DESAIN DAN IMPLEMENTASI *GRID-BASED MAP* SEBAGAI SISTEM PENGENALAN POSISI PADA KONTES ROBOT PEMADAM API INDONESIA (KRPAI) DIVISI BERODA

SKRIPSI

KONSENTRASI TEKNIK ELEKTRONIKA

Diajukan untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Teknik

Disusun Oleh:

NUR ISKANDAR JUANG

NIM. 0910630083 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

JURUSAN TEKNIK ELEKTRO

MALANG

2014

LEMBAR PERSETUJUAN

DESAIN DAN IMPLEMENTASI *GRID-BASED MAP* SEBAGAI SISTEM PENGENALAN POSISI PADA KONTES ROBOT PEMADAM API INDONESIA (KRPAI) DIVISI BERODA

SKRIPSI

Diajukan untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun Oleh:

NUR ISKANDAR JUANG
NIM. 0910630083 - 63

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	**Dosen Pembimbing II**
Waru Djuriatno, ST., MT.	**Mochammad Rif'an, ST., MT.**
NIP. 19690725 199702 1 001	**NIP. 19710301 200012 1 001**

LEMBAR PENGESAHAN

DESAIN DAN IMPLEMENTASI *GRID-BASED MAP* SEBAGAI SISTEM PENGENALAN POSISI PADA KONTES ROBOT PEMADAM API INDONESIA (KRPAI) DIVISI BERODA

SKRIPSI

Disusun Oleh:

NUR ISKANDAR JUANG

NIM. 0910630083 – 63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 21 Januari 2014

DOSEN PENGUJI

Akhmad Zainuri, ST., MT.	**Adharul Muttaqin, ST., MT.**
NIP. 19840120 201212 1 003	**NIP. 19760121 200501 1 001**

Goegoes Dwi Nusantoro, ST., MT.
NIP. 19711013 200604 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

M. Aziz Muslim, ST., MT., Ph.D
NIP. 19741203 200012 1 001

PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Desain dan Implementasi *Grid-Based Map* Sebagai Sistem Pengenalan Posisi pada Kontes Robot Pemadam Api Indonesia (KRPAI) Divisi Beroda" dengan baik. Skripsi ini disusun sebagai salah satu syarat untuk mencapai gelar Sarjana Teknik dari Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.

Penulis menyadari bahwa tanpa bantuan, bimbingan serta dorongan dari semua pihak, penyelesaian skripsi ini tidak mungkin bisa terwujud. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

- Allah SWT atas rahmat dan hidayah yang telah diberikan,
- Rasulullah Muhammad SAW, semoga shalawat serta salam selalu tercurah kepada beliau,
- Almh. Mama, terima kasih untuk segalanya yang telah diajarkan kepadaku semasa hidupmu, itu merupakan jariah yang akan terus mengalir untukmu,
- Ayah tercinta, doamu untukku adalah dukungan terbesar dalam segala urusanku. Nu', Ni', Ci terima kasih untuk segala doa dan dukungan yang merupakan sumber kekuatan terbesarku dalam menyelesaikan skripsi ini,
- Bapak Aziz Muslim, ST,.MT.,Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Hadi Suyono, ST,.MT.,Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Mochammad Rif'an, ST.,MT selaku Ketua Prodi Strata Satu Jurusan Teknik Elektro Universitas Brawijaya serta sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, saran dan kritik yang telah diberikan,
- Bapak Dr. Ir. Sholeh Hadi Pramono, MS selaku Dosen Pembimbing Akademik atas segala bimbingan, nasehat, saran, motivasi dan masukan yang telah diberikan,

- Ibu Ir. Nurussa'adah, MT selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Waru Djuriatno, ST., MT selaku Dosen Pembimbing I atas segala bimbingan, pengarahan, ide, saran serta motivasi yang telah diberikan selama pengerjaan skripsi ini,
- Bapak dan Ibu Dosen Jurusan Teknik Elektro,
- Staff Administrasi Jurusan Teknik Elektro,
- Teman-teman Ampere angkatan 2009,
- Rekan seperjuangan dalam skripsi, Jatra, Eky, Sam, Ikhsan, Mbah, Wito, Somad, Gladi, Aka, Firda, Risma, Bona, Vanti, Lintang, Rafi, Saddam dan Adeck terima kasih atas segala bantuan yang telah diberikan,
- Seluruh Keluarga Besar Anggota Tim Robot Kontes Robot Pemadam Api Indonesia, Anas, Lutfi, Imam, Zara, Desta, Manda, Guntur dan Cepi terima kasih atas segala semangat dan bantuan yang telah diberikan,
- Seluruh Keluarga Besar Tim Robot UB Jurusan Teknik Elektro atas segala bantuan alat, bahan dan masukan-masukannya yang telah di berikan,
- Sekret KRPAI, sekret KRAI, sekret KRSI, dan Laboratorium Sistem Digital yang selama ini telah menyediakan tempat bagi penulis dalam mengerjakan skripsi ini,
- Seluruh teman-teman serta semua pihak yang tidak mungkin untuk dicantumkan namanya satu-persatu, terima kasih banyak atas segala bentuk bantuan dan dukungannya.

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasan ilmu dan kendala-kendala lain yang terjadi selama pengerjaan skripsi ini. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, Januari 2014

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Nur Iskandar Juang, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Januari 2014, Desain dan Implementasi *Grid-Based Map* Sebagai Sistem Pengenalan Posisi pada Kontes Robot Pemadam Api Indonesia (KRPAI) Divisi Beroda, Dosen Pembimbing: Waru Djuriatno, ST., MT dan Mochammad Rif'an, ST., MT.

Kontes Robot Pemadam Api Indonesia (KRPAI) merupakan ajang perlombaan robotika nasional yang terdiri atas dua divisi yaitu Divisi Beroda dan Divisi Berkaki. Salah satu masalah yang sering dihadapi dalam perlombaan adalah pengenalan posisi robot. Tugas akhir ini merancang dan mengimplementasikan *grid-based map* sebagai sistem pengenalan posisi pada robot beroda. Peta yang dirancang berisi informasi mengenai jarak robot terhadap dinding bagian depan, belakang, kiri dan kanan pada posisi tertentu. Dengan menggunakan sensor ultrasonik PING))) robot mampu mengukur jarak robot terhadap dinding. Sensor *magnetic compass* digunakan robot untuk menentukan arah orientasi robot terhadap arena pertandingan.

Berdasarkan hasil pengujian sensor ultrasonik PING))), dapat diketahui rata-rata kesalahan pengukuran jarak adalah sebesar 0,23 cm dengan kesalahan terbesar adalah 0,4 cm. Hasil pembacaan sensor *magnetic compass* menunjukkan rata-rata kesalahan pembacaan sudut sebesar 1,8° dengan kesalahan terbesar sebesar 4°. Dari hasil pengujian keseluruhan, sistem pengenalan posisi menggunakan *grid-based map* dapat berfungsi sesuai dengan perancangan yang diinginkan.

Kata Kunci: *grid-based map*, KRPAI divisi beroda, *magnetic compass*, sensor ultrasonik.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kontes Robot Pemadam Api Indonesia (KRPAI) merupakan salah satu kontes robot tingkat nasional yang diadakan oleh Direktorat Jenderal Pendidikan Tinggi secara teratur setiap tahun. Kontes robot yang sebelumnya bernama Kontes Robot Cerdas Indonesia (KRCI) ini terbagi menjadi dua divisi yaitu Divisi Beroda dan Divisi Berkaki. Kedua divisi ini mempunyai tugas dan arena yang sama serta peraturan yang hampir sama.

Robot yang mengikuti Kontes Robot Pemadam Api Indonesia mempunyai tugas utama yaitu untuk memadamkan api yang terdapat pada arena pertandingan. Arena pertandingan pada Kontes Robot Pemadam Api Indonesia merupakan miniatur rumah yang terdiri dari lorong dan beberapa ruangan. Robot akan diletakkan pada sebuah arena pertandingan, kemudian robot harus dapat menyusuri arena untuk dapat mencari dan memadamkan sumber api yang berupa lilin. Selain itu robot juga mempunyai tugas tambahan yaitu kembali ke tempat atau ruangan *start* setelah berhasil memadamkan api. Kondisi arena pertandingan dapat berubah sesuai dengan hasil undian yang didapatkan. Robot harus mampu beradaptasi dengan cara mengenali lingkungan sekitar untuk dapat bergerak dan melaksanakan tugasnya. Untuk dapat memenuhi kriteria tersebut maka robot harus dilengkapi dengan sensor-sensor yang digunakan untuk mengambil data yang dibutuhkan dan kemudian data-data tersebut diolah oleh mikrokontroler. Salah satu sensor yang banyak digunakan agar robot dapat melaksanakan tugasnya adalah sensor ultrasonik. Sensor ini sering digunakan untuk mengukur jarak antara sensor dengan objek tertentu dengan memanfaatkan gelombang ultrasonik. Selain itu, bisa juga digunakan sensor kompas digital untuk menentukan orientasi posisi robot terhadap arena pertandingan.

Masalah yang masih sering dihadapi dalam KRPAI adalah kemampuan robot untuk mengetahui posisi robot itu sendiri di arena pertandingan. Kesalahan yang terjadi ketika robot menentukan posisi dapat mengakibatkan robot salah mengambil keputusan untuk dilakukan selanjutnya. Untuk itu diperlukan solusi

agar robot dapat mengetahui posisi di arena pertandingan dengan benar. Salah satu solusi yang dapat digunakan adalah dengan menanamkan peta arena pertandingan pada memori robot yang nantinya dapat digunakan oleh robot untuk dibandingkan dengan data yang diperoleh dari sensor tentang keadaan lingkungan di sekitar robot. Hasil pembandingan ini yang akan digunakan untuk menentukan posisi robot di arena pertandingan.

Dalam skripsi ini, peta yang ditanamkan pada memori robot berupa *grid-based map*, di mana masing-masing *grid* membentuk koordinat dan berisi data mengenai lingkungan sekitar robot berupa jarak robot terhadap dinding bagian depan, belakang, samping kiri dan samping kanan. Pengukuran jarak robot terdapat dinding arena pertandingan menggunakan sensor ultrasonik, sedangkan untuk mengetahui arah orientasi robot terhadap arena pertandingan digunakan modul CMPS03 *Magnetic Compass* sebagai sensor kompas digital.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, dapat disusun rumusan masalah sebagai berikut:

1) Bagaimana merancang *grid-based map* arena pertandingan KRPAI Divisi Beroda.
2) Bagaimana mengimplementasikan perangkat lunak *grid-based map* pada mikrokontroler ATmega128.
3) Bagaimana merancang sistem pengenalan posisi berdasarkan *grid-based map* yang telah dirancang.
4) Bagaimana merancang sistem pengakses sensor ultrasonik pada mikrokontroler ATmega8.
5) Bagaimana merancang sistem pengakses modul CMPS03 *Magnetic Compass* pada mikrokontroler ATmega128.

1.3. Batasan Masalah

Mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan penelitian akan diberi batasan sebagai berikut :

1) Sensor ultrasonik yang digunakan adalah sensor ultrasonik tipe PING))) untuk mengukur jarak.
2) Model lintasan/lorong yang digunakan sesuai dengan aturan dalam Kontes Robot Pemadam Api Indonesia (KRPAI) Divisi Beroda dengan mengacu pada satu mode pengacakan tanpa objek apapun dengan arah orientasi yang telah diketahui.
3) Implementasi desain menggunakan mikrokontroler lebih dari satu buah (*multi-microcontroller*).
4) Posisi-posisi yang akan dilewati robot untuk mencapai posisi tujuan sudah ditentukan terlebih dahulu.
5) *Mobile robot* yang dibuat menggunakan sistem penggerak *differential drive.*
6) Perancangan lebih ditekankan pada bagaimana robot dapat mengenali posisi robot di arena pertandingan sehingga tidak membahas pergerakan robot secara mendalam.

1.4. Tujuan

Tujuan penelitian ini adalah untuk merancang dan mengimplementasikan *grid-based map* arena pertandingan KRPAI pada robot KRPAI divisi beroda.

1.5. Sistematika Penulisan

Sistematika penulisan dalam penelitian ini sebagai berikut:

BAB I Pendahuluan

Memuat tentang latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika pembahasan

BAB II Tinjauan Pustaka

Membahas mengenai teori-teori yang mendukung dalam perancangan dan pembuatan alat atau sistem.

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Membahas mengenai penentuan spesifikasi alat beserta fungsi dan prinsip kerjanya, perancangan diagram blok, perancangan perangkat keras, perancangan perangkat lunak dan realisasi alat.

BAB V Pengujian dan Analisis

Memuat tentang aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis dilakukan pada seluruh sub sistem dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat tentang intisari dari hasil pengujian, menjawab rumusan masalah, serta memberikan saran ataupun rekomendasi untuk perbaikan kualitas penelitian pada masa yang akan datang.

BAB II
TINJAUAN PUSTAKA

Untuk memudahkan dalam memahami prinsip kerja maupun dasar-dasar perancangan sistem yang akan dibuat, maka perlu adanya penjelasan dan uraian mengenai teori penunjang yang digunakan dalam penelitian ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah:

- Kontes Robot Pemadam Api Indonesia
- Kinematika *Mobile Robot* dengan Sistem *Differential Drive*
- *Mobile Robot Localization*
- Kompas
- Sensor Ultrasonik PING)))
- Modul *Liquid Crystal Display* (LCD) Karakter
- Mikrokontroler ATmega128 (Atmel)
- Mikrokontroler ATmega8 (Atmel)

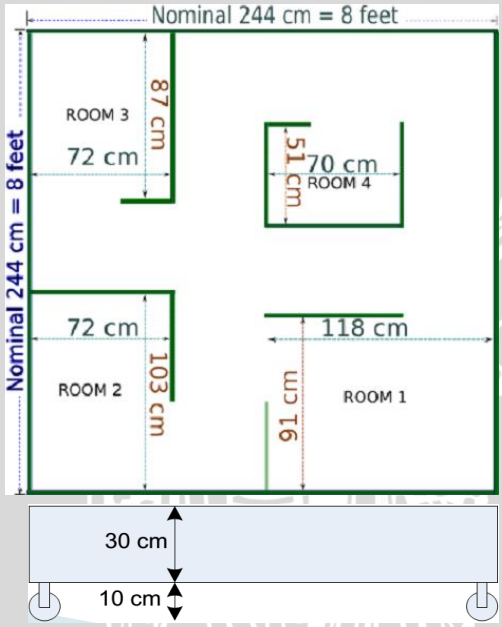
2.1. Kontes Robot Pamadam Api Indonesia

Kontes Robot Pemadam Api Indonesia merupakan pertandingan robot tingkat nasional yang diadakan oleh Direktorat Jendral Pendidikan Tinggi (Dirjen DIKTI) setiap tahun. Kontes Robot ini sebelumnya bernama Kontes Robot Cerdas Indonesia dan terdiri atas beberapa divisi yakni Divisi Senior Beroda, Divisi Senior Berkaki dan Divisi *Robo Soccer Humaniod League.* Namun pada tahun 2013, Divisi *Robo Soccer Humaniod League* berubah menjadi Kontes Robot Sepak Bola Indonesia (KRSBI) dan Kontes Robot Cerdas Indonesi berubah menjadi Kontes Robot Pemadam Api Indonesia (KRPAI) yang terdiri atas Divisi Beroda dan Divisi Berkaki. Tugas utama dari robot dalam Kontes Robot Pemadam Api Indonesia baik Divisi Beroda mapun Berkaki yaitu bergerak dari posisi *home* menelusuri arena untuk memadamkan api kemudian kembali lagi ke *home*.

Robot yang digunakan pada Kontes Robot Pemadam Api Indonesia ini memiliki dimensi panjang, lebar dan tinggi maksimum 31 cm × 31 cm × 27 cm untuk Divisi Beroda serta 46 cm × 31 cm × 27 cm untuk Divisi Berkaki. Arena

lomba baik untuk Divisi Beroda maupun Berkaki memiliki konfigurasi acak baik ruangan, posisi *home*, lokasi *hanging object* (cermin dan *sound damper*), lokasi lilin, lokasi furniture dan lokasi *uneven floor*. Selain itu, terdapat boneka anjing (*dog obstruction*) yang berfungsi untuk menghalangi perjalanan robot.

Arena lomba merupakan simulasi interior sebuah rumah yang terdiri dari 4 ruangan. Arena terbuat dari papan multipleks dengan ketebalan 1,8 s.d. 2 cm (rata-rata 1,9 cm) dan berukuran 244 cm × 244 cm × 30 cm. Di dalam arena lomba terdapat 4 ruangan dengan posisi tetap namun dua diantaranya (ruang 1 dan 4) memiliki pintu yang dapat digeser posisinya. Arena lomba memiliki roda agar dapat dipindah ataupun diputar dengan mudah. Ukuran arena lomba pada Kontes Robot Pemadam Api Indonesia ditunjukkan dalam Gambar 2.1.



Gambar 2.1 Dimensi Arena Lomba KRPAI 2013

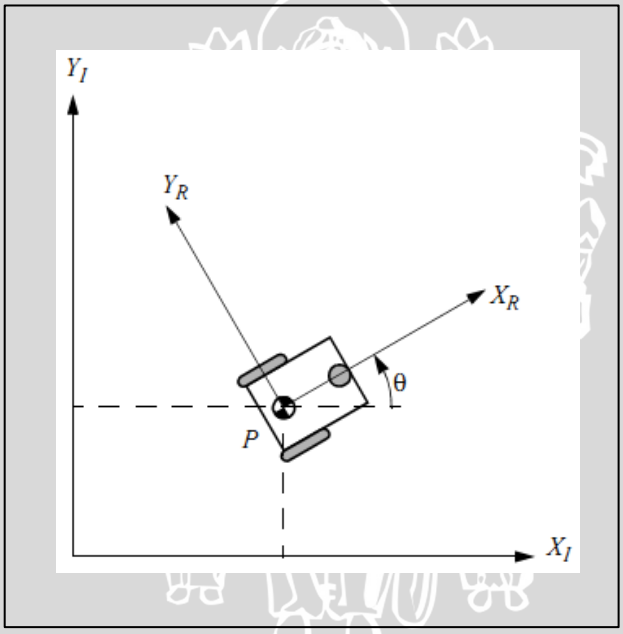
Sumber: DIKTI, 2013.

2.2. Kinematika Mobile Robot dengan Sistem *Differential Drive*

Differential drive merupakan salah satu sistem kemudi yang sering digunakan pada *mobile robot*. Sistem *differential drive* memiliki perancangan mekanik yang sederhana dibandingkan dengan sistem kemudi yang lain. Dalam sistem *differential drive*, robot bergerak menggunakan dua buah roda secara terpisah (T. Braunl, 2008: 5). Sehingga dalam sistem *differential drive* dibutuhkan minimal dua buah motor sebagai penggerak roda-roda tersebut. Dengan

mengkombinasikan kecepatan putaran motor-motornya, robot dapat dikemudikan lurus, membentuk lengkungan (*curve*), dan juga berputar (*rotation*) di tempat.

Untuk menganalisis kinematika *mobile robot* dengan sistem *differential drive*, maka robot dimodelkan menjadi suatu rangka statis di atas roda, yang mana robot akan beroperasi pada bidang horizontal. Pada bidang tersebut rangka robot memiliki tiga dimensi yaitu dua posisi dalam bidang dan satu untuk orientasi sepanjang sumbu vertikal yang ortogonal terhadap bidang. Pada kenyataannya tentu akan ditemui banyak derajat kebebasan dan fleksibilitas karena as roda, *wheel steering joints*, serta *wheel joints*. Namun dalam hal ini rangka robot yang dimodelkan hanya mengacu pada rangka statis robot itu sendiri, sehingga dapat dikatakan mengabaikan sendi (*joints*) dan derajat kebebasan (*degree of freedom*) internal terhadap robot dan roda-rodanya.



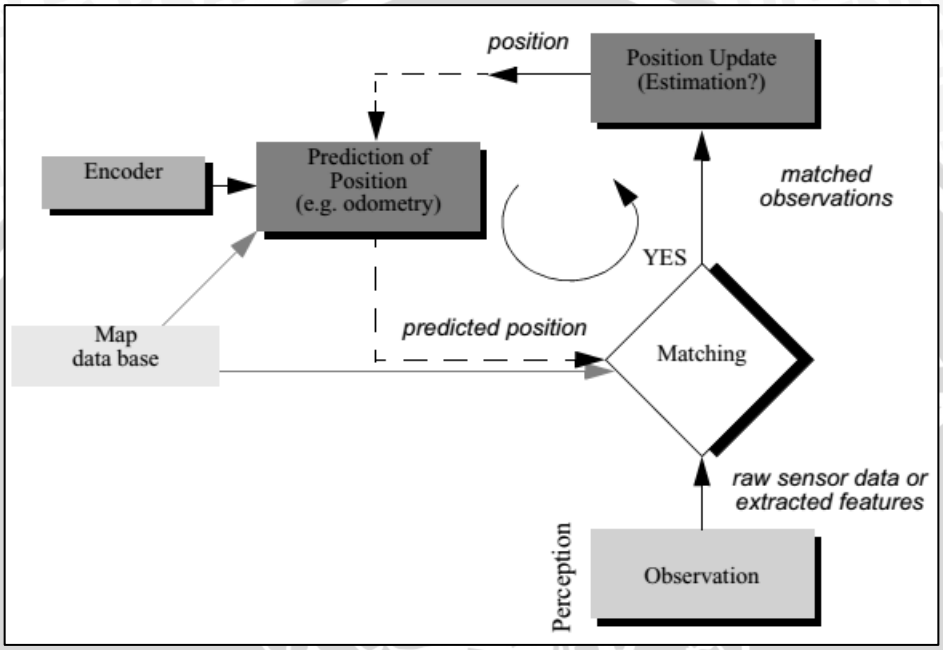
Gambar 2.2 Kerangka Acuan Global dan Lokal *Mobile Robot*

Sumber: R. Siegwart dan I.R. Nourbakhsh, 2004: 59

2.3. *Mobile Robot Localization*

Navigasi adalah salah satu kemampuan yang sangat penting bagi sebuah *mobile robot*. Agar *mobile robot* dapat bernavigasi dengan baik, ada empat hal yang harus terpenuhi yaitu robot harus dapat menerima dan mengolah data penting yang diterima oleh sensor yang digunakan (*perception*), robot harus dapat menentukan posisi keberadaannya terhadap lingkungan sekitar (*localization*),

robot harus dapat memutuskan apa yang dilakukan agar bisa mencapai tujuan (*cognition*), dan robot harus dapat mengatur keluaran motor yang digunakan agar dapat mencapai trayek yang diinginkan (*motion control*) (R. Siegwart dan I.R. Nourbakhsh, 2004: 181). Dari keempat hal di atas, *localization* telah memperoleh perhatian besar dalam penelitian beberapa dekade terakhir dan menghasilkan perkembangan yang signifikan.



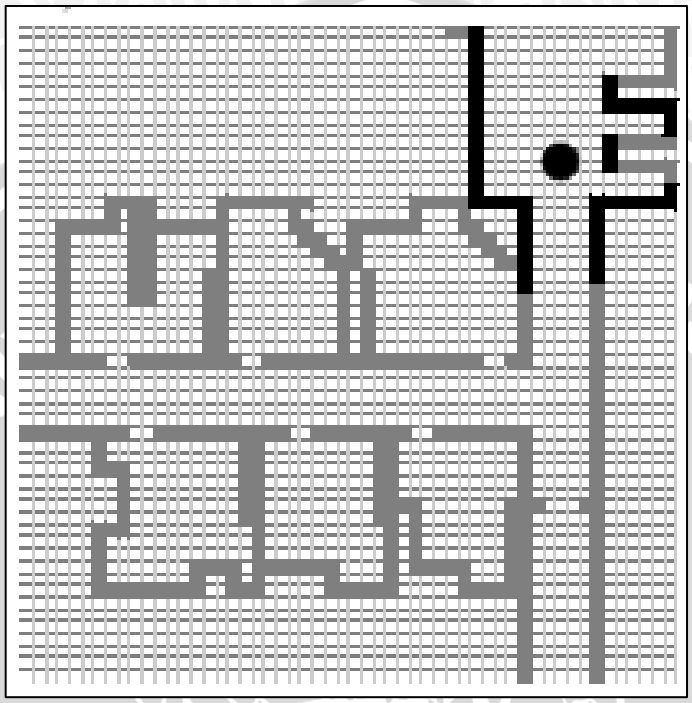
Gambar 2.3 Diagram Umum untuk *Mobile Robot Localization*

Sumber: R. Siegwart dan I.R. Nourbakhsh, 2004: 182

Mobile robot memerlukan suatu sistem navigasi untuk dapat bergerak dari suatu tempat ke tempat lain atau untuk dapat mencapai tujuan tertentu. Salah satu sistem navigasi yang dapat digunakan adalah dengan menggunakan *map-based* (*or model-based*) *navigation*, di mana *localization* dan *cognition* terdapat dalam sistem navigasi ini. Sedangkan peta yang digunakan untuk proses navigasi pada *mobile robot* bisa bermacam-macam misalnya *line-based map, grid-based map* atau *topological map.*

Grid Based-Map merupakan bagian dari *Metrics Map* yang mereprensentasikan suatu keadaan lingkungan ke dalam *grid*/blok ruang yang memiliki ukuran yang sama. Masing-masing blok/*grid cell* merupakan bidang 2 dimensi yang merepresentasikan koordinat x,y. Selain itu, masing-masing blok/*grid cell* biasanya berisi data-data tertentu. Data tersebut bisa berupa

representasi tentang keadaan lingkungan sekitar robot (jarak terhadap objek di sekitar robot) atau bisa berupa informasi tentang ada atau tidaknya objek pada koordinat tersebut (*Occupancy Grid Map*) sehingga blok/*grid* tersebut bisa ditempati oleh robot atau tidak. Contoh dari *grid-based map* terdapat dalam Gambar 2.4.



Gambar 2.4 *Grid-Based Map*

Sumber: R. Siegwart dan I.R. Nourbakhsh, 2004: 197

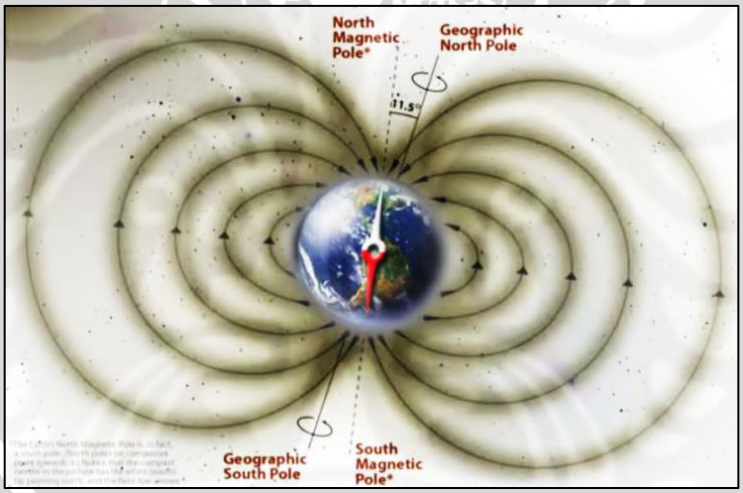
Dalam skripsi ini, dengan menggunakan *grid based-map*, robot secara eksplisit mencoba untuk mengenali keadaan lingkungan sekitar berupa jarak terhadap dinding dengan mengambil informasi yang diperoleh dari sensor, kemudian membandingkan data keadaan lingkungan sekitar hasil pembacaan sensor dengan data dari *map*/peta yang berisi keadaan lingkungan tersebut. Beberapa keuntungan yang dapat diperoleh dari penggunaan *map*/peta adalah:

- Konsep posisi dengan *map-based* membuat keyakinan sistem mengenai posisinya secara transparan tersedia untuk operator manusia.
- Keberadaan *map*/peta itu sendiri merepresentasikan sebuah medium untuk komunikasi antara manusia dengan robot: manusia dapat dengan mudah memberikan robot *map*/peta yang baru jika robot berada pada suatu lingkungan yang baru.

