dengan jarak lingkungan *maze* yang dapat terbaca oleh radiasi sensor.

5.2.5 Pengujian Dengan Posisi *Start* Dan *Finish* Yang Berbeda

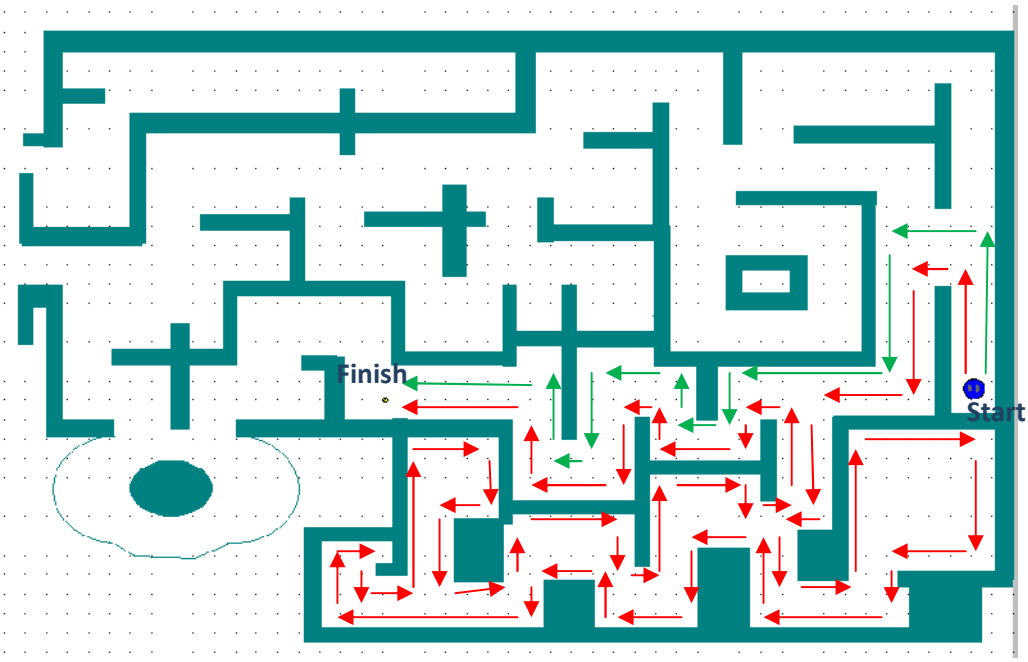
Pada pengujian ini *maze* yang digunakan masih sama, hanya posisi *start* dan *finish* yang berubah-ubah. Pengujian dilakukan dengan dua kondisi yaitu: peletakan posisi *start* yang berbeda dengan posisi *finish* masih dalam lintasan robot dan posisi *start* yang berbeda dengan posisi *finish* diluar lintasan.

1. Posisi *start* yang berbeda dengan posisi *finish* masih dalam lintasan

Dalam perubahan posisi *start* dan *finish,* pengujian dikelompokkan lagi menjadi dua, yaitu pengujian pada prioritas kiri dan pengujian pada prioritas kanan.

a. Pengujian untuk Prioritas kiri

Pengujian ini dilakukan dengan merubah koordinat dari titik *start* menjadi (48, 33) pada sumbu X dan Y. Posisi *finish* juga berubah namun tetap dalam lintasan yaitu pada koordinat (20, 34). Perubahan posisi ini dapat dilihat pada Gambar di bawah ini:



Gambar 5.25 Pengujian untuk prioritas kiri

Keterangan:

