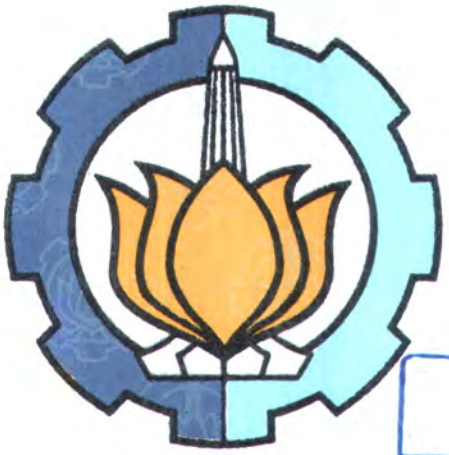
TESIS

PENJELAJAHAN DAN NAVIGASI *INDOOR MOBILE* ROBOT DALAM LINGKUNGAN YANG TIDAK DIKENALI DENGAN MENGGUNAKAN PROBABILITAS *OCCUPANCY GRIDS MAP*

Oleh :
Indrazno Siradjuddin
NRP. 2203 204 010



PROGRAM STUDI MAGISTER
BIDANG KEAHLIAN ELEKTRONIKA
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

PENJELAJAHAN DAN NAVIGASI *INDOOR MOBILE* ROBOT DALAM LINGKUNGAN YANG TIDAK DIKENALI DENGAN MENGGUNAKAN PROBABILITAS *OCCUPANCY GRIDS MAP*

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember

oleh :

Indrazno Siradjuddin
Nrp. 2203 204 010

Tanggal Ujian : 1 Pebruari 2006
Periode Wisuda : Maret 2006

Disetujui oleh Tim Penguji Tesis :

1. **Ir. Djoko Purwanto, M.Eng., Ph.D.** **(Pembimbing I)**
   NIP. 131 879 397
2. **Ir. Hendra Kusuma, M.Eng.** **(Pembimbing II)**
   NIP. 131 846 104
3. **Prof. Dr. Ir. Mauridhi Hery P., M.Eng.** **(Penguji)**
   NIP. 131 569 364
4. **Achmad Arifin, ST., M.Eng., Ph.D.**
   NIP. 132 163 669
5. **Dr. Muhammad Rivai, ST., MT.**
   NIP. 132 088 341
6. **Ir. Harris Pirngadi, MT., ID.** **(Penguji)**
   NIP. 131 843 903
7. **Rachmad Setiawan, ST., MT.** **(Penguji)**
   NIP. 132 134 651

Direktur Program Pasca Sarjana,

Prof. Ir. Happy Ratna S., MSc., Ph.D.
NIP. 130 541 829

PENJELAJAHAN DAN NAVIGASI *INDOOR MOBILE ROBOT* DALAM LINGKUNGAN YANG TIDAK DIKENALI DENGAN MENGGUNAKAN PROBABILITAS *OCCUPANCY GRIDS MAP*

Nama Mahasiswa	: Indrazno Siradjuddin
NRP	: 2203 204 010
Pembimbing	: Ir. Djoko Purwanto, M.Eng., Ph.D.
Co-Pembimbing	: Ir. Hendra Kusuma, M.Eng.

Abstrak

Kemampuan navigasi adalah hal penting yang harus dimiliki oleh sistem mobile robot. Sistem navigasi reaktif banyak digunakan dalam beberapa aplikasi seperti *wall-following* dan *line-following*, namun sistem navigasi tersebut menuntut ketersediaan peta atau jalur navigasi. Dalam beberapa aplikasi mobile robot seperti *planetary exploration*, *minesweeper*, *urban search and rescue*, ketersediaan peta menjadi hal yang sangat sulit didapatkan. Menyadari akan hal tersebut, maka mobile robot harus mampu menghasilkan peta sendiri dari hasil persepsi sensor-sensor yang digunakan dan menavigasi dirinya untuk menjelajah daerah yang tidak dikenali.

Penelitian ini membahas metode penjelajahan dan navigasi *mobile robot* pada lingkungan yang tidak dikenali. Proses penjelajahan meliputi beberapa tahapan: lokalisasi, persepsi, mapping dan navigasi. Probabilitas *occupancy grids map* adalah metode yang digunakan dalam proses penjelajahan tersebut. Metode ini berdasarkan teori bayes, yang dapat mengintegrasikan persepsi robot dalam setiap pergerakannya dari hasil pembacaan sensor-sensor yang digunakan. Mulai dari model probabilitas sensor hingga penurunan persamaan probabilitas *occupancy grids map* menghasilkan algoritma komputasi yang dapat digunakan untuk proses penjelajahan.

Algoritma probabilitas *occupancy grids map* diterapkan dalam program simulasi dan sistem navigasi pada *prototype* robot. Dari hasil simulasi, algoritma ini dapat menjelajah dengan *coverage area* rata-rata 98% dari beberapa variasi simulasi *workspace* robot.

Kata kunci: probabilitas *occupancy grids* map, *indoor mobile robot*, penjelajahan, lokalisasi dan pembuatan peta (*mapping*)

INDOOR MOBILE ROBOT EXPLORATION AND NAVIGATION IN UNKNOWN ENVIRONMENTS USING OCCUPANCY GRIDS MAP PROBABILITY

Nama Mahasiswa	: Indrazno Siradjuddin
NRP	: 2203 204 010
Pembimbing	: Ir. Djoko Purwanto, M.Eng., Ph.D.
Co-Pembimbing	: Ir. Hendra Kusuma, M.Eng.

Abstract

Navigation ability is the important thing of mobile robot system. Reactive navigation system usually used in many application, such as: wall following and line following. Unfortunately these system is demanding map or navigation route. In several application of mobile robot such as planetary exploration, minesweeper, urban search and rescue, the map is difficult to obtain. Considering those challenge, the first alternative is to use mobile robot sensors to produce a map so that they can explore unknown area.

This research try to find the appropriate exploration method and navigation for the mobile robot to explore the unknown area. Exploring process consist of several steps, they are localization, perception, mapping and navigation. Occupancy grids map probability is the method used in those exploration. These method based on bayes teory that is integrating robot perception in every movement from sensor reading result. The computation algorithm for exploring is developed from probability model until producing occupancy grids map probability equations.

The probability of occupancy grids map algorithm is used in simulation program and navigation system of robot prototype. From the simulation result, these algorithm can explore the coverage area up to 98% in the average from several variation of workspace robot simulation.

Keywords: Probability of occupancy grids map, indoor mobile robot, exploration, localization and mapping

Kata Pengantar

Puji dan syukur penulis panjatkan kepada Allah SWT yang Maha Pengasih dan Penyayang atas berkat dan karunia yang diberikan sehingga penulisan laporan ini dapat terselesesaikan. Laporan tesis ini disusun dengan judul

"PENJELAJAHAN DAN NAVIGASI *INDOOR MOBILE ROBOT* DALAM LINGKUNGAN YANG TIDAK DIKENALI DENGAN MENGGUNAKAN PROBABILITAS *OCCUPANCY GRIDS MAP*"

Segala upaya dan kemampuan telah penulis lakukan demi kesempurnaan laporan ini sebagai pertanggung jawaban ilmiah. Penulis menyadari laporan ini tidak akan dapat tersusun tanpa bantuan dan bimbingan dari berbagai pihak. Untuk itu penulis menghaturkan rasa terima kasih dari lubuk yang paling dalam kepada:

- Bapak **Ir. Djoko Purwanto, M.Eng., Ph.D.,** selaku koordinator program pasca sarjana Elektronika ITS merangkap sebagai dosen pembimbing.
- Bapak **Ir. Hendra Kusuma, M.Eng.** selaku dosen pembimbing.
- Bapak **Prof. Dr. Ir. Mauridhi Hery P., M.Eng.**, yang telah banyak memberikan saran-saran dan bantuan.
- Bapak **Rachmad Setiawan, ST., MT.,** selaku dosen wali.
- Bapak-bapak dosen penguji, yang telah banyak membantu untuk merivisi demi kesempurnaan penulisan dan banyak memberikan wacana ilmiah.
- Istriku yang tercinta dan ananda (kakak Ara dan adik Afa) yang tersayang, yang telah sabar, yang selalu berdoa, dan penuh pengorbanan yang tiada ternilai.
- Ibunda dan Ayahanda (almarhum) tercinta, yang telah banyak memberikan dorongan dan dengan kesabarannya memberikan bimbingan dan nasehat-nasehat.
- Umi dan Abah tersayang yang selalu berdoa, dan penuh kesabaran memberikan bimbingan dan nasehat-nasehat.

- Adik-adikku (Iin dan Intan) yang tersayang, yang telah banyak membantu dan selalu berdoa.
- Camik kakakku yang telah banyak membantu dan memberikan saran-saran.
- Prifan "Kribo" yang telah banyak membantu tenaga dan pikiran dalam proses perbaikan penulisan laporan Tesis.
- Kawan-kawan program pasca sarjana : Pujiono, Adam, Endah, Zaenal, Nurkholis, Bagyo, Basyir, Mardi, Morlan Pardede, Catur sebagai teman diskusi selama studi
- Semua pihak yang tidak dapat kami sebutkan satu persatu yang dengan keikhlasannya telah banyak membantu kelancaran penelitian hingga penulisan laporan Tesis ini.

Penulis menyadari bahwa laporan tesis ini jauh dari kesempurnaan mengingat keterbatasan yang penulis miliki. Untuk itu kritik dan saran yang bersifat membangun demi kesempurnaan Tesis ini sangat penulis harapkan.

Penulis berharap semoga Tesis ini bermanfaat bagi perkembangan ilmu pengetahuan khususnya bidang teknik elektronika dan robotika, dan menambah wawasan bagi yang membacanya.

Surabaya, Nopember 2006

Penulis

DAFTAR ISI

Hal

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Banyak riset yang telah dilakukan berkaitan dengan permasalahan *Path Planning* untuk *Indoor Mobile Robot* (IMR). Beberapa algoritma dikemukakan untuk penyelesaian optimasi *path* dari titik awal pergerakan menuju titik akhir / tujuan akhir pergerakan IMR, disertai dengan informasi peta (*map*) dan kondisi lingkungan yang lengkap. Akan timbul permasalahan jika *map* dan kondisi lingkungan tidak diketahui secara sebagaian maupun secara keseluruhan, maka IMR harus memiliki kemampuan untuk menjelajah (*exploration/coverage* :EC) ke-semua titik didalam suatu wilayah tertentu, dalam usahanya menuju/mencari goal/target. Aplikasi IMR yang menggunakan teknik EC diantaranya adalah pemetaan (*mapping*), inspeksi, operasi pencarian dan penyelamatan (*search and rescue*), *vacuum cleaner* , robot penyapu ranjau (*minesweeper*).

Terdapat beberapa riset yang telah menghasilkan algoritma EC baik secara teori maupun implementasi pada beberapa jenis IMR. [1,4,7,8,9,10,12], berorientasi pada pembagian area menjadi sub area, *cell decomposition*, dan *grid based*. [2,6], menggunakan metode penandaan (*marker*) pada titik atau wilayah yang telah terekplorasi. [4,5,11], menggunakan perencanaan (*path planning*) atau pola pergerakan robot: *spiral*, *line sweep*.

Algoritma EC haruslah beroperasi baik secara local maupun global konteks [3]. Operasi strategi local, keputusan pergerakan IMR selanjutnya, dibuat berdasarkan apa yang dapat dilihat IMR saat ini dan lokasi yang terdekat yang harus dituju, sebagai contoh teknik *wall-following*. Strategi local juga biasa disebut pendekatan

secara reaktif (*reactive approach*). Operasi secara global, membutuhkan penentuan wilayah atau titik yang jauh dari lokasi IMR saat ini, wilayah atau titik tersebut menjadi acuan arah pergerakan IMR. Strategi global adalah pendekatan berdasarkan model (*model based approach*).

Ketepatan proses navigasi robot sangat tergantung pada kepresisian pergerakan robot, dan ketepatan pembacaan sensor. Namun pada kenyataannya pergerakan robot dan sensor tidaklah ideal. Sehingga harus ada mekanisme atau metode yang dapat mengatasi ketidakpastian ini.

Penelitian ini menggunakan gabungan strategi local dan strategi global untuk usaha menjelajah dan me-navigasi IMR pada wilayah kerja (*workspace*) yang memilki kondisi lingkungan yang statis. Gabungan kedua metode tersebut didasarkan pada pemodelan probabilitas *occupancy grids*. Metode *occupancy grids* dikemukakan pertama kali oleh ELFES, 1989 [16,17]. Pada intinya metode *occupancy grids* merupakan model pendekatan ketidakpastian (sensor, pergerakan robot, pemetaan) dengan menggunakan metode probabilitas (*bayesian*).

1.2 Perumusan Masalah

Lokalisasi adalah salah satu komponen penting dalam sistem *autonomous vehicle* [15], dasar dari lokalisasi adalah dengan menggunakan sensor pada sistem *dead reckoning*. Permasalahan lokalisasi adalah memprediksi lokasi robot pada peta yang telah dibuat. Sehubungan dengan permasalahan eksplorasi, saat kedua informasi yaitu lokasi robot dan peta tidak diketahui. Pada kasus tersebut, posisi awal robot dimana dia mulai bergerak adalah di lokasi yang tidak diketahui dan didalam suatu wilayah yang juga tidak diketahui petanya. Pada saat itu robot harus mempunyai kemampuan untuk membuat peta dari hasil penterjemahan beberapa sensor yang

dipasang dari setiap pergerakan robot, untuk mengetahui keadaan disekelilingnya (khususnya dinding dan *obstacle*) dan juga harus mempunyai kemampuan untuk menavigasi, menentukan ke arah mana dan berapa jauh robot harus berjalan, permasalahan tersebut biasa dikenal dengan sebutan SLAM (*Simultaneous Localization and Mapping*).

Selain hal tersebut, dalam kasus ekplorasi/menjelajah robot harus mempunyai kemampuan untuk mengingat daerah mana saja atau lokasi di titik mana saja robot telah melakukan pergerakan, sehingga robot tidak terjebak dalam kondisi dimana dia hanya berputar dalam suatu wilayah tertentu saja, dan dapat menentukan arah navigasi menuju daerah yang belum pernah dilewati atau dikenali.

Penelitian ini merepresentasikan peta kedalam model 2 dimensi atau bidang koordinat XY. Peta terdiri dari bidang bidang kecil yang memilki ukuran seragam yang disebut *grid cell*. Setiap *grid cell* memiliki nilai probabilitas "kependudukan" (*occupancy*). Nilai probabilitas *occupancy* yang melekat pada setiap *grid cell* akan menunjukkan bahwa *grid cell* tersebut pernah dikenali, dilewati atau telah dijelajahi oleh robot, Peta jenis ini biasa disebut *Probabilistic Grid-Based* MAP (PGM) [16]. Terdapat 3 komponen penting permasalahan yang mendasari pembuatan peta PGM, yaitu:

- Penggabungan informasi dari hasil pembacaan beberapa sensor selama pergerakan robot, yang kemudian dipetakan dan diintegrasikan kedalam probabilitas *occupancy* pada setiap *grid cell*
- Eksplorasi. Dengan keterbatasan pendinderaan (*sensing*), IMR harus dapat menjelajahi seluruh *workspace* (contoh: sebuah rumah dengan beberapa ruangan yang tidak diketahu letak dan jumlahnya). Saat proses

eksplorasi dianggap selesai apabila IMR telah mengekplorasi seluruh ruangan atau pemberian batas waktu eksplorasi.

- Lokalisasi. Pergerakan robot tidaklah akurat dan kesalahan interpretasi terhadap pembacaan sensor odometri, mengakibatkan akumulasi kesalalahan pembacaan selama pergerakan robot. Minimilasi kesalahan pergerakan adalah dengan mendisain kontroller yang tepat dari motor penggerak roda IMR. Pemodelan kinematik terhadap posisi dan orientasi robot terhadap bidang datar dan kontruksi roda dan bahan, merupakan hal yang dapat meminimilasisi kesalahan odometri.

Workspace yang hasus diekplorasi oleh IMR memiliki kondisi lingkungan yang statis. Maksud statis adalah *obstacles* tidak bergerak (contoh: manusia). Asumsi *workspace* adalah suatu wilayah datar, permukaan lantai yang rata.

1.3 Tujuan Penelitian

Penilitian ini bertujuan untuk menerapkan metode EC untuk *workspace* yang tidak diketahui kondisi statis lingkungan yang terdapat didalamnya (*unknown environtments*). dan mengujinya pada *prototype* IMR.

1.4 Manfaat Penelitian

Manfaat secara umum penelitian ini adalah memberikan:

- Algoritma penyelesaian terhadap permasalahan ekplorasi dan navigasi IMR pada suatu "*unknown environtment*".
- Rincian disain kinematik, kontroller dan teknologi sensor yang berguna bagi kelanjutan penelitian mengenai IMR.

Dengan kemampuan yang dimilki robot untuk menjelajah, estimasi posisi (lokalisasi), dan mampu membuat *PGM*, akan meningkatkan fleksibilitas dan kegunaan dari sebuah *IMR*. Secara khusus hasil penelitian ini akan bermanfaat dalam proses pengembangan *service robot*, seperti robot *vacuum cleaner*, robot penjelajah, robot USAR (*urban search and rescue*).

BAB II
DASAR TEORI

2.1 Model *trajectory IMR differential drive*

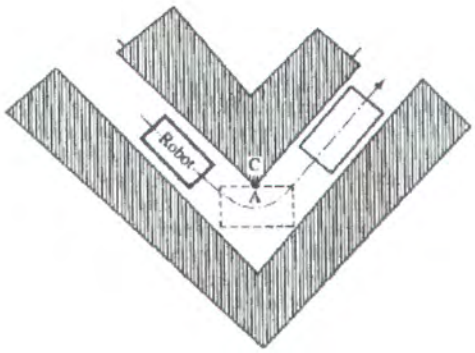
Metode yang sering digunakan dan relatif lebih mudah untuk kontrol pengemudian IMR adalah sistem *differential steering* [13]. Sistem *differential steering* (*DS*) biasa kita temukan dalam kehidupan sehari-hari, yaitu sistem pengemudian pada kursi roda. 2 roda diletakkan pada satu garis lurus, dan digerakkan secara independen pada setiap rodanya, sebagai penyeimbang biasa digunakan 1 atau lebih roda penyeimbang yang dapat bergerak ke segala arah, *caster*. Dengan memvariasikan perbedaan daya gerak atau kecepatan pada kedua roda menghasilkan efek pengemudian. Namun seberapa besar perbedaan daya gerak atau kecepatan itu menghasilkan arah dari tujuan pergerakan secara benar?.

Sebagai awal dari pemodelan kontrol pengemudian, pendekatan sederhana pada mekanisme pengemudian IMR pada belokan pada pojok persimpangan (gambar 2.1). Berdasarkan gambar 2.2, dapat ditulis persamaan sebagai berikut:

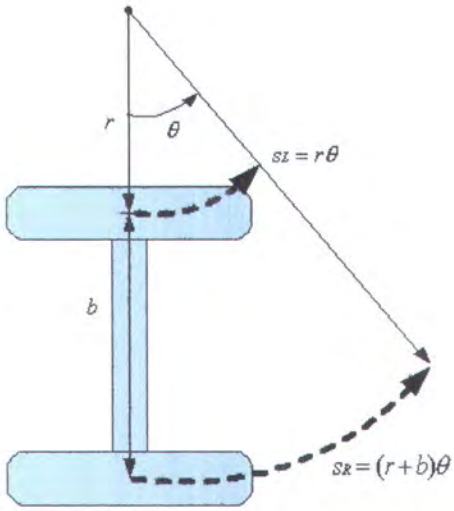
$$SL = r\theta \quad \text{.....(2.1)}$$

$$SR = (r+b)\theta \quad \text{.....(2.2)}$$

$$SM = (r+b/2)\theta \quad \text{.....(2.3)}$$



Gambar 2.1. Pergerakan IMR pada pojok persimpangan



Gambar 2.2. Path dari pergerakan memutar sistem DS

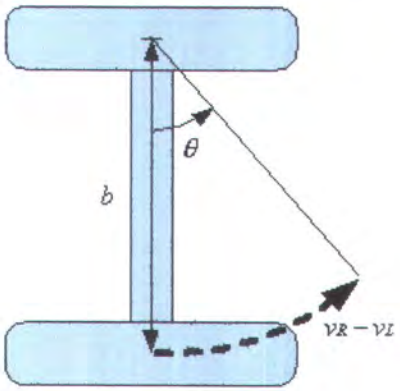
dimana Sl,SR perpindahan roda kiri dan kanan. r adalah jari roda kiri (*inner*) terhadap titik pusat putaran. b adalah jarak titik pusat roda kiri dengan titik pusat roda kanan (*outer*). θ adalah sudut putaran dalam bentuk *radian*. SM adalah kecepatan pergerakan dilihat dari titik tengah IMR.

Selanjutnya dalam bidang datar, sistem koordinat XY, didapatkan kecepatan perpindahan dalam sumbu X dan Y.

$$\frac{dx}{dt} = m(t)\cos(\theta(t)) \quad \text{.....(2.4)}$$

$$\frac{dy}{dt} = m(t)\sin(\theta(t)) \quad \text{.....(2.5)}$$

m(t) dan θ(t) merupakan fungsi kecepatan dan orientasi dalam fungsi waktu.