- *Map*/peta, jika dibuat oleh robot, dapat juga digunakan oleh manusia.

2.4. Kompas

Kompas merupakan alat bantu navigasi yang sering digunakan untuk menunjukkan arah di permukaan Bumi. Pada dasarnya, suatu benda yang memiliki sifat magnetik dan mampu bergerak bebas dapat dikatakan sebagai kompas. Pada umumnya, kompas terdiri atas jarum penunjuk magnetik yang dapat bergerak bebas pada suatu poros. Pergerakan jarum penunjuk itu akan menyelaraskan medan magnet yang terdapat terdapat pada jarum dengan medan magnet Bumi, sehingga dengan mudah dapat diperoleh informasi arah dengan menggunakan kutub utara magnet Bumi sebagai acuannya. Akan tetapi perlu diketahui, arah yang ditunjuk oleh kompas bukanlah arah utara Bumi yang sebenarnya, melainkan arah utara medan magnet Bumi. Kondisi medan magnet Bumi yang sebenarnya ditunjukkan dalam Gambar 2.5. Dalam Gambar tersebut, tampak bahwa kutub utara Bumi (*True North* atau *Geographic North Pole*), posisinya tidak berhimpitan dengan kutub utara medan magnet Bumi (*North Magnetic Pole* atau *Magnetic North*).



Gambar 2.5 Medan Magnet Bumi dan Kutub Bumi

Sumber: D.A. Wicaksono, 2009: 12

2.4.1. Kompas Digital

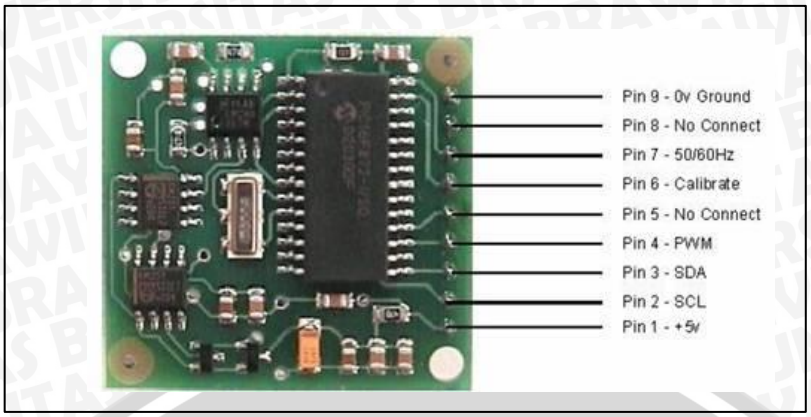
Dalam perkembangan teknologi navigasi telah ditemukan sebuah kompas digital, di mana sensor medan magnet Bumi telah menggantikan posisi jarum

penunjuk magnetik yang digunakan pada kompas magnetik. Sensor magnetik ini bekerja berdasarkan prinsip efek Hall. Secara singkat, efek Hall dapat dijelaskan sebagai berikut. Apabila suatu medan magnet melewati suatu batang metal atau lapisan film metal yang dialiri arus listrik, medan magnet tersebut akan mendorong elektron menuju ke salah satu sisi sepanjang lintasan medan magnet tersebut. Peristiwa ini menyebabkan jumlah elektron di salah satu sisi film lebih banyak dibandingkan sisi yang lainnya, sehingga dapat diukur beda tegangan di kedua sisi film tersebut. Besarnya tegangan yang terukur disebut sebagai tegangan Hall yang besarnya proporsional dengan besar arus yang mengalir pada film. Besarnya tegangan Hall juga proporsional dengan kuat medan magnet yang memotong film.

Penjelasan tentang efek Hall merupakan penjelasan secara sederhana ketika medan magnet memotong tegak lurus terhadap permukaan film. Apabila medan magnet tidak memotong tegak lurus pada film, maka yang berpengaruh adalah komponen-komponen medan magnet yang berpotongan tegak lurus terhadap permukaan film. Untuk mendapatkan hasil yang maksimal, maka dapat digunakan dua atau lebih sensor. Rasio tegangan Hall di kedua sensor tersebut dapat digunakan untuk merekonstruksi arah dan kuat medan magnet dengan perhitungan vektor.

2.4.2. Modul CMPS03 *Magnetic Compass*

Banyak jenis kompas digital termodul yang diproduksi khusus untuk keperluan navigasi, salah satunya adalah CMPS03 *magnetic compass* produksi Devantech. CMPS03 yang berukuran 3,4 cm × 3,4 cm ini menggunakan dua sensor medan magnet Philips KMZ51 yang sangat sensitif untuk mendeteksi medan magnet Bumi. Dua sensor ini dipasang saling bersilangan. Pada modul kompas telah terpasang rangkaian pengkondisi sinyal dan pemroses data keluaran sensor berupa mikrokontroler. Sehingga sensor ini dapat menghasilkan informasi arah secara langsung (dalam derajat). Modul kompas digital ini memiliki keakurasian 3-4° dengan resolusi sebesar 1° (M. C. Megasakti, 2010: 2). Konfigurasi pin dan bentuk fisik modul CMPS03 *magnetic compass* ditunjukkan dalam Gambar 2.6.



Gambar 2.6 Konfigurasi Pin Modul CMPS03 *Magnetic Compass*

Sumber: Digiware, 2007: 1

Modul CMPS03 *Magnetic Compass* memerlukan tegangan sebesar 5 V DC dan arus 15 mA pada saat beroperasi. Terdapat dua cara mengakses data modul kompas digital tersebut. Pertama dapat melalui pembacaan data PWM (*Pulse Width Modulation*) melalui pin PWM, 1 ms (untuk 0°) sampat 36,99 ms (untuk 359°). Kedua dapat melalui pembacaan data I_2C (*Inter Integrated Circuit*), yaitu melalui pin SDA (*Serial Data*) dan SCL (*Serial Clock*). Jika jalur I_2C tidak digunakan, maka pin ini harus di *pull-up* (ke +5 V) melalui resistor. Lalu komunikasi I_2C dimulai mengirimkan *start bit*, *address* modul kompas digital dengan *read/write low* (0xC0), kemudian nomor register yang akan dibaca. Kemudian diikuti dengan *start bit* lagi, *address* modul kompas dengan *read/write high* (0xC1). Lalu data dapat dibaca menggunakan satu atau dua register (8 bit atau 16 bit). Untuk register 16 bit, yang pertama kali dibaca adalah *high byte*. Melalui pembacaan *port* I_2C, data yang diakses dapat langsung digunakan yaitu 0°-359°.

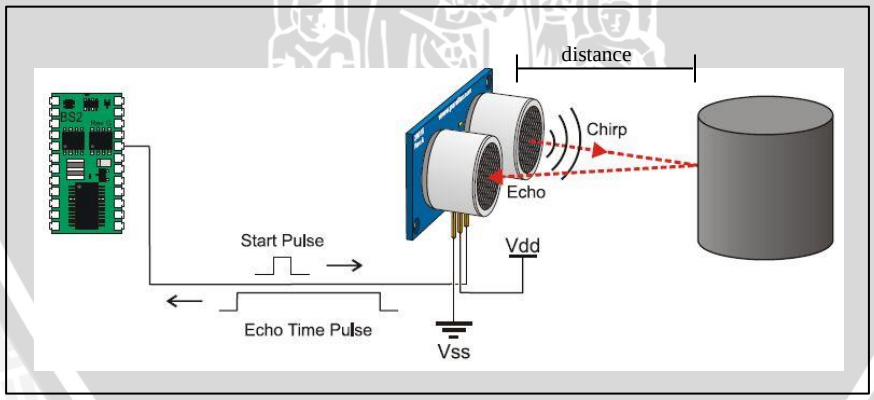
2.5. Sensor Ultrasonik PING)))

Sensor ultrasonik yang dipakai dalam penelitian ini adalah sensor PING))) produksi Parallax yang berupa modul siap pakai lengkap dengan pengirim dan penerima. Sensor ultrasonik PING))) digunakan untuk mendeteksi jarak dengan menggunakan gelombang ultrasonik berfrekuensi 40 kHz. Sinyal data sensor ultrasonik PING))) ini akan masuk ke kaki mikrokontroler. Bentuk fisik sensor ultrasonik PING))) ditunjukkan dalam Gambar 2.7.

Gambar 2.7 Sensor ultrasonik PING)))

Sumber: Parallax, 2008.

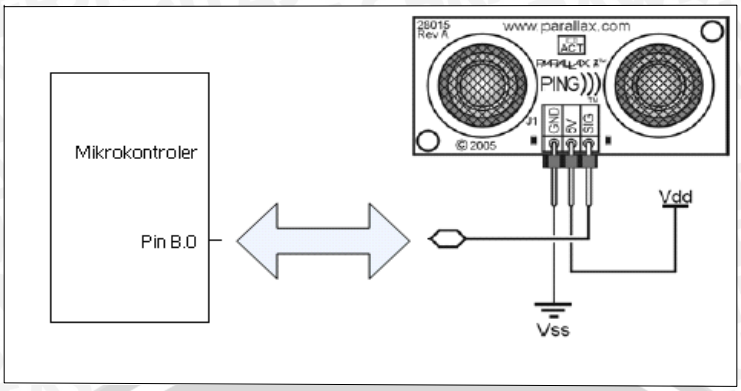
Modul PING))) mengukur jarak objek dengan cara memancarkan gelombang ultrasonik (40 kHz) selama t_{BURST} (200 µs) kemudian menunggu pantulannya. Modul PING))) memancarkan gelombang ultrasonik sesuai dengan masukan kontrol pin SIG. Gelombang ultrasonik ini melalui udara dengan kecepatan kurang lebih 344 meter per detik, mengenai objek dan memantul kembali ke modul PING))). Modul PING))) akan mengeluarkan pulsa *high* pada pin SIG selama memancarkan gelombang ultrasonik. Setelah pantulan gelombang terdeteksi, modul PING))) akan membuat pin SIG *low*. Lebar pulsa *high* (t_{IN}) ini sesuai dengan lama waktu tempuh gelombang ultrasonik untuk 2 kali jarak objek, sehingga jarak objek yang terukur adalah [(t_{IN} s × 344 m/s) / 2] meter. Ilustrasi cara kerja modul PING))) ditunjukkan dalam Gambar 2.8.



Gambar 2.8 Ilustrasi Cara Kerja Modul PING)))

Sumber: Parallax, 2008.

Sensor ultrasonik PING))) ini memiliki 3 buah kaki atau pin yakni pin VCC, pin GND dan pin SIG. Pin yang digunakan untuk mengirim data adalah pin SIG. Komunikasi mikrokontroler dengan PING))) ditunjukkan dalam Gambar 2.9.



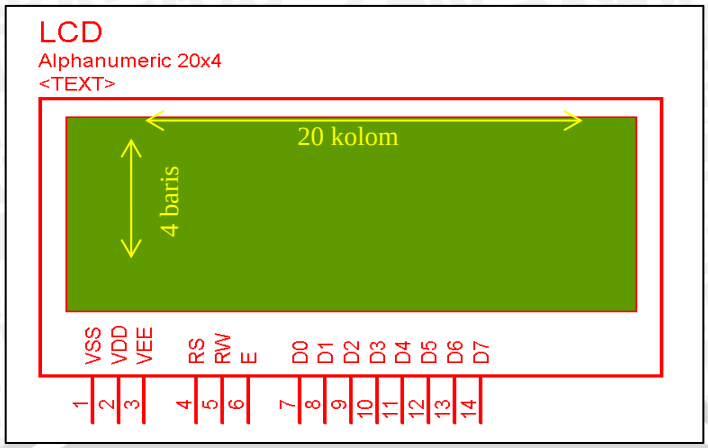
Gambar 2.9 Komunikasi mikrokontroler dengan PING)))

Sumber: Parallax, 2008.

Sensor ultrasonik PING))) ini memiliki beberapa kelebihan diantaranya bahwa sensor ini memiliki tingkat akurasi yang tinggi. Sensor ini mampu mendeteksi benda di depannya walaupun sudut yang dibentuk antara benda dan sensor tidak tegak lurus. Sensor ini juga mampu mendeteksi benda walaupun benda tersebut berukuran kecil.

2.6. Modul Liquid Crystal Display (LCD) Karakter

Liquid Crystal Display (LCD) Karakter merupakan perangkat elektronika termodul yang digunakan untuk menampilkan karakter, baik berupa karakter angka, huruf, atau karakter lainnya, sehingga tampilan tersebut dapat dilihat secara visual. Modul LCD karakter yang ada di pasaran sangatlah bermacam-macam. Jenis modul LCD karakter pada umumnya ditentukan menurut spesifikasi jumlah karakter yang dapat ditampilkan, warna karakter yang ditampilkan, dan juga warna *backlight* LCD. Meskipun memiliki spesifikasi berbeda-beda, jumlah serta fungsi pin LCD karakter tetap sama. Gambar 2.10 menunjukkan kofigurasi pin LCD karakter 20×4, dan Tabel 2.1 menunjukkan fungsi masing-masing pin LCD karakter.



Gambar 2.10 Konfigurasi Pin LCD Karakter 20×4

Sumber: Software Proteus ISIS Schematic Library, 2009

Tabel 2.1 Fungsi Masing-masing Pin LCD Karakter 20×4

No	Simbol	Level	Fungsi	
1	Vss		Power Suply	GND
2	Vcc			5 Volt
3	Vee			LCD Drive
4	RS	H/L	H: Data Input L: Ins Input	
5	R/W	H/L	H: Read L: Write	
6	E	H	Enable Signal	
7	DB0	H/L	Data Bus	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		
15	V+BL		Power Suply	4 - 4.3 Volt
16	V-BL			GND

Sumber: Manual Book LCD 20×4

Dalam penelitian ini digunakan modul LCD karakter 20x4, dengan spesifikasi sebagai berikut:

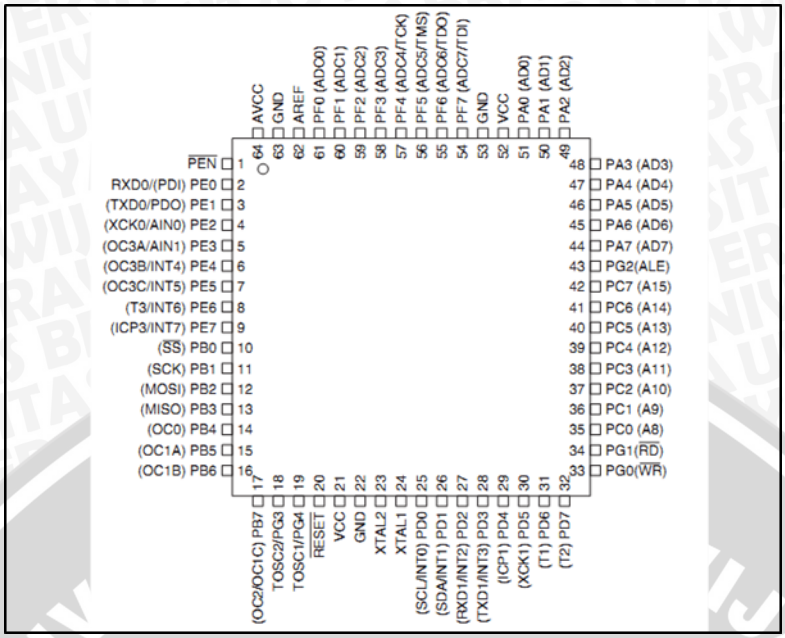
- Memiliki 20 karakter dan 4 baris tampilan.
- Memerlukan tegangan 5 V DC.
- Otomatis *reset* saat catu daya dinyalakan.

- Memiliki EEPROM 95 *full* (80 karakter).
- Menggunakan 4 bit data dan 3 bit kontrol.

2.7. Mikrokontroler ATmega128 (Atmel)

Mikrokontroler ATmega128 termasuk dalam keluarga AVR. Mikrokontroler AVR merupakan mikrokontroler berbasis arsitektur RISC (*Reduced Instruction Set Computing*) 8 bit. Mikrokontroler AVR didesain menggunakan arsitektur Harvard, di mana ruang dan jalur bus bagi memori program dipisahkan dengan memori data. Memori program diakses dengan *single-level*, di mana ketika sebuah instruksi dijalankan, instruksi lain berikutnya akan di *prefecth* dari memori program. ATmega128 memiliki *throughputs* mendekati 1 MIPS per MHz, sehingga perancang sistem dapat mengoptimalkan konsumsi daya yang diimbangi dengan kecepatan pemrosesan yang baik. Mikrokontroler ATmega128 memiliki 128K byte *In-System Programmable Flash* dengan kemampuan *Read-While-Write*, 4K byte EEPROM, 4K byte SRAM, 53 jalur I/O, 32 register, *Real Time Counter* (RTC), 4 *Timer/Counter* dengan mode *compare* dan PWM, 2 USART, *Two-wire Serial Interface*, sebuah ADC 8 kanal dengan resolusi hingga 10 bit, *Programmable Watchdog Timer* dengan *Internal Oscillator*, dan sebuah SPI. Mikrokontroler ATmega128 menyediakan pin PDO untuk MISO dan pin PDI untuk MOSI, tetapi menggunakan PDO dan PDI karena umumnya MISO dan MOSI tersebut digunakan untuk komunikasi SPI.

Masing-masing kaki dari mikrokontroler ATmega128 mempunyai fungsi tersendiri. Mikrokontroler ATmega 128 mempunyai 64 pin, konfigurasi pin ditunjukkan dalam Gambar 2.11.



Gambar 2.11 Konfigurasi Pin Mikrokontroler ATmega128

Sumber: Atmel, 2008: 2

Fungsi kaki-kaki ATmega128 adalah:

- *Port* A (Pin A0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* A adalah: *Alternate function pin External memory interface address and data bit* 0..7 (AD0..7).
- *Port* B (Pin B0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* B adalah: *Port* B0 SS (SPI *Slave Select Input*), *Port* B1 SCK (SPI *Bus Serial Clock*), *Port* B2 MOSI (SPI *Bus Master Output/Slave Input*), *Port* B3 MISO (SPI *Bus Master Input/Slave Output*), *Port* B4 OC0 (*Output Compare and* PWM *Output for Timer/Counter0*), *Port* B5 OC1A (*Output Compare and* PWM *Output A for Timer/Counter1*), *Port* B6 OC1B (*Output Compare and* PWM *Output B for Timer/Counter1*), *Port* B7 OC2/OC1C (*Output Compare and* PWM *Output for Timer/Counter2 or Output Compare and* PWM *Output C for Timer/Counter1*).
- *Port* C (Pin C0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* C adalah:

Alternate function pin External memory interface address bit 8..15 (A8..15).

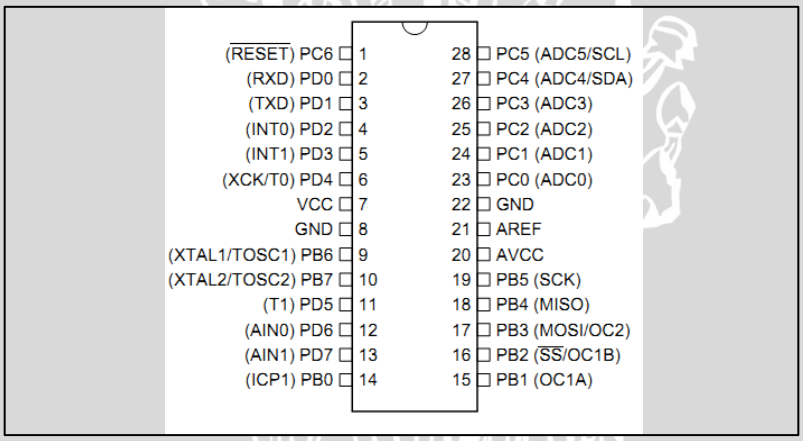
- *Port* D (Pin D0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* D adalah: *Port* D0 INT0/SCL (*External Interrupt0 Input or* TWI *Serial Clock*), *Port* D1 INT1/SDA (*External Interrupt1 Input or* TWI *Serial Data*), *Port* D2 INT2/RXD1 (*External Interrupt2 Input or* UART1 *Receive Pin*), *Port* D3 INT3/TXD1 (*External Interrupt3 Input or* UART1 *Transmit Pin*), *Port* D4 ICP1 (*Timer/Counter1 Input Capture Pin*), *Port* D5 XCK1 (USART1 *External Clock Input/Output*), *Port* D6 T1 (*Timer/Counter1 Clock Input*), *Port* D7 T2 (*Timer/Counter2 Clock Input*).
- *Port* E (Pin E0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* E adalah: *Port* E0 PDI/RXD0 (*Programming Data Input or* UART0 *Receive Pin*), *Port* E1 PDO/TXD0 (*Programming Data Input or* UART0 *Transmit Pin*), *Port* E2 AIN0/XCK0 (*Analog Comparator Positive Input or* USART0 *External Clock Input/Output*), *Port* E3 AIN1/OC3A (*Analog Comparator Negative Input or Output Compare and* PWM *Output A for Timer/Counter3*), *Port* E4 INT4/OC3B (*External Interrupt4 Input or Output Compare and* PWM *Output B for Timer/Counter3*), *Port* E5 INT5/OC3C (*External Interrupt5 Input or Output Compare and* PWM *Output C for Timer/Counter3*), *Port* E6 INT6/T3 (*External Interrupt6 Input or Timer/Counter3 Clock Input*), *Port* E7 INT7/ICP3 (*External Interrupt7 Input or Timer/Counter3 Input Capture Pin*).
- *Port* F (Pin F0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* F adalah: A/D *Converter* bit 0..7 (ADC0..7), JTAG *Interface.*
- *Port* G (Pin G0..5), merupakan saluran 5 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus yang bermacam-macam.
- Pin 21, 52 VCC, merupakan saluran masukan untuk catu daya positif sebesar 5 V DC.
- Pin 22, 53 GND, merupakan *ground* seluruh rangkaian.

- Pin 62 AREF, merupakan pin analog referensi untuk masukan ADC.
- Pin 63 GND, merupakan *ground* ADC.
- Pin 64 AVCC, merupakan catu daya untuk perangkat ADC.

2.8. Mikrokontroler ATmega8 (Atmel)

Mikrokontroler ATmega8 juga merupakan keluarga AVR 8 bit seperti mikrokontroler ATmega128. Mikrokontroler ATmega8 memiliki 8K byte *In-System Programmable Flash* dengan kemampuan *Read-While-Write*, 512 byte EEPROM, 1K byte SRAM, 23 jalur I/O, 32 register, 3 *Timer/Counter* dengan mode *compare* dan PWM, 1 USART, *Two-wire Serial Interface*, serta sebuah ADC 6 kanal dengan resolusi hingga 10 bit.

Masing-masing kaki dari mikrokontroler ATmega8 mempunyai fungsi tersendiri. Mikrokontroler ATmega8 mempunyai 28 pin, konfigurasi pin ditunjukkan dalam Gambar 2.12.



Gambar 2.12 Konfigurasi Pin Mikrokontroler ATmega8

Sumber: Atmel, 2003: 2

Fungsi kaki-kaki ATmega8 adalah:

- *Port* B (Pin B0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* B adalah: *Port* B0 ICP1 (*Timer/Counter1 Input Capture Pin*), *Port* B1 OC1A (*Timer/Counter1 Output Compare Match A Output*), *Port* B2 SS/OC1B (SPI *Slave Select Input or Timer/Counter1 Output Compare Match B Output*), *Port* B3 MOSI/OC2 (SPI *Bus Master Output/Slave Input or*

Timer/Counter2 Output Compare Match Output), *Port* B4 MISO (SPI *Bus Master Input/Slave Output*), *Port* B5 SCK (SPI *Bus Serial Clock*), *Port* B6 XTAL1, *Port* B7 XTAL2.

- *Port* C (Pin C0..6), merupakan saluran 7 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* C adalah: ADC (*Input* ADC *Channel* 0..5), *Port* C4 SDA (*Two-Wire Serial Bus Data Input/Output Line*), *Port* C5 SCL (*Two-Wire Serial Bus Clock Line*), *Port* C6 RESET.
- *Port* D (Pin D0..7), merupakan saluran 8 bit masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* D adalah: *Port* D0 RXD (USART *Input Pin*), *Port* D1 TXD (USART *Output Pin*), *Port* D2 INT0 (*External Interrupt 0 Input*), *Port* D3 INT1 (*External Interrupt 1 Input*), *Port* D4 XCK (USART *External Clock Input/Output*), *Port* D5 T1 (*Timer/Counter External Counter Input*), *Port* D6 AIN0 (*Analog Comparator Positive Input*), *Port* D7 AIN1 (*Analog Comparator Negative Input*).
- Pin 7 VCC, merupakan saluran masukan untuk catu daya positif sebesar 5 V DC.
- Pin 8 GND, merupakan *ground* seluruh rangkaian.
- Pin 21 AREF, merupakan pin analog referensi untuk masukan ADC.
- Pin 22 GND, merupakan *ground* ADC.
- Pin 20 AVCC, merupakan catu daya untuk perangkat ADC.

BAB III
METODOLOGI PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif yaitu perancangan dan implementasi sistem. Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat pada pendahuluan maka diperlukan metode penelitian. Metode penelitian yang digunakan adalah penentuan spesifikasi alat, studi literatur, perancangan dan pembuatan alat, pengujian dan analisis, serta pengambilan kesimpulan. Studi literatur dilakukan untuk mempelajari teori penunjang yang dibutuhkan dalam perancangan dan pembuatan alat.

3.1. Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut:

1) Rangka *mobile robot* berbahan dasar aluminium dan mika *acrylic*.
2) *Mobile robot* menggunakan sistem penggerak roda yang terletak di sisi kanan dan kiri badan robot, dan digerakkan oleh dua motor DC.
3) *Mobile robot* menggunakan sistem kendali *differential drive*, serta mampu melakukan gerak dasar yaitu maju, mundur dan rotasi atau berputar di tempat dengan sudut hingga 360°.
4) Pendeteksi keberadan dinding di sekitar robot menggunakan sensor ultrasonik PING))).
5) Menggunakan modul CMPS03 *magnetic compass* (resolusi: 1°, akurasi: 3-4°) untuk mengetahui arah orientasi robot.
6) Menggunakan *push button* sebagai tombol *start* yang berfungsi sebagai perangkat masukan sistem agar sistem dapat mengetahui kapan harus memulai misi.
7) Menggunakan LCD *Alphanumeric* 4×20 sebagai penampil.
8) Menggunakan sumber daya dari dua buah baterai LiIon (*Lithium Ion*) 12,5 V DC, 4500 mAh yang dipasang secara seri.