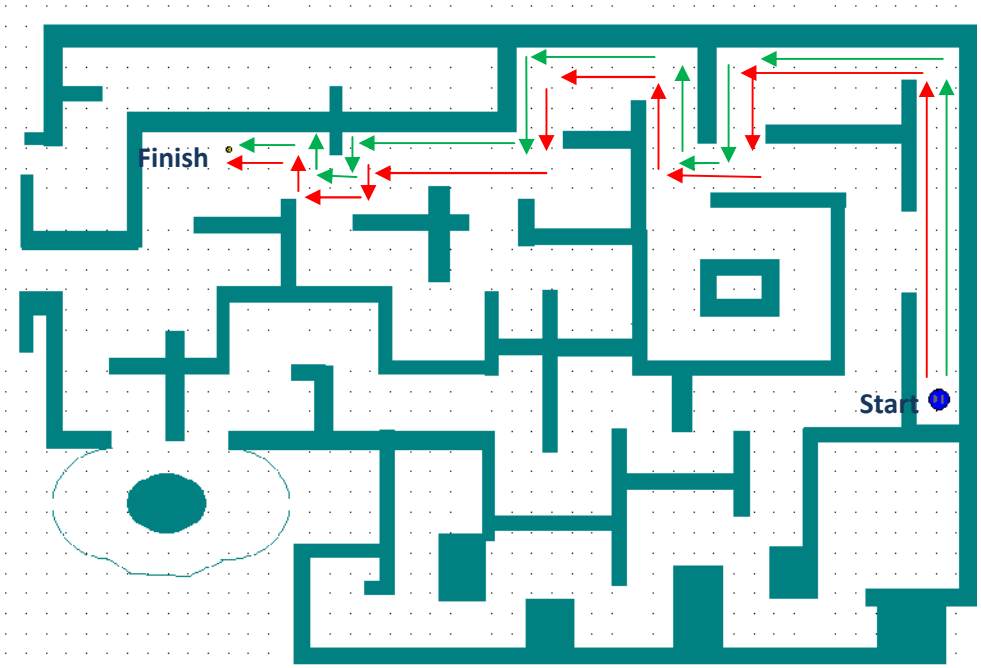
= *Wall follower*

= *Pledge*

Berdasarkan Gambar 5.25 dapat dijelaskan bahwa pengujian pada algoritma *pledge* dan *wall follower* berhasil melakukan penelusuran dengan posisi titik *start* dan *finish* yang masih berada dalam lintasan.

b. Pengujian untuk Prioritas kanan

Pengujian ini dilakukan dengan merubah koordinat dari titik *start* menjadi (48, 33) pada sumbu X dan Y. Posisi *finish* juga berubah namun tetap dalam lintasan yaitu pada koordinat (13, 21). Perubahan posisi ini dapat dilihat pada Gambar 5.26:



Gambar 5.26 Pengujian Untuk Prioritas Kanan

Keterangan:

→ = *Wall follower*

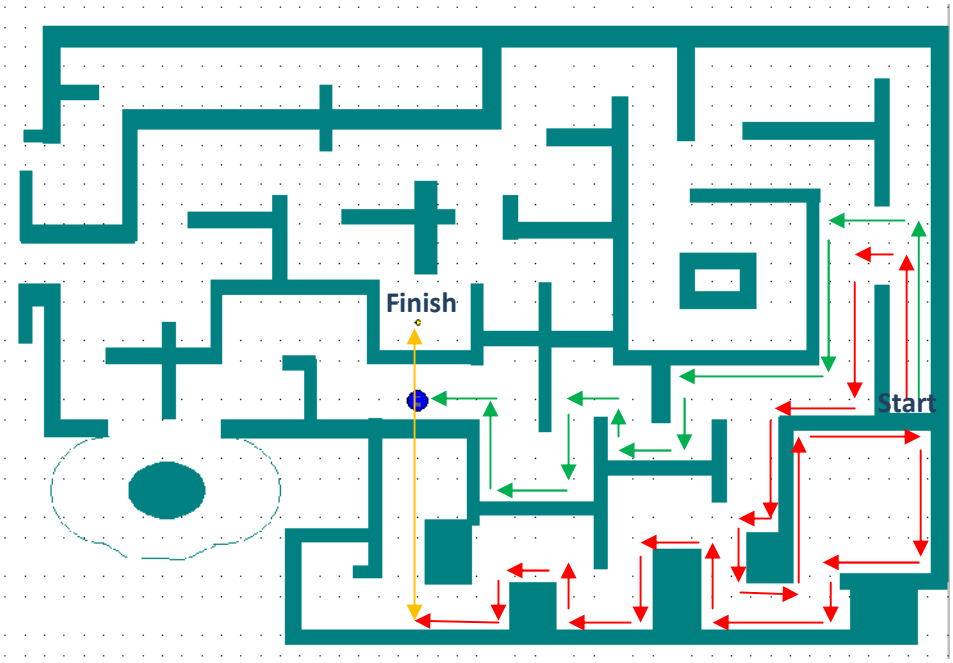
→ = *Pledge*

Berdasarkan Gambar 5.26 dapat dijelaskan bahwa pengujian pada algoritma *pledge* dan *wall follower* berhasil melakukan penelusuran dengan posisi titik *start* dan *finish* yang masih berada dalam lintasan.