Gambar 2.3. Pergerakan roda pada kecepatan yang berbeda

Kecepatan perubahan sudut perputaran dihitung dengan menggunakan persamaan 2.6.

$$d\theta/dt = (VR - VL)/b \quad \text{.....(2.6)}$$

$$dx/dt = [(VR + VL)/2]\cos(\theta(t)) \quad \text{.....(2.7)}$$

$$dx/dt = [(VR + VL)/2]\cos(\theta(t)) \quad \text{.....(2.8)}$$

Dalam bentuk diskrit persamaan 2.6 menjadi persamaan 2.9.

$$\frac{\theta_i - \theta_{i-1}}{\Delta t} = \frac{v_R - v_L}{b} = \frac{SR - SL}{b.t} = \frac{(S_{Ri} - S_{Ri-1}) - (S_{Li} - S_{Li-1})}{b.t} = \frac{\Delta S_r - \Delta S_l}{b.t} \quad \text{.....(2.9)}$$

Karena waktu Δt adalah sama dengan waktu perjalanan robot t, maka persamaan 2.9 menjadi persamaan 2.10.

$$\theta_i - \theta_{i-1} = \frac{\Delta S_r - \Delta S_l}{b} = \Delta\theta \quad \text{.....(2.10)}$$

Persamaan 2.7 dalam bentuk diskrit menjadi persamaan 2.11.

$$\frac{\Delta x}{\Delta t} = \frac{vR + vL}{2}\cos\theta = \frac{\Delta Sr\,\Delta Sl}{2.t}\cos\theta_i \quad \text{.....(2.11)}$$

t=Δt,

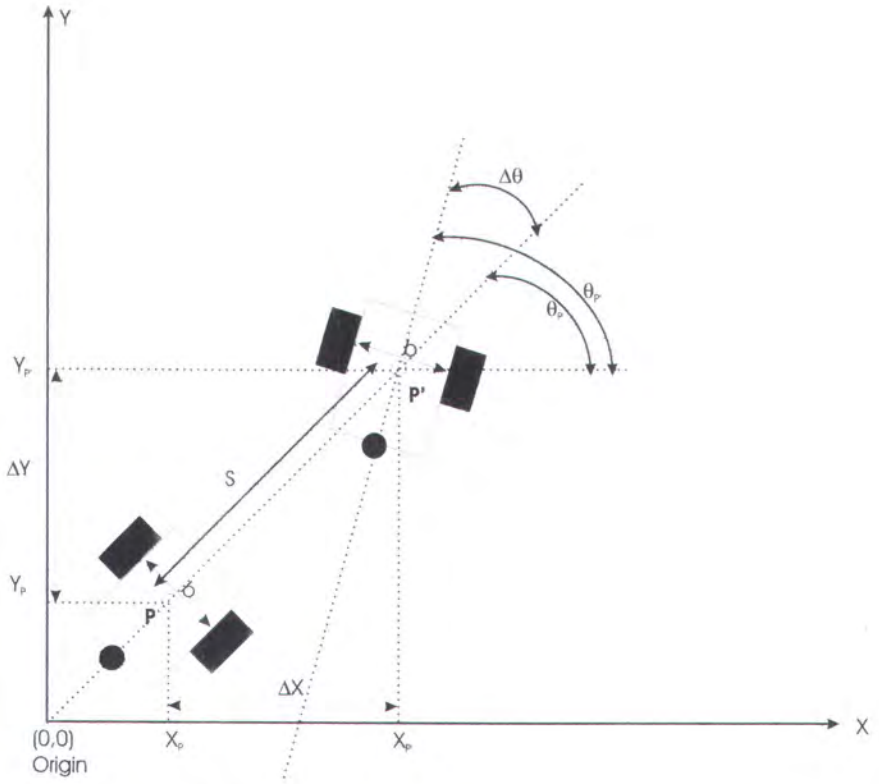
$$\Delta x = \frac{\Delta Sr + \Delta Sl}{2}\cos\theta_i \quad \text{.....(2.12)}$$

dengan mensubstitusi persamaan 2.12 dengan 2.10, maka ΔX dapat dicari dengan menggunakan persamaan 2.13.

$$\Delta x = \frac{\Delta Sr + \Delta Sl}{2}\cos(\theta_{i-1} + \frac{\Delta S_r - \Delta S_l}{b}) \quad \text{.....(2.13)}$$

Begitu juga untuk ΔY dapat dicari dengan menggunakan persamaan 2.14.

$$\Delta y = \frac{\Delta Sr + \Delta Sl}{2}\sin(\theta_{i-1} + \frac{\Delta S_r - \Delta S_l}{b}) \quad \text{.....(2.14)}$$



Gambar 2.4. Model Kinematik robot dalam bidang XY

Model kinematik robot dalam bidang XY (gambar 2.4), memberikan gambaran yang jelas tentang proses pergerakan robot dan perubahan arah orientasinya (pose). Titik origin adalah merupakan titik acuan atau biasa disebut titik origin. Robot berada di posisi P (XP,YP,θP) bergerak sejauh S menuju titik P'(XP',YP',θP'). Persamaan pergerakan robot dapat dituliskan sebagai berikut

$$P = \begin{bmatrix} X_P \\ Y_P \\ \theta_P \end{bmatrix}, \quad P' = P + \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta\theta \end{bmatrix} \qquad \text{.....(2.15)}$$

Dengan memasukkan persamaan 2.10, 2.13, 2.14 ke dalam persamaan 2.15, maka titik koordinat P' dapat dicari dengan menggunakan persamaan 2.16.

$$P' = f(x, y, \theta, \Delta S_r, \Delta S_l) = \begin{bmatrix} X_P \\ Y_P \\ \theta_P \end{bmatrix} + \begin{bmatrix} \frac{\Delta Sr + \Delta Sl}{2} \cos(\theta + \frac{\Delta S_r - \Delta S_l}{b}) \\ \frac{\Delta Sr + \Delta Sl}{2} \cos(\theta + \frac{\Delta S_r - \Delta S_l}{b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix} \quad \text{.....(2.16)}$$

2.2 *Dead Reckoning*

Dead reckoning biasa disebut odometri, adalah sistem yang sering digunakan dalam lokalisasi *mobile robot*, dimana posisi dan orientasi *mobile robot* relatif terhadap *frame* (bidang XY) dapat diketahui. Dengan mengetahui jumlah putaran motor penggerak roda, model gemotri dan kinematik *mobile robot*, dapat dihitung koordinat posisi saat ini dan perencanaan pergerakan menuju posisi selanjutnya. Sebagai sensor putaran roda biasa digunakan *optical encoder*.

Sebagai pokok bahasan yang berkembang saat ini adalah mengenai keakuratan atau kepresisian sistem *dead reckoning*. Kesalahan (*error*) pada sistem ini dapat dibagi menjadi 2 jenis berdasarkan penyebabnya: 1. *error* sistematis (*systematic error*), 2. *error non*-sistematis (*non-systematic error*) [14]. *Error* sistematis adalah disebabkan oleh faktor intenal robot sedangkan *error* non-sistematis disebabkan oleh faktor eksternal (tabel 2.1). *Error non*-sistematis bersifat acak dan tidak terakumulasi selama pergerakan *mobile robot* sebaliknya Error sistematis akan mengakumulasi *error* selama pergerakkannya, namun *error* ini dapat diminimalisasi dengan pengkalibrasian beberapa parameter penyebabnya seperti keseragaman diameter roda, model kinematik, kontrol pergerakan roda, dll.

Tabel 2.1 Penyebab *Error* pada sistem *Dead Reckoning*

Error sistematis	*Error* non-sistematis
Ketidaksamaan Diameter Roda	Permukaan lantai yang tidak rata
Peletakan roda yang tidak segaris	Slip roda yang disebabkan : Lantai yang licin (*slippery floors*) Percepatan yang terlalu besar (*skidding*) Gaya ekternal (angin, tumbukan, dll) Gaya internal (*roda caster*)
Ketidak pastian bentuk permukaan roda	
Resolusi *encoder*	
Waktu pencuplikan *encoder*	

2.3 *Occupancy Grids*

Dalam dunia sistem *autonomous robot*, robot membutuhkan kemampuan untuk mengenali lingkungan (*robot perception*) di sekitarnya berdasarkan masukan dari data sensor dan digunakan sebagai dasar untuk merencanakan gerak atau menavigasi robot (*motion planning*). Dalam hal ini motion planning dapat berupa gerak robot untuk menghindar dari tabrakan dengan suatu obyek (*obstacle avoidance*), mencari jalur terpendek atau dengan cost rendah untuk mencapai target tertentu (*path planning*).

Pemodelan lingkungan (pemetaan) direpresentasikan dalam bentuk multidimensi (2D atau 3D). Paradigma geometri ini akan sulit untuk digambarkan, dan membutuhkan kepastian yang tinggi akan keakuratan pemodelan sensor, pemodelan *trajectory*, mekanik robot, dan *noise* yang ada dalam lingkungan dimana robot berada. Namun hal tersebut akan sangat dibutuhkan jika robot memilki tugas yang komplek seperti: *planetary exploration*, *submarine exploration*, *servicing robot*, penggunaan dalam industri pertambangan dan beberapa aplikasi industri. Dari contoh-contoh tersebut diatas, robot haruslah mempunyai kemampuan untuk mengekplorasi dari suatu wilayah yang tak dikenal. Ekplorasi adalah *high-level*nya, maka tugas robot sebelum melakukan ekplorasi adalah: *Localization* (robot mengetahui posisi robot terhadap lingkungan

sekitarnya), *Mapping* (dari hasil *perception* dan *localization* robot mampu memetakan keberadaanya yang direpresentasikan dalam bentuk peta 2D atu 3D), *Path planning* (kemudian robot menentukan pergerakan berdasarkan jalur terpendek). Eksplorasi adalah kemampuan robot untuk menjelajah, dengan *step* pergerakan atau waktu yang paling optimal untuk dapat meng-*cover* daerah seluas luasnya.

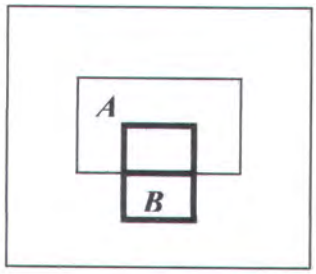
Permasalahan yang pertama kali harus dipecahkan adalah bagaimana memodelkan sensor, pergerakan robot, dan keadaan lingkungan sekitar robot yang tidak linear, sehingga pendekatan persepsi robot (*robot perception*) mendekati dengan keadaan atau persepsi yang sebenarnya.

Metode o*ccupancy grids*, pertama kali di rumuskan oleh Alberto Elfes [16,17]. Konsep dari o*ccupancy grids* adalah sebuah bidang *random* yang multidimensi yang dibentuk berdasarkan proses estimasi *stochastik* nilai *occupancy* dari setiap *cell*. Proses estimasi *cell* dilakukan dari pembacaan sensor jarak menggunakan pendekatan model probabilitas (*probabilistic model*), nilai ini akan menjadi dasar untuk meng-*capture* dari lingkungan yang tidak dikenali sebelumnya.

Dengan menggunakan aturan *Bayesian*, dapat disusun suatu algoritma untuk proses mencari nilai probabilitas *occupancy* dari setiap *cell*, dari beberapa sensor dan dari beberapa lokasi / posisi robot. Sebagai pendahuluan untuk menjelaskan teori dari *occupancy grids*, perlu dituliskan teori probabilitas dan aturan *bayesian*.

2.3.1 Probabilitas

Aksioma dari dasar teori probabilitas dapat digambarkan pada gambar 2.5 [18].



Gambar 2.5. Aksioma Probabilitas

Dapat ditulis beberapa ketetapan dari teori probabilitas dari gambar 2.5

$0 <= P(A) <= 1$ .....(2.17)

$P(True) = 1, P(False) = 0$ .....(2.18)

$P(A \text{ OR } B) = P(A) + P(B) - P(A \text{ AND } B)$ .....(2.19)

$P(\text{NOT } A) = P(\neg A) = 1 - P(A)$ .....(2.20)

Probabilitas A, memiliki nilai maksimum 100%, dan minimum 0%, atau bisa juga disebut teori ketidakpastian (*Uncertainty*). Tidak seperti pada teori 2 nilai (*bivalue logic, digital*), representasi kebenarannya dipastikan mempunyai nilai "1" atau "0", benar atau tidak benar.

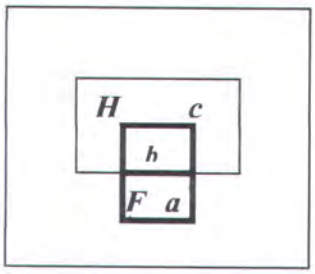
Aksioma dari teori probabilitas dapat dikembangkan sebagai berikut,

$P(A) = P(A \wedge B) + P(A \wedge \neg B)$ .....(2.21)

2.3.2 Probabilitas Kondisional

Probabilitas kondisional, sebagai contoh diberikan P(A) dan P(B), dengan fakta pada A (*prior probability*) maka berapakah kemungkinan B dengan adanya fakta tersebut, dapat dituliskan, $P(A|B)$ (*posterior probability* A).

Pada gambar 2.6, jika F adalah ungkapan kemungkinan terkena sakit flu, dan H adalah ungkapan kemungkinan terjadinya sakit kepala. Sakit kepala lebih sering terjadi, dibandingkan dengan kemungkinan terjadinya sakit flu, sehingga luas H (P(H)=1/10) lebih luas dibandingkan dengan luas F(P(F)=1/40). Dan diketahui secara statistik bahwa kemungkinan apabila saya terkena sakit flu maka 50% saya akan mengalami sakit kepala.



Gambar 2.6. Probabilitas kondisional

Dari pernyataan diatas, maka probabilitas kondisonal fakta sakit flue maka kemungkinan sakit kepala adalah sebagai berikut $P(H|F)$,

$$P(H|F) = \frac{P(H \wedge F)}{P(F)} \quad \text{.....(2.22)}$$

Analisa selanjutnya

$$P(F) = a + b \,,\, P(H) = b + c \quad \text{.....(2.23)}$$

Jadi persamaan 2.25, menjadi

$$P(H|F) = \frac{b}{a+b} \quad \text{.....(2.24)}$$

Akan timbul pertanyaan jika, seseorang mengalami sakit kepala, berapa besar kemungkinan orang tersebut akan terkena flu, tidak ada angka statistik mengenai probabilitas kondisional tersebut. Untuk mendapatkan probabilitas kondisional $P(F|H)$,

$$P(F|H) = \frac{P(H \wedge F)}{P(H)} = \frac{b}{b+c} \quad \text{.....(2.25)}$$

Dengan mengalikan persamaan 2.24, dengan faktor pengali $\left(\frac{a+b}{b+c}\right)$, maka persamaan 2.25, dapat ditulis dalam bentuk persamaan 2.26.

$$P(F|H) = P(H|F)\left(\frac{a+b}{b+c}\right) \quad \text{.....(2.26)}$$

$$P(F|H) = P(H|F)\frac{P(F)}{P(H)} \quad \text{.....(2.27)}$$

Rumus 2.27, dikenal dengan sebutan teori *Bayesian*. Teori tersebut dikemukakan oleh Thomas Bayes.

2.3.3 Independensi

Mengetahui akan sifat independensi dari suatu kejadian dengan kejadian yang lain adalah hal yang penting dalam ilmu probabilitas. Dua kejadian akan dianggap independen jika satu kejadian tidak mempunyai pengaruh terhadap kejadian yang lainnya (bukan sebab dan akibat). Dalam bentuk probabilitas kondisional, hal ini berarti dua kejadian dikatakan independen, jika dan hanya jika $P(B|A) = P(B)$ atau $P(A|B) = P(A)$,

dimana $P(B|A) = \frac{P(B \cap A)}{P(A)}$ menjadi $P(B).P(A) = P(B \cap A)$. Jika $A \cap B = B \cap A$, maka dapat ditulis seperti persamaan 2.28

$$P(B).P(A) = P(B \cap A) = P(A \cap B) \quad \text{.....(2.28)}$$

Maka jika terdapat beberapa kejadian independen A1, A2,..,An, akan dapat ditulis seperti persamaan 2.29.

$$P(A_1 \cap A_2 \cap ... \cap A_n) = P(A_1)P(A_2)....P(A_3) \quad \text{.....(2.29)}$$

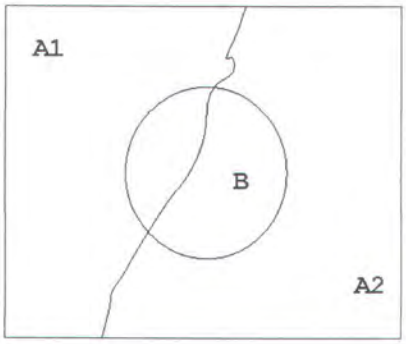
2.3.4 Probabilitas Total

Suatu himpunan semesta S pada gambar 2.7, dibagi menjadi 2 bagian A1 dan A2 yang eksklusif, dimana A1 $\cap$ A2 = 0 dan A1 $\cup$ A2 = S. Jika B adalah bagian dari S, B $\cap$ A1 dan B $\cap$ A2 adalah eksklusif, maka dalam notasi matematika seperti ditunjukkan dalam persamaan 2.30.

$$P(B) = P(B \cap A_1) + P(B \cap A_2) \quad \text{.....(2.30)}$$

Hasilnya, jika S terdapat n bagian yang eksklusif A1,A2,..,An. Jika B adalah kejadian didalam S, maka probabilitas B dapat dihitung dengan persamaan 2.31.

$$P(B) = \sum_{i=1}^{n} P(B|A_i).P(A_i) \quad \text{.....(2.31)}$$



Gambar 2.7. Diagram venn probabilitas total

2.3.5 Teori *Bayes*

Anggap suatu himpunan semesta S adalah *union* dari 2 kejadian yang eksklusif A1 dan A2. Dalam keadaan ini $A_1 = \neg A_2$. Bila terjadi kejadian B, kita bertanya bagaimana kemungkinan terjadinya kejadian A1?, dalam bentuk probabilitas kondisional, berapa $P(A_1|B)$?. Kita sudah mengetahui bahwa $A \cap B = B \cap A$, maka probabilitas B kondisonal terhadap A_1 dan A_2 adalah seperti pada persamaan 2.32 dan 2.33.

$$P(B \cap A_1) = P(A_1|B)P(B) = P(B|A_1)P(A_1) \quad \text{.....(2.32)}$$

$$P(B \cap A_2) = P(A_2|B)P(B) = P(B|A_2)P(A_2) \quad \text{.....(2.33)}$$

Dari persamaan 2.31, maka probabilitas B seperti pada persamaan 2.34.

$$P(B) = P(B|A_1)P(A_1) + P(B|A_2)P(A_2) \quad \text{.....(2.34)}$$

substistusi persamaan 2.34 ke persamaan 2.32 menjadi persamaan 2.35.

$$P(A_1|B)\left[P(B|A_1)P(A_1) + P(B|A_2)P(A_2)\right] = P(B|A_1)P(A_1)$$

$$P(A_1|B) = \frac{P(B|A_1)P(A_1)}{\left[P(B|A_1)P(A_1) + P(B|A_2)P(A_2)\right]} \quad \text{.....(2.35)}$$

Jika himpunan semesta dibagi menjadi n bagian yang ekslusif {Ai}, (i=1,2,....,n), maka bentuk persamaan 2.35 menjadi persamaan 2.36.

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{\infty} P(B|A_j)P(A_j)} \quad \text{.....(2.36)}$$

2.4 Metode Probabilitas *Occupancy Grids*

Definisi *occupancy grids* secara praktisnya adalah pembagian suatu wilayah (m x m), menjadi beberapa bagian wilayah homogen yang lebih kecil ukurannya (n x n), dimana setiap wilayah kecil tersebut disebut *cell* (C), dimana setiap *cell* memilki properti yang disebut *probabilitas occupancy* P(S(c)=OCC), notasi S mewakili dari keadaan *cell* (state). *Occupancy* secara linguistik bahasa indonesia adalah "pemilikan", pada kasus ekplorasi memiliki makna *cell* yang masuk dalam daerah cakupan sensor robot. *Occupied Cell* merupakan *cell* dimana *cell* terebut berada pada jarak maksimum sensor jarak pada robot. Sehinga pada daerah antara *cell* posisi robot dengan *cell* pada jarak maksimum sensor disebut daerah *occupancy*. Jadi daerah tersebut terdiri dari beberapa *cell* yang memilki 2 keadaan probabilitas. Probabilitas yang pertama adalah bahwa *cell* tersebut kemungkinan terletak pada jarak batas sensor P(S(C)=OCC) dan yang kedua adalah kemungkinan *cell-cell* tersebut pada jarak dibawah jangkauan maksimum sensor. Jika tidak ada *obstacle* didepan robot maka *cell* tersebut merupakan *cell* yang kosong (EMPTY =EMP), sehingga pada wilayah *occupancy* robot O(C), dan berdasar aksioma dari teori probabilitas, maka bentuk notasi persamaannya dapat ditulis seperti pada persamaan 2.37.

$$O(C) = P[(S(C) = OCC](C) \quad \text{.....(2.37)}$$

$$P[(S(C) = OCC](C) = 1 - P[(S(C) = EMP](C) \quad \text{.....(2.38)}$$

Persamaan (2.34) dan (2.35), dalam bentuk diskrit ditunjukkan dalam persamaan 2.39 dan 2.40.

$$O(C_i) = P[(S(C_i) = OCC](C_i) \quad \text{.....(2.39)}$$

$$P[(S(C_i) = OCC](C_i) = 1 - P[(S(C_i) = EMP](C_i) \quad \text{.....(2.40)}$$

Probabilitas occupancy berhubungan dengan jangkauan maksimum sensor jarak, r. Sensor jarak tidaklah selalu menghasilkan nilai yang pasti, sehingga hubungan antara sensor yang memiliki nilai ketidakpastian (*uncertainty*) bisa diselesaikan dengan teori probabilitas. Probabilitas kondisional, jika di ketahui posisi *cell* Ci, maka dengan hasil pengukuran sensor r, nilai probabilitas bahwa *cell* tersebut merupakan *cell Occupied*, dinotasikan, $P[(S(C_i) = OCC|r]$, maka daerah *occupancy* dalam bentuk diskrit,

$$O(C_i|r) = P[(S(C_i) = OCC|r](C_i) \quad \text{.....(2.41)}$$

Dengan melihat kembali aturan berantai *bayesian* (2.36), maka probabilitas cell ke-i kondisional terhadap pembacaan sensor jarak r adalah dalam persamaan 2.42.

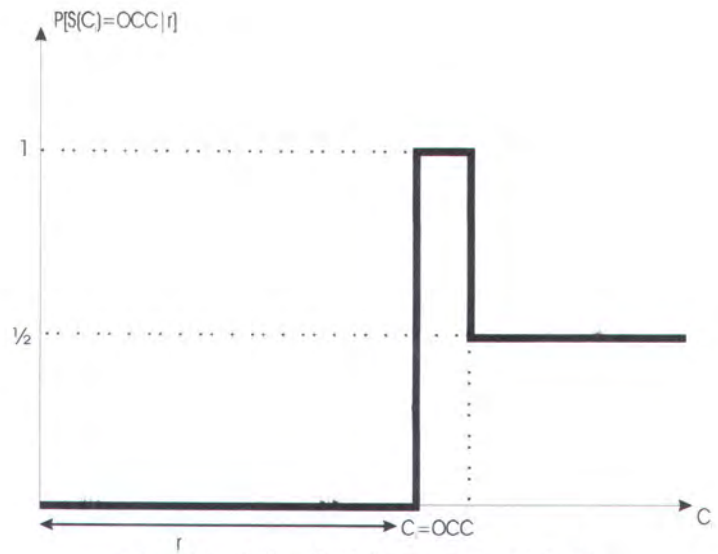
$$P[(S(C_i) = OCC|r] = \frac{P[r|S(C_i) = OCC].P[S(C_i) = OCC)]}{\sum_{S(C_i)} P[r|S(C_i) = OCC].P[S(C_i) = OCC)]} \quad \text{.....(2.42)}$$

2.4.1 Pemodelan Sensor (*Sensor Model*)

Bentuk $P[r|S(C_i) = OCC]$, tidaklah mewakili langsung dari model sensor karena sensor rentan terhadap noise hasil pembacaan, atau terdapat toleransi kesalahan, dimana toleransi ini merupakan bagian perpotongan antan *occupied* dan *empty* (2.43)

$$P(r|z) = P[r|S(C_i) = OCC \wedge S(C_k) = EMP, k < i] \quad \text{.....(2.43)}$$

Untuk sensor yang ideal, secara ilustrasi, dapat digambarkan pada gambar 2.7



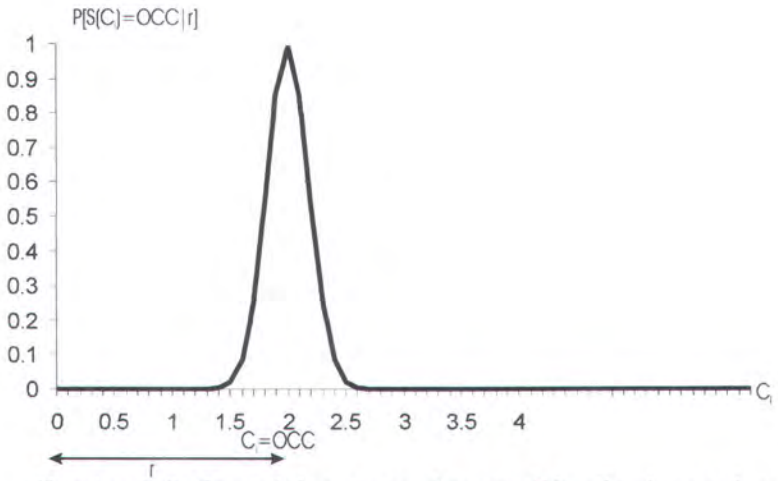
Gambar 2.8 . Model sensor ideal [17]

Metode *occupancy grid* yang dikemukakan Elfes [16][17], memberikan nilai probabilitas awal untuk semua *cell* dengan nilai ½ yang artinya belum masuk daerah *occupancy*, atau *cell* tersebut mempunyai probabilitas 50% bahwa *cell* tersebut merupakan *cell empty* dan 50 % bahwa *cell* tersebut merupakan *cell occupied*. Saat sensor jarak bekerja, dengan mengetahui jarak dari hasil pembacaan sensor ideal, maka dapat dirumuskan dalam persamaan (2.41)

$$P[S(Ci) = OCC|r] = \begin{cases} 0 & x < r, x \in Ci \\ 1 & x, r \in Ci \\ 1/2 & x > r, x \in Ci \end{cases} \quad \text{.....(2.44)}$$

Dalam memodelkan sensor biasanya bisa dilakukan dengan beberapa metode, diantaranya dengan menggunakan *distribusi gaussian*, *neural network*. Pada model sensor dengan menggunakan *distribusi gaussian* dapat dirumuskan sebagai berikut,

$$P(r|z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{r-z}{\sigma}\right)^2} \quad \text{.....(2.45)}$$

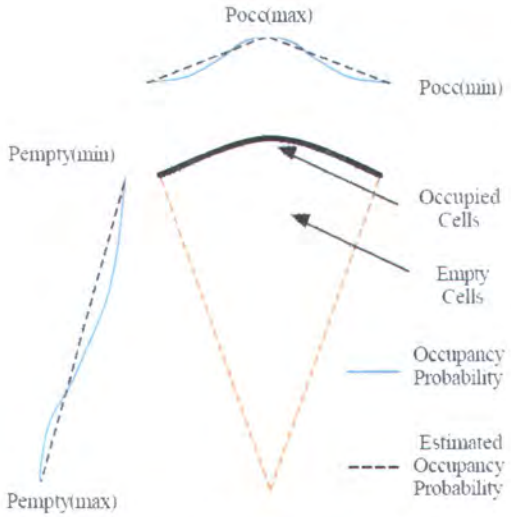


Gambar 2.9. Pemodelan sensor dengan *Distribusi gaussian*

Pada gambar 2.8, jarak maksimum r adalah 2 m dengan menggunakan *distribusi gaussian* dengan menentukan kooefesian *variant* σ2 dari hasil percobaan. Pada posisi *cell* dimana berjarak 1,5m s/d 2m mempunyai nilai probabilitas antara 0,1 s/d 1. Pada sensor yang sebenarnya, *noise* pembacaan tidak hanya terjadi pada data pembacaan jarak sensor dengan benda, namun juga terdapat *noise* pada penyimpangan sudut pembacaan (Sensor ultrasonic), maka model sensor dengan 2 variabel r dan θ, persamaan(2.43)[16,17]

$$P(r|z,\theta) = \frac{1}{\sigma_\theta \sigma_r \sqrt{2\pi}} e^{-\frac{1}{2}\left(\left(\frac{r-z}{\sigma_r}\right)^2 + \left(\frac{\theta}{\sigma_\theta}\right)^2\right)} \quad \text{.....(2.46)}$$

Secara lengkap pemodelan sensor jarak (ultrasonik) dapat digambarkan seperti pada gambar 2.9 [19].