3.2. Studi Literatur

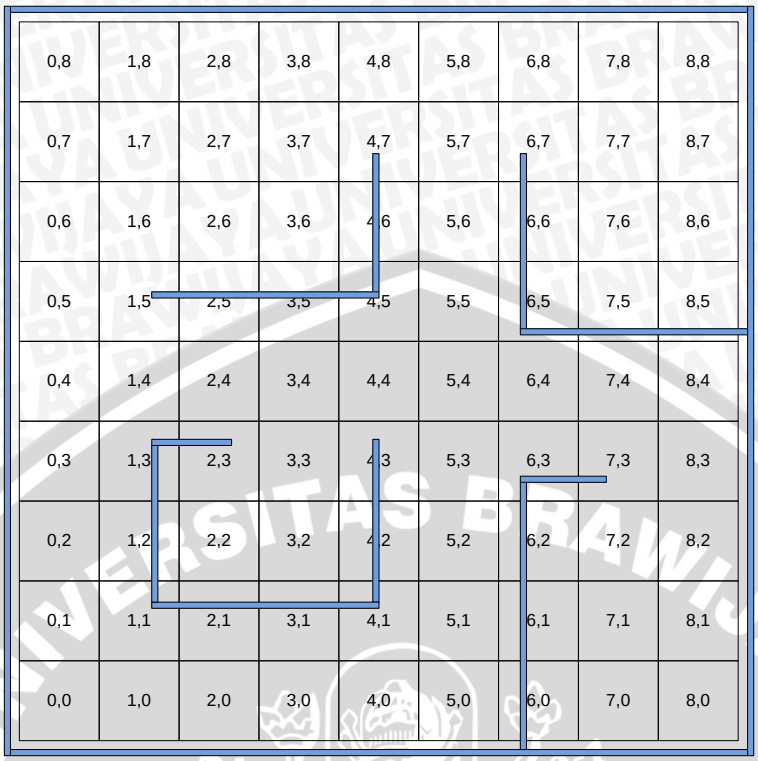
Studi literatur dilakukan untuk mempelajari teori penunjang yang dibutuhkan dalam perancangan dan pembuatan alat. Teori yang diperlukan yaitu: Kontes Robot Pemadam Api Indonesia (KRPAI), Kinematika *Mobile Robot* dengan Sistem *Differential Drive*, *Mobile Robot Localization*, Kompas, Sensor Ultrasonik PING))), Modul *Liquid Crystal Display* (LCD) Karakter, Mikrokontroler ATmega128 (Atmel), Mikrokontroler ATmega8 (Atmel).

3.3. Perancangan dan Pembuatan Alat

Perancangan dan pembuatan alat dalam penelitian ini dibagi menjadi tiga bagian, yaitu perancangan sistem pengenalan posisi, perancangan perangkat keras (*hardware*) dan perancangan perangkat lunak (*software*).

3.3.1. Perancangan Sistem Pengenalan Posisi

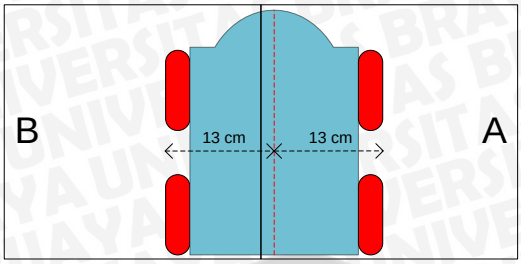
Grid Based-Map merupakan bagian dari *Metrics Map* yang mereprensentasikan suatu keadaan lingkungan ke dalam *grid*/blok ruang yang memiliki ukuran yang sama. Masing-masing blok/*grid cell* merupakan bidang 2 dimensi yang merepresentasikan koordinat x,y. Pada penelitian ini arena pertandingan KRPAI akan direpresentasikan ke dalam peta koordinat blok/*grid-based map*. Ukuran dari masing-masing blok/*grid* disesuaikan dengan dimensi robot yaitu 26 cm × 26 cm, sehingga peta tersebut akan terdiri dari blok/*grid* dengan jumlah 9 × 9. Peta tersebut kemudian diubah menjadi *array* yang membentuk blok/*grid* di mana masing-masing blok/*grid* tersebut berisi data-data mengenai keadaan lingkungan di sekitar robot. Data-data tersebut berupa data jarak robot terhadap dinding di bagian depan, belakang, sisi kanan dan sisi kiri serta apakah robot berada di dalam ruangan atau berada di lorong.



Gambar 3.1 Rancangan *Grid Based-Map* Arena KRPAI

Untuk dapat mengenali posisi di arena pertandingan, robot akan mencari data pada peta yang sudah ditanamkan pada memori robot yang sesuai atau paling mendekati dengan kondisi robot pada arena pertandingan yang diperoleh dari hasil pembacaan sensor ultrasonik. Sensor kompas akan digunakan untuk menentukan konfigurasi orientasi sensor relatif berdasarkan arah orientasi robot terhadap arena pertandingan. Proses pencarian dilakukan dengan cara membandingkan data-data tersebut sampai ditemukan posisi yang memiliki data sesuai atau paling mendekati kondisi robot sekarang. Dalam proses pembandingan, robot akan mencocokan data yang diperoleh dari hasil pembacaan sensor ultrasonik dengan data pada peta.

Data yang diperoleh dari hasil pengukuran sensor ultrasonik dibandingkan dengan data peta ± 13 cm. Maksud dari ± 13 cm ini adalah ketika badan robot berada pada dua posisi berbeda misalnya posisi A dan B, selama setengah dari badan robot masih berada pada posisi A, maka robot akan dianggap berada pada posisi A walaupun ada sebagian badan robot yang masuk ke posisi B. Ilustrasi keadaan robot yang berada pada posisi berbeda dalam waktu bersamaan ditunjukkan dalam Gambar 3.2.



Gambar 3.2 Ilustrasi Keadaan Robot Dalam Dua Koordinat Berbeda

Dari hasil pembandingan tersebut, jika ditemukan satu posisi di mana data hasil pembacaan keempat sensor ultrasonik sesuai atau mendekati empat data yang mewakili posisi tersebut, maka robot dianggap berada pada posisi tersebut. Namun apabila terdapat lebih dari satu posisi yang memenuhi persyaratan, maka dicari nilai rata-rata terkecil dari selisih antara data hasil pembacaan sensor dengan data pada peta dari masing-masing posisi. Caranya adalah dengan mencari nilai mutlak dari selisih antara keempat data hasil pembacaan sensor dengan data pada peta untuk masing-masing posisi, kemudian nilai-nilai tersebut dijumlahkan lalu dibagi dengan empat. Secara matematis dapat ditulis:

$$\Delta r_{x,y} = \frac{|s1 - d1| + |s2 - d2| + |s3 - d3| + |s4 - d4|}{4} \quad \text{...(3.1)}$$

Dengan:

$\Delta r_{x,y}$ = Nilai rata-rata posisi x,y

$s1$, $s2$, $s3$, $s4$ = Data hasil pembacaan sensor ultrasonik ke-1 sampai ke-4

$d1$, $d2$, $d3$, $d4$ = Data pada peta yang mewakili posisi x,y

Nilai rata-rata dari hasil perhitungan untuk masing-masing posisi ini kemudian dibandingkan untuk mencari nilai rata-rata terkecil. Dari hasil perhitungan, posisi dengan nilai rata-rata terkecil kemudian dianggap sebagai posisi keberadaan robot sekarang.

3.3.2. Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

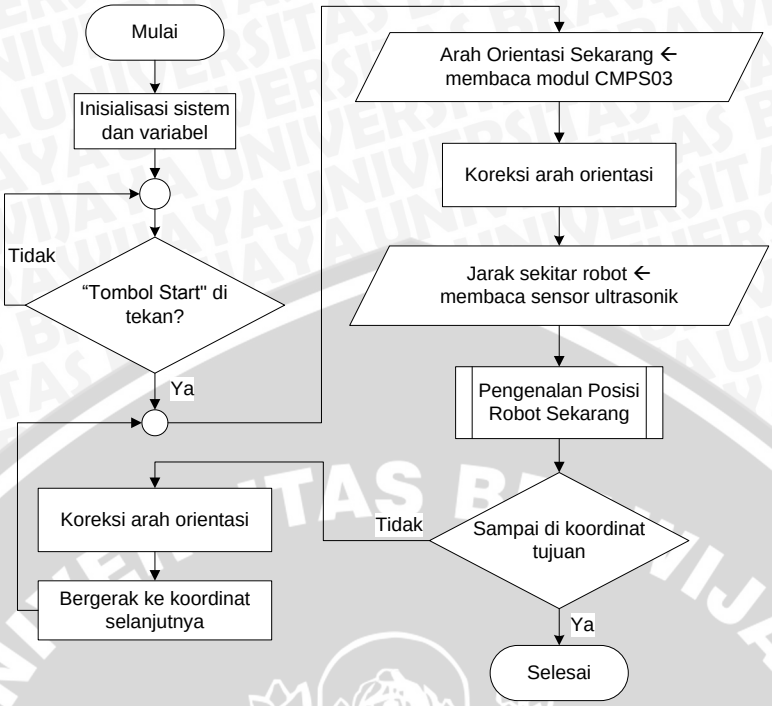
Perancangan dan pembuatan perangkat keras ini terdiri atas dua bagian yakni perancangan mekanik dan perancangan elektronik. Perancangan mekanik ditekankan pada dimensi robot, pemasangan modul CMPS03 *Magnetic Compass*

dan pemasangan sensor ultrasonik PING))). Bentuk mekanik secara umum dirancang dengan menggunakan perangkat lunak 3Ds MAX 2009.

Perancangan elektronik merupakan perancangan rangkaian antarmuka antara pemroses dengan modul-modul yang digunakan. Perancangan elektronik meliputi perancangan rangkaian antarmuka modul LCD Karakter, perancangan rangkaian antarmuka modul CMPS03 *Magnetic Compass*, perancangan rangkaian mikrokontroler pengatur sensor ultrasonik PING))), perancangan rangkaian mikrokontroler pengatur *driver* motor DC dan perancangan rangkaian mikrokontroler pemroses utama. Papan rangkaian tercetak (PCB) dirancang dengan menggunakan perangkat lunak EAGLE (*Easily Applicable Graphical Layout Editor*) versi 5.11.0.

3.3.3. Perancangan dan Pembuatan Perangkat Lunak (*Software*)

Setelah melalui proses perancangan dan pembuatan perangkat keras serta perancangan sistem pengenalan posisi, selanjutnya akan dilakukan perancangan dan pembuatan perangkat lunak. Perangkat lunak berfungsi untuk memberikan instruksi kerja kepada perangkat keras serta mengolah data yang diperoleh dari sensor untuk dapat digunakan sesuai dengan keperluan penelitian. Perangkat lunak dirancang melalui pembuatan diagram alir (*flowchart*) subsistem hingga sistem pengenalan posisi secara keseluruhan. Dalam penelitian ini, fungsi utama yang ditanamkan pada mikrokontroler pemroses utama (ATmega128) ditunjukkan dalam Gambar 3.3. Kemudian dilakukan penulisan program menggunakan *compiler* bahasa pemrograman C CVAVR (Code Vision AVR C *Compiler*) versi 2.05.0.



Gambar 3.3 Diagram Alir Fungsi Utama

Proses diawali dengan inisialisasi sistem mikrokontroler dan variable yang digunakan, kemudian menunggu perintah untuk melakukan *start*. Setelah ada perintah untuk *start*, robot akan melakukan pembacaan data sensor kompas dan melakukan koreksi arah orientasi robot. Selanjutnya robot akan melakukan pengukuran jarak bagian depan, belakang, samping kanan dan samping kiri robot. Data hasil pengukuran ini kemudian dibandingkan dengan dengan data pada peta yang sudah disimpan sebelumnya. Berdasarkan hasil pembandingan tersebut robot dapat mengetahui posisi keberadaan robot pada arena pertandingan. Jika koordinat sekarang adalah posisi tujuan robot maka misi dianggap selesai, jika tidak maka robot akan mengubah arah orientasi dan bergerak ke koordinat selanjutnya yang harus dilewati robot sampai robot dapat mencapai posisi tujuan.

3.4. Pengujian dan Analisis

Untuk mengetahui kinerja alat apakah telah sesuai dengan perancangan, maka dilakukan pengujian alat. Pengujian dilakukan secara bertahap yaitu pada masing-masing subsistem terlebih dahulu, kemudian dilanjutkan dengan

pengujian sistem secara keseluruhan. Setelah proses pengujian selesai dilanjutkan dengan proses analisis.

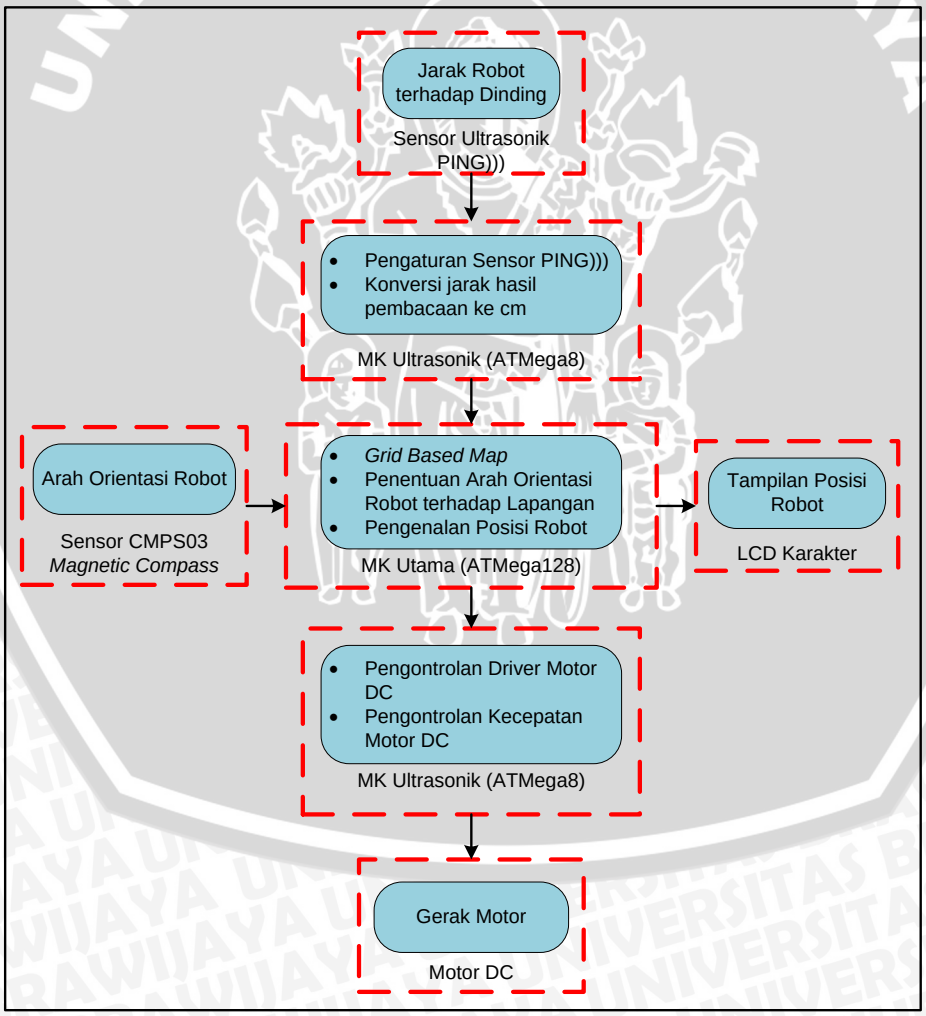
Proses analisis dilakukan setelah diperoleh data hasil pengujian alat. Data-data hasil pengujian dianalisis kemudian diambil kesimpulan dari masing-masing hasil pengujian yang dilakukan.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Perancangan dan pembuatan alat dilakukan secara bertahap dalam bentuk blok sehingga memudahkan dalam analisis. Perancangan alat terdiri atas perancangan sistem pengenalan posisi, perancangan perangkat keras dan perancangan perangkat lunak.

4.1. Perancangan Diagram Blok Sistem

Perancangan alat diawali dengan pembuatan diagram blok sistem pengenalan posisi. Diagram blok sistem pengenalan posisi ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Diagram Blok Sistem Pengenalan Posisi

Fungsi masing-masing bagian dalam diagram blok sistem tersebut adalah sebagai berikut:

1) Sensor ultrasonik PING))) berfungsi sebagai pendeteksi jarak antara robot dengan dinding yang ada si sisi depan, belakang, samping kiri dan samping kanan.
2) Mikrokontroler PING))) berupa ATmega8 berfungsi untuk mengatur proses penjadwalan pembacaan sensor PING))) agar tidak terjadi saling interferensi antara sensor dan mengirimkan data hasil pembacaan sensor ke mikrokontroler ATmega128 sebagai pemroses utama.
3) Modul CMPS03 *Magnetic Compass* berfungsi sebagai kompas digital yang dapat mengetahui arah orientasi robot dan sebagai umpan balik sudut koreksi robot serta serta pengaturan konfigurasi orientasi sensor relatif .
4) LCD *Alphanumeric* 4×20 berfungsi sebagai penampil.
5) Mikrokontroler ATmega128 berfungsi sebagai pengolah data sensor serta kendali utama.
6) Mikrokontroler ATmega8 berfungsi sebagai pemroses sinyal kendali motor dari mikrokontroler ATmega128 menuju ke driver motor kanan/kiri.
7) *Driver* motor berfungsi sebagai antarmuka mikrokontroler (*slave*) pengatur *driver* motor (ATmega8) dengan motor DC.
8) Motor DC berfungsi sebagai aktuator atau penggerak utama robot.

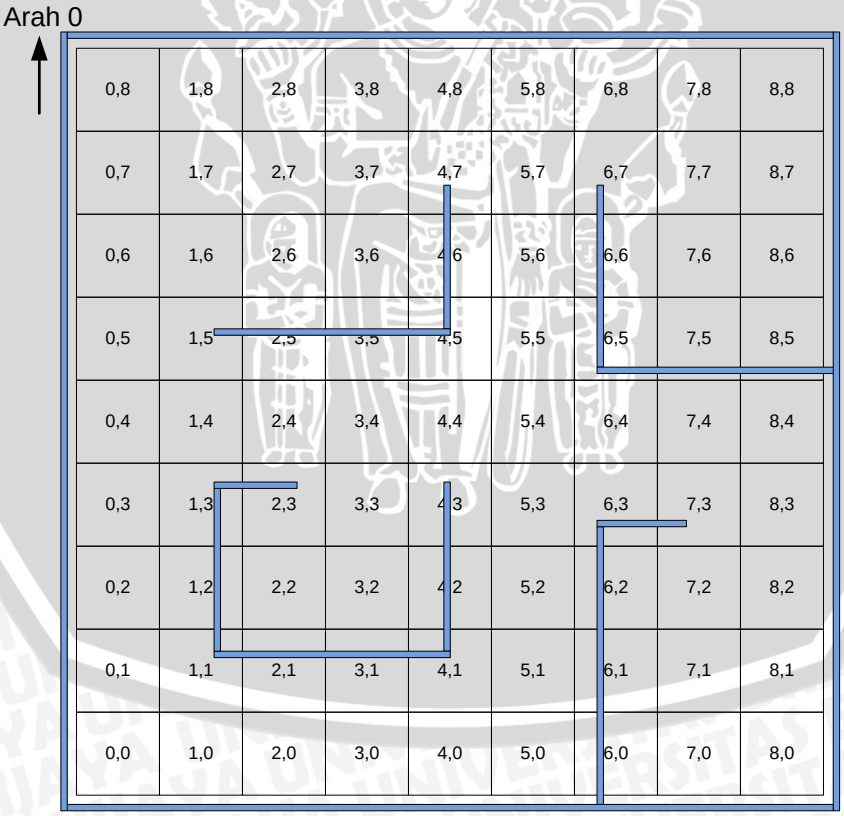
Prinsip kerja sistem ini adalah, awalnya peta/*map* arena pertandingan yang telah dibuat dimasukkan ke dalam memori robot. Ketika robot telah diaktifkan maka robot akan melakukan inisialisasi sistem serta persiapan untuk menjalankan misi. Misi utama robot adalah untuk bergerak dari posisi/koordinat *start* menuju koordinat tujuan yang terdapat dalam arena pertandingan di mana koordinat yang akan dilewati robot untuk mencapai posisi tujuan sudah ditentukan terlebih dahulu. Untuk mengetahui di mana posisi robot *start*, robot akan mengambil data dari sensor PING))) tentang lingkungan sekitar robot, kemudian data tersebut dibandingkan dengan peta yang telah ditanamkan ke dalam memori

robot sebelumnya. Setelah mengetahui posisi *start*, maka robot akan mulai bernavigasi untuk mencapai misi yang diinginkan.

4.2. Perancangan Sistem Pengenalan Posisi

4.2.1. Perancangan *Grid-Based Map*

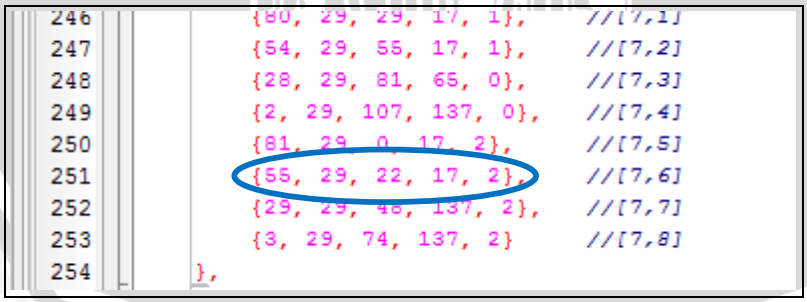
Pada perancangan ini, arena pertandingan KRPAI akan direpresentasikan ke dalam peta koordinat blok/*grid-based map*. Ukuran dari masing-masing blok/*grid* disesuaikan dengan dimensi robot yaitu 26 cm × 26 cm, sehingga peta tersebut akan terdiri dari blok/*grid* dengan jumlah 9 × 9. Peta tersebut kemudian diubah menjadi *array* yang membentuk blok/*grid* di mana masing-masing blok/*grid* tersebut berisi data-data mengenai keadaan lingkungan di sekitar robot. Data-data tersebut berupa data jarak robot terhadap dinding di bagian depan, belakang, sisi kanan dan sisi kiri serta apakah robot berada di dalam ruangan atau berada di lorong.



Gambar 4.2 Rancangan Peta Arena Pertandingan KRPAI

Untuk data-data mengenai jarak bagian depan, belakang, kiri dan kanan, data yang terdapat pada peta di memori merupakan data jarak pengukuran ketika robot diletakkan di titik tengah *grid*/blok dengan orientasi robot menghadap arah 0. Sedangkan untuk data mengenai kondisi robot berada di dalam ruangan atau di lorong, digunakan indeks 0 sampai 4 sebagai penanda. Indeks 0 digunakan ketika robot berada di lorong, indeks 1 ketika robot berada di ruang 1, indeks 2 ketika robot berada di ruang 2, indeks 3 ketika robot berada di ruang 3 dan indeks 4 ketika robot berada di ruang 4.

Berikut ini adalah contoh dari data peta untuk koordinat 7,6 yang berupa *array* berisi data mengenai jarak dan kondisi keberadaan robot pada arah orientasi 0 : {55, 29, 22, 17, 2}. Data 55 merepresentasikan jarak bagian depan robot, data 29 merepresentasikan jarak samping kanan robot, data 22 merepresentasikan jarak belakang robot, data 17 merepresentasikan jarak samping kiri robot, dan data 2 merepresentasikan kondisi robot berada di ruang 2. Jika suatu koordinat memiliki lebih dari satu kondisi pengukuran dikarenakan posisi dinding, maka peta koordinat tersebut akan berisi lebih dari satu buah *array*. Sebagai contoh adalah koordinat 7,0 di mana koordinat tersebut memiliki lebih dari satu kemungkinan pengukuran untuk bagian depan, maka data peta untuk koordinat 7,0 adalah sebagai berikut: {106, 29, 3, 17, 1}, {58, 29, 3, 17, 1}. Data {106, 29, 3, 17, 1} merupakan kemungkinan kondisi pertama dan data {58, 29, 3, 17, 1} merupakan kemungkinan kondisi kedua.



Gambar 4.3 Contoh Peta berupa *Array* Untuk Posisi dengan Satu Kemungkinan

```
539 };
540
541 unsigned char petaxy_70[2][5] =
542 {
543     {106, 29, 3, 17, 1},
544     {58, 29, 3, 17, 1}
545 };
546
```

Gambar 4.4 Contoh Peta berupa *Array* Untuk Posisi dengan lebih dari Satu Kemungkinan

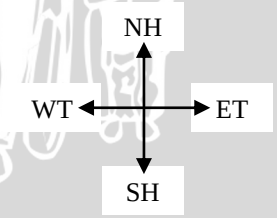
4.2.2. Variabel Masukan dan Keluaran

Variabel masukan yang akan dijadikan sebagai acuan untuk mengatahui posisi robot terhadap peta yang telah ditanamkan di dalam memori robot ada lima yaitu jarak dari pembacaan sensor ultrasonik yang berjumlah empat buah dan arah orientasi robot (sekarang) terhadap arena pertandingan, sedangkan variabel keluaran berupa posisi sekarang dan kemungkinan posisi selanjutnya.

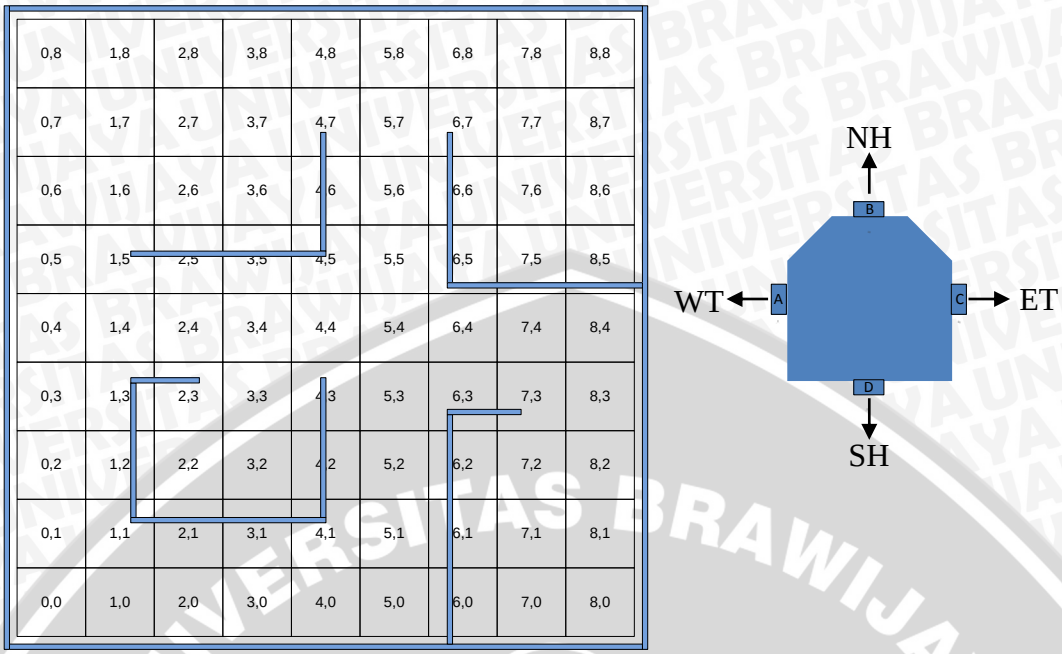
4.2.3. Perancangan Arah Orientasi Robot

Ketika robot akan bergerak ke koordinat selanjutnya maka robot perlu untuk menghadap ke arah yang dituju. Selain itu, pada saat proses pembandingan data antara data hasil pembacaan sensor ultrasonik dengan data yang ada pada peta, diperlukan referensi arah agar data yang dibandingkan sesuai. Untuk itu dirancanglah arah orientasi robot pada peneltian ini. Ada empat arah yang akan digunakan, yaitu NH (*North*), SH (*South*), ET (*East*) dan WT (*West*).

NH (*North*) : Arah 0
ET (*East*) : Arah 90
SH (*South*) : Arah 180
WT (*West*). : Arah 270



Untuk menyamakan persepsi arah, maka arah-arah tersebut ditetapkan terhadap posisi arena pada saat itu. Gambar 4.5 menunjukkan arah hadap robot berdasarkan arena KRPAI divisi beroda.



Gambar 4.5 Arah Hadap Robot Berdasarkan Arena KRPAI Divisi Beroda

Berdasarkan konfigurasi arah di atas, maka disusunlah Tabel 4.1 yang menunjukkan hubungan antara arah orientasi robot dengan orientasi sensor ultrasonik relatif bagian depan, belakang, kanan dan kiri robot.