2. Posisi *start* yang berbeda dengan posisi *finish* diluar lintasan

a. Pengujian untuk Prioritas kiri

Pengujian ini dilakukan dengan merubah koordinat dari titik *start* menjadi (48, 33) pada sumbu X dan Y. Posisi *finish* juga berubah dan berada di luar lintasan yaitu pada koordinat (23, 30). Perubahan posisi ini dapat dilihat pada Gambar 5.27:



Gambar 5.27 Pengujian untuk prioritas kiri

Keterangan:

= *Wall follower*

= *Pledge*

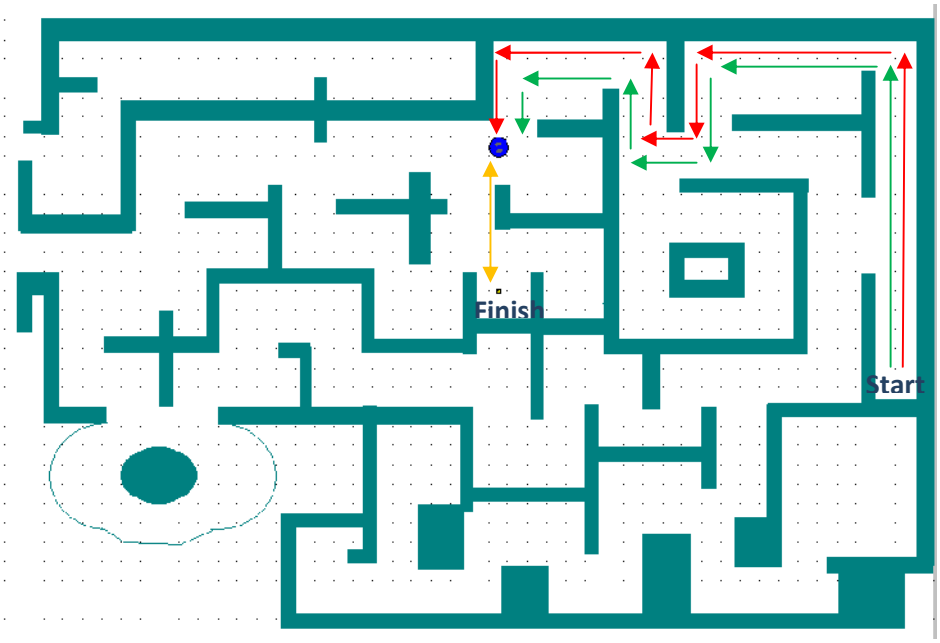
= Posisi *Finish*

Berdasarkan Gambar 5.27 dapat dijelaskan bahwa pengujian pada algoritma *pledge* dan *wall follower* tidak berhasil melakukan penelusuran sampai pada titik *finish* karena dalam simulator Mobotsim pembacaan titik *finish* dilakukan dengan melihat koordinat titik X atau Y. Seperti yang telah dijelaskan dalam bab IV tentang pembacaan *Mark* atau titik *Finish,* robot akan berhenti melakukan penelusuran ketika robot berada pada posisi yang sama dengan koordinat pembacaan *mark* atau titik *finish.* Dalam pengujian seperti pada Gambar 5.27 dapat dilihat bahwa robot menghentikan penelusuran ketika robot berada tepat pada koordinat Y dari titik *finish*, ini terjadi jika posisi *finish* berada diluar jalur penelusuran robot dan pembacaan *finish* ditetapkan melalui sumbu Y. *User* bisa saja melakukan pembacaan titik *finish* melalui sumbu X. Penetapan sumbu

mana yang dipakai dapat dilakukan berdasarkan kebutuhan dan keinginan pemakai ataupun *maze* yang digunakan.

b. Pengujian untuk Prioritas kanan

Pengujian ini dilakukan dengan merubah koordinat dari titik *start* menjadi (48, 33) pada sumbu X dan Y. Posisi *finish* juga berubah dan berada di luar lintasan yaitu pada koordinat (27.5, 29). Perubahan posisi ini dapat dilihat pada Gambar di bawah ini:



Gambar 5.28 Pengujian untuk prioritas kanan

Keterangan:

= *Wall follower*

= *Pledge*

= Posisi *Finish*

Berdasarkan Gambar 5.28 dapat dijelaskan bahwa pengujian pada algoritma *pledge* dan *wall follower* tidak berhasil melakukan penelusuran sampai pada titik *finish,* sama halnya dengan prioritas kiri. Pembacaan titik *finish* dilakukan berdasarkan sumbu X atau Y, ketika robot berada tepat