Gambar 2.10. Pemodelan sensor jarak (Ultrasonic)

Probabilitas bahwa *cell* didepan sensor adalah merupakan *cell occupied* $P[(S(C_i) = OCC|r]$ mempunyai batas POCC(Max) dan POCC(Min) dimana distribusi titik antara POCC(Max) dan POCC(Min) adalah fungsi *distribusi Gaussian*. Probabilitas $P[(S(C_i) = EMP|r]$ memilki batas PEMP(Max) dan PEMP(Min), makin jauh *cell* dan mendekati *cell occupied* maka nilai $P[(S(C_i) = EMP|r]$ bernilai kecil dengan batas PEMP(Min). *Cell* dimana tempat posisi robot berada memiliki nilai $P[(S(C_i) = EMP|r]$ tertinggi = PEMP(Max).

Pada setiap keadaan robot bergerak dalam tujuannya untuk eksplorasi, maka nilai *probabilitas occupancy* dari setiap *cell* juga akan berubah, untuk itu perlu dilakukan peng-*update*-an nilai probabilitas *occupancy* dari setiap *cell*-nya. Peng-*update*-an berdasarkan urutan sekuensial dari beberapa hasil pembacaan sensor, {r}t={r1,r2,...,rt}, sehingga rumus rantai *bayesian* (2.36) dapat ditulis kembali sebagai berikut [16, 17],

$$P\left[(S(C_i)=OCC|\{r\}_{t+1}\right]=\frac{P\left[r_{t+1}|S(C_i)=OCC\right]P\left[S(C_i)=OCC)|\{r\}_t\right]}{\sum_{S(C_i)}P\left[r_{t+1}|S(C_i)=OCC\right]P\left[S(C_i)=OCC)|\{r\}_t\right]} \quad \text{.....(2.47)}$$

2.4.2 Lokalisasi Posisi Robot (*Localization*)

Keakuratan pembuatan peta matrik (2D) sangatlah tergantung dari seberapa tepat perkiraan posisi robot dalam *map* tersebut. Slip dan kesalahan odometri seringkali menyebabkan estimasi posisi robot. Identifikasi dan koreksi terhadap kesalahan ini menjadi hal yang sangat penting dalam kasus ini. Dari uraian diatas, bisa kita simpulkan ada 2 hal pokok dalam lokalisasi:

Odometri atau *encoder* pada roda robot. Odometri untuk waktu tempuh yang pendek memang bisa diandalkan keakuratannya, namun untuk ekplorasi daerah yang relatif lebih luas akan menyebabkan akumulasi kesalahan yang cukup besar. Informasi yang dapat diberikan oleh *encoder* pada kedua roda adalah jarak yang dtempuh oleh kedua roda kiri dan kanan (*Differential Drive Robot*), dari informasi ini dapat digambarkan posisi robot dan pose robot dalam bidang 2D (koordinat x,y,θ), lihat persamaan inverse kinematik *differential drive* (2.16)

Kesesuaian *Local Map* dan *Global Map* (*Map Matching*). Pada saat sensor robot mulai bekerja, robot sudah dapat mengkonstruksi keadaan di sekelilingnya yang direpresentasikan dalam bidang 2D, disebut peta lokal (*local map*). Pada kasus ekplorasi, robot diletakkan pada suatu area yang tidak dikenali (*Global Map Unknown*), oleh sebab itu pada saat pertama robot membaca sensor, *local map* adalah sama dengan *global map*. Karena letak awal relatif terhadap *global Map* adalah tidak diketahui, maka asumsi yang bisa dilakukan adalah robot awal diletakkan pada titik 0(2D: 0,0) atau disebut titik asal

(origin). Kemudian *local map* dibentuk berdasarkan jangkauan maksimum dari sensor jarak yang terpasang pada robot (luasan *local map* selalu tetap), seiring dengan pergerakan robot, pengetahuan robot akan keadaan disekeliling robot juga akan bertambah, *global map* bertambah luas. *Local map* dan *global map* dibentuk dari daerah *occupancy* robot, dengan menerapakan teknik *probabilitas occupancy* pada pembahasan dasar teori sebelumnya. Dengan menentukan korelasi antara *local map* terhadap *global map*, posisi robot dalam bidang xy dapat diestimasi. Dalam ilmu *computer vision*, Seperti gambar (*grey level*) *PCB* dengan ukuran gambar X x Y, terdapat gambar lubang solder dengan ukuran (Xm x Ym). Posisi tengah dari lubang solder relatif terhadap gambar *PCB* dapat ditentukan dengan mengkorelasikan *kernel* gambar lubang solder dengan gambar *PCB*. Dari proses korelasi ini nilai *grey level*, setelah di *treshold* akan mempunyai perbedaan nilai yang besar, sehingga dapat ditetentukan posisi tengah dari lubang solder [21]. Informasi dari *Map Matching* ini merupakan sumber informasi kedua dari mencari posisi robot (*localization*) selain dengan menggunakan odometri.

2.4.3 Eksplorasi (*Exploration*)

Maksud eksplorasi adalah robot bergerak dengan *cost* seminim mungkin untuk mencapai daerah *coverage* yang seluasnya. Untuk itu robot harus mempunyai kemampuan untuk memperhitungkan jarak terpendek (*minimum cost path*) untuk mencari daerah yang belum terekplorasi. Untuk mencari *cost* yang minimum menggunakan nilai *probabilitas occupancy* pada setiap *cell* pada daerah *occupancy* robot. Algoritma iterasi untuk mencari *cost* ini disebut algoritma *dynamic programming*. Pseudo algoritma ini dapat didiskripsikan sebagai berikut:

Inisialisasi. Inisialisasi disini adalah memberikan nilai awal pada daerah *occupancy* robot. Memberikan nilai *cost* = 0 untuk *cell* yang belum terekplorasi dan *cost* = ~ untuk *cell* yang sudah diekplorasi. *Cell* yang sudah tereksplorasi adalah *cell* yang paling tidak nilai probabilitas *occupancy*-nya telah di-*update* 1 kali.

$$V_{x,y} = \begin{cases} 0, & \texttt{Jika C(x,y) belum tereksplorasi} \\ \infty, & \texttt{Jika C(x,y) telah tereksplorasi} \end{cases} \quad \text{.....(2.48)}$$

Iterasi : *Update* nilai Vx,y. Untuk semua *cell* yang telah dieksplorasi,

$$V_{x,y} = \underset{\substack{\gamma=-1,0,1 \\ \lambda=-1,0,1}}{Min} \{V_{x+\gamma,y+\lambda} + \text{Prob}[C_i = OCC, (x+\gamma, y+\lambda)] \quad \text{.....(2.49)}$$

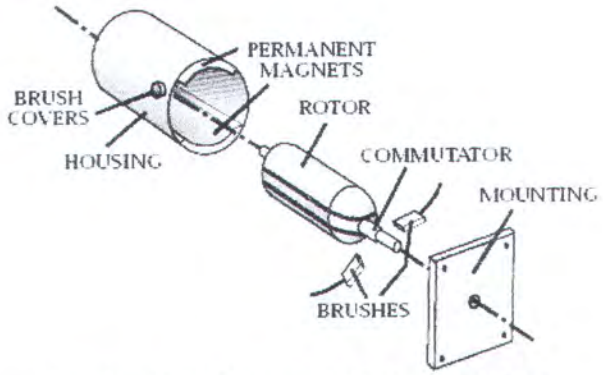
Vx,y dengan nilai *cost* yang paling rendah merupakan arah dan tujuan pergerakan robot selanjutnya. Setelah robot melakukan pergerakan ke *Cell* Ci(x,y), maka lakukan *reset* untuk semua daerah yang terekplorasi dengan memberikan nilai = ~

$$V_{x,y} = \infty \quad \text{, jika } V_{x,y} \leq \underset{\substack{\gamma=-1,0,1 \\ \lambda=-1,0,1}}{Min} \{V_{x+\gamma,y+\lambda} + \text{Prob}[C_i = OCC, (x+\gamma, y+\lambda)] \quad \text{.....(2.50)}$$

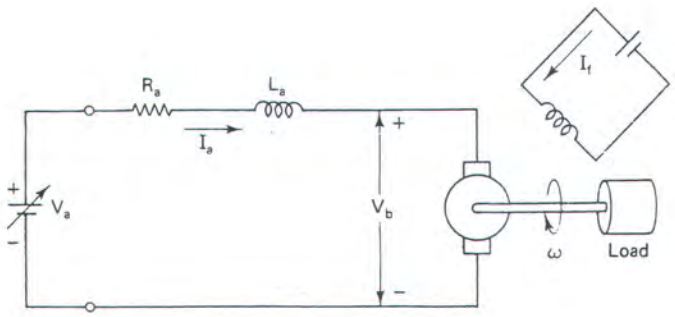
Untuk menghemat komputasi, maka daerah untuk mencari *cost* tersebut dibatasi dengan *bounding box*, sehingga pencarian jalur terpendek berdasarkan iterasi *dynamic programming* hanya terfokus pada area yang kecil saja.

2.5 Model dan Kontroler Motor *DC*

2.5.1 Model Motor DC



Gambar 2.11. Konstruksi Motor DC [22]



Gambar 2.12. Model motor DC [23]

Kontruksi motor DC (gambar 2.11), Model motor DC (gambar 2.12), kumparan motor (bagian yang berputar, rotor) dapat dimodelkan dalam bentuk rangkaian Ra dan La yang dipasang secara seri. Vb merupakan tegangan induksi balik (*back* EMF) yang ditimbulkan oleh kumparan yang berputar pada medan magnet yang permanen. Kumparan motor berputar dengan kecepatan ω rad/detik, memutar beban (*load*).

$$V_b = K_b \times \omega \quad \text{.....(2.51)}$$

Kb = Konstanta Induksi

Medan magnet ditimbulkan oleh magnet permanen, yang juga menghasilkan arus induksi If pada kumparan motor, jadi besarnya torsi pada kumparan motor dipengaruhi oleh arus Ia dan arus induksi If,

$$T_m = K.I_f.I_a \quad \text{.....(2.52)}$$

K = Konstanta kesebandingan torsi dengan arus induksi If dan arus kumparan Ia.

Persamaan model motor (gambar 2.12),

$$I_a = \frac{V_a - V_b}{R_a} \quad \text{.....(2.53)}$$

Dengan mensubstitusi pers. 2.52 ke pers. 2.53, maka menjadi persamaan 2.54.

$$T_m = K.I_f.\frac{V_a - V_b}{R_a} \quad \text{.....(2.54)}$$

Kemudian mensubstitusi pers 2.51 ke pers. 2.54, maka menjadi persamaan 2.55.

$$T_m = K.I_f.\left(\frac{V_a}{R_a} - \frac{K_b.\omega}{R_a}\right) \quad \text{.....(2.55)}$$

Magnet permanen menyebabkan arus induksi If dengan nilai yang tetap, K.If = Ki. Torsi motor Tm , adalah gaya gerak motor untuk memutar beban motor (*Load*), dengan gaya gesekan Td, dan gaya akibat perlawanan induksi magnet permanen Tf, jadi dalam bentuk matematika,

$$T_m = T_l + T_d + T_f = K_i.I_a \quad \text{.....(2.56)}$$

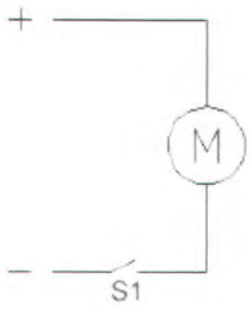
Dari pers. (2.55) dan (2.56) dapat diketahui bahwa torsi motor berbanding lurus dengan arus kumparan Ia atau berbanding lurus dengan tegangan motor Va dan kecepatan putar motor ω.

2.5.2 Kontroler Motor DC

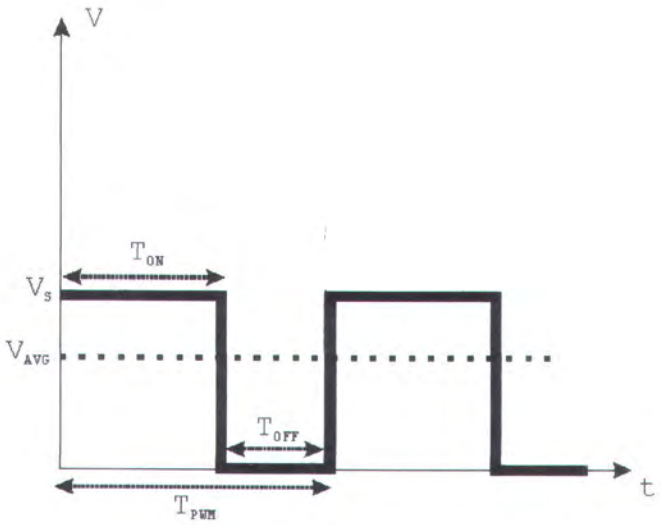
Kecepatan putaran motor DC adalah sebanding dengan tegangan sumber DC yang diberikan ke motor [24]. Misalnya, untuk kecepatan putaran maksimum dibutuhkan tegangan DC 12V, maka saat tegangan DC diturunkan menjadi 6V, maka motor akan berputar dengan ½ kali kecepatan maksimumnya.Untuk tegangan sumber DC yang konstan, pengaturan tegangan motor dapat dilakukan dengan metode PWM (*Pulse Width Modulation*). PWM juga biasa disebut dengan PDM (*Pulse Duration Modulation*) [25].

Prinsip kerja PWM (gambar 2.18) adalah dengan mengatur lama waktu *switch* S "ON" dan waktu *switch* S "OFF", sehingga dengan cara pengaturan tersebut, tegangan rata-rata yang dirasakan oleh motor M akan bervariasi dari 0V sampai dengan tegangan maksimum sumber DC (Vs). Pada gambar 2.19, periode sinyal PWM adalah jumlah dari lama waktu *switch* S "ON" dan waktu *switch* S "OFF" , pers. (2.65).

Pembuatan PWM dalam rancangan ini tidak digunakan sebagai kontrol kecepatan seperti pada sistem kontrol *close loop*. PWM digunakan untuk mengkalibrasikan kecepatan motor roda kanan dan kiri dalam kaitannya untuk memperoleh konfigurasi kecepatan putar yang tepat untuk manuver robot (bergerak maju 30 cm, pivot ke kanan 90° dan pivot ke kiri 90°). Pengkalibrasian harus dilakukan karena karakteristik motor dan beban torsi pada masing-masing motor berbeda. Ketidakseimbangan desain mekanis dan beban roda dapat menyebabkan perbedaan karakteristik tersebut.



Gambar 2.13. Prinsip PWM



Gambar 2.14. Periode sinyal PWM

$$T_{PWM} = T_{ON} + T_{OFF} \quad \text{.....(2.57)}$$

Tegangan rata-rata VAVG,

$$V_{AVG} = \frac{1}{T_{PWM}} \int_{0}^{T_{PWM}} V_{PWM}.dt \quad \text{.....(2.58)}$$

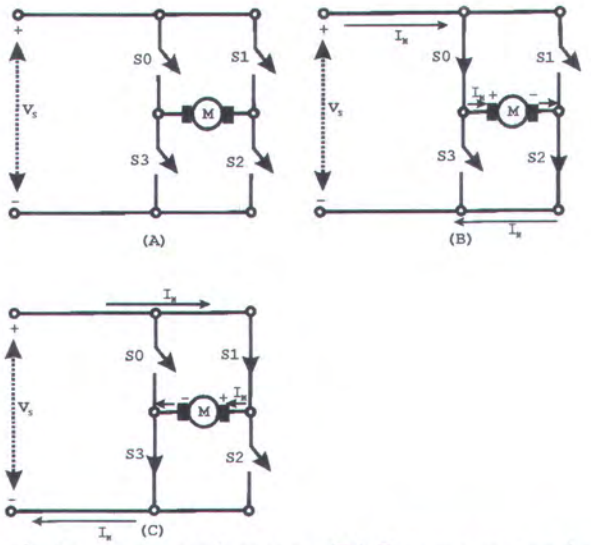
$$V_{AVG} = \frac{1}{T_{PWM}} \left(\int_{T_{ON}} V_S.dt + \int_{T_{OFF}} 0.dt \right) \quad \text{.....(2.59)}$$

Dari persamaan 2.59, jika TON=TOFF, maka tegangan rata-rata VAVG,

$$V_{AVG} = \frac{V_S . T_{ON}}{T_{PWM}} = \frac{V_S}{2} \quad \text{.....(2.60)}$$

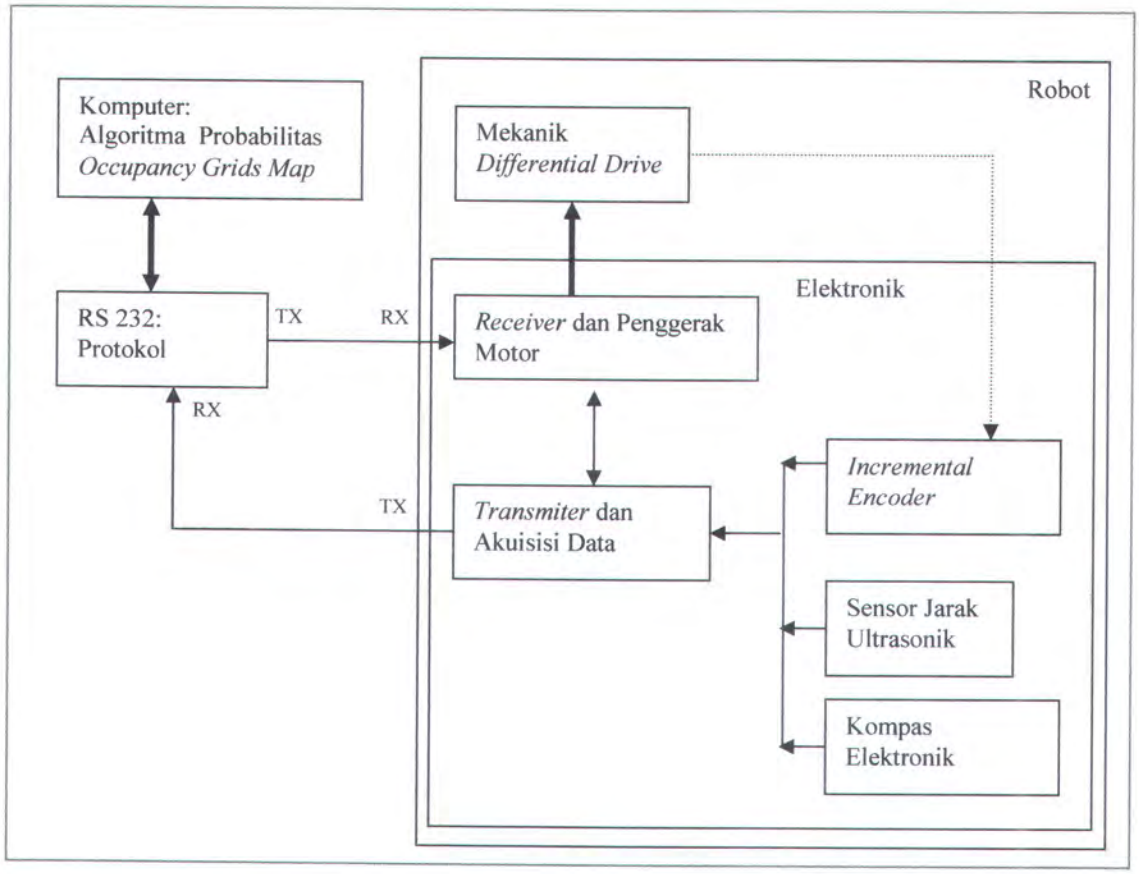
Dengan membalik polaritas VS, maka arah arus akan berubah (gambar 2.13), sehingga menyebabkan arah putaran motor DC juga akan berubah. Karena motor pada setiap penggerak roda robot harus dapat bergerak dalam dua arah, maju atau mundur, searah jarum jam atau berlawanan arah jarum jam (*Clock Wise* CW *atau Counter Clock Wise* CCW), maka perlu ditambahkan rangkaian elektronika yang dapat membalik arah putaran motor.

Gambar 2.15 adalah rangkaian penggerak motor yang umum digunakan untuk menggerakkan motor dengan 2 arah putaran. Prinsip dasar dari rangkaian penggerak motor dapat digambarkan seperti gambar 2.20. Saat switch S0, S2 "ON" dan S1, S2 "OFF", arus IM mengalir dengan arah seperti pada gambar 2.15 (B). Menyebabkan motor DC berputar dengan arah putaran searah jarum jam. Begitu juga sebaliknya, saat S0,S1 "OFF" dan S1,S2 "ON", arah arus IM menyebabkan polaritas tegangan pada motor berubah, sehinnga menyebabkan arah putaran motor berlawanan arah jarum jam, gambar 2.15 (C).