Tabel 4.1 Hubungan Orientasi Robot dan Orientasi Sensor Relatif

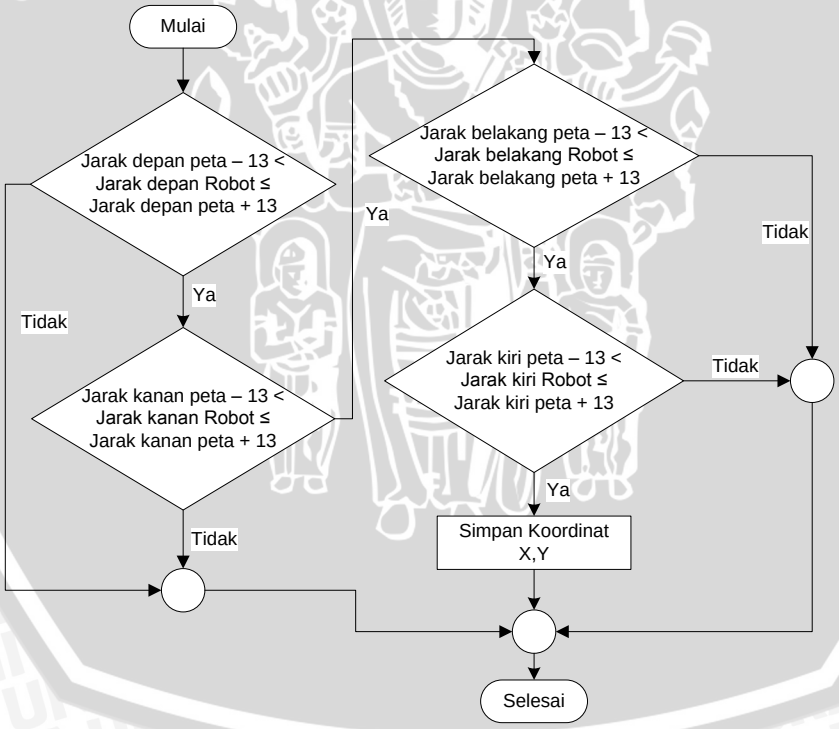
Orientasi Robot	**Orientasi Sensor Relatif**			
	Depan	**Belakang**	**Kiri**	**Kanan**
NH	Sensor B	Sensor D	Sensor A	Sensor C
ET	Sensor A	Sensor C	Sensor D	Sensor B
SH	Sensor D	Sensor B	Sensor C	Sensor A
WT	Sensor C	Sensor A	Sensor B	Sensor D

Sebagai contoh, ketika robot berada pada koordinat 5,2 dan arah orientasi robot menghadap ke arah ET (*East*), maka urutan data sensor yang akan dibandingkan dengan data yang berada pada peta yang telah disimpan di dalam memori mikrokontroler adalah data sensor A dengan data depan, data sensor B dengan data samping kanan, data sensor C dengan data belakang dan data sensor D dengan data samping kiri.

Posisi *start* robot pada arena pertandingan disesuaikan dengan keadaan ketika lomba. Saat robot melakukan *start* di lorong, maka posisi *start* robot berada

pada koordinat 5,0, sedangkan ketika robot melakukan *start* di dalam ruangan, maka posisi *start* robot dapat berada pada salah satu koordinat yang terdapat pada ruangan tersebut.

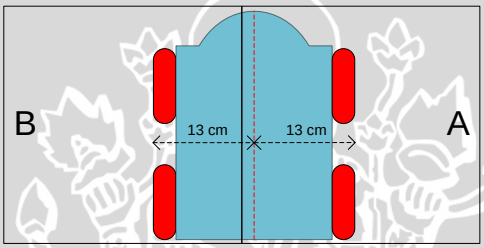
Untuk dapat mengetahui posisi robot di arena pertandingan, robot membutuhkan data berupa arah orientasi robot terhadap arena pertandingan dan data jarak robot bagian depan, belakang, samping kiri dan samping kanan robot. Data mengenai arah orientasi robot diperoleh dari data sensor CMPS03 *magnetic compass* sedangkan data jarak diperoleh dari hasil pembacaan sensor ultrasonik. Berdasarkan data arah orientasi robot, maka dapat ditentukan orientasi sensor ultrasonik relatif di mana data sensor ultrasonik berdasarkan urutan tersebut akan dibandingkan dengan data yang terdapat pada peta. Diagram alir proses pembandingan antara data hasil pembacaan sensor ultrasonik PING))) terhadap lingkungan di sekitar robot dengan data yang ada di peta secara umum ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Diagram Alir Proses Pembandingan Data

Data hasil pembacaan sensor ultrasonik bagian depan, belakang, kiri dan kanan robot dibandingkan dengan data peta yang sudah dimasukkan sebelumnya di dalam memori robot. Caranya adalah dengan mencocokkan satu per satu data

hasil pembacaan sensor PING))) bagian depan, belakang, kiri dan kanan yang didapat dari pengukuran pada posisi robot sekarang dengan data peta yang merepresentasikan jarak depan, belakang, kiri dan kanan koordinat tertentu yang terdapat pada memori robot. Dalam proses pencocokan ini, data hasil pembacaan sensor ultrasonik dibandingkan dengan data pada peta ± 13 cm. Maksud dari ± 13 cm ini adalah ketika badan robot berada pada dua koordinat berbeda misalnya koordinat A dan B, selama setengah dari badan robot masih berada pada koordinat A, maka robot akan dianggap berada pada koordinat A walaupun sudah ada badan robot yang masuk ke koordinat B. Proses ini dilakukan sampai robot menemukan koordinat dengan data yang cocok dengan data keadaan lingkungan robot sekarang. Ilustrasi keadaan robot yang berada pada koordinat berbeda dalam waktu bersamaan ditunjukkan dalam Gambar 4.7.



Gambar 4.7 Ilustrasi Keadaan Robot Dalam Dua Koordinat Berbeda

4.3. Perancangan Perangkat Keras (*Hardware*)

Perancangan dan pembuatan perangkat keras terdiri atas perancangan mekanik serta perancangan elektronik. Perancangan mekanik terdiri atas perancangan bentuk mekanik robot. Sedangakan perancangan elektronik terdiri atas perancangan rangkaian antarmuka modul *magnetic compass*, perancangan rangkaian antarmuka modul LCD karakter, perancangan rangkaian mikrokontroler pengatur *driver* motor DC, perancangan rangkaian mikrokontroler pengatur sensor ultrasonik PING))) dan perancangan rangkaian mikrokontroler pemroses utama.

4.3.1. Perancangan Mekanik Robot dan Pemasangan Sensor

Berdasarkan peraturan Kontes Robot Pemadam Api Indonesia (KRPAI) Divisi Beroda tahun 2013, batasan dimensi robot ketika robot dalam keadaan

berhenti, berjalan, bermanuver maupun saat robot mematikan lilin adalah sebagai berikut:

- Panjang maksimum : 31 cm
- Lebar maksimum : 31 cm
- Tinggi maksimum : 27 cm

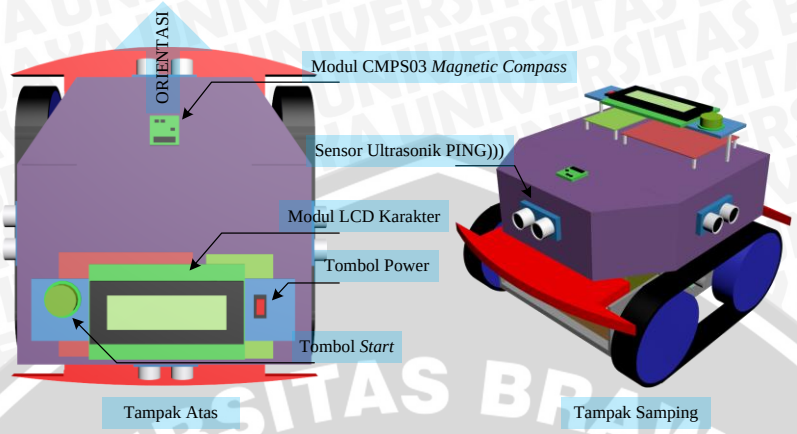
Dengan adanya batasan dimensi tersebut, maka desain mekanik robot beroda dirancang agar tidak melebihi batasan yang telah ditetapkan. Desain mekanik robot pada penelitian ini dirancang dengan dimensi panjang × lebar × tinggi sebasar 26 cm × 26 cm × 25 cm.

Selain perancangan mekanik robot, perancangan pemasangan sensor juga perlu dilakukan. Hal ini dimaksudkan agar sensor yang digunakan dapat berfungsi dengan baik. Sensor kompas dalam skripsi ini pemasangannya perlu ditentukan sedemikian rupa sesuai dengan prinsip kerjanya. Dalam skripsi ini digunakan modul *magnetic compass* sebagai sensor kompas. Modul tersebut memiliki karakteristik mudah terpengaruh oleh medan magnet di sekelilingnya. Adanya medan magnet selain medan magnet tersebut, dapat menyebabkan kesalahan dalam pembacaan arah sebenarnya. Untuk meminimalisir gangguan tersebut, maka perlu dilakukan beberapa hal, antara lain penggunaan bahan selain logam pada daerah pemasangan sensor kompas, pemasangan sensor yang tidak boleh terlalu dekat dengan motor dan orientasi pemasangan sensor kompas harus sesuai dengan orientasi robot. Orientasi modul *magnetic compass* tampak atas ditunjukkan dalam Gambar 4.8.

Gambar 4.8 Orientasi Modul CMPS03 *Magnetic Compass*

Untuk pemasangan sensor ultrasonik, posisi sensor diletakkan segaris dengan titik tengan robot. dalam perancangan ini digunakan empat buah sensor ultrasonik yang diletakkan di bagian samping kiri (Sensor A), depan (Sensor B),

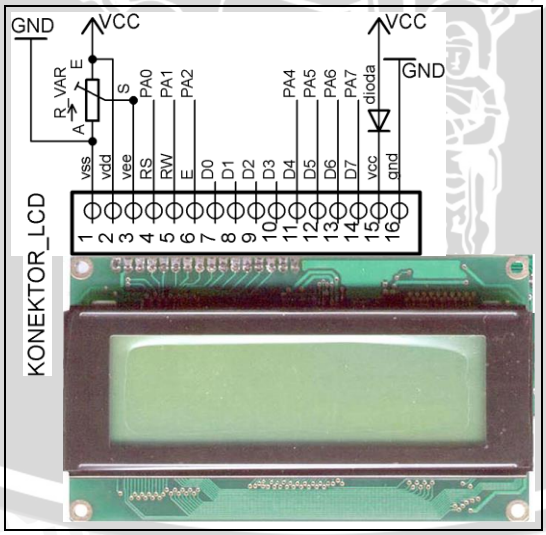
samping kanan (Sensor C) dan belakang (Sensor D). Perancangan bentuk mekanik dan pemasangan sensor ditunjukkan dalam Gambar 4.9.



Gambar 4.9 Bentuk Mekanik dan Pemasangan Sensor

4.3.2. Perancangan Rangkaian Antarmuka Modul LCD Karakter

Dalam sistem ini, modul LCD karakter akan diakses langasung oleh mikrokontroler pemroses utama (ATmega128) sebagai perangkat penampil. Agar dapat diakses oleh mikrokontroler pemroses utama, maka di dalam sistem ini dirancang rangkaian antarmuka modul LCD karakter. Rangkaian antarmuka modul LCD karakter ditunjukkan dalam Gambar 4.10.



Gambar 4.10 Perancangan Rangkaian Antarmuka Modul LCD Karakter

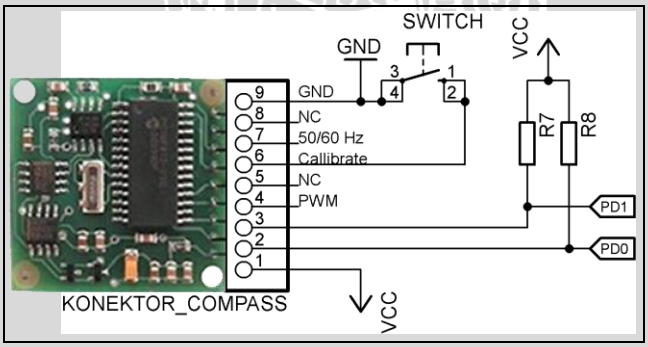
Pada rangkaian antarmuka modul LCD karakater, terdapat tujuh pin penting yang dihubungkan ke mikrokontroler pemroses utama yaitu RS (*Register*

Select) ke PA0, RW (*Read/Write*) ke PA1, E (*Enable*) ke PA2, dan jalur data D4-D7 ke PA4-PA7.

Pin ke-3 dari modul LCD karakter digunakan untuk mengatur tingkat ketajaman karakter yang ditampilkan oleh LCD. Untuk memudahkan dalam mengatur tingkat ketajaman tersebut, maka pada pin ke-3 ini dipasang variabel resistor dengan nilai 10 kΩ. Sedangkan pin ke-15 digunakan untuk mengatur *backlight* LCD karakter. Pada kaki tersebut dipasang dioda IN4002 untuk membatasi tegangan yang masuk agar tegangan tersebut sesuai dengan rentang nilai tegangan *backlight* yang diizinkan yakni berkisar antara 4-4,3 V DC.

4.3.3. Perancangan Rangkaian Antarmuka Modul CMPS03 *Magnetic Compass*

Dalam sistem ini, modul CMPS03 *magnetic compass* akan diakses secara langsung oleh mikrokontroler pemroses utama (ATmega128) sebagai perangkat masukan. Modul *magnetic compass* akan diakses oleh mikrokontroler pemroses utama dengan menggunakan jalur I_2C. Untuk dapat diakses oleh mikrokontroler pemroses utama melalui jalur I_2C, maka di dalam sistem ini dirancang rangkaian antarmuka modul *magnetic compass*. Rangkaian antarmuka modul *magnetic compass* ditunjukkan dalam Gambar 4.11.



Gambar 4.11 Perancangan Rangkaian Antarmuka Modul *Magnetic Compass*

Pada rangkaian antarmuka ini, terdapat dua pin modul *magnetic compass* yang harus dihubungkan ke mikrokontroler pemroses utama, yaitu pin SCL dan SDA. Pin SCL dan SDA dihubungkan ke PD0 dan PD1 pada mikrokontroler pemroses utama. Dari *datasheet* mikrokontroler pemroses utama (ATmega128), diketahui $V_{OL(TWI)}$ = 0,4 volt dan (I_{OL}) *sink current* = 3 mA, sehingga dibutuhkan

pengondisian dengan memasang resistor *pull-up* pada kedua pin tersebut. Nilai minimal resistor *pull-up* dapat ditentukan melalui perhitungan sebagai berikut:

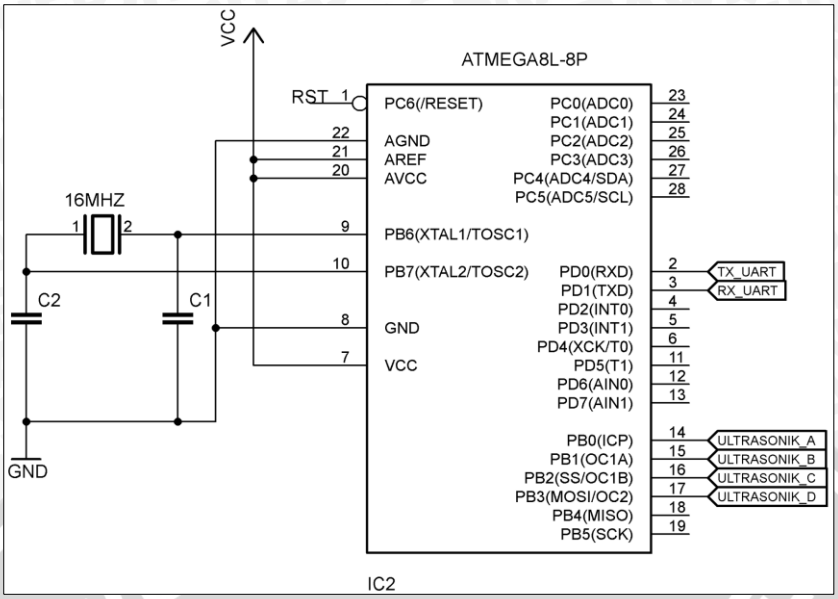
$$R_{pull-up_{min}} = \frac{Vcc - 0{,}4}{3mA} \quad \text{...(4.1)}$$

$$= \frac{5 - 0{,}4}{3mA} = 1533{,}33\ \Omega = 1{,}533\ k\Omega$$

Dari perhitungan tersebut diperoleh nilai resistor *pull-up* minimal sebesar 1,533 kΩ, sehingga dalam perancangan ini digunakan resistor 1,8 kΩ. Kalibrasi modul *magnetic compass* dapat dilakukan dengan menghubungkan pin ke 6 (*callibrate*) modul *magnetic compass* ke GND (*ground*). Untuk mempermudah proses kalibrasi, maka dipasang *push button* di antara pin ke 6 dengan pin ke 9. Sehingga ketika *push button* ditekan, pin ke 6 (*callibrate*) akan terhubung ke pin 9 (*ground*).

4.3.4. Perancangan Rangkaian Mikrokontroler Pengatur Sensor Ultrasonik PING)))

Dalam sistem ini, digunakan empat buah sensor ultrasonik PING))) yang digunakan untuk mengukur jarak. Sensor ini memancarkan gelombang ultrasonik dan menerima kembali pantulan gelombang ultrasonik yang dipancarkan untuk mengetahui jarak tertentu. Agar tidak terjadi interverensi antara gelombang ultrasonik yang dipancarkan oleh masing-masing sensor ultrasonik, maka diperlukan rangkaian mikrokontroler untuk mengatr sensor ultrasonik tersebut. Rangkaian mikrokontroler ini nantinya akan mengirimkan data hasil pembacaan sensor ultrasonik melalui jalur komunikasi UART. Rangkaian mikrokontroler pengatur sensor ultrasonik PING))) ditunjukkan dalam Gambar 4.12.



Gambar 4.12 Rangkaian Mikrokontroler Pengatur Sensor Ultrasonik PING)))

Pada rangkaian ini, digunakan mikrokontroler ATmega8 sebagai mikrokontroler pengatur sensor ultrasonik PING))). Mikrokontroler ATmega8 mempunyai 3 port, dengan 21 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. Pada perancangan ini, pin yang yang digunakan adalah:

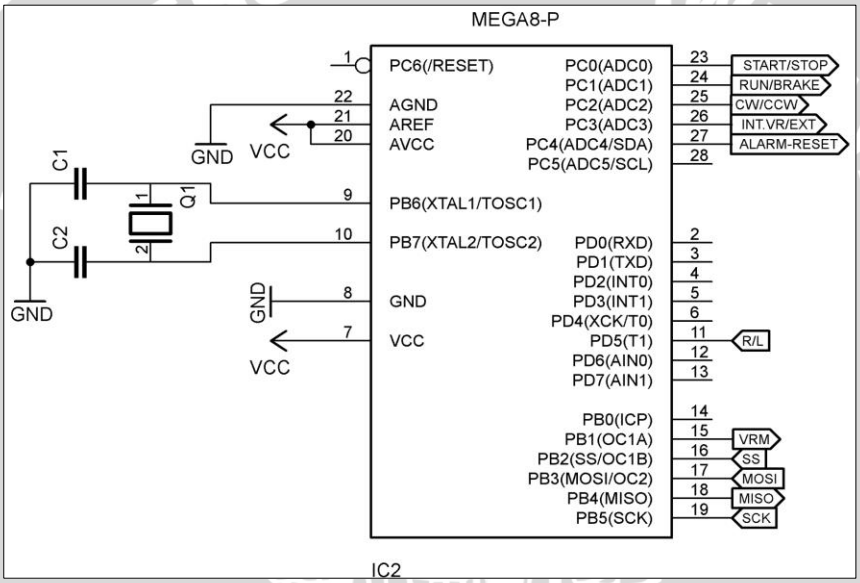
Pin B.0 = digunakan untuk antarmuka sensor ultrasonik A
Pin B.1 = digunakan untuk antarmuka sensor ultrasonik B
Pin B.2 = digunakan untuk antarmuka sensor ultrasonik C
Pin B.3 = digunakan untuk antarmuka sensor ultrasonik D
Pin B.6 = dihubungkan dengan kaki osilator 16 MHz
Pin B.7 = dihubungkan dengan kaki osilator 16 MHz
Pin D.0 – D.1 = Tx dan Rx untuk transmisi data UART

4.3.5. Perancangan Rangkaian Mikrokontroler Pengatur *Driver* Motor DC

Dalam penelitian ini, *mobile robot* yang digunakan memiliki dua buah motor DC yang masing-masing dikendalikan oleh *driver* motor. Untuk mengendalikan arah putaran motor DC, *driver* motor membutuhkan masukan pada pin START/STOP, RUN/BRAKE, CW/CCW, INT.VR/EXT.VR dan ALARM-RESET berupa logika H (*high*) atau L (*low*). Sedangkan untuk

mengendalikan kecepatan putaran motor DC, *driver* motor membutuhkan masukan pada pin VRM berupa PWM (*Pulse Width Modulation*).

Untuk menyederhanakan proses yang dilakukan di dalam mikrokontroler utama, dirancang rangkaian pengatur *driver* motor DC yang berfungsi khusus untuk mengatur masukan-masukan yang dibutuhkan oleh *driver* motor DC. Mikrokontroler yang digunakan untuk mengontrol *driver* motor DC adalah mikrokontroler ATmega8 yang berkedudukan sebagai *slave*, sedangkan mikrokontroler pemroses utama (ATmega128) berkedudukan sebagai *master*. Rangkaian mikrokotroler pengatur *driver* motor DC ditunjukkan dalam Gambar 4.13.



Gambar 4.13 Perancangan Rangakaian Mikrokontroler Pengatur *Driver* Motor DC

Mikrokontroler ATmega8 mempunyai 3 port, dengan 21 jalur I/O yang dapat diprogram menjadi masukan atau keluaran. Pada perancangan ini, pin yang yang digunakan adalah:

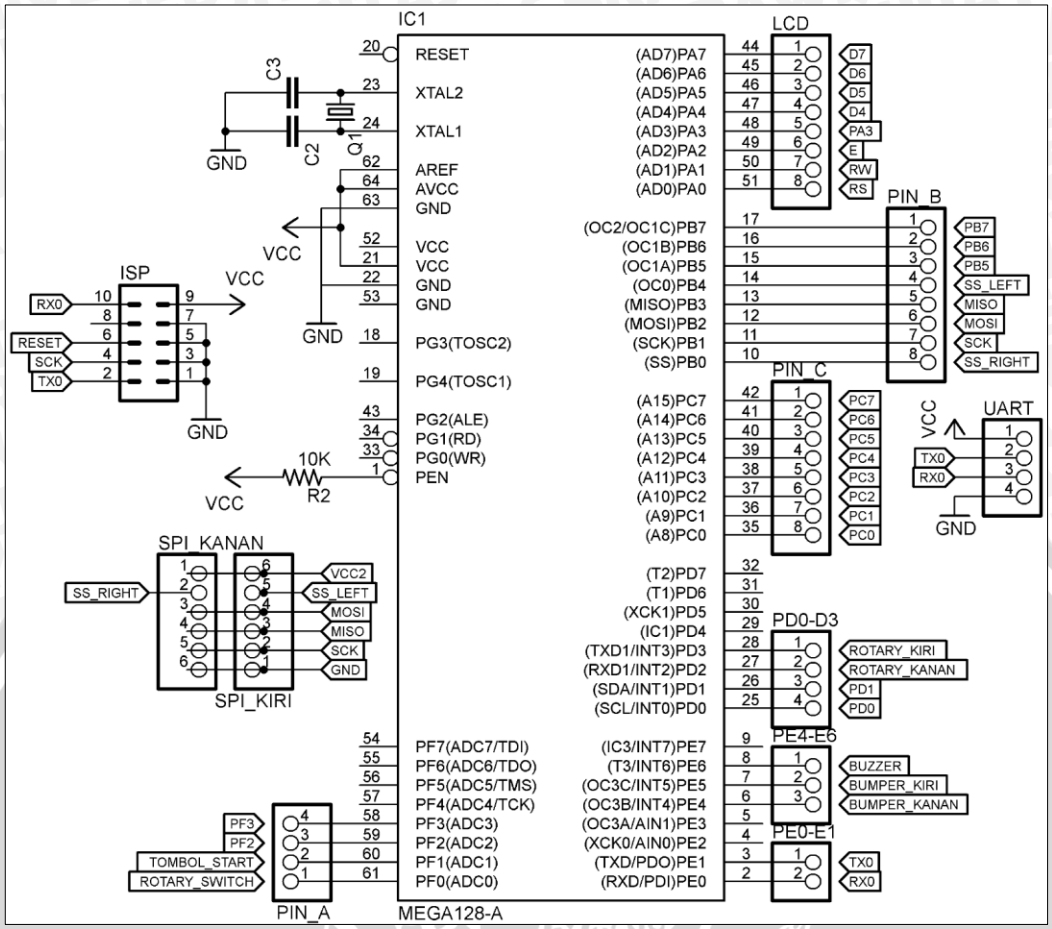
Pin C.0 = dihubungkan dengan pin START/STOP pada modul *driver* motor
Pin C.1 = dihubungkan dengan pin RUN/BRAKE pada modul *driver* motor
Pin C.2 = dihubungkan dengan pin CW/CCW pada modul *driver* motor
Pin C.3 = dihubungkan dengan pin INT.VR/EXT.VR pada modul *driver* motor
Pin C.4 = dihubungkan dengan pin ALARM-RESET pada modul *driver* motor
Pin B.1 = dihubungkan dengan pin VRM pada modul *driver* motor

Pin B.2 = digunakan sebagai jalur SPI dengan mikrokontroler utama (*SS-SPI*)
Pin B.3 = digunakan sebagai jalur SPI dengan mikrokontroler utama (*MOSI-SPI*)
Pin B.4 = digunakan sebagai jalur SPI dengan mikrokontroler utama (*MISO-SPI*)
Pin B.5 = digunakan sebagai jalur SPI dengan mikrokontroler utama (*SCK-SPI*)
Pin B.6 = dihubungkan dengan kaki osilator 16 MHz
Pin B.7 = dihubungkan dengan kaki osilator 16 MHz
Pin D.5 = digunakan sebagai masukan untuk menentukan mikrokontroler pengendali motor DC kanan atau kiri

Mikrokontroler pengatur *driver* motor DC (ATmega8) berkomunikasi dengan mikrokontroler pemroses utama (ATmega128) melaui jalur SPI (*Serial Peripheral Interface*). Terdapat 4 jalur penting dalam komunikasi dengan menggunakan jalur SPI yaitu jalur SS, MOSI, MISO dan SCK. Jalur SS, MOSI, MISO dan SCK masing-masing menghubungkan pin SS, MOSI, MISO dan SCK pada mikrokontroler pemroses utama dengan pin PB2 (SS), PB3 (MOSI), PB4 (MISO), dan PB5 (SCK) pada mikrokontroler pengatur *driver* motor DC.

4.3.6. Perancangan Rangkaian Mikrokontroler Pemroses Utama

Mikrokontroler yang digunakan sebagai mikrokontroler pemroses utama dalam sistem ini adalah sebuah mikrokontroler ATmega128. Fungsi dari mikrokontroler pemroses utama ini adalah sebagai pemroses data-data dari sensor (CMPS03 *magnetic compass* dan sensor ultrasonik PING))), mengatur pergerakan motor melalui pengiriman perintah kepada mikrokontroler pengatur *driver* motor DC, serta mengatur keluaran LCD karakter dan *buzzer*. Rangkaian mikrokontroler pemroses utama ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Rangkaian Mikrokontroler Pemroses Utama

Perancangan rangkaian mikrokontroler pemroses utama disesuaikan dengan jenis pengaturan komunikasi dengan perangkat keras lainnya. Untuk berkomunikasi dengan mikrokontroler pengatur sensor ultrasonik maka digunakan jenis komunikasi serial UART. Untuk berkomunikasi dengan modul CMPS03 *magnetic compass* maka digunakan jenis komunikasi serial I_2C. Untuk berkomunikasi dengan mikrokontroler pengatur *driver* motor DC maka digunakan jenis komunikasi serial SPI. Dan untuk berkomunikasi dengan modul LCD karakter maka digunakan jenis komunikasi paralel.