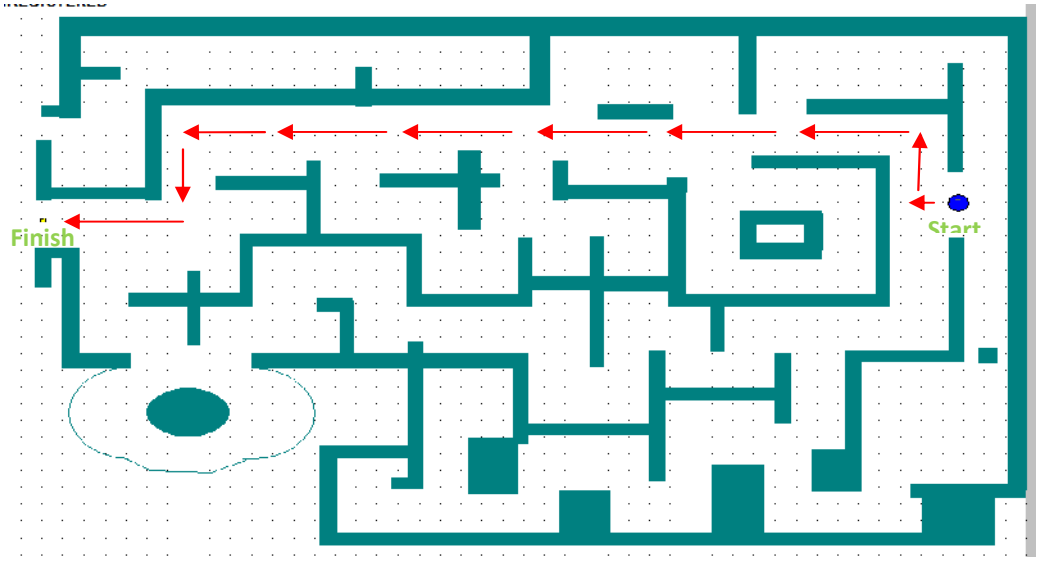
pada sumbu X atau Y maka penelusuran pun berhenti. Dalam pengujian ini titik *finish* ditetapkan pada sumbu Y.

5.2.6 Pengujian Dengan Kondisi *Maze* Yang Berbeda

Untuk melihat apakah algoritma *pledge* dan *wall follower* dapat bekerja sesuai dengan konsep penelusurannya perlu dilakukan pengujian terhadap kondisi *maze* yang berbeda. *Maze* yang digunakan terdiri dari dua yaitu *maze* yang dimodifikasi dari *maze* yang telah dirancang sebelumnya dan digunakan pada saat pengujian algoritma dan *maze* yang baru dengan kondisi rintangan yang jauh berbeda dari *maze* yang digunakan sebelumnya.

1. Pengujian pada algoritma *pledge* dengan *right wall priority*

   Pengujian Algoritma *pledge* dengan *right wall priority* menggunakan *maze* yang telah dimodifikasi dapat dilihat pada Gambar dibawah ini:



Gambar 5.29 Pengujian Algoritma *Pledge* dengan *Right Wall Priority*

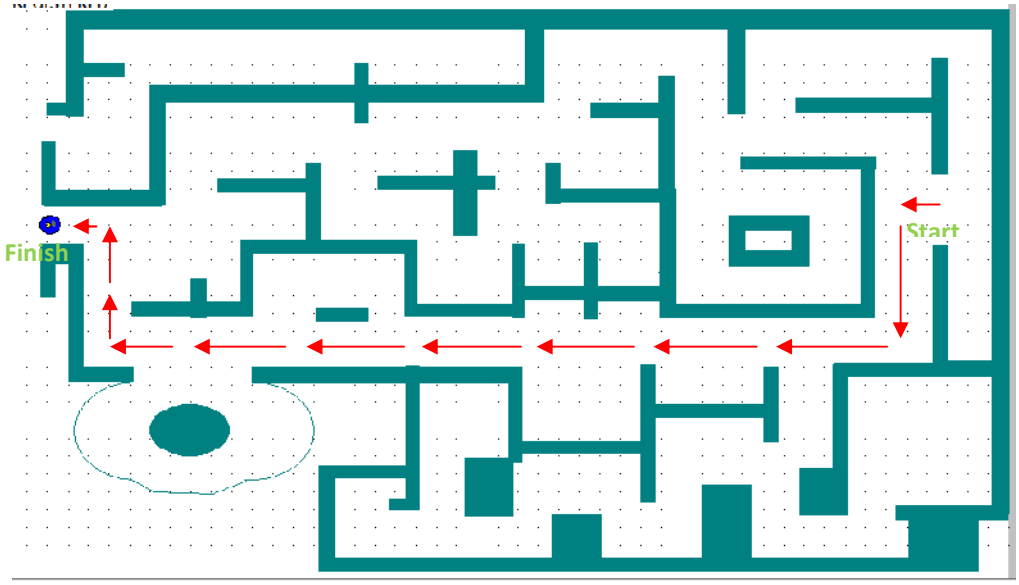
Berdasarkan Gambar 5.29 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *pledge* dengan *right wall priority*. Arah utama robot adalah arah dimana robot pertama kali dijalankan. Halangan pertama dijumpai pada simpangan pertama yang memiliki dua simpang yaitu kiri dan kanan, karena menggunakan *right wall priority* maka robot harus berbelok ke

kanan. Pada simpangan kedua robot menemui halangan di dinding sebelah kanan, maka robot harus berbelok ke kiri dan saat itu robot berjalan sesuai dengan arah utama, karena itu robot akan tetap berjalan lurus walaupun disebelah kanan terdapat persimpangan. Robot terus berjalan sampai robot menemukan halangan berupa dinding, karena dinding sebelah kanan juga tidak terdapat jalan maka robot langsung berbelok ke kiri dan mengikuti dinding dengan metode *wall follower* sampai robot menemukan titik *finish* dan proses penelusuran dihentikan.