Gambar 2.15. (A) Rangkaian penggerak motor, (B) Putaran motor dengan arah CW, (C) Putaran motor dengan arah CCW

BAB III
PERANCANGAN

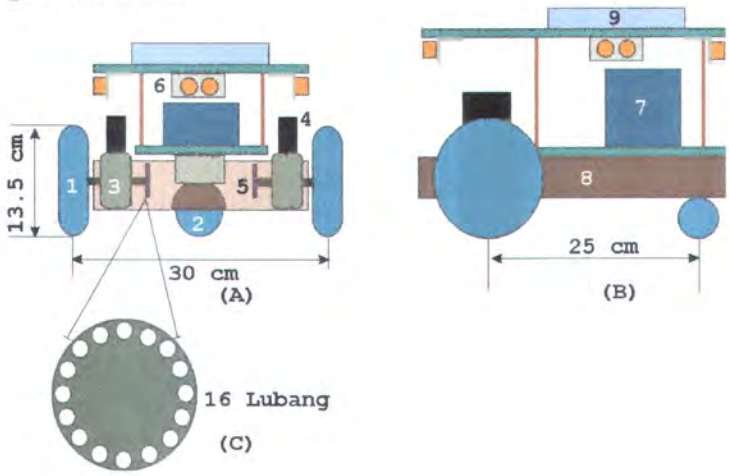


Gambar 3.1 Diagram blok perancangan sistem robot

Perancangan robot terdiri dari 2 bagian: perancangan mekanik dan elektronik, seperti pada gambar 3.1. Pada bagian blok elektronik terdapat modul-modul: "*receiver* dan penggerak motor", "transmiter dan akuisisi data", incremental encoder, sensor jarak ultrasonik dan kompas elektronik. Modul *receiver* akan menerima data berupa perintah dari komputer, data berupa perintah tersebut kemudian akan divalidasi. Jika proses validasi berhasil, maka modul ini akan melaksanakan 2 hal yaitu: pertama adalah

mengirimkan sinyal kepada penggerak motor untuk menggerakkan kedua motor dengan kecepatan dan arah putar sesuai dengan perintah yang dikirimkan komputer, dan hal yang kedua adalah mengirimkan sinyal untuk memberitahu modul "*transmiter* dan akuisisi data" untuk memulai mengaktifkan sensor dan pembacaan data dari sensor-sensor yang digunakan. Sinyal dari modul "*receiver* dan penggerak motor" kembali dikirimkan sebagai tanda robot telah berhenti bergerak, dan "modul *transmiter* dan data akuisisi" berhenti membaca data dari sensor-sensor kemudian mengirimkan hasil pembacaaan data sensor-sensor ke komputer. Program didalam komputer akan memvalidasi data yang dikirimkan robot dengan format protokol yang akan dibahas dalam bab ini. Data yang valid kemudian diproses dalam algoritma probabilitas *occupancy grids map*. Algoritma ini menghasilkan arah navigasi robot selanjutnya dalam usahanya untuk menjelajah daerah yang tidak dikenali. Komunikasi antara komputer dengan robot adalah menggunakan *port* serial. Untuk mengantisipasi kesalahan data akibat *noise*, maka perlu untuk dirancang protokol komunikasi data. Pada pembahasan mengenai perancangan ini akan dijelaskan detail dari masing-masing blok dalam gambar 3.1.

3.1 Perancangan Mekanik



Gambar 3.2. Mekanik robot. (A) tampak depan, (B) tampak samping, (C) *optical encoder*

Perancangan mekanik robot (gambar 3.2), terdiri dari beberapa komponen dengan keterangan sebagai berikut:

1) Roda kiri dan roda kanan memiliki diameter 13.5 cm.
2) Roda castor, atau *freewheel* yang dapat bergerak bebas.
3) *Gearbox*, berfungsi untuk mengurangi torsi pada beban roda dan badan robot.
4) Motor DC 12V, untuk penggerak roda kiri dan kanan.
5) *Optical encoder* (gambar 2.1 (C)), berupa piringan dengan 16 lubang, yang berfungsi untuk mengukur jarak pergerakan robot, dilihat dari roda kiri dan roda kanan.
6) Sepasang *tranduser ultrasonic* (TX dan RX), sebagai sensor jarak. Terdapat 4 pasang *tranduser ultrasonic* yang diletakkan pada bagian: depan, kanan, kiri dan belakang robot.
7) Baterai 12V,7A dan 6V,4A yang digunakan sebagai catudaya 2 motor DC 12 V dan catudaya rangkaian kontrol dan komunikasi robot.
8) Platform robot berbentuk T terbuat dari bahan aluminium dengan ketebalan 3mm.
9) Rangkaian kontrol robot terdiri dari 2 modul: modul pertama terdiri dari *receiver* dan penggerak motor atau roda robot, modul kedua terdiri dari rangkaian pembacaan sensor-sensor robot (2 incremental encoder, 4 pasang *tranduser ultrasonik*, 1 kompas elektronik) dan *transmitter*. Kedua modul tersebut menggunakan mikrokontroler ATMEL AVR 8535L (ATMEGA8535L) sebagai pusat kontrollernya.

Ukuran yang terpenting dari mekanik robot adalah jarak antara roda 30 cm, panjang robot 25 cm, dan diameter roda 13.5 cm. Resolusi *encoder* 16 lubang adalah,

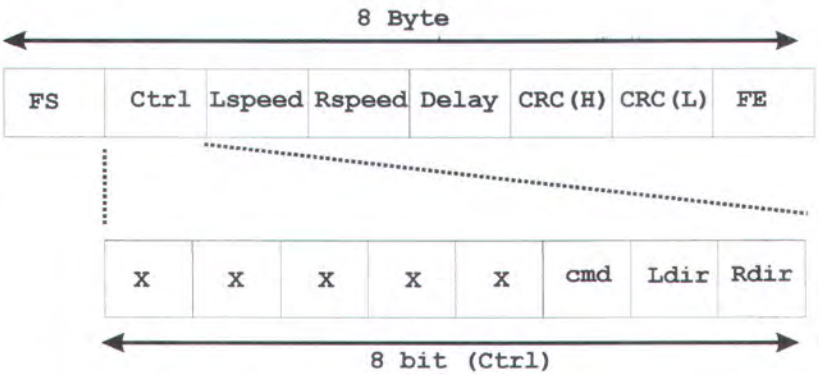
$$\text{Re}solusiEncoder = \frac{\pi.D}{16} = \frac{3.14 \times 13.5}{16} = \frac{42.39}{16} = 2.649375 \quad \text{.....(3.1)}$$

Jadi 1 pulsa yang terbaca dari pembacaan sensor *optical encoder* adalah sama dengan pergerakan 2.649375 cm.

3.2 Perancangan Protokol Komunikasi

Pada prinsipnya robot bersifat pasif, algoritma eksplorasi di komputer menghasilkan perintah (*command*) yang harus dikerjakan oleh robot, command bisa dibedakan menjadi 2 jenis, yaitu *command* untuk pergerakan robot (navigasi robot), dan *command* untuk permintaan data sensor. Pada layer dasar menggunakan protokol RS232, pada tingkat aplikasi menggunakan format protokol dengan aturan yang *custom* (bukan standard industri). Perancangan protokol dibedakan antara protokol untuk *command* dan protokol data sensor. Namun untuk menjamin validitas data komunikasi, aturan ke dua protokol yang digunakan mirip dengan protokol standar industri MODBUS.

- Format paket *command* seperti pada gambar 3.3.

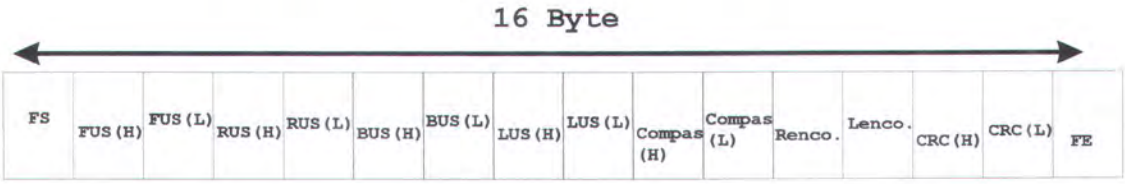


Gambar 3.3. Format paket *command*

- FS, *Frame Start*. Tanda diawalinya paket data command. Menggunakan bilangan 02h, bilangan tersebut hanya merupakan kesepakatan aturan saja, dan tidak mutlak menggunakan bilangan tersebut.
- Ctrl, *Control*. Tidak semua bit dipakai, hanya bit ke-0 sampai bit ke-2 saja yang digunakan. Bit ke-0 (Rdir, right direction) dan bit ke-1 (Ldir, left direction), digunakan untuk memberikan perintah arah putaran roda kanan dan roda kiri robot, dimana nilai "1" berarti bergerak maju dan "0" untuk bergerak mundur. Bit ke-2 (cmd, command) digunakan untuk menentukan jenis perintah, cmd=1 berarti permintaan agar robot mengirimkan data dari sensor (data request), dan cmd=0 berarti perintah robot untuk melakukan pergerakan.
- LSpeed dan RSpeed, adalah data PWM untuk menggerakkan motor penggerak roda kiri dan kanan. Memiliki jangkauan nilai 0-255, 0 berarti motor berhenti dan 255 motor berputar dengan kecepatan maksimum.
- *Delay*, waktu tunda, atau lama waktu motor berputar dengan kecepatan tertentu (LSpeed dan Rspeed). Waktu untuk pergerakan kedua motor adalah sama.
- CRC (*cyclic redundant code*), kode CRC yang digunakan adalah CRC 16 bit (CRCH (H= MSByte)) dan CRCL (L=LSByte)). CRC digunakan untuk memvalidasi paket data bersama dengan kode FS (*frame start*) dan FE (*frame end*). Algoritma CRC akan dijelaskan pada pembahasan tersendiri.

- FE (*frame end*), merupakan kode 1 byte untuk menandai akhior dari paket data, sama halnya dengan kode FS, aturan kode hanya merupakan kesepakatan disain saja. Kode FS yang digunakan adalah 03h.

- Format paket data sensor seperti pada gambar 3.4



Gambar 3.4. Format paket data sensor

- FS (*frame start*) dan FE (*frame end*) sama dengan nilai yang terdapat pada format paket *command*. (FS=02h dan FE=03h)
- XUS adalah data sensor jarak ultrasonik 2 Byte, XUSH dan XUSL. Dengan X adalah simbol untuk 4 sensor jarak ultrasonik yang diletakkan di bagian depan (X=F), kanan (X=R), belakang (X=B), dan kiri robot (X=L).
- Kompas(H) dan Kompas(L), adalah data 2 byte untuk hasil pembacaan sensor Kompas elektronik.
- Renco dan Lenco, adalah *data incremental encoder* pada roda kanan dan kiri.

Sedangkan algoritma sistem komunikasi antara pusat kendali (komputer) dengan robot adalah sebagai berikut:

Initialization
Ack=false
While Ack=false
Send command(data request)

```
            Delay_transmission()
            Ack=receive(RcvData)
        End
Robot position (x,y,θ)=RcvData
Repeat
Navigation
        Move()=Occupancy Grids(Robot position(x,y,θ)
        {Move Again:}
        Send command(Move())
Ack=False
        While Ack=false
            Send command(data request)
            Delay_transmission()
            Ack=receive(RcvData)
        End
        If Robot position (x,y,θ)=RcvData then
            goto {Move Again}
        End if
        Robot position (x,y,θ)=RcvData
Until Stop Criteria
```

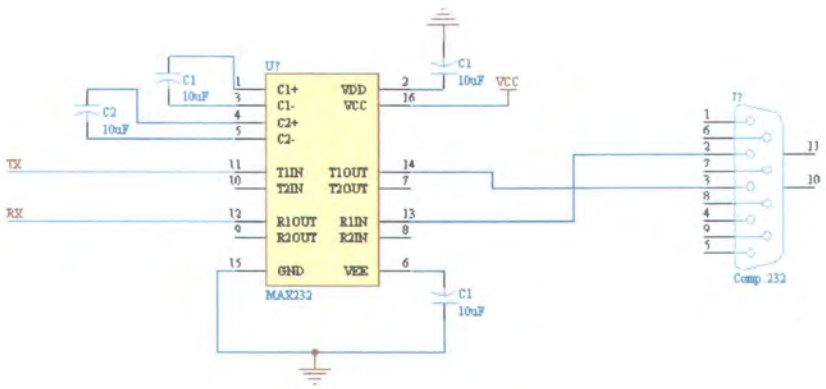
Dalam algoritma diatas, pada baris ACK=*receive*(RcvData) akan bernilai benar "*true*" jika *frame* data (RcvData) adalah valid dengan menguji nilai FS dan FE serta nilai CRC. Algoritma untuk menghitung nilai CRC dan menguji nilai CRC adalah sebagai berikut,

- Untuk perhitungan CRC:

```
ValData[] = DATA []
N = Packet Lenght
CRC = FFFFh
for i = 1 to N-3
        CRC = CRC XOR ValData[i]
        for j = 1 to 8
            if (CRC AND 1) = 1 then
                CRC = CRC SHR 1
                CRC = CRC XOR A001h
            else
                CRC = CRC SHR 1
            end if
        next j
next i
```

  - Untuk menguji nilai CRC:

RcvCRC = DATACRC (DATA[])
RealCRC = CRC(DATA[])
if RcvCRC = RealCRC then
    Valid=true
else
    Valid=false
End if

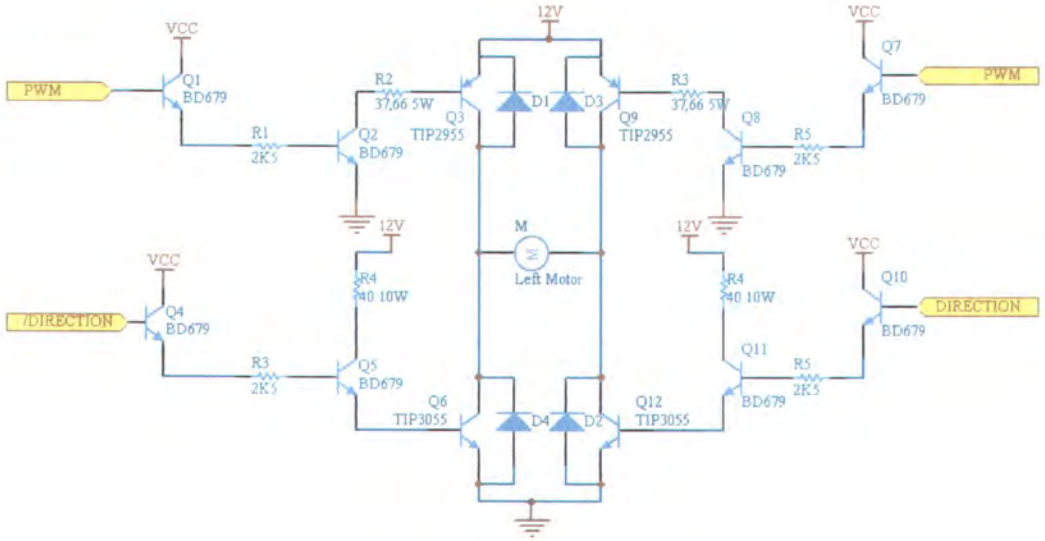


Gambar 3.5. RS232 *Interface*

Gambar 3.5 adalah gambar rangkaian *interface* komunikasi serial RS232. Fungsi dari rangkaian ini adalah mengkonversi level tegangan pada data serial digital TX dan RX menjadi level tegangan standar RS232 oleh IC Max232.

3.3 Perancangan Elektronik

3.3.1 Rangkaian *Receiver* dan Penggerak Motor



Gambar 3.6. Rangkaian penggerak motor

Gambar 3.6 adalah rangkaian penggerak motor dengan nilai pada setiap komponennya dihasilkan dari perhitungan. Sebelum melakukan analisa perhitungan, terlebih dahulu mengetahui karakter dari motor DC yang akan digunakan sebagai penggerak roda.

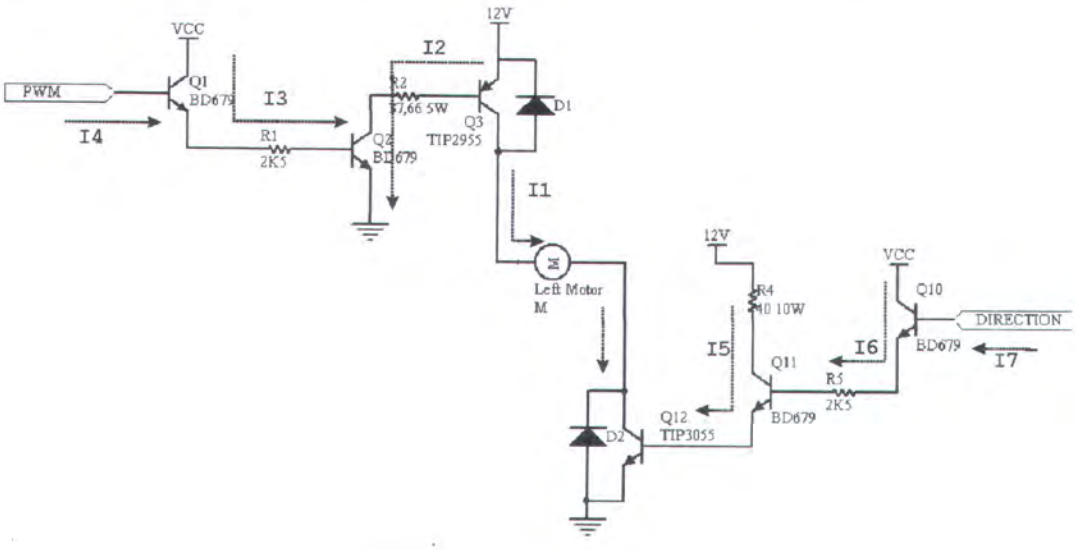
Pada perencanaan ini, motor DC yang digunakan tidak dilengkapi dengan spesifikasi yang jelas, untuk itu perlu dilakukan beberapa pengukuran. Praktisnya motor DC secara mekanik harus dipasang terlebih dahulu pada robot, untuk mensimulasikan torsi yang harus dihasilkan oleh motor untuk menggerakkan robot secara benar.

$$T_M = T_{GB} + T_W + T_R \quad \text{.....(3.2)}$$

Torsi motor harus menanggung beban putar dari gearbox, beban putar pada roda dan beban robot. Saat robot bergerak dengan kecepatan tertentu dengan memberikan Va

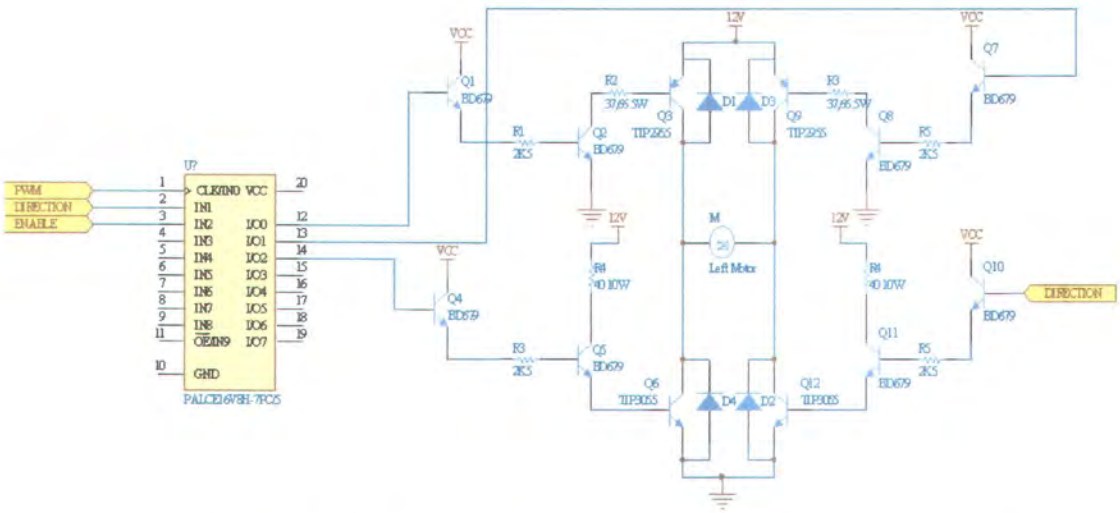
= 12 Volt, arus yang mengalir Ia = 2.5 A (lihat persamaan 2.61 dan gambar 2.17). Ra dari hasil pengukuran adalah 2Ω. Dengan menganggap tegangan jatuh pada kumparan motor La = 0V, maka tegangan balik EMF Vb=12V – (2.5A x 2Ω)= 7V. Dengan menggunakan persamaan-persamaan pada BAB II bagian model motor dan pengukuran torsi motor, akan didapat karakteristik motor secara lengkap, namun dengan acuan arus Ia saja, sudah dapat dilakukan proses perencanaan dan perhitungan rangkaian penggerak motor.

Untuk perencanaan menggunakan asumsi Ia = 3A. Maka transistor yang digunakan untuk penggerak motor (gambar 3.6, Q3,Q9,Q6 dan Q12) harus memilki karakter arus *collector* IC maksimum sama atau lebih besar dari 3A. Dalam perancangan ini digunakan pasangan transistor NPN TIP3055 dan PNP TIP2955. Masukan basis pada transistor Q4 adalah komplemen masukan basis Q10, hal ini untuk memastikan arah arus motor bergerak pada arah yang benar. Perhitungan dapat dilakukan pada satu sisi penggerak saja, sisi penggerak yang lain mempunyai nilai yang sama (gambar 3.7)



Gambar 3.7. Analisa arus

Karena transistor yang digunakan hanya sebagai proses *switching* saja, maka analisa daerah kerja transistor pada saturasi dan *cutoff* saja. Dari hasil perhitungan didapatkan nilai komponen seperti pada gambar 3.6



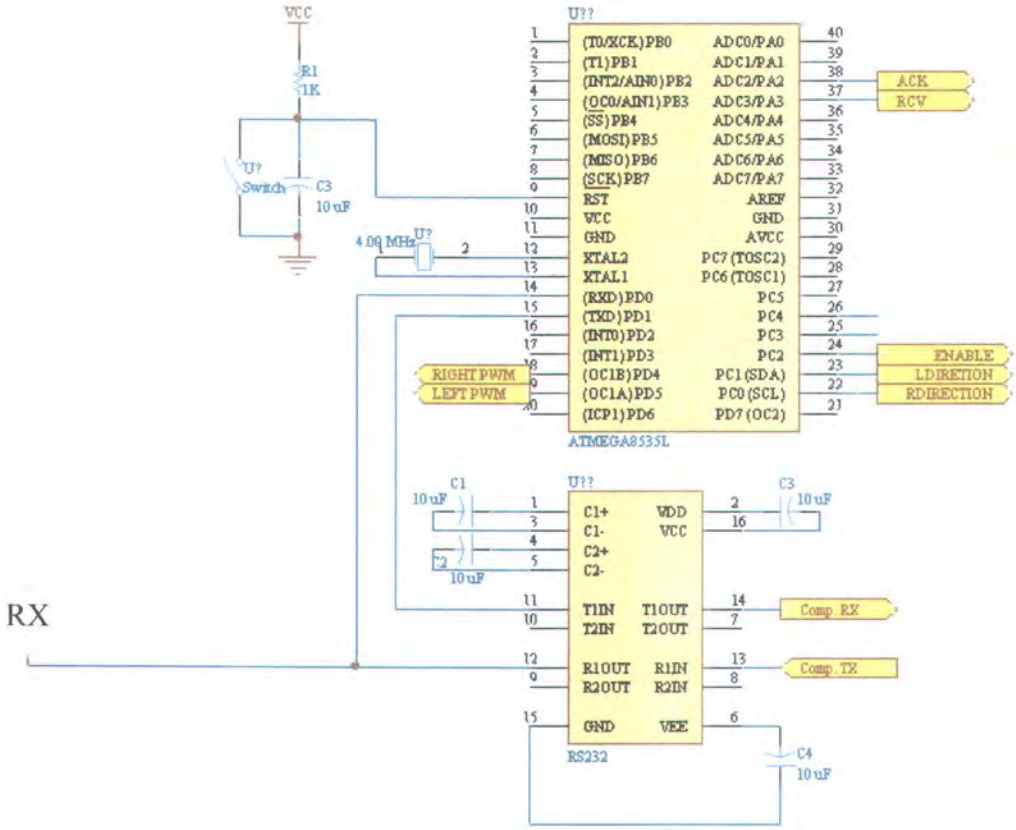
Gambar 3.8. GAL16V8 sebagai *decoder* untuk rangkaian penggerak motor

Gambar 3.8, menggunakan GAL sebagai decoder untuk rangkaian penggerak motor, untuk mengurangi pemakaian gerbang TTL standar, dan untuk mempermudah proses perencanaan. Berikut adalah isi program VHDL decoder rangkaian penggerak motor:

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
entity hbridgec is port(
    pwm : in std_logic;
    dir : in std_logic;
    en : in std_logic;
    y0: out std_logic;
    y1: out std_logic;
    y2: out std_logic);
    attribute pin_numbers of hbridgec: entity is
    "pwm:1 dir:2 en:3 y0:13 y1:12 y2:14";
```

```
end hbridgec;

architecture archhbridgec of hbridgec is
begin
   y2<=en and not (dir);
   y0<= not(dir) and pwm;
   y1<=dir and pwm;
end archhbridgec ;
```



Gambar 3.9. Unit *receiver*

Pada gambar 3.9, data serial RX yang dikirimkan oleh komputer, divalidasi oleh mikrontroler ATMEGA8535, jika data valid sesuai kesepakatan aturan protokol, mikrokontroler kemudian melaksanakan perintah yang telah diterima. Algoritma kontroler adalah sebagai berikut:

```
DO
        if valid(RcvData)=true then
              ACK=1
```

```
            if requestData(RcvData)=true then
                RCV=1
            else
                RCV=0
                MoveRobot(RcvData)
            end if
            wait(ms)
            valid = false
            ACK = 0
            RCV = 0
        else
            ACK=0
            listen
        end if
    LOOP

    Sub MoveRobot(RcvData):
        Time=0
        Delay = Delay(RcvData)
        LeftPWM = LeftPWM(RcvData)
        LDirection = LeftDirection(RcvData)
        RightPWM = RightPWM(RcvData)
        RDirection = RigtDirection(RcvData)
        Repeat
            PD4 = RightPWM
            PD5 = LeftPWM
        Until Time=Delay
    Return
```

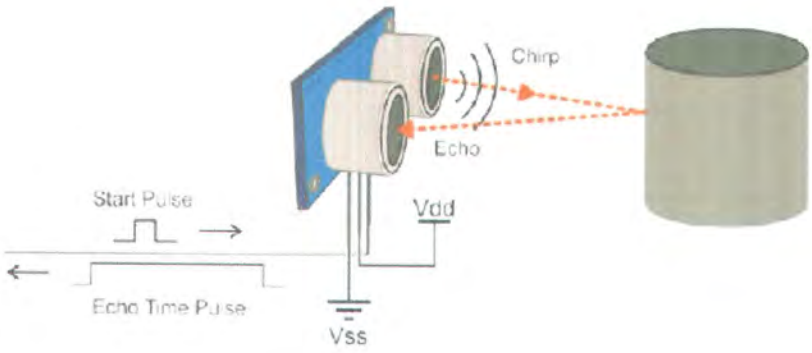
Implementasi algoritma mikrokontroler ATMEGA8535 menggunakan BASCOM AVR ® ver.1.11.

3.3.2 *Transmitter* dan akuisisi data

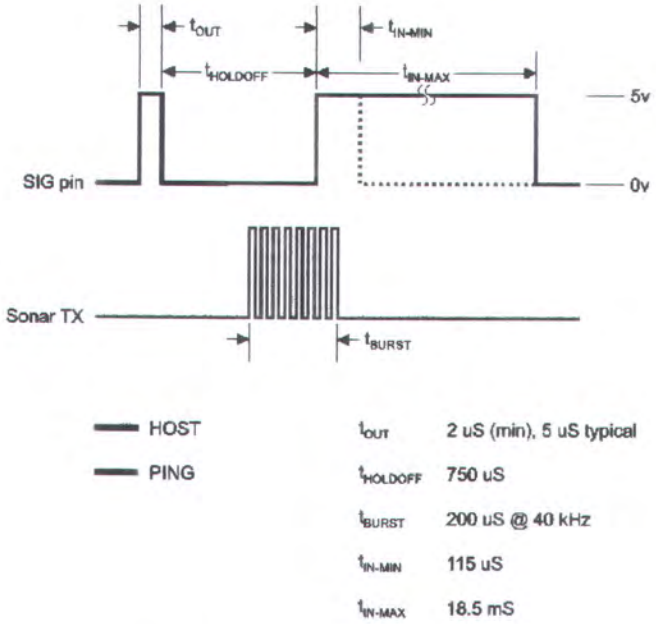
3.3.2.1 Sensor Jarak dengan Ultrasonik

Gambar 3.10. *PING)))* ™Ultrasonic Sensor

Sensor jarak yang digunakan dalam perencanaan ini merupakan modul jadi *PING)))* produk dari Parallax Inc (gambar 3.10). Modul sensor ini mempunyai jangkauan pengukuran jarak 3 cm – 3.3 m. Prinsip kerja dari sensor ini (gambar 3.11) adalah dengan mengirimkan sinyal *start pulse* (pin SIG) modul akan mengirimkan sinyal 40 KHz (sinyal 40KHz ini diistilahkan sebagai *chirp*) ke *tranduser* ultrasonik. Sesaat setelah *start pulse* telah diberikan, modul *PING)))* akan membuat tegangan pada jalur atau pin SIG menjadi logik "*high*", jika terdapat obyek dibawah jangkauan jarak maksimumnya, maka hasil pantulan sinyal *chirp* yang diterima *tranduser ultrasonik* akan membuat pin SIG kembali menjadi logik "*low*", waktu sinyal saat *high* dan kembali menjadi low disebut *echo time pulse*. Lama periode sinyal ini sebanding dengan waktu yang dibutuhkan sinyal chirp saat dipancarkan sampai pantulannya diterima *tranduser ultrasonik*. Gambar 3.12 adalah diagram waktu *PING)))*.



Gambar 3.11. Prinsip kerja *PING))) ™*



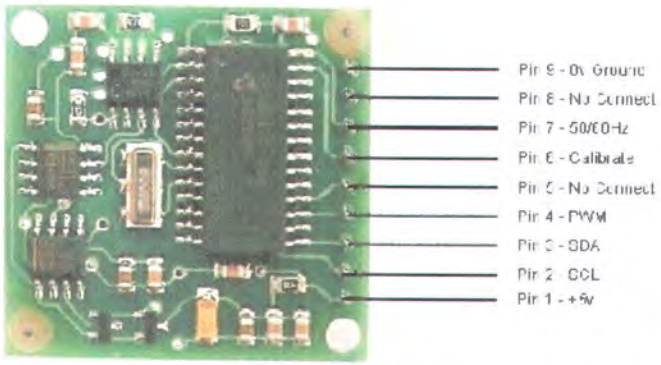
Gambar 3.12. Diagram waktu *PING))) ™*

Kecepatan rambat suara ultrasonik adalah 1 cm /29.034 µs (344,424 m/s). Untuk menghitung periode *sinyal echo*, dibutuhkan *counter* yang digerakkan oleh *pulsa clock* dengan periode sesuai kepresisian pengukuran jarak yang diinginkan. Anggap kepresisian pengukuran jarak adalah 1 mm, maka dibutuhkan sinyal *clock* dengan periode sinyal 2.9034 µs. Jumlah cacahan *counter* ini adalah sebanding dengan 2 kali jarak antara

sensor dengan benda didepannya, sehingga untuk mendapatkan jarak antara sensor dengan benda sesungguhnya adalah dengan membagi 2 jumlah cacahan *counter*. Berikut algoritma mikrokontroler untuk membaca sensor *PING)))* ™

Set FreqClock
Config portX.0 = output
PulseOut (PortX.0, 5 µS) {Start pulse}
Config portX.0 = input
PulseIn (Tick,PortX.0) {Tick = Echo Time Pulse}
TickDistance = Tick/2
{FreqClock.Alpha = faktor kalibrasi}
CmDistance = Tick.FreqClock.Alpha

3.3.2.2 Kompas Elektronik



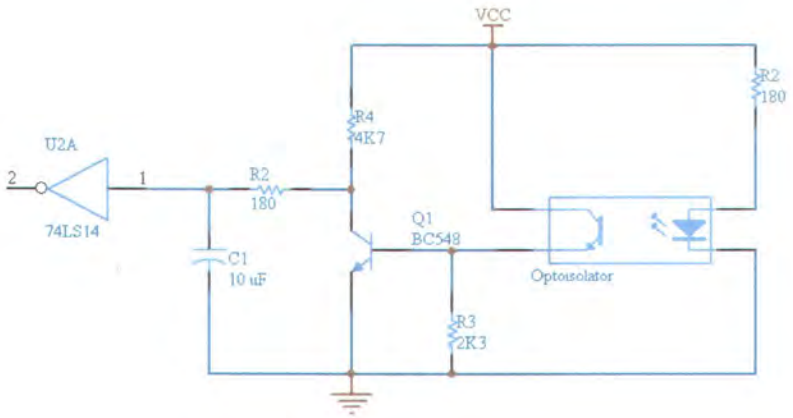
Gambar 3.13. CMPS03, Modul Kompas Elektronik

Kompas sebagai penunjuk pose robot menggunakan modul CMPS03 (gambar 3.13), yang bekerja relatif terhadap arah medan magnet bumi. Pin 4 PWM, adalah sinyal keluaran PWM, dimana periode sinyal pada saat "*HIGH*" adalah merepresentasikan penyimpangan sudut terhadap arah utara bumi. Pulsa positif PWM ini memiliki lebar pulsa antara 1 ms (00, utara) sampai dengan 359.9 ms(359,90). Dengan menggunakan sinyal *clock* 10us untuk menghitung periode sinyal positif PWM akan didapat ketelitian 0.10 . Berikut algoritma sederhana pembacaan sensor CMPS03.

Set FreqClock
Config portX.0 = input

PulseIn (Tick,PortX.0) {Tick = Positif PWM}
Theta = Tick/FreqClock.Alpha
{FreqClock.Alpha = faktor kalibrasi}

3.3.2.3 Incremental Encoder



Gambar 3.14. Rangkaian *Encoder* Odometri

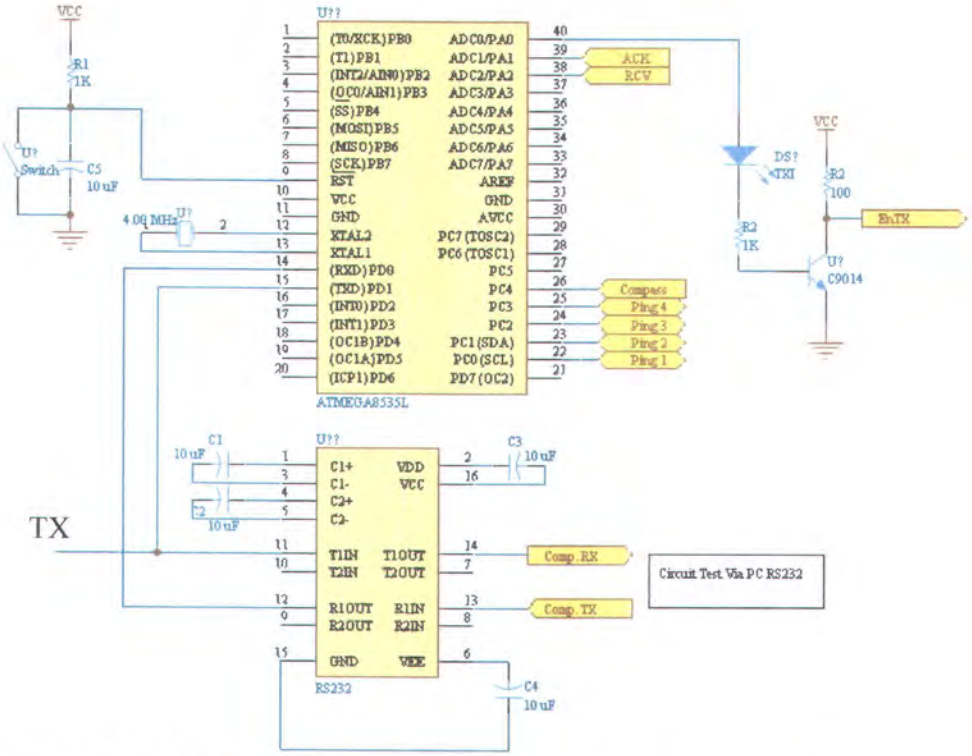
Pada gambar 3.14, antara dioda led dengan basis *phototransistor* pada *optocoupler* terdapat *optical encoder* (gambar 3.1 (c)), pulsa yang dihasilkan untuk membias basis transitor BC818, keluaran pada kaki kolektor membalik pulsa input basis. Keluaran ini kemudian masuk *low pass filter* RC, untuk menghilangkan sinyal *bouncing* pada pembacaan *optical encoder*. Sinyal ini dibalik lagi keluaranya oleh schmit trigger 74LS14. Keluaran sinyal schmit trigger merupakan *pulsa clock* sebagai acuan perhitungan jarak yang dihitung secara program oleh mikrokontroler ATMEGA8535 dengan memanfaatkan fungsi *counter*. Berikut algoritma perhitungan *encoder* dengan menggunakan fungsi *counter*

```
config Timer_X = Counter, Edge=Rising
if IsStartCount = true then
        Start Counter_X
        Wait IsStopCount = True
```

Stop Counter_X
Encoder = Counter_X
end if

3.3.2.4 Transmitter

Gambar 3.15 adalah rangkaian mikrokontroler ATMEGA8535L sebagai unit pembaca data sensor dan unit kontrol *transmitter*. Mikrokontroler akan membaca sensor jika pada unit kontrol atau receiver memberi tanda pada ACK=1, artinya unit *receiver* telah menerima *command* yang valid dan segera melaksanakan *command* tersebut, jika *command* tersebut adalah *command* untuk navigasi maka tanda pada RCV=0, dengan demikian unit pembaca sensor sudah memulai pembacaan *pulsa encoder*. Pembacaan sensor ultrasonik dan kompas menunggu sinyal ACK =0, yang artinya robot telah bergerak dan sekarang pada posisi berhenti. Pada posisi berhenti inilah *encoder* telah mencacah jarak perjalanan robot, dan sensor ultrasonik dan kompas memulai bekerja. Jadi data kompas dan ultrasonik didapat saat robot pada kondisi berhenti.



Gambar 3.15. Unit kontrol untuk pembacaan sensor dan *transmitter*

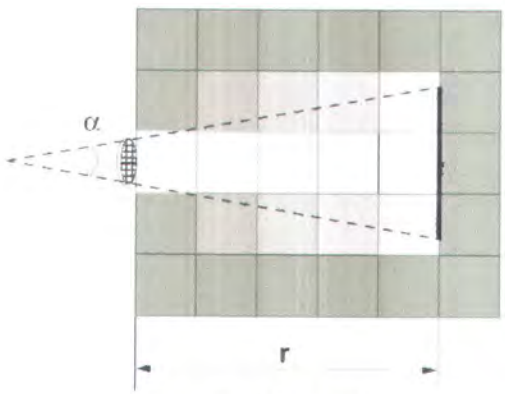
Setelah unit kontrol pembacaan sensor dan *transmitter* telah mendapatkan data, data di simpan. Selama ini *transmitter* TLP434A masih dalam keadaan tidak aktif, *disable*, EnTX = "0". Data yang disimpan tersebut akan ditransmisikan melalui *transmitter* TLP434A, jika *command* adalah merupakan *command* untuk permintaan data, dengan tanda RCV=1. Berikut algoritma dari unit kontrol pembacaan sensor dan *transmitter*,

```
const isDataRequest = 1
do
        wait ACK=1
        if RCV=isDataRequest then
                Enable Transmiter
                Send DATA[]
                waitms
                Disable Transmiter
        else
```

```
            Start Odometri
            wait ACK=0
            Stop Odometri,DATA[]
            GetDataSensor DATA[]
        end if
loop
```

3.4 Algoritma Probabilitas *Occupancy Grids Map*

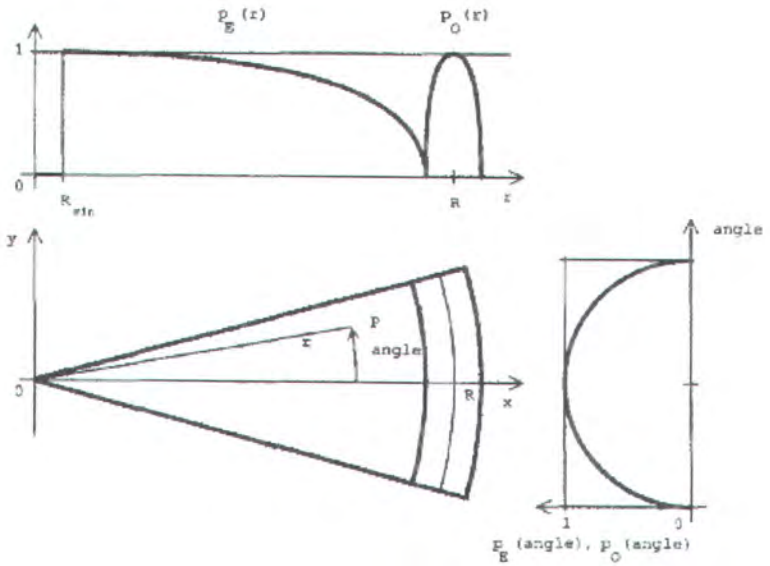
Pada *occupancy grids map*, peta dibagi menjadi *grid* Ci,j (dengan koordinat pusat *grid* [xi,yi]) dengan ukuran *grid* yang seragam mxm. Seperti yang dijelaskan pada bab terdahulu, bahwa nilai probabilitas *occupancy* dari suatu *cell* P(OCCx,y) ditentukan dari model sensor jarak. Model sensor jarak didasarkan pada konsep *field of view*, FOV gambar 3.16. Dimana FOV mempunyai 2 parameter yaitu jarak maksimum sensor r dan sudut persepsi sensor α. Sebagai contoh, dengan menggunakan model *PING)))™* berdasarkan *datasheet* α=200 dengan jarak maksimum r = 3 m.



Gambar 3.16. FOV

Selanjutnya perhitungan probabilitas *occupancy* pada setiap *grid cell* dalam area FOV, berdasarkan 2 parameter (r dan α) dapat digunakan pendekatan linear, atau gaussian (gambar 2.9 dan gambar 2.10). Dalam perancangan ini menggunakan

pendekatan fungsi kuadrat dimana secara grafis menyerupai bentuk grafik distribusi normal atau gaussian (Gambar 3.17).



Gambar 3.17. Perancangan model sensor

Keterangan gambar 3.17:

- PE(r)=Probabilitas Empty
- PO(r)=Probabilitas *Occupancy*
- P = (x,y,α) merupakan titik yang berada pada daerah FOV
- R = Hasil pembacaan sensor jarak
- S = Posisi sensor/robot (xs,ys,αs)
- δ = Jarak dari P ke S
- ε = Deviasi error sensor jarak
- α = Sudut simpangan titik P dilihat dari S
- Ω = Sudut simpangan sensor (spesifikasi sensor)

Probabilitas kondisional *occupancy cell* x,y jika diberikan fakta hasil pembacaan sensor jarak R $P(OCC_{x,y}|R^T)$, dimana T adalah data sensor jarak ke T, tergantung dari dua parameter R dan α,

$$P(OCC_{x,y}|R^T) = O_R(\delta)O_\alpha(\alpha) \quad \text{.....(3.3)}$$

$$O_R(\delta) = \begin{cases} 1-\left(\dfrac{\delta - R}{\varepsilon}\right)^2, \delta \in [R-\varepsilon, R+\varepsilon] \\ 0, \delta < R-\varepsilon \end{cases} \quad \text{.....(3.4)}$$

$$O_\alpha(\alpha) = \begin{cases} 1-\left(\dfrac{2\alpha}{\Omega}\right)^2, \alpha \in \left[-\dfrac{\Omega}{2}, \dfrac{\Omega}{2}\right] \\ 0, \alpha < -\dfrac{\Omega}{2} atau \alpha > \dfrac{\Omega}{2} \end{cases} \quad \text{.....(3.5)}$$

Jika nilai total probabilitas *cell* x,y dari t kali pembacaan sensor adalah $P(OCC_{x,y}|R^{(t)})$, dengan t = 1,2,...T, menjadi $P(OCC_{x,y}|R^1, R^2, ..., R^T)$. Dari aksioma probabilitas seperti dijelaskan pada dasar teori probabilitas (BAB 2), maka berikut dapat diturunkan persamaan untuk proses mengupdate probabilitas dari setiap *cell* dan dari t pembacaan sensor jarak.

$$P(OCC_{x,y}|R^1, ..., R^T) = 1 - P(\neg OCC_{x,y}|R^1, ..., R^T) \quad \text{.....(3.6)}$$

Dari aturan probabilitas kondisional,

$$P(OCC_{x,y}|R^1, ..., R^T) = P(R^T|OCC_{x,y}, R^1, ..., R^{T-1}).P(OCC_{x,y}|R^1, ..., R^{T-1}) \quad \text{.....(3.7)}$$

$$P(\neg OCC_{x,y}|R^1, ..., R^T) = P(R^T|\neg OCC_{x,y}, R^1, ..., R^{T-1}).P(\neg OCC_{x,y}|R^1, ..., R^{T-1}) \quad \text{.....(3.8)}$$

dengan membagi persamaan 3.7 dengan persamaan 3.8, didapat

$$= \frac{P(R^T \mid OCC_{x,y}, R^1,..., R^{T-1}).P(OCC_{x,y} \mid R^1,..., R^{T-1})}{P(R^T \mid \neg OCC_{x,y}, R^1,..., R^{T-1}).P(\neg OCC_{x,y} \mid R^1,..., R^{T-1})}$$

$$= \frac{P(R^T \mid OCC_{x,y})P(R^1),..., P(R^{T-1}).P(OCC_{x,y} \mid R^1,..., R^{T-1})}{P(R^T \mid \neg OCC_{x,y})P(R^1),..., P(R^{T-1}).P(\neg OCC_{x,y} \mid R^1,..., R^{T-1})}$$

$$= \frac{P(R^T \mid OCC_{x,y})P(OCC_{x,y} \mid R^1,..., R^{T-1})}{P(R^T \mid \neg OCC_{x,y})P(\neg OCC_{x,y} \mid R^1,..., R^{T-1})} \quad \text{.....(3.9)}$$

$$P(R^T \mid OCC_{x,y}) = \frac{P(OCC_{x,y} \mid R^T)P(R^T)}{P(OCC_{x,y})} \quad \text{.....(3.10)}$$

$$P(R^T \mid \neg OCC_{x,y}) = \frac{P(\neg OCC_{x,y} \mid R^T)P(R^T)}{P(\neg OCC_{x,y})} \quad \text{.....(3.11)}$$

Persamaan 3.11 dan 3.10 substistusi ke persamaan 3.9

$$= \frac{\frac{P(OCC_{x,y} \mid R^T)P(R^T)}{P(OCC_{x,y})} P(OCC_{x,y} \mid R^1,..., R^{T-1})}{\frac{P(\neg OCC_{x,y} \mid R^T)P(R^T)}{P(\neg OCC_{x,y})} P(\neg OCC_{x,y} \mid R^1,..., R^{T-1})}$$

$$= \frac{P(OCC_{x,y} \mid R^T)P(\neg OCC_{x,y})P(OCC_{x,y} \mid R^1,..., R^{T-1})}{P(\neg OCC_{x,y} \mid R^T)P(OCC_{x,y})P(\neg OCC_{x,y} \mid R^1,..., R^{T-1})} \quad \text{.....(3.12)}$$

Jika, persamaan 3.7 dibagi dengan persamaan 3.8 adalah

$$\frac{P(OCC_{x,y} \mid R^1,..., R^T)}{P(\neg OCC_{x,y} \mid R^1,..., R^T)} = A \quad \text{.....(3.13)}$$

dan substitusi persamaan 3.13 ke persamaan 3.6

$$P(OCC_{x,y} \mid R^1,..., R^T) = 1 - \frac{P(OCC_{x,y} \mid R^1,..., R^T)}{A} \quad \text{.....(3.14)}$$

$$\frac{P(OCC_{x,y}|R^1,...,R^T)(A+1)}{A}=1$$

$$P(OCC_{x,y}|R^1,...,R^T)=\frac{A}{1+A} \quad \text{.....(3.15)}$$

Substitusi persamaan 3.15 ke persamaan 3.14 (bentuk sebelah kanan persamaan), maka

$$P(OCC_{x,y}|R^1,...,R^T)=1-\frac{1}{1+A} \quad \text{.....(3.16)}$$

dengan mensubstitusi A dengan persamaan 3.12,

$$P(OCC_{x,y}|R^1,...,R^T)=1-\frac{1}{1+\dfrac{P(OCC_{x,y}|R^T)P(\neg OCC_{x,y})P(OCC_{x,y}|R^1,..,R^{T-1})}{P(\neg OCC_{x,y}|R^T)P(OCC_{x,y})P(\neg OCC_{x,y}|R^1,..,R^{T-1})}}$$

$$=1-\frac{1}{1+\dfrac{\left[P(OCC_{x,y}|R^T)\right]\left[1-P(OCC_{x,y})\right]\left[P(OCC_{x,y}|R^1,..,R^{T-1})\right]}{\left[1-P(OCC_{x,y}|R^T)\right]\left[P(OCC_{x,y})\right]\left[1-P(OCC_{x,y}|R^1,..,R^{T-1})\right]}}$$

$$=1-\left(1+\frac{\left[P(OCC_{x,y}|R^T)\right]}{\left[1-P(OCC_{x,y}|R^T)\right]}\frac{\left[1-P(OCC_{x,y})\right]}{\left[P(OCC_{x,y})\right]}\frac{\left[P(OCC_{x,y}|R^1,..,R^{T-1})\right]}{\left[1-P(OCC_{x,y}|R^1,..,R^{T-1})\right]}\right)^{-1} \quad \text{.....(3.17)}$$

Jika keadaan awal untuk semua *cell* pada *workspace* adalah memiliki nilai probabilitas 0.5 (belum tereksplorasi), maka

$$\frac{\left[1-P(OCC_{x,y})\right]}{\left[P(OCC_{x,y})\right]}=1 \quad \text{.....(3.18)}$$

Substitusi persamaan 3.18 ke persamaan 3.17

$$=1-\left(1+\frac{\left[P(OCC_{x,y}|R^T)\right]}{\left[1-P(OCC_{x,y}|R^T)\right]}\frac{\left[P(OCC_{x,y}|R^1,..,R^{T-1})\right]}{\left[1-P(OCC_{x,y}|R^1,..,R^{T-1})\right]}\right)^{-1} \quad \text{.....(3.19)}$$

dari persamaan 3.19, dapat ditulis kembali dalam bentuk k (pergerakan robot dari 0,1...,T) sebagai berikut:

$$P(OCC_{x,y}|R)[k] = 1 - \left(1 + \frac{[P(OCC_{x,y}|R)[k]]}{[1 - P(OCC_{x,y}|R)[k]]} \frac{[P(OCC_{x,y}|R)[k-1]]}{[1 - P(OCC_{x,y}|R)[k-1]]}\right)^{-1} \quad \text{.....(3.20)}$$

Dari apa yang telah diuraikan diatas dapat dibuat algoritma untuk probabilitas *occupancy grids map*, sebagai berikut:

All Cell $P(OCC_{x,y})[] = 0.5$ *'Initialitation*
k=0
$P(OCC_{x,y}|R)[0] = 0.5$
'Exploration
While not StopCriteria
inc k
GetSensorData '2 Encoder, Compass, 4 Range sensor
SetPosition 'Inverse Kinematic Model
$P(OCC_{x,y}|R)[k] = O_R(\delta)O_\alpha(\alpha)$ *'Sensor Perception*
'Update

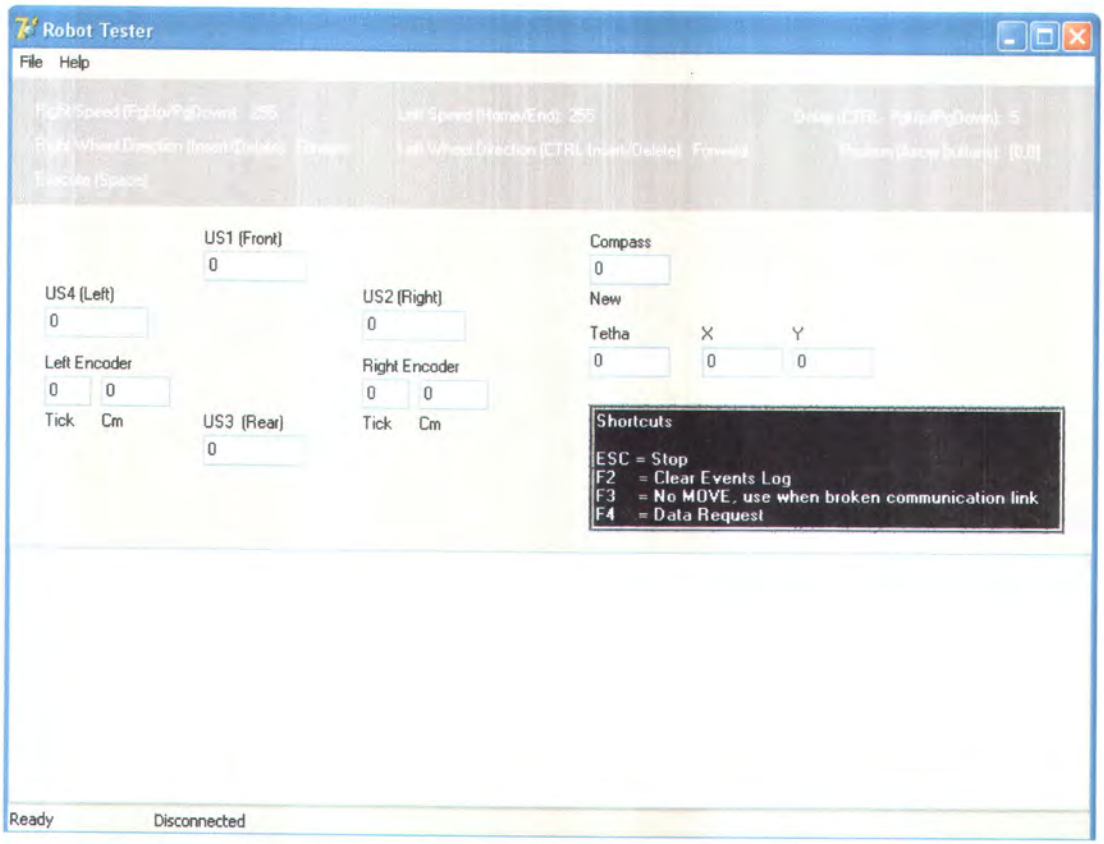
$$P(OCC_{x,y}|R)[k] = 1 - \left(1 + \frac{[P(OCC_{x,y}|R)[k]]}{[1 - P(OCC_{x,y}|R)[k]]} \frac{[P(OCC_{x,y}|R)[k-1]]}{[1 - P(OCC_{x,y}|R)[k-1]]}\right)^{-1}$$

'Mapping & Localization
SetLocalMAP
GlobalMAP= GlobalMAP + SetLocalMAP
'Path Planning
$V_{x,y} = \underset{\substack{\gamma=-1,0,1 \\ \lambda=-1,0,1}}{Min} \{V_{x+\gamma,y+\lambda} + P[OCC_{(x+\gamma,y+\lambda)}]$
Direction [Forward,Right,Left]= Dir (Vx,y)
MoveRobot Direction
loop

BAB IV
PENGUJIAN DAN ANALISA

4.1 Pergerakan Robot dan Komunikasi

Untuk mendapatkan karakteristik pembacaan sensor, pergerakan robot, dan komunikasi robot maka perlu dibuat program untuk menguji hal-hal tersebut. Pada gambar 4.1 merupakan gambar GUI dari program robot tester.



Gambar 4.1. GUI program robot tester

Pergerakan robot dalam penelitian ini dibatasi hanya dalam 3 model pergerakan: maju, pivot kanan dan maju, pivot kiri dan maju. Memang dengan hanya dibatasi 3 model

pergerakan tersebut akan menyebabkan keterbatasan gerak robot, alasan dipilihnya 3 model pergerakan tersebut adalah untuk meminimalisasi kesalahan gerak yang tidak dapat diprediksi.

Dengan menggunakan program *robot tester*, dan metode "*trial and error*" untuk mencari nilai variabel-variabel pergerakan yang mendekati hasil manuver : gerak maju, pivot kanan, dan pivot kiri.

Pada gambar 4.2, menunjukkan pengiriman paket data ke robot. Berurut dari kiri ke kanan,

- 02 : *Frame Start*
- 03: Byte Ctrl, bit-0 (RDir) dan bit–1 (LDir) bernilai 1, bit-3 (cmd) bernilai 0, artinya roda kanan dan roda kiri bergerak maju, serta komputer sedang memerintahkan robot untuk bergerak ditandai dengan cmd=0
- FF : Roda kiri berputar dengan kecepatan penuh (255)
- FF: Roda kanan berputar dengan kecepatan penuh (255)
- 05: *Delay*, jumlah iterasi dalam program mikrokontroller untuk menggerakkan kedua roda.
- 47 dan EC : adalah hasil perhitungan CRC 16 bit dari 02 03 FF FF 05
- 03: *Frame Stop*

Gambar 4.2. GUI pengiriman paket data

Dari hasil *trial and error*, nilai dari variabel-variabel untuk menggerakkan robot maju sejauh 30 cm adalah (gambar 4.3)

- E4 : Roda kiri bergerak maju dengan kecepatan putar = 228
- 73 : Roda kanan bergerak maju dengan kecepatan putar = 115
- 0D : *Delay* atau jumlah iterasi = 13