Mikrokontroler ATmega128 memiliki 7 *port* dengan 53 jalur yang dapat diprogram menjadi *input* atau *output* yaitu Port A, Port B, Port C, Port D, Port E, Port F, dan Port G. Pada perancangan ini, pin-pin yang digunakan adalah sebagai berikut:

Pin A.0 = digunakan sebagai antarmuka dengan LCD sebagai pin RS

Pin A.1 = digunakan sebagai antarmuka dengan LCD sebagai pin RW
Pin A.2 = digunakan sebagai antarmuka dengan LCD sebagai pin E
Pin A.4 = digunakan sebagai antarmuka dengan LCD sebagai pin D4
Pin A.5 = digunakan sebagai antarmuka dengan LCD sebagai pin D5
Pin A.6 = digunakan sebagai antarmuka dengan LCD sebagai pin D6
Pin A.7 = digunakan sebagai antarmuka dengan LCD sebagai pin D7
Pin B.0 = digunakan sebagai jalur SPI dengan mikrokontroler pengatur *driver* motor DC kanan (*SS-SPI*)
Pin B.1 = digunakan sebagai jalur SPI dengan mikrokontroler pengatur *driver* motor DC (*SCK-SPI*)
Pin B.2 = digunakan sebagai jalur SPI dengan mikrokontroler pengatur *driver* motor DC (*MOSI-SPI*)
Pin B.3 = digunakan sebagai jalur SPI dengan mikrokontroler pengatur *driver* motor DC (*MISO-SPI*)
Pin B.4 = digunakan sebagai jalur SPI dengan mikrokontroler pengatur *driver* motor DC kiri (*SS-SPI*)
Pin D.2 = digunakan sebagai masukan sensor putaran roda kanan
Pin D.3 = digunakan sebagai masukan sensor putaran roda kiri
Pin E.0 = Rx untuk komunikasi UART dengan mikrokotroler pengatur sensor ultrasonik
Pin E.1 = Tx untuk komunikasi UART dengan mikrokotroler pengatur sensor ultrasonik
Pin E.6 = digunakan sebagai keluaran pengatur *buzzer*
Pin F.0 = dihubungkan dengan *rotary switch* sebagai saklar pilih atau saklar *mode*
Pin F.1 = dihubungkan dengan *push button* sebagai *manual start*
Pin XTAL1 = dihubungkan dengan kaki osilator kristal 16 MHz
Pin XTAL2 = dihubungkan dengan kaki osilator kristal 16 MHz

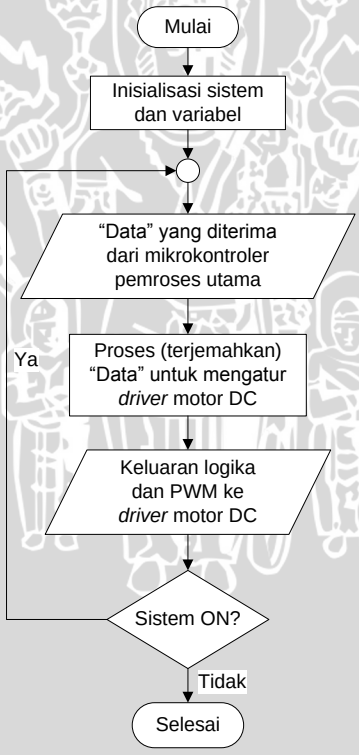
4.4. Perancangan Perangkat Lunak (*Software*)

Perancangan dan pembuatan perangkat lunak terdiri atas perancangan perangkat lunak mikrokontroler pengatur *driver* motor DC, perancangan

perangkat lunak mikrokontroler pengatur sensor ultrasonik PING))) dan perancangan perangkat lunak pemroses utama.

4.4.1. Perancangan Perangkat Lunak Mikrokontroler Pengatur *Driver* Motor DC

Mikrokontroler pemroses utama (ATmega128) memerintahkan mikrokontroler pengatur *driver* motor (ATmega8) untuk mengatur kerja modul *driver* motor dalam proses menggerakkan motor DC. Mikrokontroler pemroses utama tersebut mengirimkan perintah melalui jalur SPI. Untuk memproses (menerjemahkan) data yang diterima dari mikrokontroler pemroses utama, maka dalam penelitian ini dirancang perangkat lunak pengatur *driver* motor DC. Gambar 4.15 menunjukkan diagram alir perangkat lunak pengatur *driver* motor DC.



Gambar 4.15 Diagram Alir Perangkat Lunak Pengatur *Driver* Motor DC

Data yang diterima oleh mikrokontroler pengatur *driver* motor memiliki rentang nilai -128 hingga 127 dalam desimal. Data tersebut dirancang sehingga dapat merepresentasikan perintah yang diinginkan dalam mikrokontroler pemroses utama. Perintah berupa data tersebut diterima oleh mikrokontroler

pengatur *driver* motor, kemudian data tersebut diproses dan dijadikan sebagai acuan untuk mengatur kerja modul *driver* motor DC. Modul *driver* motor DC diatur oleh mikrokontroler pengatur *driver* motor melalui pemberian logika pada pin START/STOP, RUN/BRAKE, CW/CCW, INT.VR/EXT.VR, ALARM-RESET dan pemberian PWM pada pin VRM.

4.4.2. Perancangan Perangkat Lunak Mikrokontroler Pengatur Sensor Ultrasonik PING)))

Penggunaan sensor ultrasonik pada robot ini bertujuan untuk mengetahui jarak robot terhadap dinding bagian depan, belakang, kiri dan kanan robot. Sensor ultrasonik ini dikendalikan menggunakan mikrokontroler ATmega8. Penggunaan mikrokontroler ini bertujuan untuk menghindari kemungkinan terjadinya gangguan pada mikrokontroler pemroses utama akibat penambahan algoritma sensor ultrasonik. Sehingga dengan penggunaan mikrokontroler yang terpisah dari mikrokontroler utama, proses pengukuran jarak dengan sensor ultrasonik tidak akan terganggu oleh algoritma lain, di mana hal ini dapat terjadi jika pengendalian sensor ultrasonik juga dilakukan oleh mikrokontroler pemroses utama.

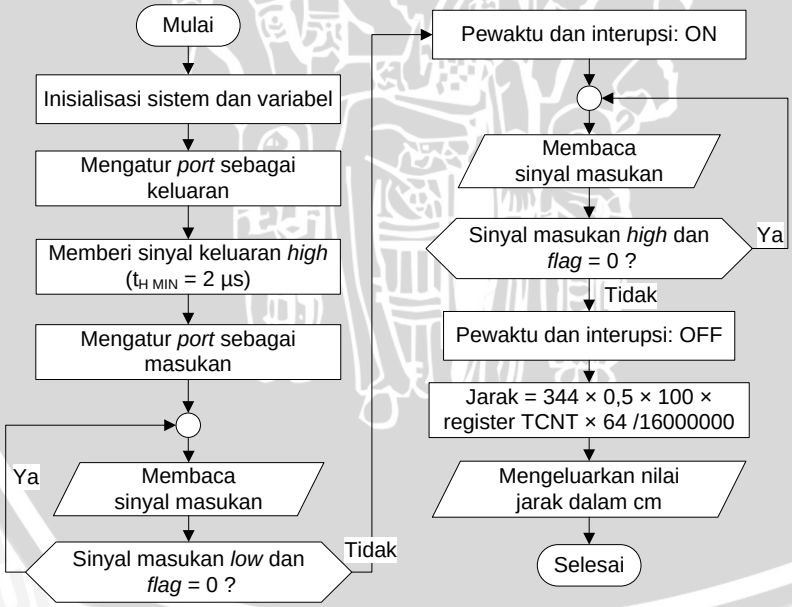
Akan tetapi, penggunaan sensor ultrasonik dalam jumlah banyak memungkinkan terjadinya gangguan *crosstalk* karena pantulan gelombang dari sensor satu ke sensor lainnya (Borenstein, 1995:1). Untuk menghindari terjadinya *crosstalk*, digunakan metode penjadwalan (*schedulling*) denga cara mengaktifkan sensor ultarsonik secara bergantian dengan selang waktu yang telah ditentukan. Selain itu, pada saat pengaktifan sensor, diusahakan agar tidak ada dua atau lebih sensor yang berdekatan aktif pada selang waktu yang dekat. Tabel 4.2 menunjukkan urutan aktivasi sensor ultrasonik.

Tabel 4.2 Urutan aktivasi sensor ultrasonik pada robot

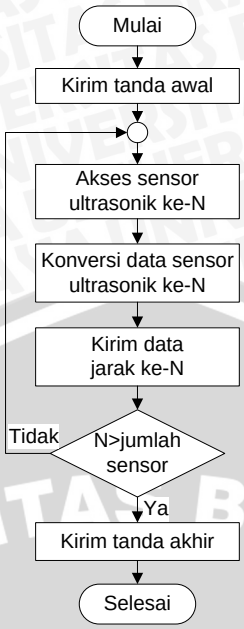
Sensor	Aktif (ms)
A	8
C	8
B	8
D	8
Total	32

Dimensi arena pertandingan seluas 244 cm × 244 cm dan dimensi lintasan yang memiliki lebar 46 cm menyababkan pantulan gelombang ultrasonik pada dinding lintasan sehingga menimbulkan kesalahan pengukuran jarak oleh sensor ultrasonik untuk jarak lebih dari 1 meter. Lama waktu aktif pada sensor sebesar 8 ms sebanding dengan jarak terbaca sebesar 137 cm. Ketika jarak yang diukur kurang dari 137 cm maka waktu aktif sensor kurang dari 8 ms. Sedangkan bila jarak yang diukur melewati 137 cm maka sensor selanjutnya akan aktif setelah melewati waktu aktif sensor sebelumnya dan data yang diproses adalah 137 cm.

Pengaturan sensor ultrasonik ini menggunakan *timer 1* pada mikrokontroler ATmega8. Data dari setiap sensor ultrasonik yang telah dikonversi menjadi besaran jarak dalam centimeter kemudian dikirim melalui komunikasi serial UART dengan *baudrate* 1 Mbps dari mikrokontroler pengatur sensor ultrasonik menuju mikrokontroler pemroses utama. Gambar 4.16 menunjukkan diagram alir program pembacaan sensor ultrasonik dan Gambar 4.17 menunjukkan diagram alir proses pengaturan sensor ultrasonik.



Gambar 4.16 Diagram Alir Proses Pembacaan Sensor Ultrasonik



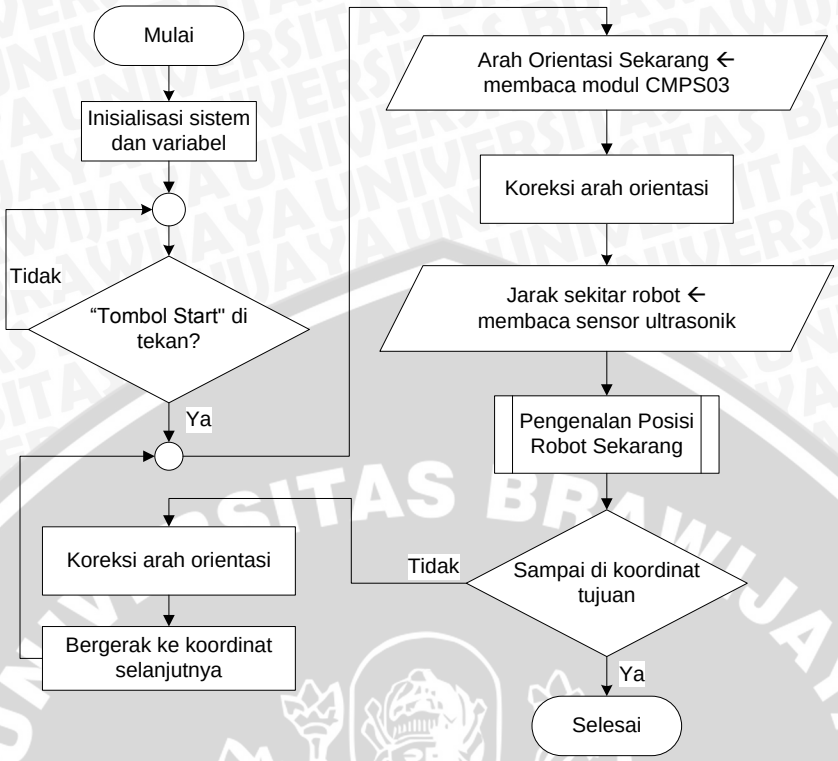
Gambar 4.17 Diagram Alir Proses Pengaturan Sensor Ultrasonik

4.4.3. Perancangan Perangkat Lunak Pemroses Utama

Perancangan perangkat lunak pemroses utama terdiri atas perancangan perangkat lunak fungsi utama sistem, perancangan perangkat lunak pemroses data modul CMPS03 *magnetic compass* dan perancangan perangkat lunak sistem pengenalan posisi.

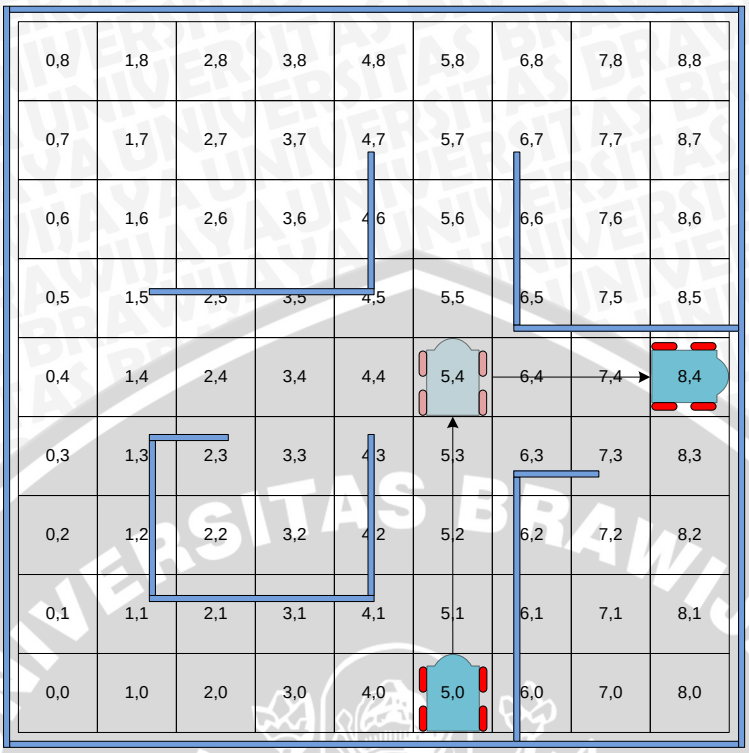
4.4.3.1. Perancangan Perangkat Fungsi Utama Sistem

Dalam penelitian ini, sistem pengenalan posisi berdasarkan *grid-based map* dirancang dengan maksud agar robot dapat mengenali posisi robot di arena pertandingan. Proses pengenalan posisi tersebut dilakukan melalui pembacaan arah orientasi robot sekarang, pengukuran jarak di sekitar robot, melakukan pengenalan posisi robot sekarang, melakukan koreksi arah orietasi robot, serta memberikan perintah gerak sebagai aksi pengaturan. Untuk melakukan serangkaian proses tersebut maka dalam penelitian ini dirancang perangkat lunak yang berperan sebagai fungsi utama dalam sistem pengenalan posisi berdasarkan *grid-based map*. Digram alir perangkat lunak fungsi utama sistem pengenalan posisi berdasarkan *grid-based map* ditunjukkan dalam Gambar 4.18.



Gambar 4.18 Diagram Alir Perangkat Lunak Fungsi Utama Sistem Pengenalan Posisi Berdasarkan *Grid-Based Map*

Agar robot dapat sampai pada koordinat tujuan, maka robot harus bergerak dari satu koordinat ke koordinat selanjutnya yang berdekatan. Hal ini akan terus dilakukan oleh robot hingga robot sampai ke koordinat tujuan. Gambar 4.19 menunjukkan ilustrasi pergerakan robot dari posisi *start* ke posisi tujuan robot.



Gambar 4.19 Ilustrasi Pergerakan Robot dari Posisi *Start* ke Posisi Tujuan Robot

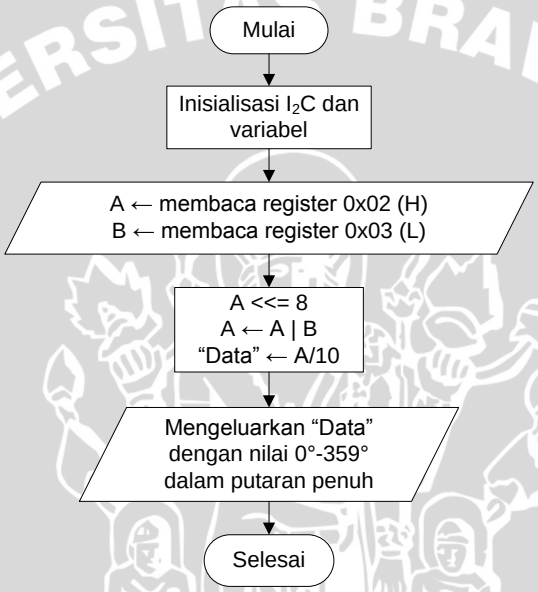
Pada ilustrasi di atas, posisi *start* robot berada pada koordinat 5,0 dan posisi tujuan robot berada pada koordinat 8,4. Agar robot dapat bergerak dari koordinat 5,0 ke koordinat 8,4, maka robot akan terlebih dahulu bergerak dari koordinat 5,0 ke koordinat 5,4 melewati koordinat 5,1, 5,2, dan 5,3 untuk menghindari dinding, setelah itu robot bergerak dari koordinat 5,4 menuju koordinat tujuan 8,4 melewati koordinat 6,4 dan 7,4.

4.4.3.2. Perancangan Perangkat Lunak Pemroses Data Modul CMPS03 *Magnetic Compass*

Modul CMPS03 *magnetic compass* berfungsi sebagai kompas digital yang dapat mengkonversi besaran fisik arah mata angin menjadi besaran elektrik yang kemudian diolah (di dalam modul tersebut) sehingga menghasilkan data hasil pembacaan arah mata angin yang berupa nilai sudut yang dapat dibaca oleh mikrokontroler. Data yang dihasilkan oleh modul disimpan dalam dua bentuk yaitu data 8 bit dan data 16 bit. Data 8 bit disimpan dengan alamat 0x01 sedangkan data 16 bit disimpan dalam dua buah register dengan alamat 0x02 (H *byte*) dan 0x03 (L *byte*). Data dalam register dengan alamat 0x01 bernilai 0-255

dalam skala penuh. Sedangkan data dalam register dengan alamat 0x02 dan 0x03 jika digabungkan bernilai 0-3599 dalam skala penuh.

Data yang tersimpan dalam register-register tersebut diakses melalui jalur I_2C oleh mikrokontroler pemroses utama (ATmega128). Data yang telah diakses kemudian diproses lebih lanjut untuk mendapatkan data dalam satuan derajat. Untuk melakukan serangkaian proses tersebut maka dalam penelitian ini dirancang perangkat lunak pemroses data modul CMPS03 *magnetic compass*. Diagram alir perangkat lunak pemroses data modul CMPS03 *magnetic compass* ditunjukkan dalam Gambar 4.20.



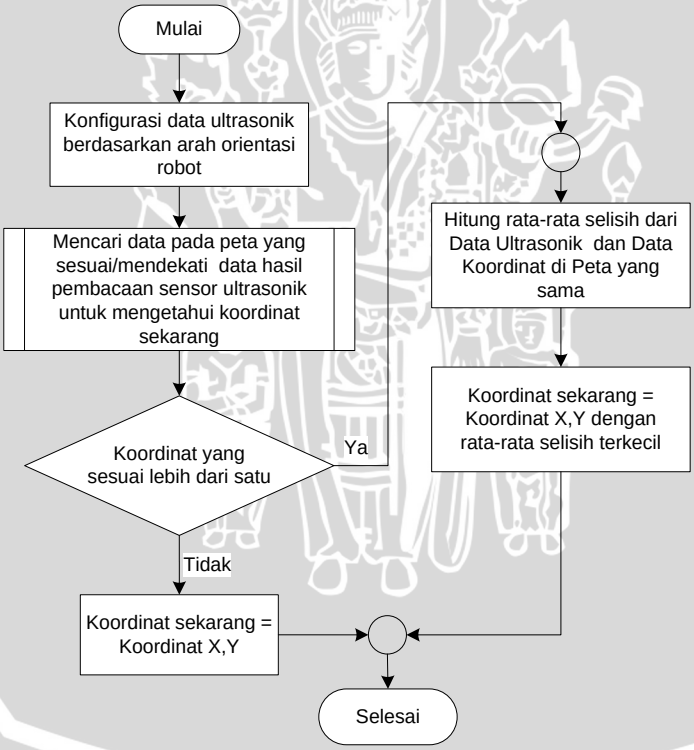
Gambar 4.20 Diagram Alir Perangkat Lunak Pemroses Data Modul CMPS03 *Magnetic Compass*

Proses pertama pada perangkat lunak pemroses data modul CMPS03 *magnetic compass* adalah inisialisasi *library* I_2C (i2c.h) dan variabel-variabel yang dibutuhkan. Proses selanjutnya yaitu membaca data yang terdapat dalam register modul CMPS03 dengan alamat 0x02 (H *byte*) dan 0x03 (L *byte*), lalu data hasil pembacaan tersebut disimpan dalam dua variabel berbeda (A untuk H *byte* dan B untuk L *byte*). Selanjutnya adalah proses penggabungan dua data yang masing-masing tersimpan dalam variabel A dan B lalu data hasil penggabungan disimpan kembali ke vriabel A. Proses penggabungan variabel A dan B diawali dengan menggeser data pada variabel A sebnya 8 bit ke kiri dengan menggunakan operator manipulasi bit *shift left* "<<". Selanjutnya variabel A di OR dengan

variabel B lalu disimpan kembali ke variabel A, sehingga variabel A berisi data 16 bit dari hasil pembacaan modul CMPS03 *magnetic compass* tersebut. Untuk dapat mereprestasikan hasil pembacaan arah dalam bentuk sudut, maka data dalam variabel A dibagi 10 kemudian disimpan ke variabel "Data". Proses terakhir adalah mengeluarkan "Data". "Data" memiliki rentang 0°-359,9° dengan resolusi 0,1°, namun dalam penelitian ini "Data" dikeluarkan dalam rentang 0°-359° dengan resolusi 1°.

4.4.3.3. Perancangan Perangkat Lunak Pengenalan Posisi

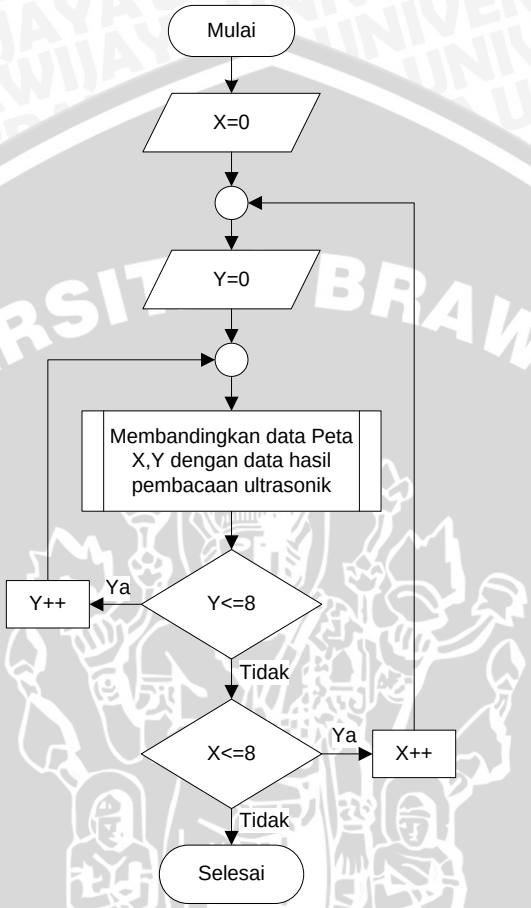
Pada pengenalan posisi robot saat *start*, robot tidak memiliki acuan mengenai koordinat di sekitar robot, untuk itu diperlukan pembandingan data hasil pembacaan sensor ultrasonik dengan seluruh data koordinat di peta. Diagram alir untuk proses tersebut ditunjukkan dalam Gambar 4.21.



Gambar 4.21 Diagram Alir Penentuan Posisi saat *Start*

Jika pada saat pembandingan hanya ditemukan satu data koordinat yang sama maka koordinat sekarang adalah koordinat dengan data sama tersebut. Namun jika terdapat lebih dari satu koordinat dengan data yang sama maka dilakukan perhitungan untuk mencari rata-rata selisih terkecil dari data koordinat

yang sama dengan data sekarang. Koordinat dengan data yang memiliki nilai rata-rata selisih terkecil dianggap sebagai koordinat posisi sekarang. Diagram alir proses pencarian data pada peta saat penentuan posisi *start* ditunjukkan dalam Gambar 4.22.

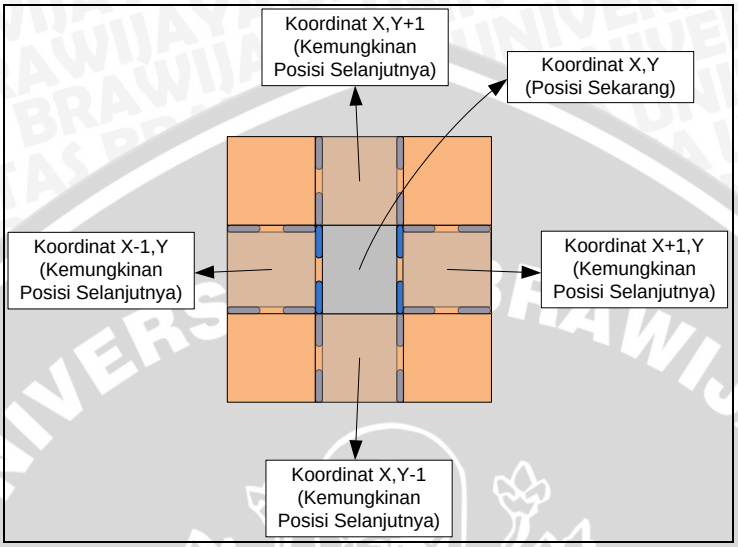


Gambar 4.22 Diagram Alir Proses Pencarian Data pada Peta saat Penentuan Posisi *Start*

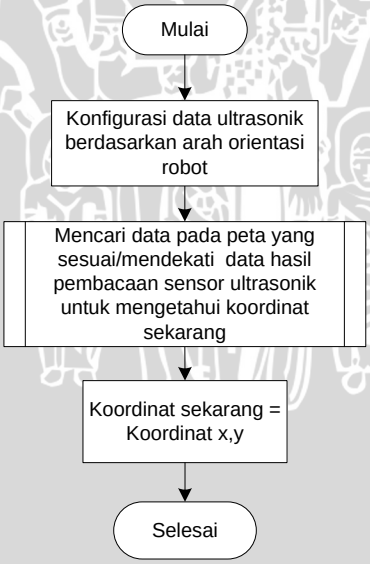
Berdasarkan diagram alir di atas, proses pencarian data yang sesuai dengan data hasil pembacaan sensor ultrasonik dilakukan dengan melakukan pembandingan data antara data ultrasonik dengan data pada peta 0,0. Proses ini dilanjutkan sampai dengan data peta 0,8. Setalah sampai dengan peta 0,8, proses pembandingan dilanjutkan lagi dari data peta 1,8 sampai data peta 1,8. Hal ini dilanjutkan sampai data ultrasonik dibandingkan dengan data peta 8,8 sesuai dengan peta yang dirancang. Setalah selesai membandingkan dengan data peta 8,8 maka proses pencarian data untuk mengenenali posisi *start* selesai.