Kondisi ini jelas berbeda dengan pengujian algoritma *pledge* dengan *right wall priority* yang ditunjukkan pada Gambar 5.2 dikarnakan kondisi *maze* pada Gambar 5.29 memiliki lintasan lurus searah titik *finish* yang lebih panjang dibanding dengan kondisi *maze* pada Gambar 5.2. Dengan kondisi *maze* seperti Gambar 5.29 jelas bahwa waktu dan jarak tempuh lebih cepat dibandingkan dengan *maze* pada Gambar 5.2. Pada Gambar 5.2 waktu tempuh untuk mencapai titik *finish* adalah 5 menit 59 detik, sedangkan pada *maze* pada Gambar 5.29 waktu tempuh hanya 4 menit 13 detik.

2. Pengujian pada algoritma *pledge* dengan *left wall priority*

Pengujian Algoritma *pledge* dengan *left wall priority* menggunakan *maze* yang telah dimodifikasi dapat dilihat pada Gambar dibawah ini:



Gambar 5.30 Pengujian Algoritma *Pledge* dengan *Left Wall Priority*

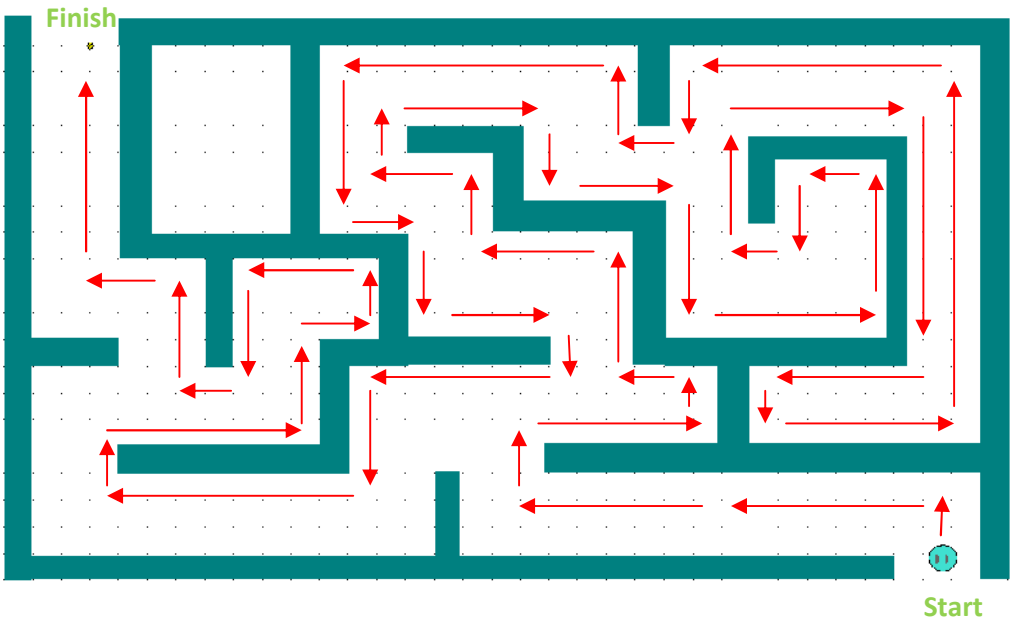
Berdasarkan Gambar 5.30 dapat dijelaskan bahwa robot melakukan penelusuran menggunakan algoritma *pledge* dengan *left wall priority*. Arah utama robot adalah arah dimana robot pertama kali dijalankan. Halangan pertama dijumpai pada simpangan pertama yang memiliki dua simpang yaitu kiri dan kanan, karena menggunakan *left wall priority* maka robot harus berbelok ke kiri. Pada simpangan kedua robot menemui halangan di dinding sebelah kiri, maka robot harus berbelok ke kanan dan saat itu robot berjalan sesuai dengan arah utama, oleh karena itu robot akan tetap berjalan lurus walaupun disebelah kiri terdapat persimpangan. Robot terus berjalan sampai robot menemukan halangan berupa dinding, karena dinding sebelah kiri juga tidak terdapat jalan maka robot langsung berbelok ke kanan dan mengikuti dinding dengan metode *wall follower* sampai robot menemukan titik *finish* dan proses penelusuran dihentikan.