```
Moving forward...
Sending: 02 03 E4 73 0D 46 F9 03
```

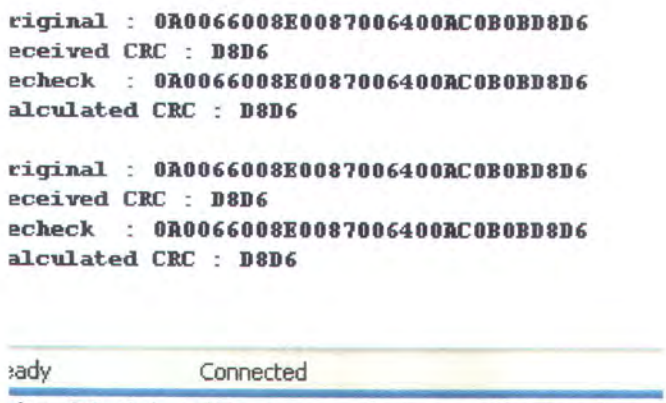
Gambar 4.3. Paket data untuk gerak maju 30 cm

Setelah robot bergerak maju sejauh 30 cm, maka saatnya komputer untuk meminta data pembacaan sensor, dengan membuat bit-3 = 1 pada Ctrl. Contoh perintah untuk meminta data dari robot seperti pada gambar 4.4

```
Data request
Sending: 02 07 00 00 00 B4 5C 03
```

Gambar 4.4. Perintah permintaan data

Selanjutnya robot akan merespon dengan mengirimkan data-data sensor , selama waktu penantian sampainya data yang dikirim oleh robot ke komputer, unit *transmiter* pada komputer di *disable*, untuk menghindari terjadinya *cross talk*. Setelah data diterima, program didalam komputer akan memvalidasi paket data yang datang dengan memeriksa nilai *Frame Start*, *Frame End* dan 2 byte CRC. Gambar 4.5 adalah contoh paket data yang diterima oleh komputer, setelah robot melakukan pergerakan maju 30 cm.



Gambar 4.5. Paket data sensor robot yang diterima komputer

Pada saat hasil validasi menghasilkan bahwa paket data yang diterima oleh komputer adalah data yang valid, program akan menampilkan bahwa nilai CRC yang diterima adalah sama dengan nilai CRC dari hasil hitungan (gambar 4.5). Kemudian program menterjemahkan deretan data pda paket data tersebut menjadi informasi nilai-nilai yang dihasilkan oleh sensor robot. Seperti pada gambar 4.6, dapat dijabarkan sebagai berikut:

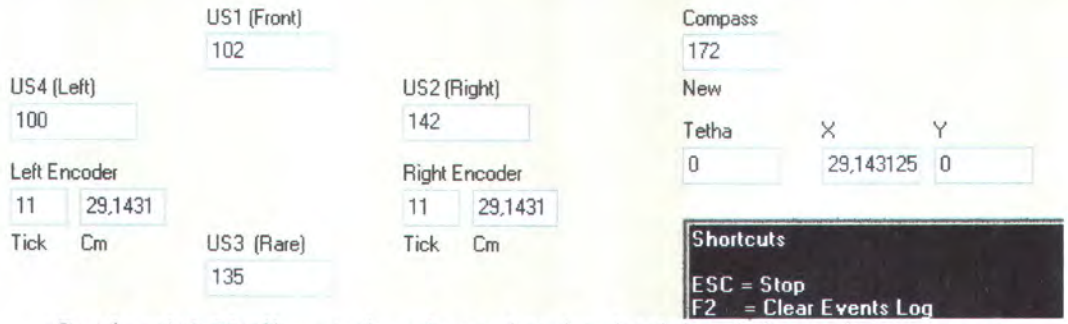
- US1(Front) = 102. Didepan robot pada jarak 102 cm terdapat *obstacle*.
- US2 (Right) = 142, US3 (Rare) =135, US4(Left) = 100. Nilai nilai tersebut menunjukkan jarak *obstacle* dilihat dari sudut pandang masing-masing sensor
- *Left Encoder* (*Tick*) = 11. Nilai *tick* dikalikan dengan faktor pengali 2.649375 menghasilkan nilai cm = 29.1431 cm. Faktor pengali 2.649375 didapat dari keliling roda dibagi dengan jumlah lubang pada *encoder* = 16. Keliling roda = π X Diameter roda = π X 13.5= 42.39 cm
- *Right Encoder* (*Tick*)=11
- Tetha = 0. Nilai tersebut dihasilkan dari fungsi kinematik, dengan contoh cuplikan program sebagai berikut

Tetha := (180)((FRightSign* DistanceR - FLeftSign * DistanceL)/RobotWidth)*

- X = 29.1431. Gerak translasi pada sumbu X adalah 29.1431 cm, mendekati 30 cm.
- Y =0. Gerak translasi pada sumbu Y adalah 0 cm. Hasil gerak translasi pada sumbu X dan sumbu Y juga dengan menggunakan rumus kinematik, dengan cuplikan program sebagai berikut

```
X:=(( DistanceR+DistanceL)/2)*Cos((pi/180)*(Tetha));
Y:=(( DistanceR+DistanceL)/2)*sin((pi/180)*(Tetha));
```

- Kompas = 172. Pose robot 172^0 berlawanan arah jarum jam dari arah utara.



Gambar 4.6. Hasil penterjemahan paket data ke dalam informasi data sensor

Berikut adalah rangkuman perintah robot:

- ```
  Moving forward...
  Sending: 02 03 E4 73 0D 46 F9 03
  ```
- ```
  Moving pivot right...
  Sending: 02 02 FF FF 09 BE ED 03
  ```
- ```
  Moving pivot left...
  Sending: 02 01 FF 78 0B CB 0F 03
  ```
- ```
  Data request
  Sending: 02 07 00 00 00 B4 5C 03
  ```

Sistem kontrol pergerakan pada robot adalah *open loop*, dengan kata lain setiap kali robot diperintahkan untuk bergerak, didalam sistem internal kontroler pada robot tidak terdapat umpan balik yang akan mengoreksi kesalahan gerak tersebut. Jadi, asumsi robot dapat bergerak sesuai dengan perintah yang diberikan. Salah atau benar pergerakan robot, dan berdasarkan informasi pembacaan sensor, algoritma navigasi yang akan menghitung kemana pergerakan robot selanjutnya. Kekonsistenan transmisi data dengan menggunakan saluran kabel yang panjang memegang peranan penting dalam proses navigasi ini.

4.2 Simulasi Software

Program simulasi dibuat untuk mensimulasikan dengan pendekatan pada keadaan sebenarnya dari sistem penjelajahan ini, terdapat beberapa ketentuan dalam membuat program simulasi ini yaitu:

- Pergerakan robot. Dalam mensimulasikan pergerakan robot, pergerakan pada masing-masing roda didekati dengan menggunakan fungsi *random gaussian*. Dimana fungsi *random gaussian* memerlukan 2 masukan, yaitu *mean* dan deviasi. *Mean* artinya nilai tengah dimana fungsi *gaussian* sama dengan 1 dan deviasi bisa juga disebut konstanta *covariance* (τ) pada fungsi *gaussian*.

```
rWheel:=Round(RandG(30,1));
lWheel:=Round(RandG(30,1));
Kinematic(rWheel,lWheel);
```

rWheel dan *lWheel* pada cuplikan program diatas mewakili gerak translasi roda kanan dan roda kiri. Karena data translasi roda robot diperoleh dari jumlah cacahan *encoder* maka untuk merepresentasikan keadaan yang sebenarnya digunakan fungsi *Round( )* untuk membuat hasi fungsi *random* menjadi bentuk

integer.

- Pose robot. Sudut hadap robot atau pose robot diperoleh dengan hasil hitungan persamaan kinematik dengan masukan nilai jarak translasi roda kanan dan roda kiri. Pergerakan robot atau navigasi robot hanya dalam 3 arah pergerakan, yaitu: gerak maju, pivot kekanan, pivot kekiri. Pada gerak maju robot disimulasikan bergerak sejauh 30 cm atau selebar badan robot. Gerak pivot ke-kanan, roda kanan bergerak mundur dan roda kiri bergerak maju dengan jarak translasi ¼ lebar badan robot. Gerak pivot ke-kiri , roda kiri bergerak mundur dan roda kanan bergerak maju dengan jarak translasi ¼ lebar badan robot. Berikut adalah cuplikan kode program untuk jenis manuver robot.

```
procedure Kinematic(Const rWheel,lWheel:integer);
begin
        dTetha:=180*(rWheel-lWheel)/ROBOT_DIAMETER;
        dX:=((rWheel+lWheel)/ROBOT_DIAMETER)*
                Cos((pi/180)*(angle+dTetha));
        dY:=((rWheel+lWheel)/ROBOT_DIAMETER)*
                sin((pi/180)*(angle+dTetha));
end;
procedure goForward;
begin
        rWheel:=Round(RandG(30,1));
```