Setelah mengetahui koordinat posisi *start*, maka kita dapat mengetahui koordinat berikutnya yang akan dilalui robot di mana koordinat tersebut dapat

digunakan sebagai acuan dalam menentukan posisi koordinat robot selanjutnya. Gambar 4.23 mengilustrasikan empat kemungkinan koordinat selanjutnya dari posisi robot, dan Gambar 4.24 menunjukkan diagram alir dari proses penentuan posisi robot setelah *start*.



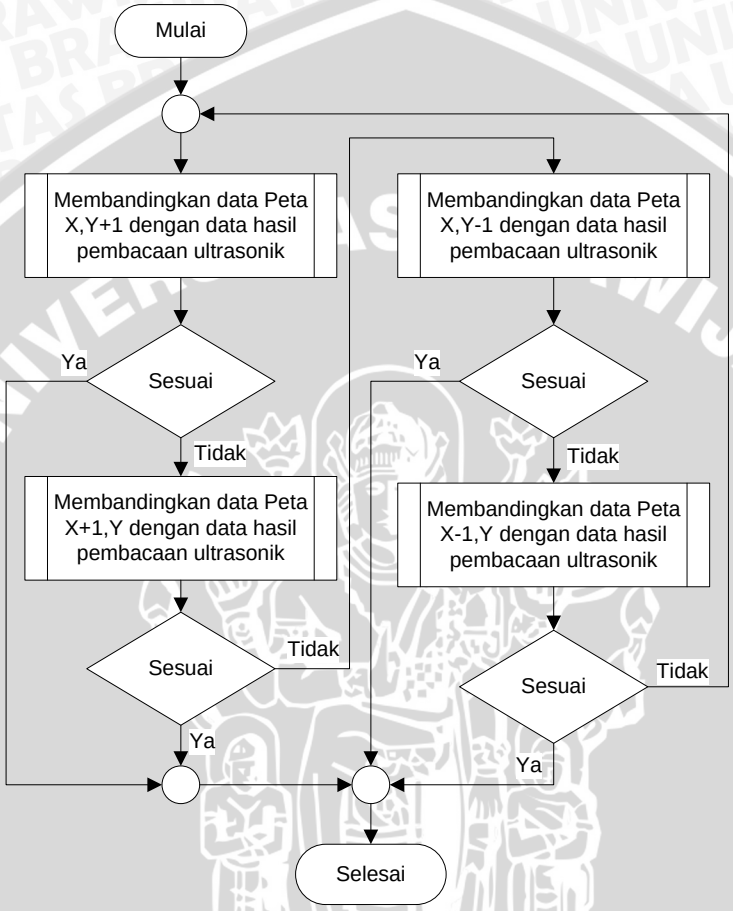
Gambar 4.23 Ilustrasi Posisi Robot dan Kemugkinan Posisi Selanjutnya



Gambar 4.24 Diagram Alir Penentuan Posisi setelah *Start*.

Berdasarkan Gambar 4.23, ada empat kemungkinan posisi selanjutnya yang akan dilalui robot untuk mencapai koordinat tujuan robot. Oleh karena itu pada saat proses pembandingan data yang diperoleh dari sensor ultrasonik dengan data di peta, tidak perlu dilakukan proses pembandingan terhadap semua data

yang ada di peta, tetapi cukup dengan empat data koordinat tersebut. Koordinat [X,Y] yang didapat dari hasil pembandingan tersebut selanjutnya dianggap sebagai koordinat di mana robot sedang berada sekarang. Diagram alir proses pencarian data pada peta saat penentuan posisi setelah *start* ditunjukkan dalam Gambar 4.25.



Gambar 4.25 Diagram Alir Proses Pencarian Data pada Peta saat Penentuan Posisi Setelah *Start*

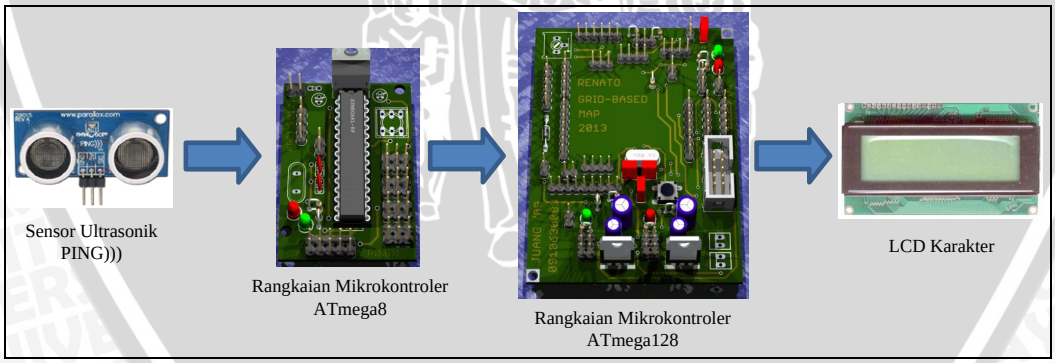
BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui kinerja sistem, apakah sistem telah sesuai dengan perancangan. Pengujian dilakukan tiap blok sistem kemudian secara keseluruhan. Pengujian terdiri atas:

1) Pengujian sensor ultrasonik PING)))
2) Pengujian modul CMPS03 *magnetic compass*
3) Pengujian pengenalan posisi
4) Pengujian sistem secara keseluruhan

5.1. Pengujian Sensor Ultrasonik PING)))

Pengujian sensor ultrasonik PING))) ini dibagi menjadi beberapa bagian. Pertama adalah pengujian hasil pembacaan sensor ultrasonik. Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah pembacaan sensor sesuai dengan jarak sensungguhnya. Pengujian ini dilakukan dengan menghubungkan sensor ultrasonik PING))), mikrokontroler pengatur sensor ultrasonik PING))) (ATmega8) dan modul LCD. Diagram blok pengujian sensor ultrasonik PING))) ditunjukkan dalam Gambar 5.1.

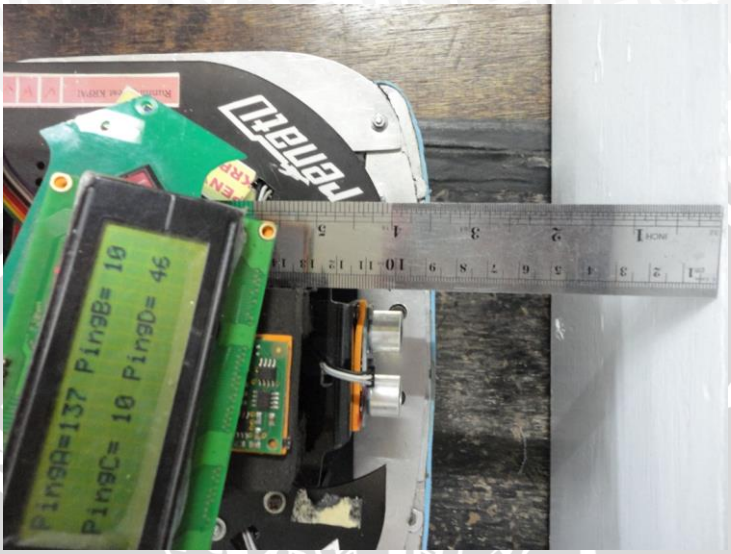


Gambar 5.1 Diagram Blok PengujianSensor Ultrasonik PING)))

Pengujian hasil pembacaan sensor ultrasonik dilakukan dengan prosedur sebagai berikut:

1) Menghidupkan catu daya elektronik pada robot, kemudian memilih mode pengujian sensor ultrasonik PING))) melalui *rotary switch*.
2) Meletakkan robot dengan posisi sensor ultrasonik sejajar dengan dinding.

3) Menggeser posisi robot hingga hasil pembacaan sensor yang ditampilkan pada LCD menunjukkan jarak yang diinginkan.
4) Mengukur jarak antara sensor ultrasonik dengan dinding, seperti yang ditunjukkan dalam Gambar 5.2.



Gambar 5.2 Pengukuran Jarak antara Sensor Ultrasonik PING))) dan Dinding

5) Mencatat hasil pengukuran antara sensor ultrasonik dengan dinding.
6) Mengulang prosedur 2 sampai prosedur 5 sampai diperoleh 10 hasil pengujian dengan kelipatan jarak 5 cm.
7) Mengulangi prosedur 1 sampai prosedur 6 untuk semua sensor ultrasonik yang digunakan.

Perangkat *timer 1* pada mikrokontroler pengatur ultrasonik digunakan untuk menghitung lama waktu aktif sinyal jawaban dari sensor ultrasonik. Selanjutnya data berupa lama waktu aktif sinyal dikonversi ke dalam jarak terbaca dalam centimeter (cm) dan ditampilkan menggunakan LCD. Hasil pengujian yang diperoleh melalui beberapa kali pengambilan data ditunjukkan pada Tabel 5.1.

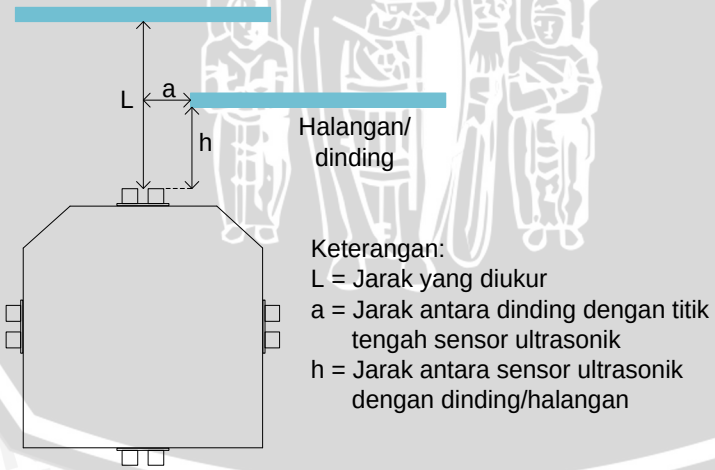
Tabel 5.1 Data Hasil Pengujian Sensor Ultrasonik PING)))

Pengujian ke-	**Jarak Uji (cm)**	**Jarak Terukur**				**Tampilan LCD**
		Sensor A	**Sensor B**	**Sensor C**	**Sensor D**	
1	5	5,1	5,2	5,2	5,1	5
2	10	10,1	10,1	10,3	10,1	10
3	15	15,1	15,1	15,1	15,4	15

4	20	20,4	20,2	20,2	20,2	20
5	25	25,4	25,1	25,4	25,1	25
6	30	30,3	30,2	30,3	30,3	30
7	35	35,4	35,2	35,1	35,3	35
8	40	40,4	40,3	40,3	40,1	40
9	45	45,4	45,4	45,3	45,2	45
10	50	50,3	50,2	50,1	50,3	50

Berdasarkan Tabel 5.1 dapat diketahui bahwa kesalahan rata-rata yang terjadi pada saat pembacaan sensor ultrasonik adalah sebesar 0,23 cm. Kesalahan pembacaan terbesar yaitu sebesar 0,4 cm. Pada pengujian, kesalahan pembacaan yang terjadi berupa hasil pengukuran yang lebih besar pada nilai desimal di belakang tanda koma, sedangkan pada nilai desimal di depan tanda koma bernilai sama sengan jarak yang diuji. Kesalahan tersebut diharapkan tidak memberikan pengaruh pada kinerja sistem yang dirancang karena pada sistem hanya digunakan data jarak dengan nilai desimal di depan tanda koma.

Selanjutnya dilakukan pengujian pengaruh posisi dinding/halangan terhadap hasil pembacaan sensor ultrasonik PING))). Hal ini dilakukan dengan tujuan untuk mengetahui pengaruh posisi dinding terhadap hasil pembacaan sensor ultrasonik. Gambar 5.3 menunjukkan ilustrasi pengujian.



Gambar 5.3 Ilustrasi Pengujian Pengaruh Posisi Dinding terhadap Hasil Pembacaan Sensor Ultrasonik

Pengujian ini dilakukan dengan prosedur sebagai berikut:

1) Menghidupkan catu daya elektronik pada robot, kemudian memilih mode pengujian sensor ultrasonik PING))) melalui *rotary switch*.
2) Meletakkan robot dengan posisi sensor ultrasonik sejajar dengan dinding dengan jarak tertentu (h).
3) Menggeser posisi robot hingga sensor ultrasonik tidak terhalangi oleh dinding.
4) Melihat hasil pembacaan sensor ultrasonik yang ditampilkan pada LCD.
5) Menggeser robot hingga hasil pembacaan sensor ultrasonik sesuai dengan jarak yang diukur (L).
6) Mengukur jarak antara dinding dengan titik tengah sensor ultrasonik (a).
7) Mengulangi prosedur 1 sampai prosedur 6 sampai diperoleh 5 hasil pengujian dengan kelipatan jarak antara sensor ultrasonik dengan dinding (h) sebesar 10 cm.

Tabel 5.2 menunjukkan data hasil pengujian pengaruh dinding/halangan terhadap sensor ultrasonik.

Tabel 5.2 Data Hasil Pengujian Pengaruh Posisi Dinding/Halangan terhadap Hasil Pembacaan Sensor Ultrasonik PING)))

Pengujian ke-	**Jarak yang diukur (L) (cm)**	**Jarak Dinding (h) (cm)**	**Jarak Dinding dari Titik Tengah Robot (a) ketika Hasil Pembacaan Sesuai Jarak yang diukur (cm)**
1	100	10	2
2	100	20	5
3	100	30	6
4	100	40	8
5	100	50	11

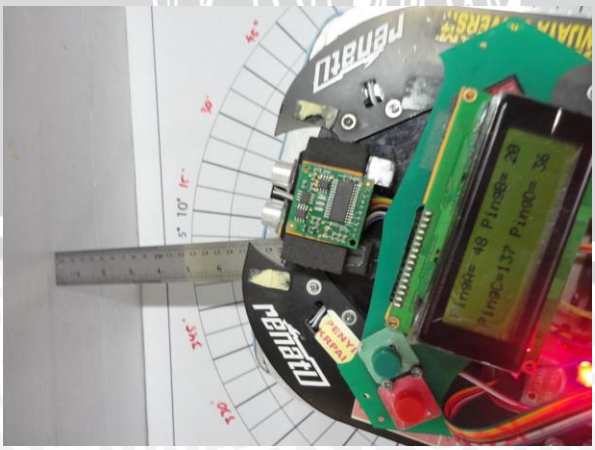
Berdasarkan Tabel 5.2, semakin jauh jarak dinding terhadap sensor maka pengaruh posisi dinding terhadap hasil pembacaan sensor ultrasonik semakin besar. Hal ini mungkin saja dikarenakan gelombang yang dipancarkan semakin melebar seiring bertambahnya jarak. Untuk mengatasi hal tersebut maka pada peta yang digunakan perlu ditambahkan data hasil pembacaan sensor ultrasonik yang mungkin terbaca akibat adanya pengaruh dinding pada arena pertandingan.

Kemudian dilakukan pengujian pembacaan sensor ultrasonik yang membentuk sudut tertentu terhadap dinding. Tujuan dari pengujian ini adalah untuk mengetahui pengaruh sudut yang dibentuk antara sensor dengan dinding terhadap hasil pembacaan sensor ultrasonik. Gambar 5.4 menunjukkan ilustrasi pengujian.

Gambar 5.4 Ilustrasi Pengujian Pembacaan Sensor Ultrasonik yang Membentuk Sudut Tertentu terhadap Dinding

Pengujian ini dilakukan dengan prosedur sebagai berikut:

1) Menghidupkan catu daya elektronik pada robot, kemudian memilih mode pengujian sensor ultrasonik PING))) melalui *rotary switch*.
2) Meletakkan robot dengan posisi sensor ultrasonik sejajar dengan dinding dengan jarak yang diukur (L).
3) Memutar robot hingga posisi antara sensor ultrasonik dengan dinding membentuk sudut tertentu (α), seperti yang ditunjukkan dalam Gambar 5.5.



Gambar 5.5 Posisi Robot yang Membentuk Sudut Tertentu (α) terhadap Dinding

4) Mencatat hasil pembacaan sensor ultrasonik yang ditampilkan pada LCD.
5) Mengulangi prosedur 1 sampai prosedur 5 sampai diperoleh 10 hasil pengujian dengan kelipatan α sebesar 5°.

Tabel 5.3 menunjukkan data hasil pengujian pengujian pembacaan sensor ultrasonik yang membentuk sudut tertentu terhadap dinding.

Tabel 5.3 Data Hasil Pengujian Pembacaan Sensor Ultrasonik Dengan Sudut Tertentu

Pengujian Ke-	Jarak Terhadap Dinding (L) (cm)	Sudut yang Dibentuk (α) (°)	Hasil Pembacaan Sensor Ultrasonik (cm)
1	20	0	20
2	20	5	20
3	20	10	20
4	20	15	20
5	20	20	21
6	20	25	21
7	20	30	21
8	20	35	22
9	20	40	100
10	20	45	100

Berdasarkan data pada Tabel 5.3, hasil pembacaan sensor ultrasonik masih sesuai dengan jarak yang diukur pada sudut 0°-15°. Pada sudut 20°-30°, hasil pembacaan sensor ultrasonik memiliki kesalahan pembacaan sebesar 1 cm dari jarak yang diukur. Kesalahn pembacaan meningkat menjadi 2 cm pada sudut 35°. Sedangkan pada sudut 40°-45° hasil pembacaan sensor sudah tidak sesuai dengan jarak yang diukur. Hal ini mungkin dikarenakan gelombang ultrasonik yang dipancarkan oleh sensor tidak memantul kembali ke sensor dikarenakan sudut pantulan yang terlalu besar.

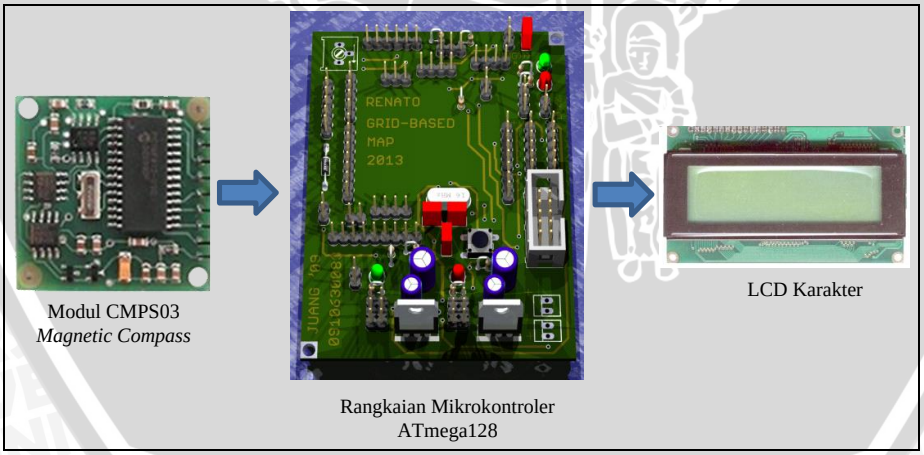
5.2. Pengujian Modul CMPS03 *Magnetic Compass*

Pengujian modul CMPS03 *magnetic compass* bertujuan untuk mengetahui akurasi pembacaan arah mata angin yang dibaca dalam bentuk nilai sudut 0°-359° oleh modul tersebut. Akurasi adalah ukuran yang menyatakan nilai maksimum dari keseluruhan kesalahan yang diperkirakan muncul dalam pengukuran suatu variabel. Pada umumnya akurasi dinyatakan dengan nilai

inakurasi (ketidak telitian). Hasil yang diharapkan dalam pengujian ini adalah modul CMPS03 *magnetic compass* mampu bekerja dengan akurasi pembacaan arah sebesar ± 5°. Nilai akurasi dalam penelitian ini dinyatakan berdasarkan nilai kesalahan terbesar dari keseluruhan pengujian. Nilai kesalahan diperoleh dengan menghitung selisih antara nilai arah yang terbaca dengan nilai arah yang sebenarnya.

Dalam pengujian akurasi modul CMPS03 *magnetic compass* dibutuhkan perangkat lain yang dapat digunakan sebagai acuan. Pada pengujian ini digunakan kompas konvensional sebagai acuannya. Kompas konvensional digunakan sebagai acuan karena memiliki prinsip keja yang sama seperti modul CMPS03 *magnetic compass* yaitu dapat menentukan arah berdasarkan pengaruh medan magnet Bumi. Kompas konvensional digunakan sebagai acuan penetuan arah 0° (utara) saja. Sedangkan untuk mempermudah pengujian maka digunakan papan uji sebagai acuan selanjutnya.

Dalam pengujian ini, sistem minimal yang dibutuhkan yang dibutuhkan terdiri atas modul CMPS03 *magnetic compass*, rangkaian mikrokontroler pemroses utama (ATmega32), dan modul LCD karakter. Diagram blok pengujian modul CMPS03 *magnetic compass* ditunjukkan dalam Gambar 5.6.



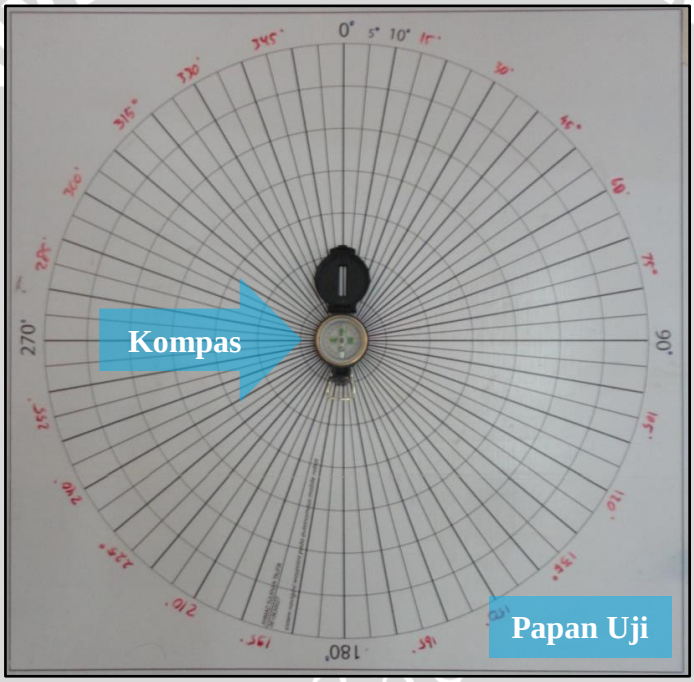
Gambar 5.6 Diagram Blok Pengujian Modul CMPS03 *Magnetic Compass*

Data hasil pembacaan arah yang dihasilkan oleh modul CMPS03 diolah oleh perangkat lunak pemroses data modul CMPS03 *magnetic compass* yang terdapat dalam mikrokontroler pemroses utama. Data yang diolah kemudian

ditampilkan pada modul LCD dengan satuan derajat. Data yang ditampilkan memiliki rentang nilai 0°-359° dengan resolusi 1°.

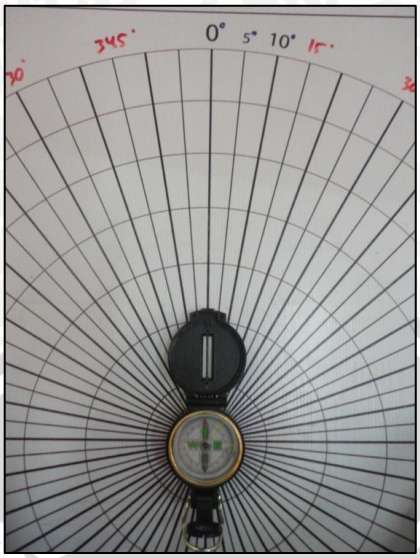
Pengujian modul CMPS03 *magnetic compass* dilakukan dengang prosedur sebagai berikut:

1) Menentukan posisi papan uji sebagai acuan dalam pengujian, melalui prosedur:
   a) Memposisikan kompas di atas papan uji dengan posisi kompas terhadap papan uji seperti ditunjukkan dalam Gambar 5.7, serta mengkondisikan papan uji tersebut pada bidang yang datar agar kompas ataupun modul CMPS03 *magnetic compass* dapat bekerja secara normal.



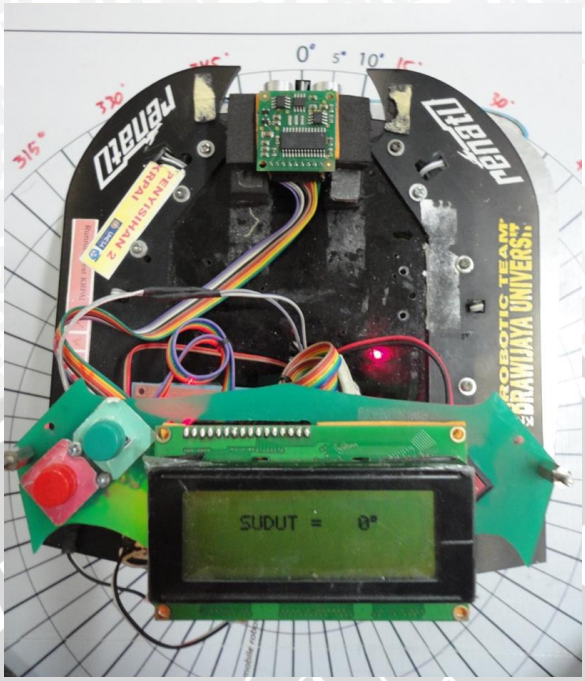
Gambar 5.7 Posisi Kompas Terhadap Papan Uji

   b) Menetapkan posisi papan uji dengan cara mengkondisikan papan uji hingga penunjukkan jarum utara kompas mengarah ke nilai 0° pada papan uji, seperti ditunjukkan dalam Gambar 5.8.



Gambar 5.8 Penentuan Posisi Papan Uji Sebagai Acuan dalam Pengujian

2) Menghidupkan catu daya elektronik pada robot, kemudian memilih mode pengujian modul CMPS03 *magnetic compass* melalui *rotary switch*.
3) Memposisikan arah hadap modul CMPS03 *magnetic compass* ke arah tertentu dengan mangacu pada papan uji (dengan catatan tidak mengubah posisi papan uji).
4) Mencatat arah hadap modul (berupa nilai sudut yang terdapat pada papan uji) serta mencatat data hasil pembacaan modul CMPS03 *magnetic compass* yang tertampil pada modul LCD, seperti yang ditunjukkan dalam Gambar 5.9.