Penelusuran dengan *maze* yang telah dimodifikasi ini juga sangat berbeda dengan *maze* yang digunakan pada Gambar 5.3. Dengan lintasan yang lebih sederhana seperti pada Gambar 5.30, jelas bahwa waktu dan jarak tempuh lebih cepat dibandingkan dengan penelusuran pada *maze* yang digunakan pada Gambar 5.3, dengan selisih waktu simulasi sebesar 3 menit 34 detik, karena pada penelusuran dengan menggunakan *maze* pada Gambar 5.3 waktu simulasi adalah 6 menit 23 detik, sedangkan dengan *maze* pada Gambar 5.30 waktu simulasi hanya 3 menit 57 detik.

Dari dua penelusuran algoritma *pledge* dengan *right* dan *left wall priority* yang telah diuji dengan menggunakan *maze* dengan kondisi jalur yang berbeda dari *maze* yang sebelumnya digunakan, dapat dilihat bahwa algoritma ini dapat bekerja dengan baik sesuai dengan konsep penelusuran yang diterapkan.

3. Pengujian pada algoritma *wall follower* dengan *right hand rule*

Pengujian Algoritma *Wall follower* dengan *right hand rule* menggunakan *maze* yang berbeda dapat dilihat pada Gambar 5.31:

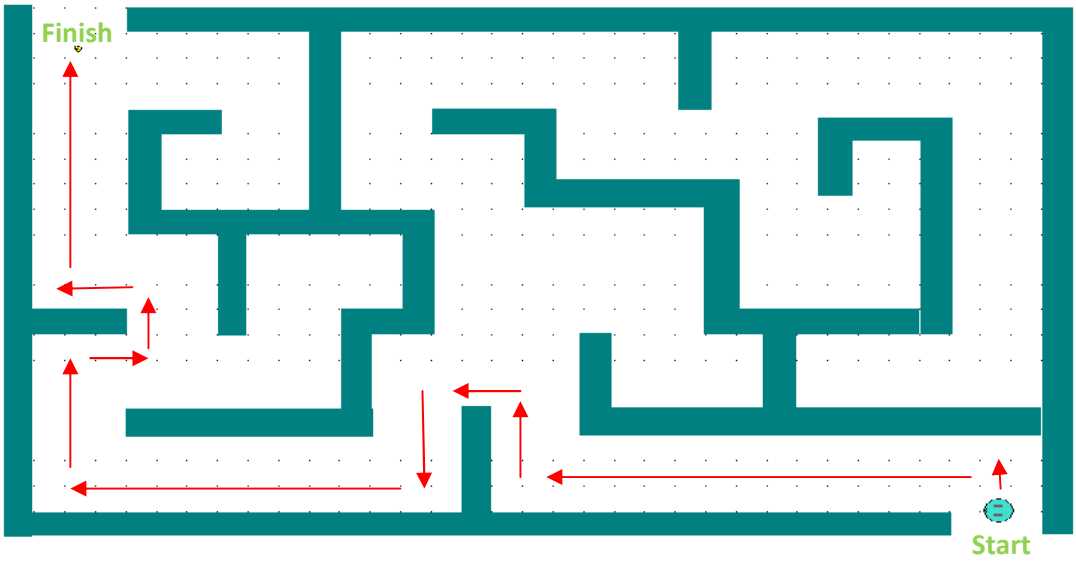


Gambar 5.31 Pengujian Algoritma *Wall Follower* Dengan *Right Hand Rule*

Berdasarkan Gambar 5.31 dapat dijelaskan bahwa *maze* yang digunakan jauh berbeda dengan *maze* yang digunakan pada Gambar 5.4. *Maze* yang digunakan pada Gambar 5.31 tidak memiliki rintangan melingkar, sehingga robot dapat melakukan penelusuran keseluruhan *maze* sampai menuju titik *finish* dengan penelusuran *right hand rule* pada algoritma *wall follower*. Namun rute penelusuran sangat panjang, karena robot menelusuri keseluruhan lintasan *maze* dan tentunya waktu yang dibutuhkan juga jauh lebih lama.

4. Pengujian pada algoritma *wall follower* dengan *left hand rule*

Pengujian Algoritma *wall follower* dengan *left hand rule* menggunakan *maze* yang berbeda dapat dilihat pada Gambar 5.32:



Gambar 5.32 Pengujian Algoritma *Wall Follower* Dengan *Left Hand Rule*

Berdasarkan Gambar 5.32 dapat dijelaskan bahwa *maze* yang digunakan juga jauh berbeda dengan *maze* yang digunakan pada Gambar 5.5. *Maze* yang digunakan pada Gambar 5.32 memiliki rintangan yang lebih sederhana.