```
        lWheel:=Round(RandG(30,1));
        Kinematic(rWheel,lWheel);
 end;
procedure pivotRight;
begin
        rWheel:= Round(RandG(7.5,1));
        rWheel:=-rWheel;
        lWheel:= Round(RandG(7.5,1));
        Kinematic(rWheel,lWheel);
end;
procedure pivotLeft;
begin
        rWheel:= Round(RandG(7.5,1));
        lWheel:= Round(RandG(7.5,1));
        lWheel:=-lWheel;
        Kinematic(rWheel,lWheel);
end;
```

- *Workspace* dan *obstacle* hanya merupakan abstraksi untuk visualisasi saja. Saat awal abstraksi tersebut tidaklah diketahui robot, robot hanya melakukan proses *updating occupancy* saja berdasarkan pembacaan sensor-sensor-nya, dari hasil proses penjelajahan dan navigasi robot, robot akan memilki persepsi sendiri terhadap *workspace* dan *obstacle* yang terdeteksi. Persepsi ini akan terlihat sebagai bentuk peta dan *history* pergerakan robot.

Pengujian software dilakukan dengan membuat file *MAP* dengan meletakkan partisi simulasi dinding dengan ukuran sama dengan 1 *grid cell* pada koordinat X,Y. Secara *default* ukuran *workspace* 140 x 140 *grid* dengan ukuran *grid* 30 cm x 30 cm. Hal-hal yang perlu diperhatikan dalam simulasi proses penjelajahan dan navigasi robot ini adalah :

- Persentase daerah yang tidak dapat dieksplorasi robot terhadap abstraksi *workspace* atau peta yang disediakan. Jika persentase daerah yang belum tereksplorasi adalah 0% maka robot mempunyai persepsi peta yang sama dengan abtraksi peta yang diberikan.

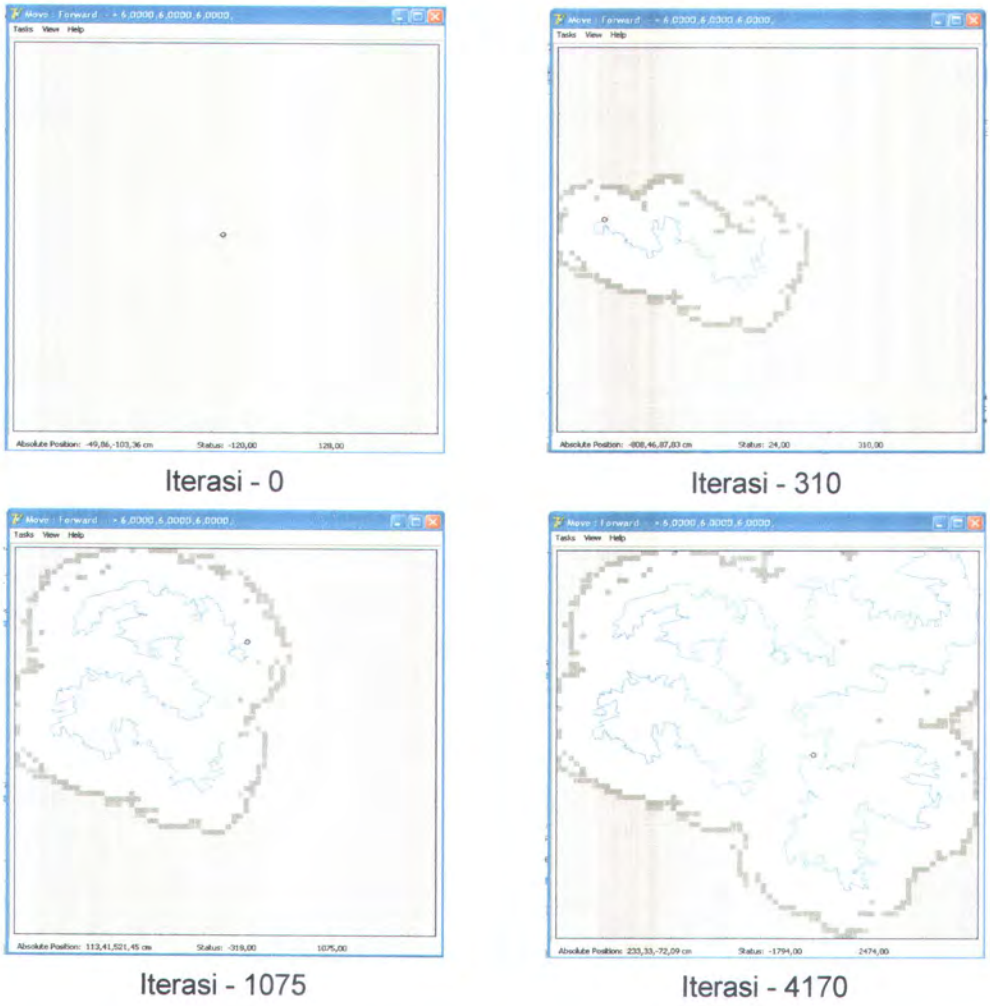
- Waktu iterasi. Jumlah waktu iterasi menunjukkan efektifitas navigasi robot dalam proses eksplorasinya.
- Pengujian terhadap suatu *workspace* atau peta perlu dilakukan lebih dari 1 kali, karena keputusan navigasi robot dapat berbeda-beda setiap proses *running*. Hal ini dapat disebabkan oleh beberapa faktor yaitu:
  - Proses pembulatan nilai *occupancy*, berakibat pada proses pemilihan arah navigasi berdasarkan nilai *occupancy* pada setiap *view* sensor.
  - Terdapat 2 jenis sistem koordinat yaitu: koordinat absolut dan koordinat *grid cell*. Koordinat *absolut*, merupakan koordinat yang dihasilkan oleh pergerakan robot dalam bentuk satuan panjang cm. Sedangkan koordinat *grid cell* adalah sistem koordinat yang digunakan untuk proses perhitungan *occupancy*, jadi terdapat proses pemetaan dan translasi dari koordinat *absolut* ke koordinat *grid cell* dan begitu juga sebaliknya, oleh karena itu terdapat juga faktor pembulatan angka untuk proses pemetaan dan translasi ini.

Pada intinya program simulasi merupakan gambaran *real* penggunaaan metode *occupancy grids map* pada robot yang sebenarnya untuk mengekplorasi dan menavigasi pada daerah yang tidak dikenali. Pergerakan robot yang sebenarnya mengalami *noise* pergerakan, oleh karena itu ada kemungkinan pada saat robot bergerak dekat dengan *obstacle* robot akan mengalami tabrakan dengan *obstacle* padahal pada saat itu algoritma telah memerintahkan robot untuk bergerak menghindar. Proses tabrakan dan mereposisi robot setelah terjadinya tabrakan sangatlah sulit untuk dimodelkan, sehingga pada program simulasi tidak menggunakan model tabrakan robot dengan *obstacle*. Untuk

menghindari tabrakan pada simulasi dapat dilakukan dengan memperkecil *step* pergerakan. Jika pada kondisi aslinya robot bergerak dengan step-30 cm, atau ukuran 1 *grid* dalam peta simulasi, maka pergerakan robot dalam simulasi dibuat lebih kecil dari 30 cm. Pada pengujian, simulasi menggunakan *step* pergerakan 10 cm atau 1/3 dari badan robot atau 1/3 dari ukuran *grid.*

4.2.1 Pengujian 1

Gambar 4.7, adalah *workspace* yang digunakan untuk pengujian 1. Warna abu-abu 50% mewakili daerah yang belum terekplorasi, robot selalu pada posisi tengah (hanya abstraksi saja: robot tidak tahu kalau posisinya berada ditengah *workspace*). Luas *workspace* 140x140 ((140 * 30cm) * (140*30cm)) atau 609.6 m^2 dan tanpa *obstacle* atau dinding didalam *workspace*.



Gambar 4.7. Pengujian simulasi *software* – 1

Tabel 4.1 Data Pengujian simulasi *software* -1

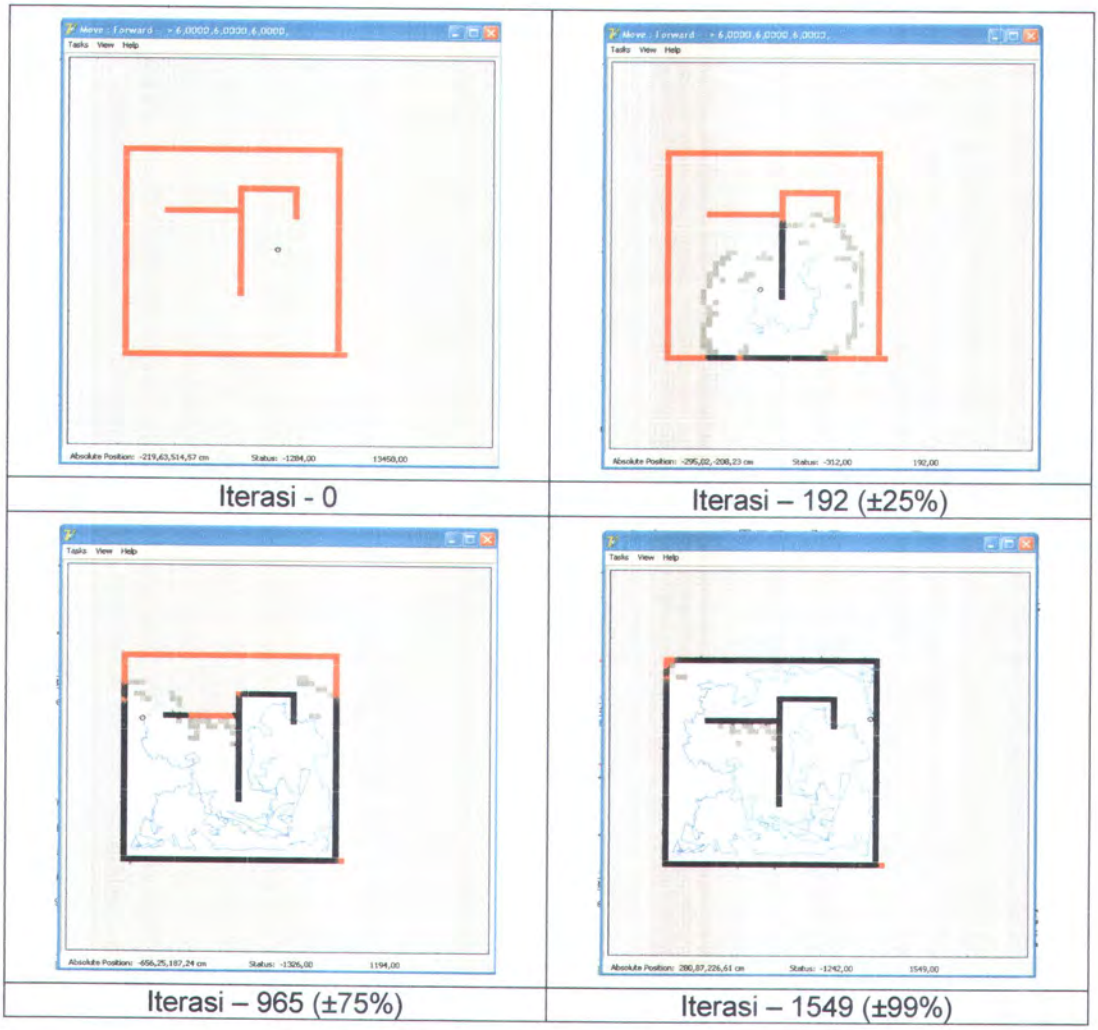
Luas Wilayah (m^2)	Jumlah Ruangan dan Koridor	Jumlah Iterasi	Persentase Wilyah yang Tereksplorasi	Perbandingan Kenaikan % Luas Wilayah dan Jumlah Iterasi
609,6	1	310	20%	0.39329
609,6	1	1075	40%	0.159
609,6	1	4170	80%	0.078

Dari tabel 4.1 dapat diketahui bahwa terjadi peningkatan jumlah iterasi yang cukup tinggi saat eksplorasi 40% wilayah menjadi ekplorasi 80% wilayah yaitu 3095 iterasi. Apabila dihitung dari nilai perbandingan antara kenaikan persentase wilayah yang

terekplorasi dengan kenaikan jumlah iterasi, maka terjadi penurunan nilai perbandingan. Hal ini menunjukkan penurunan efektifitas navigasi saat wilayah ekplorasi telah mencapai 20%.

4.2.2 Pengujian 2

Pengujian 2, menggunakan abtraksi peta mengadopsi dari peta pada kontes robot devisi expert 2005. Blok warna merah adalah abstraksi peta atau *workspace* robot. Ukuran peta 36 * 36 *grid cell* (116.64 m^2).



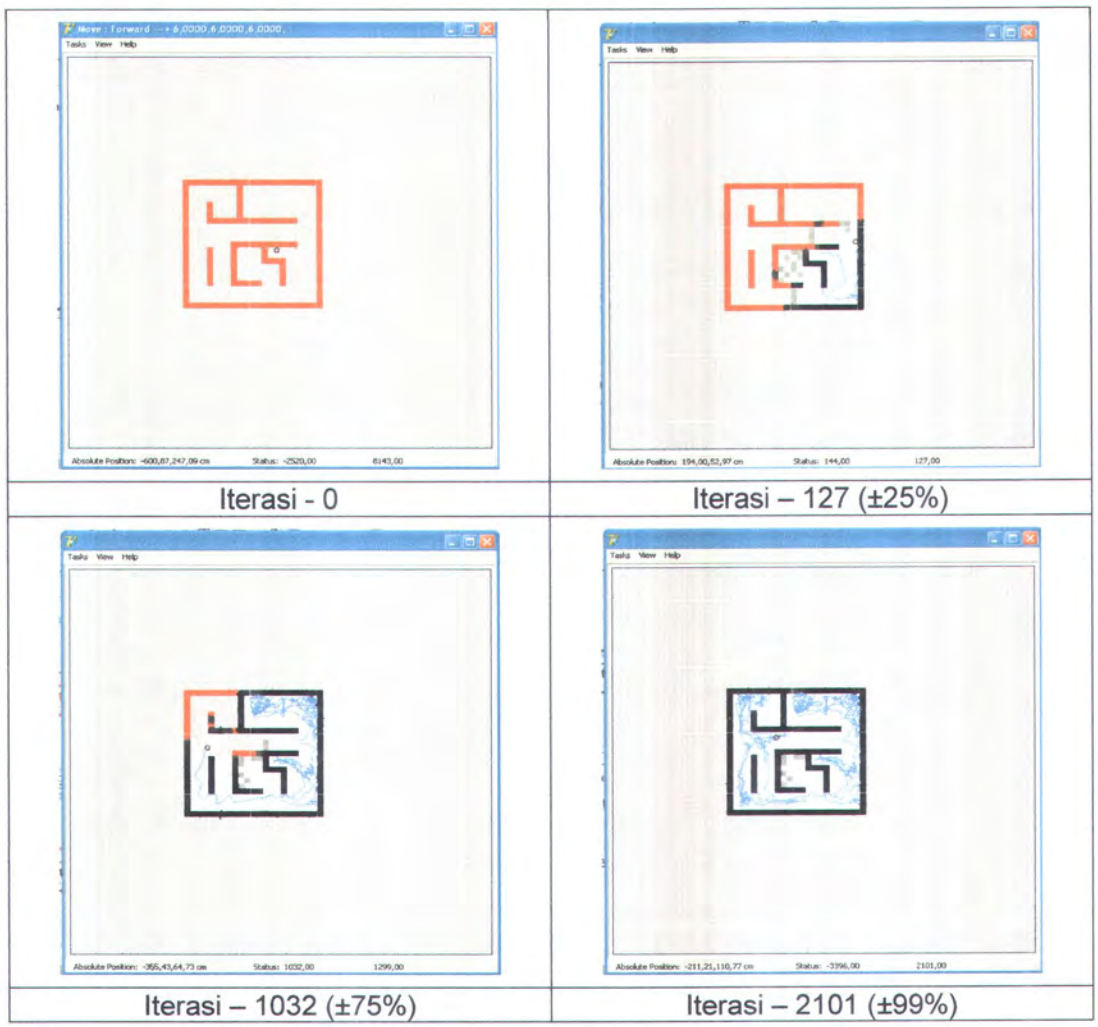
Gambar 4.8. Pengujian simulasi software - 2

Pada saat iterasi-0, robot belum mulai bergerak, peta abstaraksi ditandai dengan warna merah, warna merah merupakan tanda pada *cell* tersebut terdapat *obstacle* atau dinding. Saat iterasi – 192, daerah *occupancy* robot kurang lebih ¼ bagian dari *workspace* yang dibatasi dengan dinding atau *obstacle*. Daerah *occupancy* menjadi kurang lebih ¾ bagian dari *workspace* pada iterasi – 965 dan kurang lebih 99% dari *workspace* telah tereksplorasi pada iterasi 1549. Pada setiap iterasi robot selalu mengalami 4 tahap : persepsi (*local map*), hitung *occupancy*, navigasi, lokalisasi posisi robot dan membentuk *global map*. Perubahan dinding merah (abstrak) menjadi warna hitam (persepsi robot) menunjukkan proses *map building*.

Tabel 4.2 Data Pengujian simulasi *software* -2

Luas Wilayah (m^2)	Jumlah Ruangan dan Koridor	Jumlah Iterasi	Persentase Wilyah yang Tereksplorasi	Perbandingan Kenaikan % Luas Wilayah dan Jumlah Iterasi
116.64	3	192	25%	0.15
116.64	3	965	75%	0.075
116.64	3	1549	99%	0.047

4.2.3 Pengujian 3 (bentuk peta mengadopsi peta KRCI devisi Expert, 2005)



Gambar 4.9. Pengujian simulasi software – 3

Ukuran peta pengujian 3 adalah 23 *grid* * 23 *grid* =47.61 m^2.

Tabel 4.3 Data Pengujian simulasi *software* -3

Luas Wilayah (m^2)	Jumlah Ruangan dan Koridor	Jumlah Iterasi	Persentase Wilyah yang Tereksplorasi	Perbandingan Kenaikan % Luas Wilayah dan Jumlah Iterasi
47,61	8	127	25%	0,09372
47,61	8	1032	75%	0,026304
47,61	8	2101	99%	0,010689

4.2.4 Pengujian 4 (bentuk peta mengadopsi peta KRCI, 2005)

Ukuran peta pengujian 4 adalah 23 *grid* * 23 *grid* = 47.61 m^2.

Gambar 4.10. Pengujian simulasi software – 4

Tabel 4.4 Data Pengujian simulasi *software* -4

Luas Wilayah (m^2)	Jumlah Ruangan dan Koridor	Jumlah Iterasi	Persentase Wilyah yang Tereksplorasi	Perbandingan Kenaikan % Luas Wilayah dan Jumlah Iterasi
47,61	8	140	25%	0,085018
47,61	8	1968	75%	0,013022
47,61	8	2101	99%	0,019599

Dari hasil pengujian simulasi software -1 sampai dengan ke – 4, dapat dianalisa bahwa algoritma *occupancy grids map* dapat menyelesaikan kasus eksplorasi dengan nilai rata-rata daerah *occupancy* atau daerah *coverage*-nya adalah sama dengan 98%. Jumlah iterasi menunjukkan total pergerakan robot atau manuver robot, namun belum tentu sebanding dengan jarak tempuh robot karena terdapat manuver robot yang hanya bergerak pivot, dimana gerak pivot, titik pusat robot tetap tidak bergeser. Namun dengan melihat jumlah iterasi dan bentuk *path* robot, secara intuitif dapat disimpulkan bahwa: jumlah iterasi sangat dipengaruhi oleh jumlah ruangan dalam *workspace* dibanding dengan pengaruh dari luas *workspace*. Dengan kata lain jumlah iterasi akan makin besar jika ruangan bertambah pada luas *workspace* yang sama, Jumlah iterasi juga menunjukkan efektifitas navigasi dalam usaha robot untuk melakukan eksplorasi.

4.3 Simulasi Dengan *Mobile Robot*

Pengujian simulasi eksplorasi dan navigasi *mobile robot* dengan *occupancy grids* dilakukan dengan operator. Operator disini hanya melakukan perintah sesuai dengan hasil algoritma. Algoritma menghasilkan arah atau navigasi robot yang harus dilaksanakan oleh operator dengan cara melakukan penekanan tombol keyboard yang sesuai. Hal ini menyebabkan sistem eksplorasi dan navigasi robot tidak bekerja secara otomatis. Salah satu sebab yang mungkin adalah sistem komunikasi yang lemah.

Tahapan operator untuk membantu proses navigasi dan algoritma sistem dijelaskan sebagai berikut:

- Saat pertama, operator mengirimkan perintah untuk mengambil data awal sensor dengan cara menekan tombol F6. Komputer akan memerintahkan

robot untuk membaca data awal. Penekanan terus dilanjutkan sampai led pada setiap sensor jarak yang terpasang menyala sekejap. Kondisi led yang menyala sekejap, menandakan bahwa robot telah membaca sensor. Setelah dipastikan robot telah membaca sensor, langkah kemudian adalah menekan tombol F4. Komputer selanjutnya mengirimkan perintah permintaan data. Penekanan terus dilakukan hingga komputer menerima data valid dari paket data yang dikirimkan oleh robot.

- Setelah data valid diterima oleh robot, program akan melaksanakan algoritma *occupancy grids map*. Algoritma *occupancy grid map* pada intinya terdapat tahapan:
  - Algoritma akan melaksanakan rutin *Localization*. Dari hasil rutin ini dihasilkan posisi robot saat ini (x,y,θ). (Catatan: posisi awal robot disebut titik origin, dan direpresentasikan sebagai titik tengah abstraksi *workspace*). Penggalan kode program sebagai berikut:

```
//Start Occupancy Grids Map Method
 //Localization
 angle := FRobotView.GetCurrentRobotAngle;
 x := FRobotView.GetHorizontalRobotPosition;
 y := FRobotView.GetVerticalRobotPosition;
```

  - Dari hasil pembacaan sensor odometri, program melakukan proses perhitungan persamaan kinematik. Hasil perhitungan ini diperlukan untuk mengupdate posisi dan pose robot akibat pergerakan robot..

```
//Kinematic
Tetha := (180)*((RightWheel - LeftWheel)/RobotWidth);
dX:=((RightWheel+LeftWheel)/2)*Cos((pi/180)*(angle+Tetha));
dY:=((RightWheel+LeftWheel)/2)*sin((pi/180)*(angle+Tetha));
```

  - Proses *Updating* posisi dan pose robot

```
//Update Robot Localization
FRobotView.SetRobotPosition(x+dx,y+dy);
 FRobotView.SetRobotAngle(angle+Tetha);
angle := FRobotView.GetCurrentRobotAngle;
x := FRobotView.GetHorizontalRobotPosition;
 y := FRobotView.GetVerticalRobotPosition;
```

- *Perception*. Setelah posisi robot akibat pergerakan robot diketahui, selanjutnya algoritma melakukan interpretasi data sensor jarak, relatif terhadap posisi robot dan pose robot, ke dalam bentuk *occupancy map*.