Gambar 5.9 Tampilan Hasil Pembacaan Modul CMPS03 *Magnetic Compass* pada Modul LCD

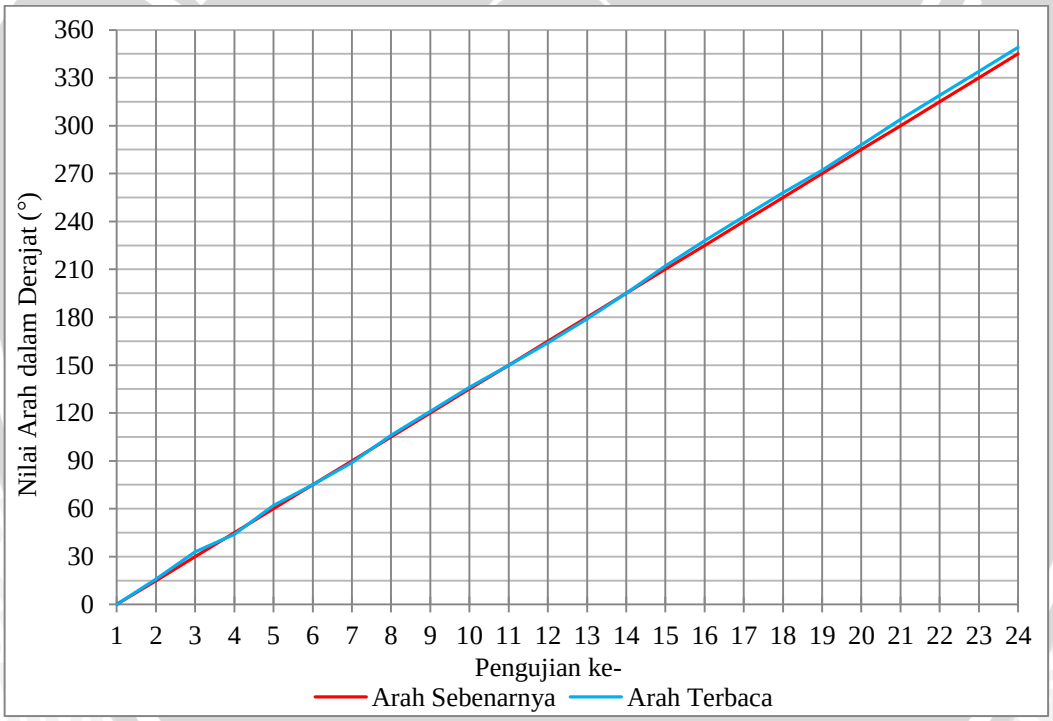
5) Mengulang prosedur ke 3 hingga ke 4 sampai diperoleh 24 data hasil pengujian pada arah hadap modul yang berbeda-beda.

Data hasil pengujian modul CMPS03 *magnetic compass* ditunjukkan dalam Tabel 5.4 dan disajikan dalam bentuk grafik seperti ditunjukkan dalam Gambar 5.10.

Tabel 5.4 Data Hasil Pengujian Modul CMPS03 *Magnetic Compass*

No.	Arah Sebenarnya	Arah Terbaca	Kesalahan
1	0°	0°	0°
2	15°	16°	-1°
3	30°	33°	-3°
4	45°	44°	1°
5	60°	62°	-2°
6	75°	75°	0°
7	90°	89°	1°
8	105°	106°	-1°
9	120°	121°	-1°
10	135°	136°	-1°
11	150°	150°	0°
12	165°	164°	1°

13	180°	179°	1°
14	195°	195°	0°
15	210°	212°	-2°
16	225°	228°	-3°
17	240°	243°	-3°
18	255°	258°	-3°
18	270°	272°	-2°
20	285°	288°	-3°
21	300°	304°	-4°
22	315°	319°	-4°
23	330°	334°	-4°
24	345°	349°	-4°
Kesalahan Rata-rata = 1,8°			
Kesalahan Terbesar = 4°			



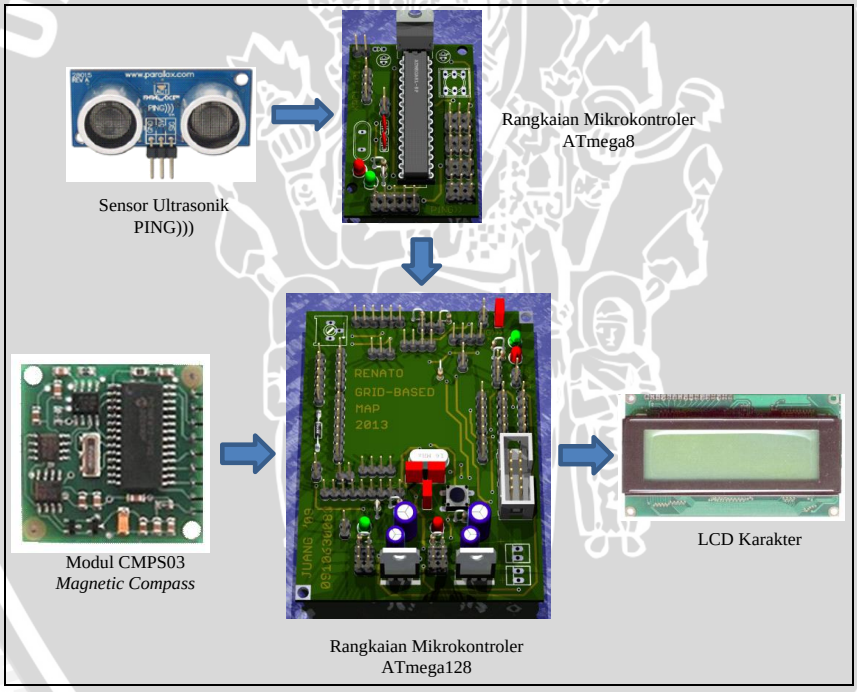
Gambar 5.10 Grafik Hasil Pengujian Modul CMPS03 *Magnetic Compass*

Berdasarkan data hasil pengujian, dapat diketahui nilai kesalahan terbesar dari pembacaan arah menggunakan modul CMPS03 *magnetic compass* adalah sebesar 4°. Sehingga berdasarkan data hasil pengujian tersebut dapat diketahui bahwa dalam penelitian ini, modul CMPS03 *magnetic compass* yang digunakan memiliki akurasi sebesar ± 4° dengan kesalahan pembacaan arah rata-rata sebesar

1,8°. Berdasarkan hasil tersebut maka modul CMPS03 *magnetic compass* dapat difungsikan sebagai penunjuk arah mata angin serta dapat digunakan sebagai umpan balik untuk menentukan arah orientasi robot terhadap arena pertandingan.

5.3. Pengujian Pengenalan Posisi

Pengujian pengenalan posisi bertujuan untuk mengetahui kemampuan robot dalam mengolah data hasil pembacaan sensor ultrasonik dan membandingkan data tersebut dengan peta yang sudah disimpan di dalam memori mikrokontroler pemroses utama (ATmega128). Sistem minimal yang dibutuhkan dalam pengujian ini terdiri atas sensor ultrasonik PING))), mikrokontroler pengatur sensor ultrasonik PING))) (ATmega8), modul CMPS03 *magnetic compass*, mikrokontroler pemroses utama (ATmega128) dan LCD karakter. Diagram blok pengujian pembacaan peta ditunjukkan dalam Gambar 5.11.



Gambar 5.11 Diagram Blok Pengujian Pengenalan Posisi

Pengujian pengenalan posisi dilakukan dengan prosedur sebagai berikut:

1) Meletakkan robot pada arena pertandingan dengan prosedur:
   a) Robot diletakkan pada koordinat tertentu yang telah diketahui oleh operator.

b) Arah orientasi robot dihadapkan pada salah satu dari empat arah orientasi yang mengacu pada arena pertandingan.

2) Menghidupkan catu daya elektrik pada robot, kemudian memilih mode pengujian pembacaan peta melalui *rotary switch*.
3) Memposisikan arah hadap robot ke salah satu dari empat arah orientasi dengan mengacu pada arah orientasi arena pertandingan (dengan catatan sudah melakukan kalibrasi arah orientasi arena sebelumnya).
4) Mancatat data hasil pembacaan peta (berupa koordinat) yang tertampil pada modul LCD karakter.
5) Mengulang prosedur 1 sampai 4 hingga diperoleh 10 data hasil pengujian pengenalan posisi dengan arah orientasi dan robot koordinat yang berbeda-beda.

Data hasil pengenalan posisi ditunjukkan dalam Tabel 5.5

Tabel 5.5 Data Hasil Pengujian Pengenalan Posisi

Pegujian ke-	Orientasi Robot	Posisi Sesungguhnya	Hasil Pengenalan Posisi
1	NH	0,0	0,0
2	ET	0,2	0,2
3	SH	0,8	0,8
4	WT	2,2	2,2
5	NH	5,0	5,0
6	ET	5,4	5,4
7	SH	5,8	5,8
8	WT	6,4	1,0
9	NH	8,0	8,0
10	SH	8,4	8,4

Berdasarkan Tabel 5.5, robot dianggap sudah mampu untuk mengenali posisi keberadaan robot di arena pertandingan. Adapun kesalahan yang terjadi pada pembacaan di posisi 6,4 diakibatkan adanya ketidaksesuaian antara data hasil pembacaan sensor dengan data yang terdapat pada peta. Hal ini bisa disebabkan karena kurangnya data pada peta yang mencakup kemungkinan pembacaan sensor ultrasonik mengalami kesalahan pengukuran akibat adanya pengaruh posisi dinding. Untuk mengatasi hal tersebut maka pada peta perlu ditambahkan

informasi tambahan mengenai jarak robot terhadap dinding dengan memperhatikan kemungkinan pengaruh posisi dinding terhadap hasil pembacaan sensor ultrasonik.

5.4. Pengujian Sistem Secara Keseluruhan

Prinsip kerja dari sistem pengenalan posisi berdasarkan *grid-based map* pada skripsi ini secara umum adalah sebagai berikut. Pertama, robot diletakkan pada arena pertandingan di mana arah orientasi robot terhadap lapangan pertandingan telah dikalibrasi sebelumnya. Jika arah orientasi robot belum dikalibrasi, maka proses kalibrasi dapat dilakukan dengan cara melakukan pembacaan sensor *magnetic compass* ketika robot menghadap pada arah 0, arah 90, arah 180 dan arah 270. Setelah itu, hasil pembacaan tersebut dimasukkan pada program dan disimpan pada memori mikrokontroler utama, seperti pada Gambar 5.12.

```
arah_0 = 283;
arah_90 = 5;
arah_180 = 95;
arah_270 = 180;
```

Gambar 5.12 Arah Orientasi Robot berdasarkan Orientasi Arena Pertandingan

Setelah itu robot akan melakukan pembacaan arah orientasi robot menggunakan sensor *magnetic compass*. Jika arah orientasi robot telah sesuai dengan yang arah orientasi yang disimpan pada memori mikrokontroler utama, maka robot tidak perlu melakukan koreksi, namun jika tidak, maka robot akan melakukan koreksi dengan cara berputar di tempat hingga arah orientasi robot sesuai dengan arah orientasi yang telah disimpan di dalam memori mikrokontroler utama. Kemudian, robot akan melakukan pembacaan jarak menggunakan sensor ultrasonik PING))) untuk mengetahui robot terhadap dinding di sekitar robot. Jarak hasil pembacaan sensor ultrasonik ini kemudian akan dibandingkan dengan peta berupa *array* yang udah di simpan dalam memori mikrokontroler pemroses utama. Agar proses pembandingan sesuai, maka arah orientasi robot berdasarkan hasil pembacaan dari sensor *magnetic compass* ini digunakan sebagai acuan untuk melakukan konfigurasi orientasi sensor relatif. Gambar 5.13 menunjukkan potongan program yang berfungsi untuk menentukan konfigurasi sensor relatif

dan Gambar 5.14 menunjukkan potongan program pembandingan antara data hasil pembacaan sensor ultrasonik dengan data pada memori mikrokontroler utama.

```
if (heading==NORTH){
    data[0]=jarak_B;
    data[1]=jarak_C;
    data[2]=jarak_D;
    data[3]=jarak_A;}
else if (heading==EAST){
    data[0]=jarak_A;
    data[1]=jarak_B;
    data[2]=jarak_C;
    data[3]=jarak_D;}
else if (heading==SOUTH){
    data[0]=jarak_D;
    data[1]=jarak_A;
    data[2]=jarak_B;
    data[3]=jarak_C;}
else if (heading==WEST){
    data[0]=jarak_C;
    data[1]=jarak_D;
    data[2]=jarak_A;
    data[3]=jarak_B;}
```

Gambar 5.13 Program Konfigurasi Sensor Relatif

```
if((x==0)&&(y==0)){
    for(c = 0; c<2; c++){
        if(((petaxy_00[c][0]-13)<=data[0])&&(data[0]<(petaxy_00[c][0]+13))){
            if(((petaxy_00[c][1]-13)<=data[1])&&(data[1]<(petaxy_00[c][1]+13))){
                if(((petaxy_00[c][2]-13)<=data[2])&&(data[2]<(petaxy_00[c][2]+13))){
                    if(((petaxy_00[c][3]-13)<=data[3])&&(data[3]<(petaxy_00[c][3]+13))){
                        ok=1;
                        for(i=0;i<4;i++){
                            peta[a][b][i]=petaxy_00[c][i];};

}}}}}}
else if((x==0)&&(y==1)){
    for(c = 0; c<4; c++){
        if(((petaxy_01[c][0]-13)<=data[0])&&(data[0]<(petaxy_01[c][0]+13))){
            if(((petaxy_01[c][1]-13)<=data[1])&&(data[1]<(petaxy_01[c][1]+13))){
                if(((petaxy_01[c][2]-13)<=data[2])&&(data[2]<(petaxy_01[c][2]+13))){
                    if(((petaxy_01[c][3]-13)<=data[3])&&(data[3]<(petaxy_01[c][3]+13))){
                        ok=1;
                        for(i=0;i<4;i++){
                            peta[a][b][i]=petaxy_01[c][i];};
}}}}}}
else if((x==0)&&(y==3)){
    for(c = 0; c<3; c++){
        if(((petaxy_03[c][0]-13)<=data[0])&&(data[0]<(petaxy_03[c][0]+13))){
            if(((petaxy_03[c][1]-13)<=data[1])&&(data[1]<(petaxy_03[c][1]+13))){
                if(((petaxy_03[c][2]-13)<=data[2])&&(data[2]<(petaxy_03[c][2]+13))){
                    if(((petaxy_03[c][3]-13)<=data[3])&&(data[3]<(petaxy_03[c][3]+13))){
                        ok=1;
```

Gambar 5.14 Potongan Program Proses Pembandingan Antara Data Hasil Pembacaan Sensor Ultrasonik dengan Data pada Memori Mikrokontroler Utama.

Dari hasil pembandingan ini akan diperoleh koordinat yang dianggap sebagai posisi *start*. Sebagai contoh digunakan koordinat [5,0] untuk posisi *start* dan koordinat [8,2] sebagai posisi tujuan. Untuk bisa sampai ke posisi [8,2] dari posisi [5,0] robot harus melewati posisi [5,1], [5,2], [5,3], [5,4], [6,4], [7,4], [8,4] dan [8,3]. Oleh karena itu setelah mengetahui posisi *start* robot akan melakukan pengecekan arah orientasi untuk menyesuaikan arah orientasi robot dengan arah orientasi di mana koordinat selanjutnya yang akan dilewati robot berada. Sebagai contoh, koordinat [5,1] berada pada arah 0 dari koordinat [5,0]. Jika arah orientasi robot menghadap ke arah 0, maka robot tidak perlu melakukan koreksi arah orientasi, namun jika tidak, maka robot akan melakukan koreksi sampai arah orientasi robot menghadap pada arah 0.

Setelah arah orientasi robot sesuai dengan arah orientasi di mana koordinat selanjutnya berada, maka robot akan bergerak sambil terus melakukan pembandingan antara data yang diperoleh dari hasil pembacaan sensor ultrasonik ketika robot bergerak dengan data peta yang telah disimpan pada memori mikrokontroler pemroses utama. Jika robot berhasil mengenali koordinat [5,1], maka robot akan terus melanjutkan misinya untuk mengenali posisi selanjutnya hingga robot berhasil sampai pada posisi tujuan. Namun jika robot gagal mengenali posisi [5,1] atau posisi-posisi selanjutnya, maka robot dianggap gagal menjalankan misinya.

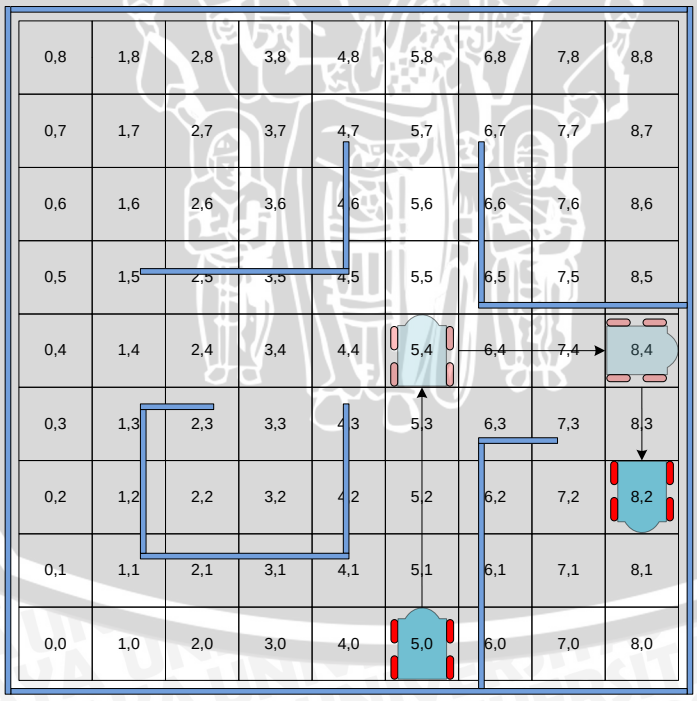
Pengujian sistem secara keseluruhan bertujuan untuk mengetahui tingkat keberhasilan dari sistem pengenalan posisi berdasakan *grid-based map* yang diterapkan pada robot beroda. Tingkat keberhasilan dinyatakan berdasarkan kemampuan robot untuk mengenali setiap posisi yang akan dilewati oleh robot untuk mencapai posisi tujuan yang telah ditentukan. Hal ini dilakukan dengan cara mencatat hasil pengenalan posisi berupa koordinat yang ditampilkan pada LCD karakter dan membandingkannya dengan koordinat yang akan dilalui oleh robot untuk mencapai posisi tujuan yang telah ditetapkan sebelumnya.

Pengujian sisitem secara keseluruhan dilakukan sebanyak lima kali dengan kondisi posisi yang berbeda-beda untuk setiap pengujian. Hasil yang diharapkan dalam pengujian ini adalah sistem pengenalan posisi berdasarkan *grid-based map* ini mempu membantu robot untuk mengenali posisi keberadaan robot

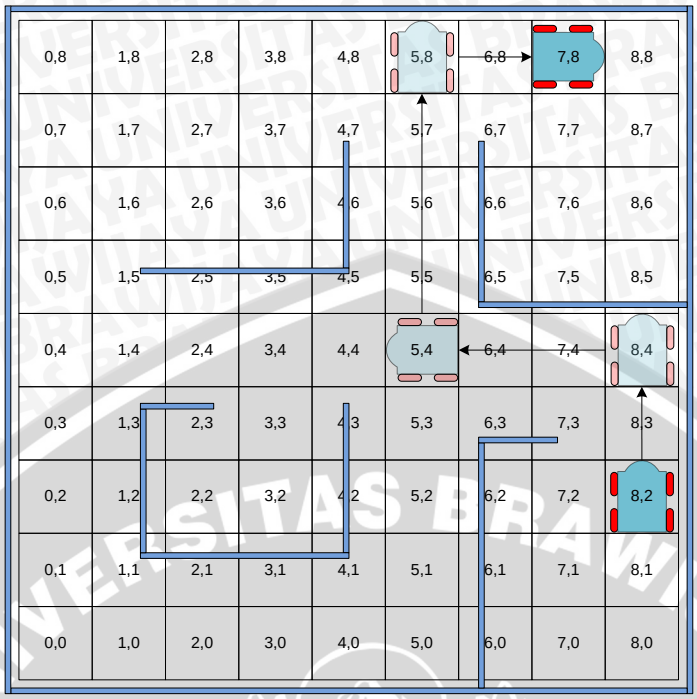
di arena/arena pertandingan KRPAI. Pengujian sistem secara keseluruhan dilakukan melalui prosedur sebagai berikut:

1) Menentukan koordinat acuan yang akan dijadikan sebagai posisi awal, posisi tujuan dan posisi yang akan dilewati robot berdasarkan *grid-based map* yang telah dirancang.
2) Meletakkan robot pada posisi awal, di mana robot arah hadap robot mengarah pada satu dari empat arah yang telah ditentukan, kemudian memilih mode untuk pengujian keseluruhan.
3) Menghidupkan robot dan menekan tombol *start*.
4) Mencatat hasil pembacaan posisi robot yang tertampil pada LCD.
5) Mengulang prosedur ke 2 sampai 5 hingga seluruh tahap pengujian selesai.

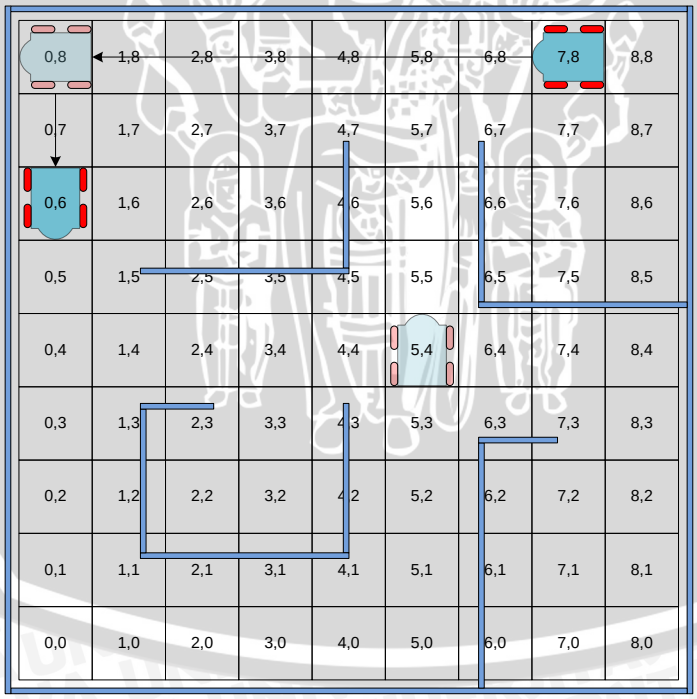
Berdasarkan langkah kedua pada prosedur pengujian, maka kita perlu menentukan posisi awal, posisi tujuan dan posisi yang akan dilewati oleh robot untuk setiap pengujian. Gambar 5.15 sampai Gambar 5.19 menunjukkan perencanaan posisi awal, posisi tujuan dan posisi yang akan dilewati robot untuk 5 kali proses pengujian.



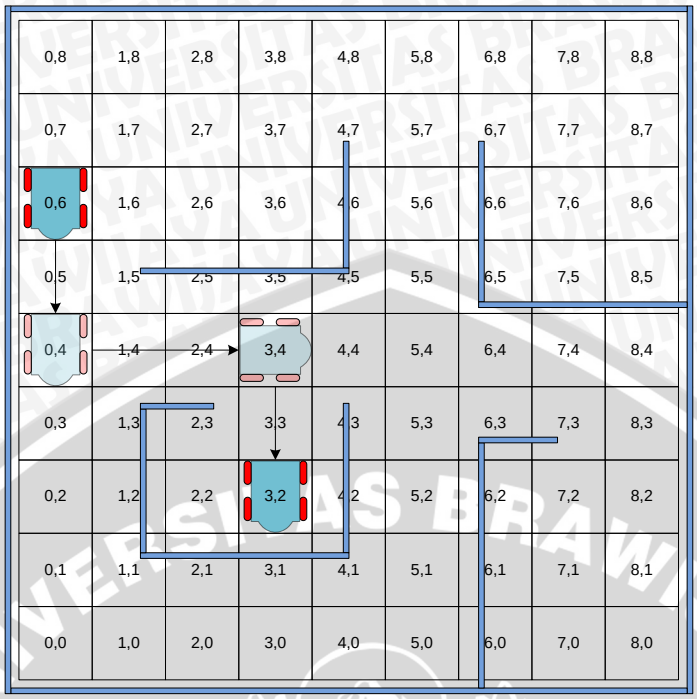
Gambar 5.15 Posisi Awal, Posisi Tujuan dan Posisi yang akan Dilewati Oleh Robot untuk Pengujian Pertama



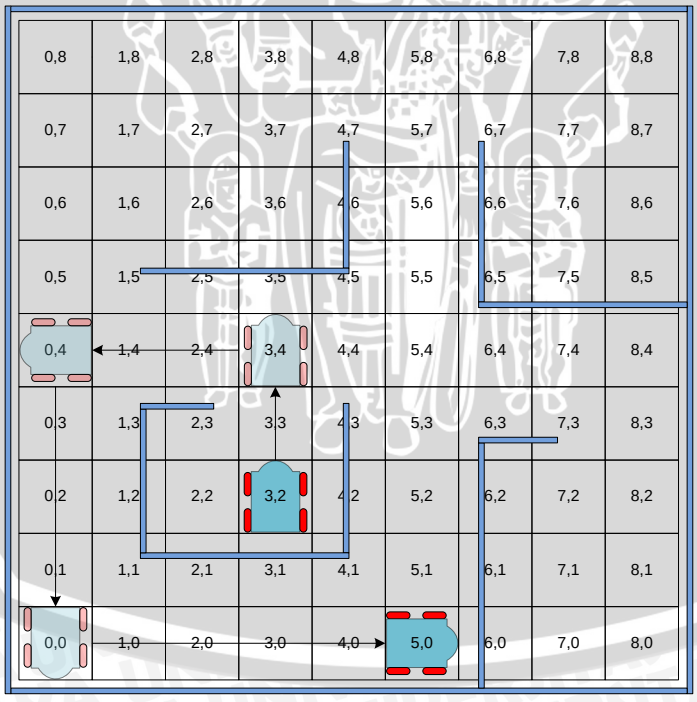
Gambar 5.16 Posisi Awal, Posisi Tujuan dan Posisi yang akan Dilewati Oleh Robot untuk Pengujian Kedua



Gambar 5.17 Posisi Awal, Posisi Tujuan dan Posisi yang akan Dilewati Oleh Robot untuk Pengujian Ketiga



Gambar 5.18 Posisi Awal, Posisi Tujuan dan Posisi yang akan Dilewati Oleh Robot untuk Pengujian Keempat



Gambar 5.19 Posisi Awal, Posisi Tujuan dan Posisi yang akan Dilewati Oleh Robot untuk Pengujian Kelima

Data hasil pengujian sistem secara keseluruhan untuk pengujian ke-1 sampai pengujian ke-5 ditunjukkan dalam Tabel 5.6.

Tabel 5.6 Data Hasil Pengujian Sistem Secara Keseluruhan

Pengujian ke-	Tampilan LCD			Keterangan Tambahan
	Posisi *Start*	Posisi yang Dilewati	Posisi *Stop*	
1	5,0	5,1 5,2 5,3 5,4 6,4 7,4 8,4 8,3	8,2	Robot berputar dua kali pada posisi 7,4 saat koreksi arah menuju koordinat 8,4. Berhasil sampai tujuan
2	8,2	8,3 8,4 7,4 6,4 5,4 5,5 5,6 5,7 5,8 6,8	7,8	Robot berputar sekali pada posisi 5,7 saat koreksi arah menuju koordinat 5,8. Berhasil sampai tujuan
3	7,8	6,8 5,8 4,8 3,8 2,8 1,8 0,8 0,7	0,6	Berhasil sampai tujuan
4	0,6	0,5 0,4 1,4 2,4 3,4 3,3	3,2	Robot berputar dua kali pada posisi 0,5 saat koreksi arah menuju koordinat 0,4, sekali pada posisi 1,4 saat koreksi arah menuju koordinat 2,4. Berhasil sampai tujuan
5	3,2	3,3 3,4 2,4	5,0	Berhasil sampai tujuan

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

Berdasarkan hasil pengujian, dapat diketahui robot berhasil menggunakan peta yang telah dirancang untuk mengenali posisi robot di arena pertandingan. Selain itu robot juga berhasil mencapai tujuan yang dikehendaki. Adapun robot berputar pada posisi 5,4 saat koreksi arah menuju koordinat 6,4 dan pada posisi 5,8 saat koreksi arah menuju koordinat 6,8 dikarenakan proses perputaran robot yang terlalu cepat sehingga pembacaan data kompas menjadi sedikit terlambat dibandingkan dengan perputaran robot.