Pada penelusuran dengan algoritma *wall follower* menggunakan konsep *left hand rule,* robot dapat dengan mudah menemukan titik *finish*, tidak seperti penelusuran pada algoritma *wall follower* dengan *right hand rule* yang melakukan penelusuran pada semua lintasan *maze* untuk mencapai titik *finish.* Untuk itu dapat disimpulkan bahwa algoritma *wall follower* dengan *left hand rule* lebih unggul dalam menyelesaikan *maze* seperti Gambar 5.31 atau 5.32 dibandingkan dengan algoritma *wall follower* dengan *right hand rule.*

5.3. Kesimpulan Pengujian

Dari pengujian-pengujian yang telah dilakukan dapat ditarik beberapa kesimpulan, yaitu:

1. Dalam penelusuran *maze* yang dirancang dalam tugas akhir ini, algoritma yang paling cepat melakukan penelusuran dan mencapai titik *finish* adalah algoritma *pledge* dengan *right wall priority* dan jalur

terpendek untuk mencapai titik *finish* juga ditunjukkan pada hasil penelusuran dengan algoritma *pledge* dengan *right wall priority*.

2. Nilai *radiation cone* sangat mempengaruhi akurasi pembacaan sensor saat penelusuran dilakukan, nilai *radiation cone* yang cocok untuk semua algoritma adalah sebesar 80°. Semakin besar *radiation cone* yang digunakan, maka semakin akurat pembacaan sensor terhadap rintangan.
3. Nilai jarak sensor ke dinding yang paling tepat adalah antara 0.9-1 meter. Nilai ini digunakan sesuai dengan kondisi *maze* yang dilalui. Nilai ini juga bisa berubah jika kondisi *maze* yang digunakan berbeda ukuran jarak antara dindingnya dengan *maze* yang digunakan dalam tugas akhir ini.
4. Ada dua pengujian dengan titik *start* dan *finish* yang berbeda, yaitu dengan posisi *finish* yang berada di dalam lintasan dan posisi *finish* di luar lintasan. Untuk posisi *finish* di dalam lintasan robot berhasil melakukan penelusuran dan sampai di titik *finish.* Sedangkan untuk posisi *finish* yang berada di luar lintasan, robot tidak dapat mencapai titik *finish* namun tetap berhenti sesuai koordinat *mark* atau titik *finish.*
5. Dengan menggunakan *maze* yang berbeda, algoritma *pledge* dan *wall follower* berhasil melakukan penelusuran dan mencapai titik *finish*. Namun waktu tempuh berbeda dengan pengujian algoritma dengan *maze* sebelumnya karena bentuk *maze* yang digunakan berbeda.

BAB VI

PENUTUP

6.1 Kesimpulan

Setelah menyelesaikan serangkaian tahapan-tahapan terhadap perancangan algoritma *pledge* pada robot *wall maze* yang dimulai dari pengumpulan data tentang algoritma *pledge* hingga pada tahapan pengujian, maka dapat diambil beberapa kesimpulan diantaranya adalah sebagai berikut:

1. Hasil penerapan algoritma *pledge* untuk menyelesaikan *maze* pada robot *wall maze* memiliki tingkat keberhasilan sebesar 90%, karena pada pengujian dengan posisi *finish* diluar lintasan robot tidak dapat mencapai titik *finish* seperti yang diinginkan, namun robot tetap berhenti sesuai koordinat Y dari titik *finish*.
2. Algoritma *pledge* yang dimodifikasi memiliki 2 kemungkinan prioritas yang dapat digunakan yaitu *Right wall priority* dan *Left wall priority*. Tidak ada keunggulan ditiap prioritasnya, yang membedakan hanya pada saat pengambilan keputusan ketika robot menemui rintangan berbentuk persimpangan.
3. Algoritma *wall follower* digunakan sebagai pembanding kinerja algoritma *pledge*. Algoritma ini memiliki dua konsep penelusuran yaitu *right hand rule* dan *left hand rule.*
4. Dengan bentuk *maze* yang digunakan dalam tugas akhir ini, dapat dilihat bahwa algoritma *pledge* lebih unggul dalam melakukan penelusuran dibandingkan dengan algoritma *wall follower,* ini disebabkan:
    a. Algoritma *wall follower* melakukan penelusuran hanya berdasarkan dinding kanan dan kiri, sehingga hampir keseluruhan dinding pada bagian kiri ditelusuri.

b. Penelusuran dengan *wall follower* memakan waktu yang lebih lama dibanding algoritma *pledge* yang melakukan penelusuran berdasarkan arah utama, sehingga tidak perlu melakukan penelusuran terhadap keseluruhan dinding yang diprioritaskan.