```
//Robot perception
rangePerception(FrontSonar,RightSonar,RearSonar,LeftSonar);
```

- *Updating* nilai *occupancy*. Setelah persepsi robot terhadap lingkungan disekitarnya telah diperoleh, maka selanjutnya algoritma melaksanakan proses *updating* nilai *occupancy* pada setiap *cell* dalam jangkauan sensor relatif terhadap posisi robot saat ini (X,Y).

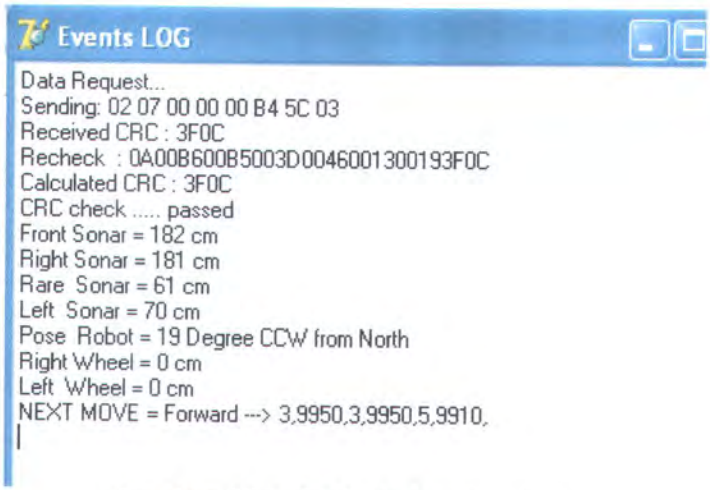
```
//Update Occupancy Cell
UpdateBayesCells(X,Y);
```

- Setelah proses peng-*update*-an nilai occupancy, selanjutnya algoritma menghitung nilai minimum pada setiap *cell occupancy* pada daerah *occupancy* ketiga arah: depan, kanan dan kiri. Nilai yang terkecil, merupakan pilihan arah navigasi selanjutnya. Kemudian arah navigasi ditampilkan pada window *events log*.

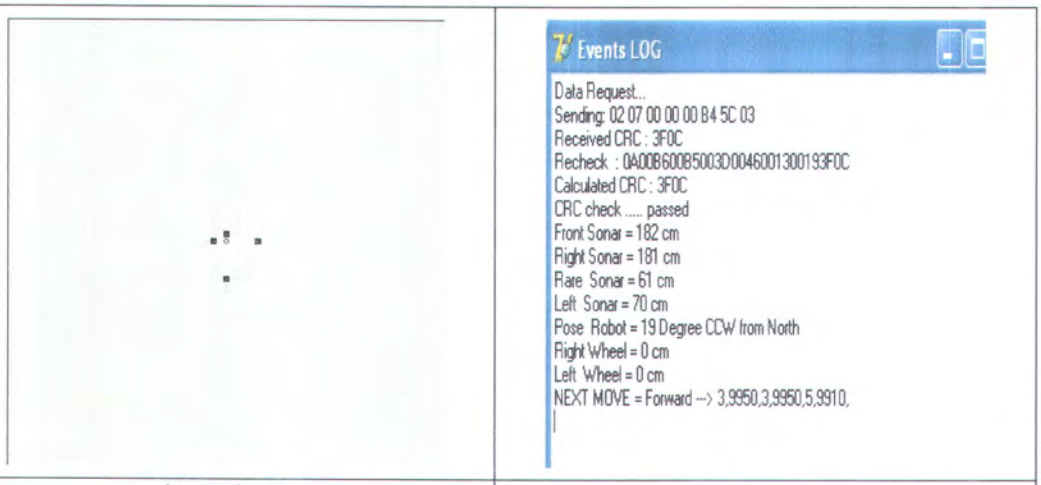
- Operator melakukan bantuan penekanan tombol navigasi sesuai dengan pesan dalam window *events log* (Gambar 4.11). Tombol navigasinya adalah:

panah atas (gerak maju 30 cm), panah kanan (gerak pivot ke kanan), panah kiri (gerak pivot ke kiri).

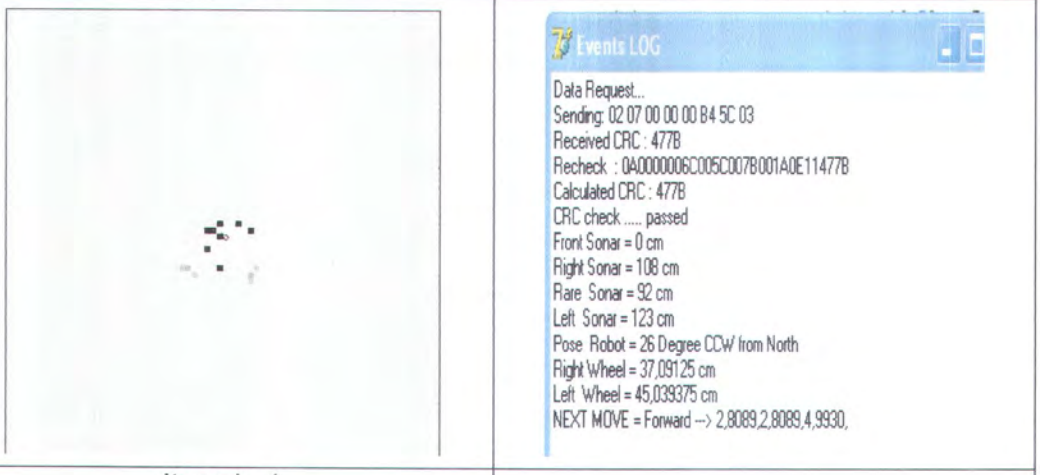
- Penekanan tombol navigasi yang sama terus dilakukan sampai robot menghasilkan gerakan tertentu.
- Setelah robot melakukan pergerakan, operator menekan tombol F4, untuk meminta data sensor dari robot. Penekanan tombol berulang-ulang sampai data valid diterima komputer.
- Dan seterusnya operator melakukan navigasi yang sesuai dengan hasil algoritma dan meminta data robot. Secara berulang kali operator melakukan navigasi, sampai peta yang ditampilkan pada program sesuai dengan *workspace* robot. (gambar 4.12).



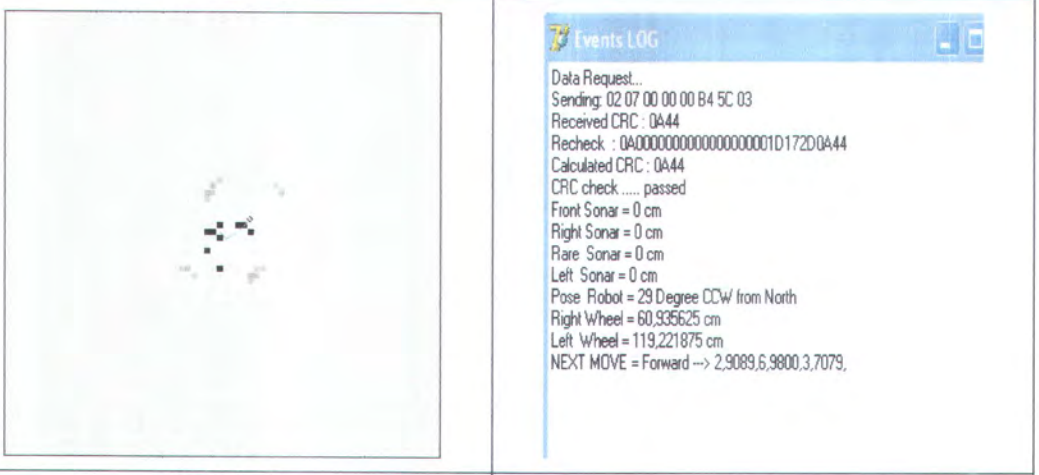
Gambar 4.11. GUI window events log



Iterasi - 0



Iterasi - 1



Iterasi - 2

Gambar 4.12. Proses eksplorasi dan navigasi mobile robot.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil pengujian dapat ditarik kesimpulan sebagai berikut:

- Terjadi penurunan efektifitas navigasi untuk eksplorasi yaitu rata-rata tejadi penurunan 50%, pada saat wilayah ekplorasi telah mencapai 20% dari total wilayah *workspace*. Untuk luas wilayah 609,6 m^2 terjadi penurunan dari 39% menjadi 16%, untuk luas wilayah 116,64 m^2 terjadi penurunan dari 15% menjadi 7.5%, untuk luas wilayah 47,61 m^2 terjadi penurunan 9.3% menjadi 2.6%. Hal ini disebabkan keterbatasan persepsi robot dalam usahanya mengenali lingkungan disekitarnya. Dengan hanya menggunakan 4 sensor jarak pada posisi di depan, kanan, kiri dan belakang robot, maka persepsi robot hanya dapat dibentuk pada arah hadap masing-masing sensor.
- Jumlah ruangan dan koridor serta luas wilayah *workspace* mempengaruhi efektifitas navigasi untuk ekplorasi.
- Pada luas wilayah workspace yang sama dengan jumlah ruangan dan koridor yang sama namun berbeda tata letak, nilai efektifitas navigasi tidak mengalami perubahan yang cukup berarti, yaitu antara 1% hingga 1.5%.
- Secara umum, daerah yang dapat diekplorasi mencapai 98% dari total wilayah *workspace*.
- Manuver robot yang hanya dapat bergerak maju, pivot ke kanan dan ke kiri sebesar 90^0 putaran, menyebabkan robot rentan terhadap kondisi "*tie*" atau

"*deadlock*". Hal ini dapat dilihat dari path yang dibentuk pada saat proses simulasi berlangsung, dimana terdapat suatu keadaan robot hanya bergerak dalam suatu area kecil saja.

- Persepsi robot yang dihasilkan dipengaruhi oleh jumlah sensor jarak yang terpasang dengan arah yang berbeda. Persepsi robot mempengaruhi pengambilan keputusan untuk navigasi dengan mencari fungsi minimum dari total *probabilitas occupancy cell* pada daerah *local map*.

5.2 Saran

Sebagai saran untuk perbaikan dan keberlanjutan penelitian ini adalah:

- Perbaikan sistem komunikasi. Sistem komunikasi yang baik akan menjamin kelancaran data, dan proses penyelesaian algoritma navigasi. Dengan menggunakan 2 *tranceiver* yang berbeda frekuensi modulasinya akan dapat memperbaiki performansi komunikasi data antara komputer dengan robot.
- Menggunakan lebih dari 4 sensor jarak, sehingga akan dihasilkan kecepatan proses eksplorasi dan navigasi robot.
- Manuver robot yang dapat bergerak ke segala arah. Hal ini dapat dilakukan dengan perbaikan disain kontroler dan mekanik robot yang lebih bagus.
- Selain memodelkan sensor untuk mencari nilai *occupancy*, juga perlu untuk memodelkan karakteristik pergerakan robot, sehingga ketepatan perhitungan lokalisasi dapat didekati.

DAFTAR PUSTAKA

[1] Anthony Stentz (1996). **Map-Based Strategies For Robot Navigation In Unknown Environments**, Proc. AAAI 1996 Spring Symposium On Planning With Incomplite Information For Robot Problems.

[2] Maxim A. Batalin, Et All (2003). **Efficient Exploration Without Localization**, Intl. Conference On Robotics And Automation (ICRA2003), May 12-17, Taipei-Taiwan.

[3] P Newman, M Bosse, J. Leonard. **Autonomous Feature-Based Exploration**, Massachusetts Institute Of Technology.

[4] Torvald Ersson And Xiaoming Hu. **Path Planning And Navigation Of Mobile Robots In Unknown Environments**, Optimization And Systems Theory, Center For Autonomous Systems, Royal Institute Of Technology, SE 100 44 Stockholm Sweden.

[5] Scott Burlington, Gregory Dudek. **Spiral Search As An Efficient Mobile Robotic Search Technique**, Center For Inteligent Machines Mcgill University, 3480 Rue Univeristy, Montreal Canada.

[6] Maxim A. Batalin, Gaurav S Sukhatme (2004). **Coverage, Exploration And Deployment By A Mobile Robot And Communication Network**, Tellecomunication Systems Journal, Special Issue On Wireless Sensor Networks, University Of Southern California, USA.

[7] Robert Grabowski, Pradeep Khosla, Howie Choset (2003). **Autonomous Exploration Via Regions Of Interest**, Proc. Intl Conference On Intelligent Robots And Systems,October, Las Vegas, Nevada.

[8] Anthony Stentz (1994). **Optimal And Efficient Path Planning For Partially-Known Environtments**, Proc. IEEE International Conference On Robotic And Automation, May, Pittsburg,PA 15213

[9] Cyrill Stachniss, Wolfram Burgard. **Exploring Unknown Environtments With Mobile Robots Using Coverage Maps**, University Of Freiburg, Departement Of Computer Science, Germany.

[10] Cyrill Stachniss, Wolfram Burgard, Dirk Hahnel. **Exploration With Active Loop-Closing For FastSLAM**, University Of Freiburg, Departement Of Computer Science, Germany.

[11] Wesley H. Huang (2001). **Optimal Line-Sweep-Based Decompositions For Coverage Algorithms**, Proc IEEE International Conference On Robotic And Automation.

[12] Amir Pirzadeh, Wesley Snyder (1990). **A Unified Solution To Coverage And Search In Explored And Unexplored Terrains Using Indirect Control**, Proc. IEEE International Conference On Robotics And Automation, May 1990.

[13] G.W. Lucas. **A Tutorial And Elementary Trajectory Model For Differential Steering System Of Robot Wheel Actuators**.

[14] Philip R.K (2001). **Development and Implementation of a Self Building Global Map for Autonomous Navigation**, Thesis, Virginia Polythecnic Institute and State University, April 2004, Virginia

[15] Eduardo Nebot (2002). **Simultaneous Localization and Mapping**, Australian Centre for Field Robotics, The University of Sydney, Australia, July 2002

[16] A.Elfes (1989). **Using Occupancy Grids for Mobile Robot Perception and Navigation**, IEEE Computer, 1989

[17] A.Elfes (1990). **Accupancy Grids: A stochastic Spatial Representation for Active Robot Perception**, Proceeding of the Sixth Conference on Uncertanty in AI, AAAI, July 1990, California, USA

[18] Andrew W. Moore(2001), **Probabilistic and Bayesian Analitics**,Source Material, Agustus 2001, Carnegie Mellon University

[19] Vassilis Varveropoulos, **Robot Localization and Map Construction Using Sona Data**, The Rossum Project, http://rossum.sourceforge.net

[20] Malcolm Ryan, **Motion Planning**, Source Material, http://

[21] Djoko Purwanto, **Visual Perception Based Robot Navigation**, Source Material, Departement of Electrical Engineering ITS, Surabaya, Indonesia

[22] **A Handbook Explaining The Basics of Motion**, Baldor Electric Company, Arkansas, USA, 1994

[23] Gayagwad Ramakant, Leonard Sokoloff (1988), **Analog and Digital Control System**, 1988, Prentice-Hall, New Jersey, USA

[24] Speed Controller, September 2001, http://www.unn.ac.uk/~epje2/robotinfo/homepages.which.net/ paul.hills/Motors/Motors.html

[25] James T. Humphries, Leslie P.Sheets (1983), **Industrial Electronic**, 1983, Breton Publishers Division of Wadsworth.inc, Massachusetts, USA

PARALLAX INC.
599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

PING)))™ Ultrasonic Range Finder (#28015)

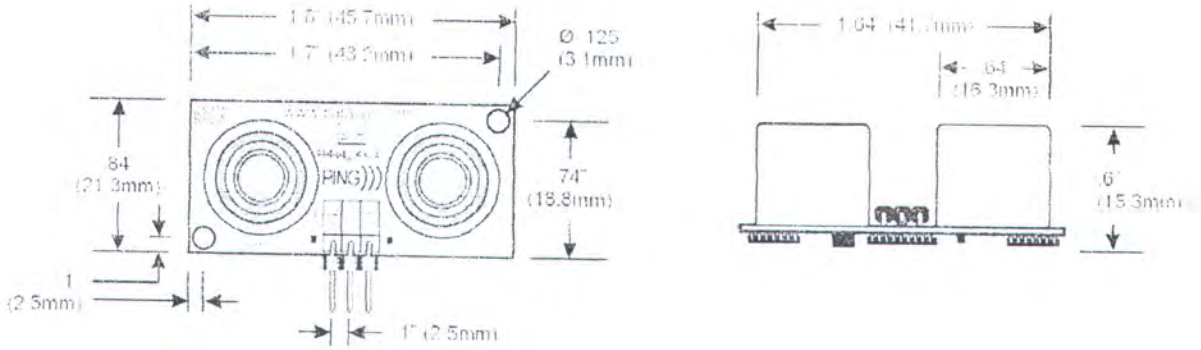
The Parallax PING))) ultrasonic range finder provides precise, non-contact distance measurements from about 3 cm (1.2 inches) to 3 meters (3.3 yards). It is very easy to connect to BASIC Stamp or Javelin Stamp microcontrollers, requiring only one I/O pin.

The Ping sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width the distance to target can easily be calculated.

Features

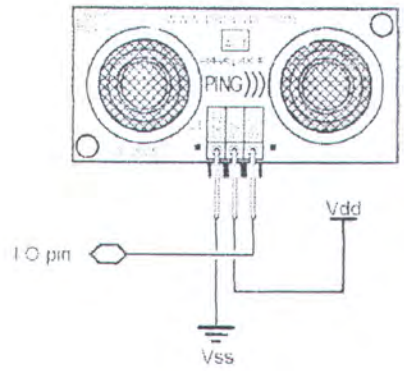
- Supply Voltage – 5 vdc
- Supply Current – 30 mA typ; 35 mA max
- Range – 3 cm to 3 m (1.2 in to 3.3 yrds)
- Input Trigger – positive TTL pulse, 2 uS min, 5 uS typ.
- Echo Pulse – positive TTL pulse, 115 uS to 18.5 mS
- Echo Hold-off – 750 uS from fall of Trigger pulse
- Burst Frequency – 40 kHz for 200 uS
- Burst Indicator LED shows sensor activity
- Delay before next measurement – 200 uS
- Size – 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)

Dimensions



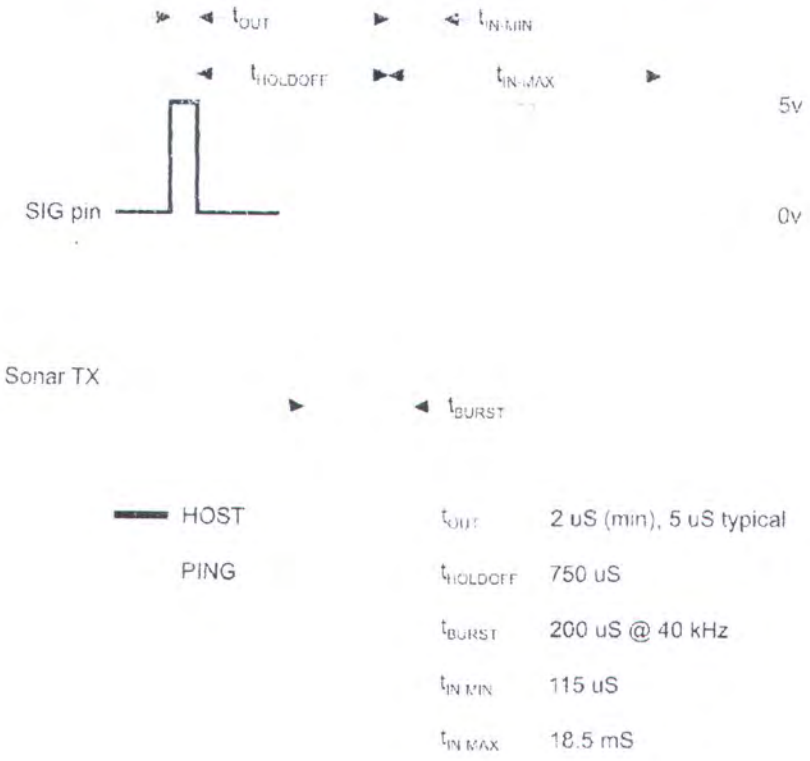
Connection to a Microcontroller

The PING))) sensor has a male 3-pin header used to supply power (5 vdc), ground, and signal. The header allows the sensor to be plugged into a solderless breadboard, or to be located remotely through the use of a standard servo extender cable. Standard connections are show in the diagram below:



Theory of Operation

The Ping sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air at about 1130 feet per second, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

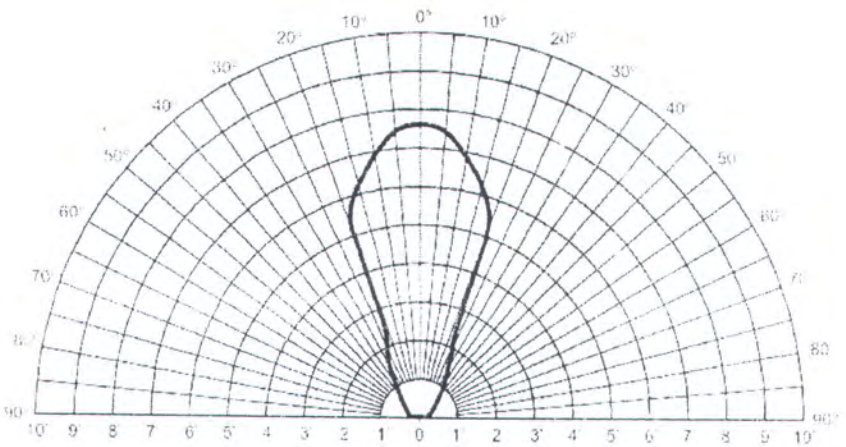


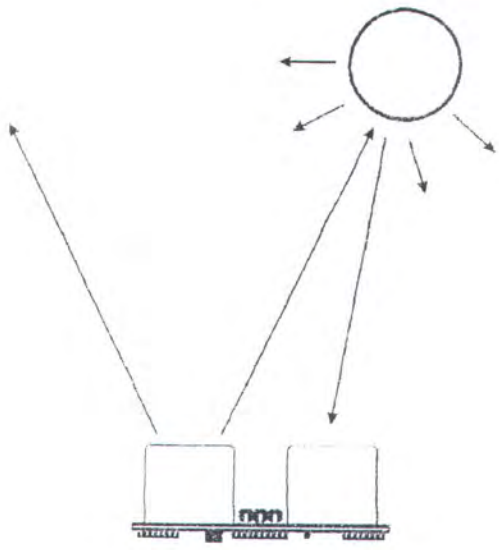
Test Data

The following test data is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

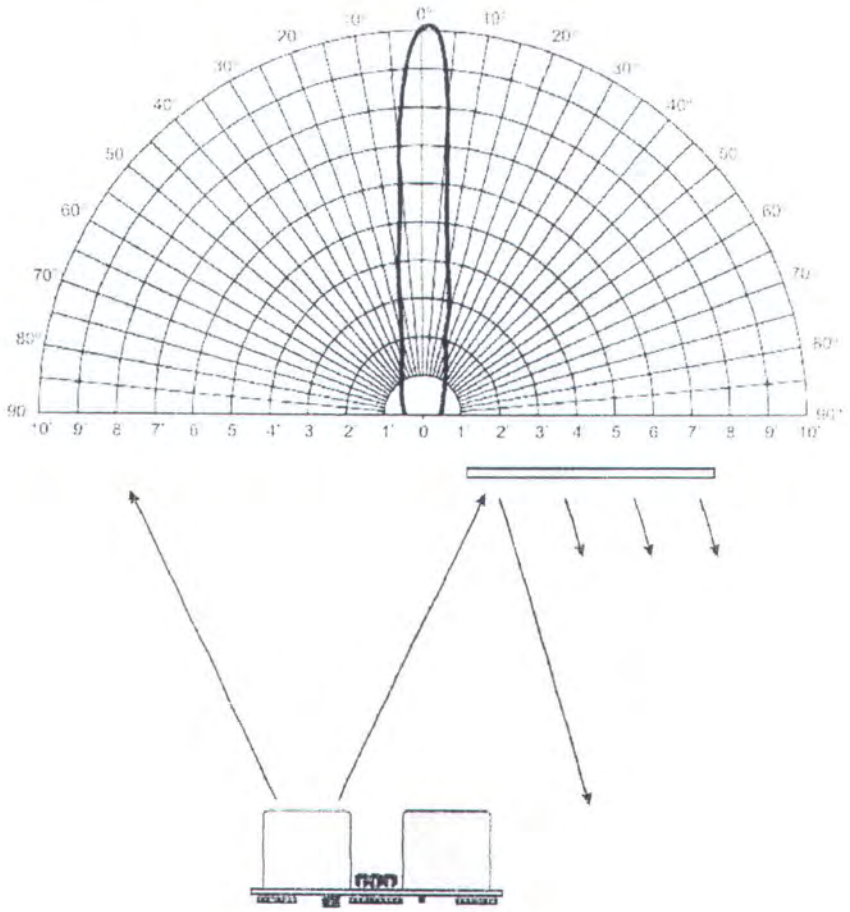
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation





Test 2

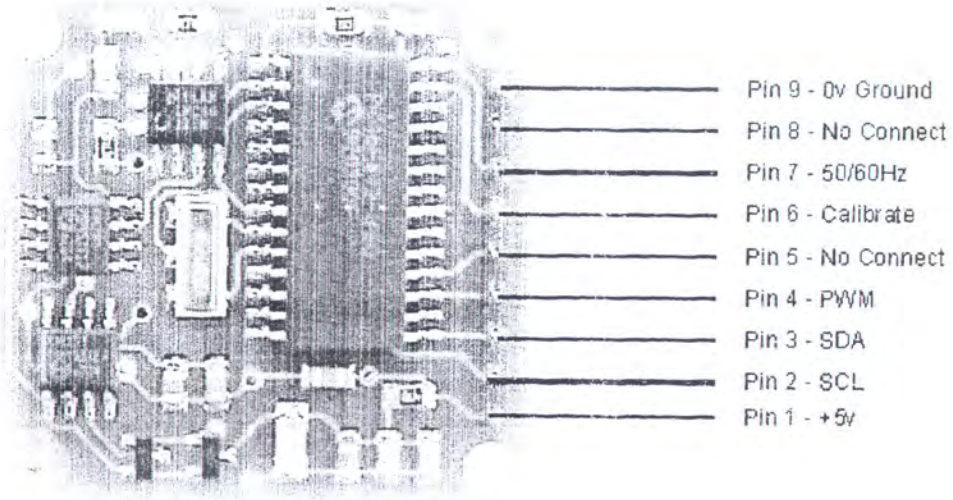
Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
- target positioned parallel to backplane of sensor



CMPS03 - Robot Compass Module

This compass module has been specifically designed for use in robots as an aid to navigation. The aim was to produce a unique number to represent the direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field.

Connections to the compass module



The compass module requires a 5v power supply at a nominal 15mA.
There are two ways of getting the bearing from the module. A PWM signal is available on pin 4, or an I2C interface is provided on pins 2,3.

The PWM signal is a pulse width modulated signal with the positive width of the pulse representing the angle. The pulse width varies from 1mS (0°) to 36.99mS (359.9°) – in other words 100uS/° with a +1mS offset. The signal goes low for 65mS between pulses, so the cycle time is 65mS + the pulse width - ie. 66ms-102ms. The pulse is generated by a 16 bit timer in the processor giving a 1uS resolution, however I would not recommend measuring this to anything better than 0.1° (10uS). Make sure you connect the I2C pins, SCL and SDA, to the 5v supply if you are using the PWM, as there are no pull-up resistors on these pins.

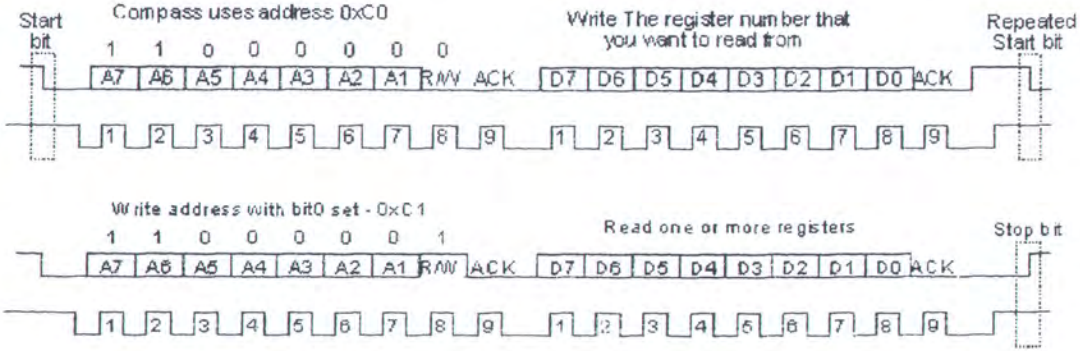
Pin 2,3 are an I2C interface and can be used to get a direct readout of the bearing. If the I2C interface is not used then these pins should be pulled high (to +5v) via a couple of resistors. Around 47k is ok, the values are not at all critical.

The I2C interface does not have any pull-up resistors on the board, these should be provided elsewhere, most probably with the bus master. They are required on both the SCL and SDA lines, but only once for the whole bus, not on each module. I suggest a value of 1k8 if you are going to be working up to 400KHz and 1k2 or even 1k if you are going up to 1MHz. The compass is designed to work at up to the standard clock speed (SCL) of 100KHz, however the clock speed can be raised to 1MHZ providing the following precaution is taken;
At speeds above around 160KHz the CPU cannot respond fast enough to read the I2C data. Therefore a small delay of 50uS should be inserted either side of writing the register address. No delays are required anywhere else in the sequence. By doing this, I have tested the compass module up to 1.3MHz SCL clock speed.

Pin 7 is an input pin selecting either 50Hz (low) or 60Hz (high) operation. I added this option after noticing a jitter of around 1.5° in the output. The cause was the 50Hz mains field in my workshop. By converting in synchronism with the mains frequency this was reduced to around 0.2° . An internal conversion is done every 40mS (50Hz) or every 33.3mS (60Hz). The pin has an on-board pull-up can be left unconnected for 60Hz operation. There is no synchronism between the PWM or I2C outputs and the conversion. They both retrieve the most recent internal reading, which is continuously converted, whether it is used or not.

Pin 6 is used to calibrate the compass. The calibrate input (pin 6) has an on-board pull-up resistor and can be left unconnected after calibration.

Pins 5 and 8 are No Connect. Actually pin 8 is the processor reset line and has an on-board pull-up resistor. It is there so that we can program the processor chip after placement on the PCB..



I2C communication protocol with the compass module is the same as popular eeprom's such as the 24C04.. First send a start bit, the module address (0XC0) with the read/write bit low, then the register number you wish to read. This is followed by a repeated start and the module address again with the read/write bit high (0XC1). You now read one or two bytes for 8bit or 16bit registers respectively. 16bit registers are read high byte first. The compass has a 16 byte array of registers, some of which double up as 16 bit registers as follows;

Register	Function
0	Software Revision Number
1	Compass Bearing as a byte, i.e. 0-255 for a full circle
2,3	Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-359.9 degrees.
4,5	Internal Test - Sensor1 difference signal - 16 bit signed word
6,7	Internal Test - Sensor2 difference signal - 16 bit signed word
8,9	Internal Test - Calibration value 1 - 16 bit signed word
10,11	Internal Test - Calibration value 2 - 16 bit signed word
12	Unused - Read as Zero
13	Unused - Read as Zero
14	Unused - Read as Undefined
15	Calibrate Command - Write 255 to perform calibration step. See text.

Register 0 is the Software revision number (8 at the time of writing). Register 1 is the bearing converted to a 0-255 value. This may be easier for some applications than 0-360 which requires two bytes. For those who require better resolution registers 2 and 3 (high byte first) are a 16 bit unsigned integer in the range 0-3599. This represents 0-359.9°. Registers 4 to 11 are internal test registers and 12,13 are unused. Register 14 is undefined. Don't read them if you don't want them - you'll just waste your I2C bandwidth. Register 15 is used to calibrate the compass.