5. Tampilan 2D pada simulator Mobotsim V.10 mengakibatkan kurang akuratnya visualisasi terhadap halangan atau dinding yang dirancang melalui objek-objek yang tersedia. Tinggi dari halangan atau dinding tidak dapat ditentukan sehingga ketika robot bertemu dengan setiap objek yang dirancang kedalam *worl frame*, robot akan membacanya sebagai halangan dan menghindarinya.

6.2 Saran

Untuk penelitian kedepannya penulis memiliki beberapa saran yang dapat dikembangkan pada penelitian selanjutnya, seperti:

1. Analisa penelusuran terhadap *maze* ini dapat diterapkan dengan menggunakan algoritma-algoritma penelusuran dan menggunakan simulator lainnya, seperti simulator dengan tampilan 3D agar bentuk robot dan *maze* yang digunakan dapat divisualisasikan dengan lebih jelas.
2. Pengembangan robot *wall maze* juga bisa dilakukan dengan menambahkan kemampuan lain kepada robot, misalnya mengembangkannya menjadi robot pemadam api, sehingga robot memiliki kecerdasan bukan hanya dalam menelusuri jalur dan mencari titik *finish* tetapi robot tersebut juga dapat memadamkan api sesuai dengan konsep kecerdasan yang nantinya akan ditanamkan.

DAFTAR PUSTAKA

Anita, Nur Syafidtri. *Robot Micromouse Dengan Menggunakan Algoritma.* [online] *available* www.gunadarma.ac.id/library/articles/.../Artikel_21105199.pdf, diakses tanggal 3 januari 2012.

Budiharto, Widodo, *Membuat Sendiri Robot Cerdas*, Elex Media Komputindo, Jakarta, 2006.

Budiharto, Widodo, *10 Proyek Robot Spektakuler*, Elex Media Komputindo, Jakarta, 2008.

Darmawan, Arif, Hendriawan, Akhmad dan Akbar, Reesa. *Penerapan Algoritma Pledge untuk Menyelesaikan Maze pada Line Follower Robot.* Tugas Akhir EEPIS-ITS, 2010.

David, Giessel, *Building a Mouse*, UAF MicroMouse, 2007

Harapan, Samudra, Bekti, *Pencarian Shortest Path Dinamik Dengan Algoritma Bellman-Based Flood-Fill Dan Implementasinya Pada Robot Micromouse*, Tugas akhir ITB, 2009.

Hartanto, Dwi dan Raharjo, Suwanto. *Visual Downloader untuk Microcontroller AT89C2051* . Andi Offset, Yogyakarta, 2005.

Indrawan, Gede. *Perancangan dan Implementasi Kecerdasan Buatan Robot Pencari Jalur Berbasis Microcontroler Basic Stamp.* Tesis UI, 2008.

Iqbal, M, Nugraha, Hendriawan, Akhmad dan Akbar Reesa. *Penerapan Algoritma Maze Mapping untuk menyelesaikan Maze pada Line Tracer.* Tugas Akhir EEPIS-ITS, 2009.

Kamphans, Tom dan Lengetepe, Elmer. *The Pledge Algorithm Reconsidered under Errors in Sensors and Motion.* Departmen of computer science, Bonn, Germany, 2004.

Kusumadewi, Sri, *Artificial Intelligence: Teknik dan Aplikasinya*, Graha Ilmu, Yogyakarta, 2003.

Mishra, Swati. *Maze Solving Algorithm for Micro Mouse*. IEEE International Conference on Signal Image Technology and Internet Based Systems: 2008.

Pitowarno, Endra, *Robotika Desain, Kontrol dan Kecerdasan Buatan,* Andi Offset, Yogyakarta, 2006.

Putra, Agfianto Eko. *Pengendalian Mobile Robot (MOBOT) Dengan MOBOTSIM v1.0,* Gaya Media, Yogyakarta, 2005.

Putra, Agfianto Eko. *MOBOTSIM v1.0: Software Pembelajaran Pengendalian Mobile Robot*. [online] *available* http://agfi.staff.ugm.ac.id/blog/index.php/2009/01/mobotsim-v10-software-pembelajaran-pengendalian-mobile-robot/, diakses tanggal 29 Desember 2011.

Reiners, Paul D. *Robots, Mazes, And Subsumption Architecture*. [online] *available* http://www.ibm.com/developerworks/java/library/j-robots/, diakses tanggal 3 Januari 2012.

Setiawan, Sandi, *Artificial Intelligence*, Andi Offset, Yogyakarta, 1993.

Suparman, *Mengenal Artificial Intelligence*, Andi Offset, Yogyakarta, 1991.

Suyanto, *Artificial Intellegence: Searching, Reasoning, Planning, Learning*. Informatika Bandung, Bandung, 2007.