Secara garis besar, peta yang dirancang sudah dapat membantu robot dalam mengenali posisi robot di arena pertandingan walaupun masih terdapat kelemahan dalam sistem pengenalaan posisi yang dirancang, seperti arah orientasi lapangan yang harus diketahui terlebih dahulu dan tidak adanya *obstacle* pada arena pertandingan. Hasil pengujian keseluruhan menunjukkan bahwa robot mampu mengenali posisinya di arena pertandingan ketika robot bergerak mencapai tujuan melewati jalur yang sudah ditetapkan sebelumnya.

BAB VI
PENUTUP

6.1. Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan, maka dapat disimpulkan beberapa hal sebagai berikut:

1. Arena pertandingan KRPAI dapat diubah ke dalam *grid-based map* dengan ukuran 26 cm × 26 cm tiap *grid*. Total *grid* yang dihasilkan adalah sebanyak 81 *grid*. Masing-masing *grid* berisi data jarak robot terhadap dinding bagian depan, belakang, kiri dan kanan.
2. Untuk mengimplementasikan *grid-based map* ke dalam mikrokontroler ATmega128 dapat dilakukan dengan mengubah data tiap *grid* ke dalam bentuk *array*, lalu diprogram ke dalam mikrokontroler ATmega128.
3. Untuk mengenali posisi robot di arena pertandingan, data yang diperoleh robot dari sensor ultrasonik mengenai jarak robot terhadap dinding dibandingkan dengan data peta yang sudah dimasukkan ke dalam mikrokontroler dalam bentuk *array*. Modul CMPS03 *Magnetic Compass* digunakan sebagai penentu referensi arah sehingga dalam proses pembandingan data yang dibandingkan sesuai antara data yang diperoleh dari sensor ultrasonik PING))) dengan data peta yang telah disimpan di dalam memori mikrokontroler. Dari hasil pengujian dapat diketahui tingkat keberhasilan dari sistem pengenalan posisi yang dirancang sebesar 90%.
4. Sensor ultrasonik PING))) dapat diakses menggunakan fungsi *port* I/O dengan bantuan *timer1* dari mikrokontroler ATmega8. Untuk mengakses sensor, I/O yang digunakan untuk mengakses dikonfigurasi sebagai *port* keluaran kemudian memberikan sinyal keluaran *high* selama 2 µs. Setelah itu, *port* I/O yang digunakan diubah menjadi pin masukan yang digunakan untuk membaca sinyal dari sensor ultrasonik PING))). Jika sinyal masukan yang diterima adalah logika *high*, maka fungsi *timer1* akan diaktifkan dan digunakan untuk menghitung lama waktu sampai sinyal masukan berubah menjadi *low*. Lama waktu perubahan sinyal

masukan dari *high* ke *low* inilah yang nantinya akan dikonversi menjadi jarak hasil pengukuran sensor ultrasonik PING))). Dari hasil pengujian, sensor ultrasonik PING))) memiliki tingkat keakuratan dalam pembacaan jarak terhadap objek berupa dinding dengan kesalahan rata-rata sebesar 0,23 cm dengan kesalahan terbesar 0,4 cm.

5. Modul CMPS03 *Magnetic Compass* dapat diakses menggunakan jalur komunikasi I_2C. Untuk memperoleh data mengenai hasil pembacaan arah mata angin berupa sudut dapat dilakukan dengan cara mengakses dua buah register dengan alamat 0x02 (H *byte*) dan 0x03 (L *byte*). Data dalam register dengan alamat 0x02 dan 0x03 jika digabungkan bernilai 0-3599 dalam skala penuh. Data hasil penggabungan dari dua buah register ini kemudian dibagi dengan 10 untuk mendapatkan hasil pembacaan sudut sebesar 0°-359,9°. Dari hasil pengujian, modul CMPS03 *Magnetic Compass* memiliki kesalahan rata-rata pembacaan sudut sebesar 1,8° derajat dengan kesalahan pembacaan terbesar ± 4°.
6. Penggunaan *Grid-Based Map* sebagai sistem pengenalan posisi yang dirancang mampu membantu robot untuk mengetahui posisi di arena pertandingan.

6.2. Saran

Dalam perancangan dan pembuatan alat ini masih terdapat banyak kekurangan. Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Membuat peta dengan *grid* yang lebih kecil, sehingga pengenalan posisi lebih tepat.
2. Menambahkan sistem *path planning*, sehingga robot dapat memilih jalur yang akan dilewati secara otomatis.
3. Menambahkan sistem pengingat posisi yang telah dilewati robot.
4. Penggunaan metode lain yang lebih baik, sehingga sistem dapat diterapkan tanpa batasan tertentu (dapat memperbaiki kelemahan sistem).

DAFTAR PUSTAKA

Akbar, Arnas Elmiawan. 2013. *Implementasi Sistem Navigasi Wall Following Menggunakan Kontroler PID dengan Metode Tuning pada Robot Kontes Robot Cerdas Indonesia (KRCI) Divisi Senior Beroda*. Malang: Skripsi Jurusan Teknik Elektro FT-UB.

Atmel. 2008. *ATMEGA128/ATMEGA128L, 8-bit AVR with 128 Kbytes in System Programable Flash.* http://www.atmel.com/products/microcontrollers/avr/default.aspx?tab=documents, diakses tanggal 24 Juni 2013.

Atmel. 2003. *ATMEGA8/ATMEGA8L, 8-bit AVR with 8 Kbytes in System Programable Flash.* http://www.atmel.com/products/microcontrollers/avr/default.aspx?tab=documents, diakses tanggal 24 Juni 2013.

Borenstein, J. 1996. *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan Press.

Braunl, Thomas. 2008. Embedded Robotics, Mobile Robot Design and Application with Embedded Systems, Third Edition. Verlag Berlin Heidelberg: Springer.

DIKTI. 2012. *Panduan Kontes Robot Pemadam Api Indonesia (KRPAI)*. Jakarta: DIKTI

Gapaiasa, Y. 2011. *Implementasi Sensor Kompas Digital untuk Memonitor Arah Muatan Roket*. Malang: Skripsi Jurusan Teknik Elektro FT-UB

Hitachi. 2006. *LMB162 Specification*. http://datasheetcatalog.com, diakses tanggal 24 Juni 2013.

McComb, Gordon dan Myke Predko. 2006. *Robot Builder's Bonanza, Third Edition*. New York: McGraw-Hill.

Megasakti, M. C. 2010. *Rancang Bangun Auto Tracking dengan Menggunakan Microcontroller, GPS, SAT Finder dan Digital Compass untuk Sinkronisasi Azimuth Antena Terhadap Satelit Cakrawarta-2*. Depok: Skripsi Program S1 Ekstensi Teknik Elektro FT-UI.

Parallax. 2008. *PING))) Ultrasonic Distance Sensor*. California: Parallax.

Siegwart, R. dan Illah R. N. 2004. *Introduction to Autonomous Mobile Robots*. London: MIT Press.

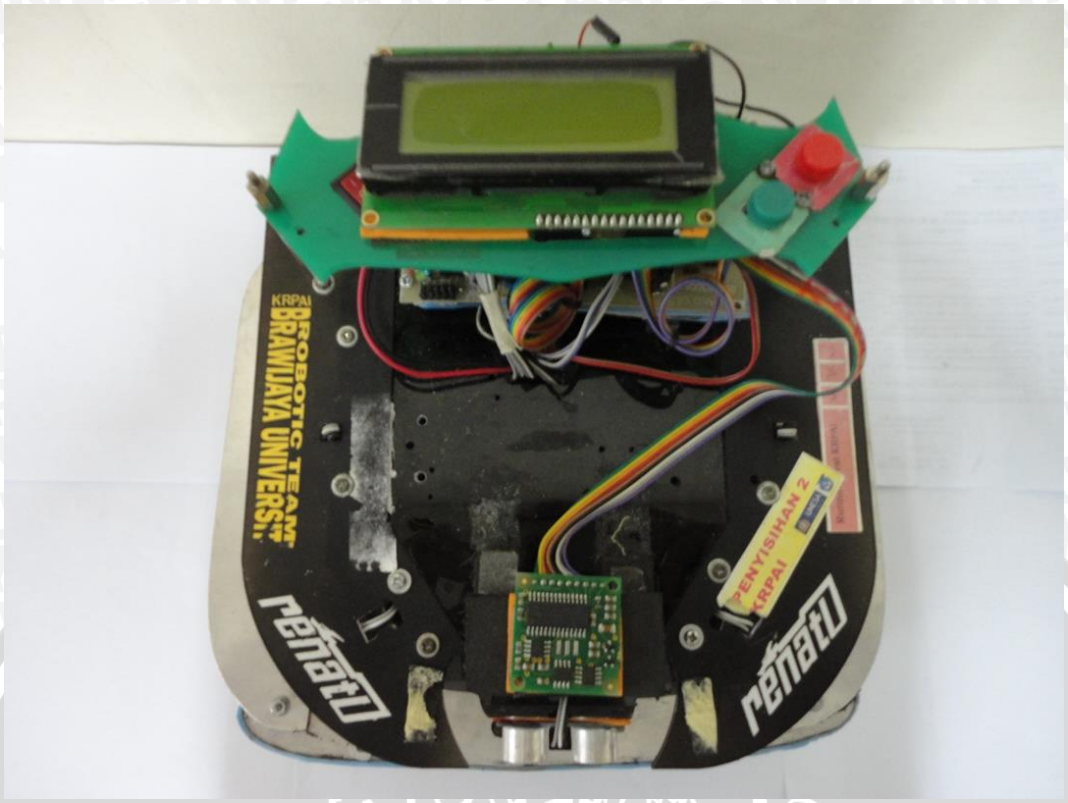
Taufik, Ahmad Sulkhan. 2013. *Sistem Navigasi Waypoint pada Autonomous Mobile Robot*. Malang: Skripsi Jurusan Teknik Elektro FT-UB.

Tim Digiware. 2008. *CMPS03-Magnetic Compass*. Surabaya: Digiware.

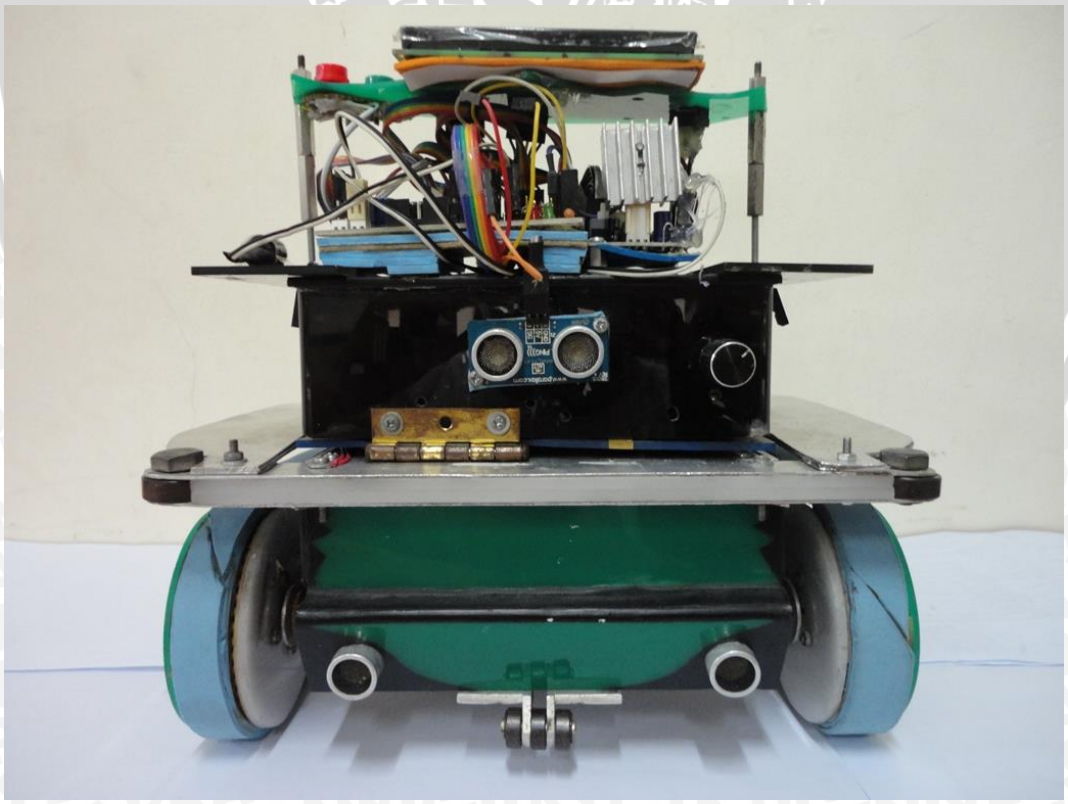
Zulkarnain, M. Y. 2011. *Sistem Pemetaan Posisi pada Robot Kontes Robot Cerdas Indonesia (KRCI) Divisi Senior Beroda dengan Metode Decision Tree*. Malang: Skripsi Jurusan Teknik Elektro FT-UB.

LAMPIRAN 1

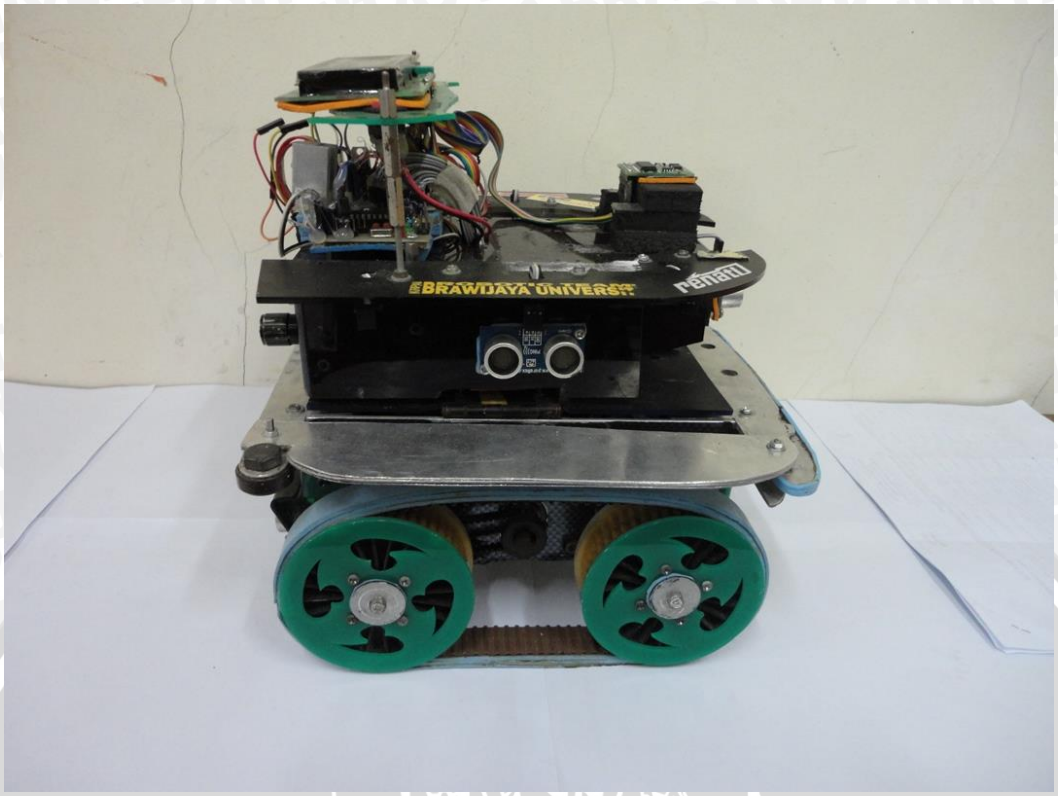
FOTO ALAT



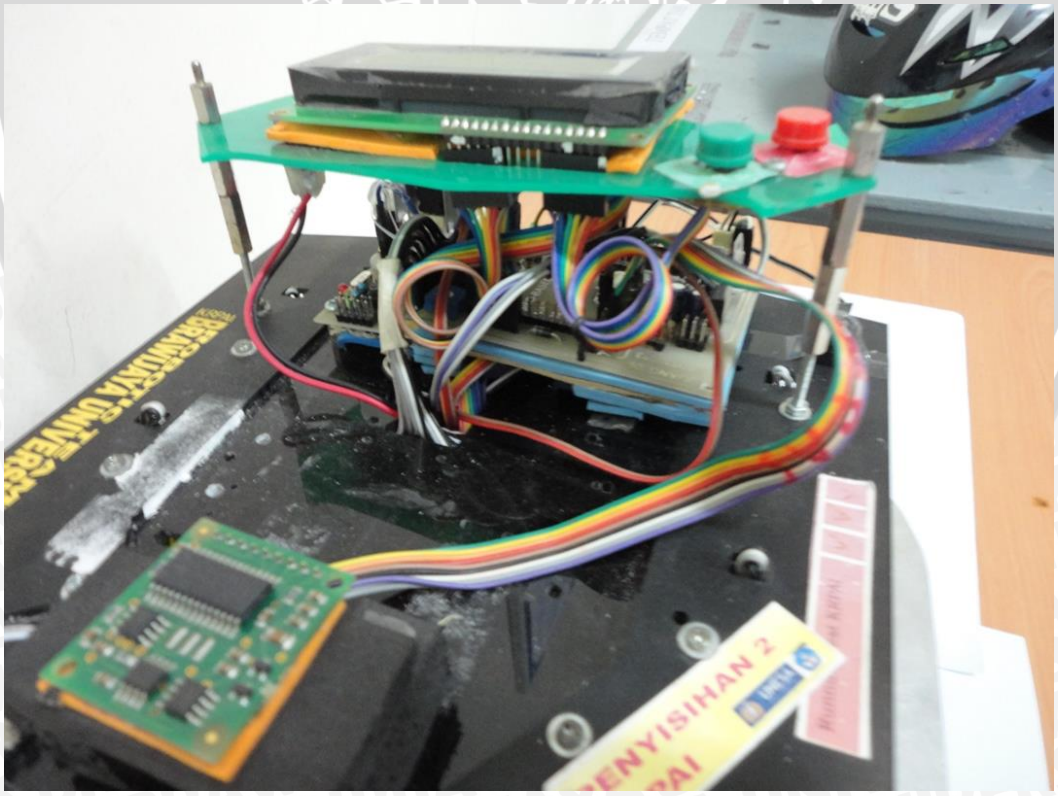
Gambar Robot Tampak Atas.



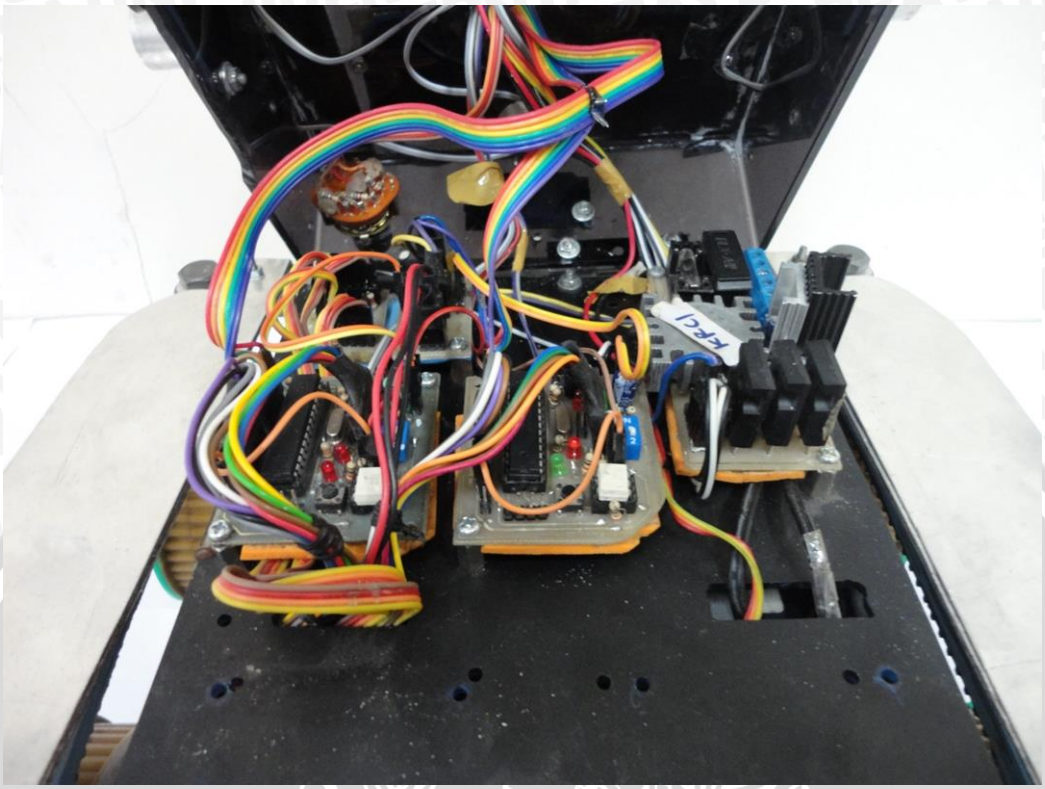
Gambar Robot Tampak Belakang.



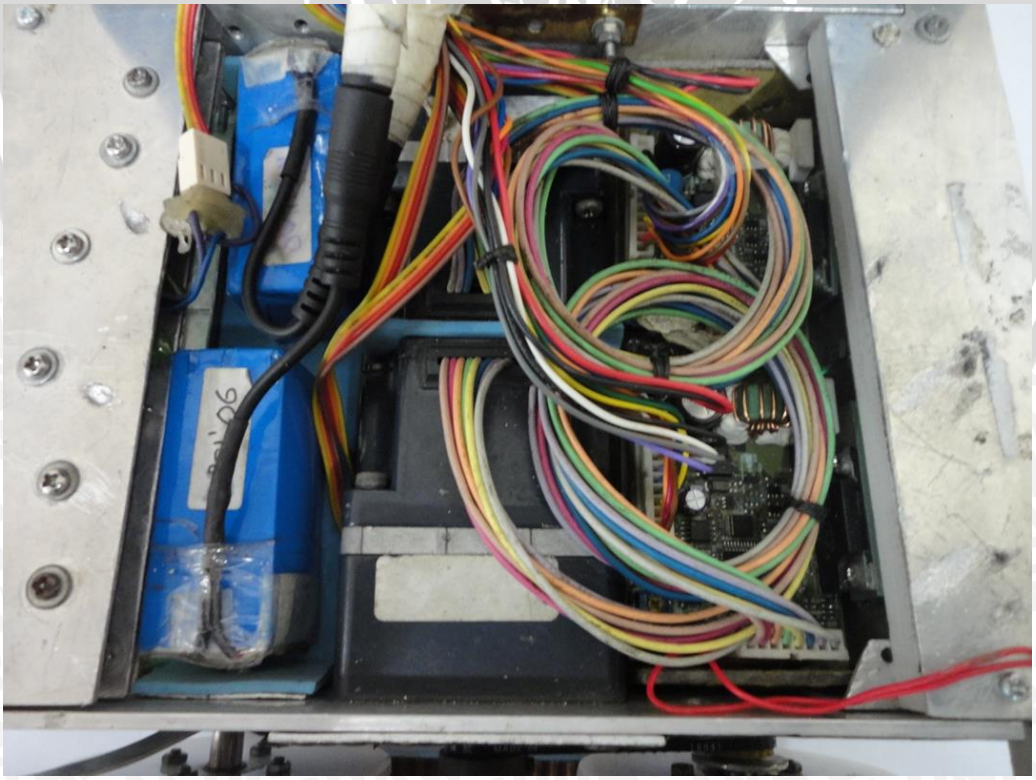
Gambar Robot Tampak Samping



Gambar Modul CMPS03 *Magnetic Compass*, Mikrokontroler Utama, dan Mikrokontroler Pengatur Sensor Ping))).



Gambar Mikrokontroler Pengatur *Driver* Motor



Gambar *Driver* Motor DC, Motor DC dan Baterai LiIon

LAMPIRAN 2

LISTING PROGRAM

Listing Program Mikrokontroler Utama (ATmega128)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Pengenalan Posisi dengan Grid-Based Map
Version : 1.1
Date    : 7/12/2013
Author  : Nur Iskandar Juang
Company : Teknik Elektro Universitas Brawijaya
Comments:


Chip type               : ATmega128
Program type            : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 1024
*****************************************************/

#include <mega128.h>
#include <spi.h>
#include <i2c.h>
#include <alcd.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <delay.h>

#define ADC_VREF_TYPE 0x20
#define ss_Right      PORTB.4
#define ss_Left       PORTB.0
#define brake         120,120
#define COMPASS_WRITE 192
#define COMPASS_READ  193
#define LR  0
#define R1  1
#define R2  2
#define R3  3
#define R4  4
#define NORTH 0
#define EAST  1
#define SOUTH 2
#define WEST  3
#define NOL 0

#define NEGATIF 1
#define POSITIF 2
#define sys_on  sys.tc=1;sys.timer=2000
#define sys_off  sys.tc=0;sys.timer=2000
#define sisi_kanan   1
#define sisi_kiri    2

//===== I2C COMPASS =====//
// I2C Bus functions
#asm
   .equ __i2c_port=0x12 ;PORTD
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

//===== UART PING))) =====//
#ifndef RXB8
#define RXB8 1
#endif

#ifndef TXB8
#define TXB8 0
#endif

#ifndef UPE
#define UPE 2
#endif

#ifndef DOR
#define DOR 3
#endif

#ifndef FE
#define FE 4
#endif

#ifndef UDRE
#define UDRE 5
#endif

#ifndef RXC
#define RXC 7
#endif

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<DOR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART0 Receiver buffer
```

```
#define RX_BUFFER_SIZE0 6
char rx_buffer0[RX_BUFFER_SIZE0];

#if RX_BUFFER_SIZE0 <= 256
unsigned char rx_wr_index0,rx_rd_index0,rx_counter0;
#else
unsigned int rx_wr_index0,rx_rd_index0,rx_counter0;
#endif

// This flag is set on USART0 Receiver buffer overflow
bit rx_buffer_overflow0;

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART0 Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter0==0);
data=rx_buffer0[rx_rd_index0++];
#if RX_BUFFER_SIZE0 != 256
if (rx_rd_index0 == RX_BUFFER_SIZE0) rx_rd_index0=0;
#endif
#asm("cli")
--rx_counter0;
#asm("sei")
return data;
}
#pragma used-
#endif

//===== ADC =====//
// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}

//===== FUNGSI PROTOTYPE =====//
//===== MAP BASED =====//
int data[5];
char data_peta[10][2];
int x_prev,x_now,x_next,y_prev,y_now,y_next,x_des,y_des;
int heading,sudut,sudut_koreksi,sudut_error;
char koor_x,koor_y;
int a,b,c,d,e,f,g=0,i;
bit peta_ok=0,peta_cocok=0,ok;

eeprom int arah_0=280,arah_90=15,arah_180=105,arah_270=193;
eeprom int arah_0_l,arah_0_h,arah_90_l,arah_90_h,arah_180_l,arah_180_h,arah_270_l,arah_270_h;
eeprom int koreksi_0_l,koreksi_0_h,koreksi_90_l,koreksi_90_h;
eeprom int koreksi_180_l,koreksi_180_h,koreksi_270_l,koreksi_270_h;

//peta buat putchar_8
unsigned char peta [9][9][5] = {
    {
        {137, 137, 3, 3, 0},    //[0,0]
        {137, 137, 29, 3, 0},   //[0,1]
        {137, 17, 55, 3, 0},    //[0,2]
        {133, 17, 81, 3, 0},    //[0,3]
        {107, 137, 107, 3, 0},  //[0,4]
        {81, 137, 133, 3, 0},   //[0,5]
        {55, 89, 137, 3, 3},    //[0,6]
        {29, 103, 137, 3, 3},   //[0,7]
        {3, 137, 137, 3, 3}     //[0,8]
    },
    {
        {137, 111, 3, 29, 0},   //[1,0]
        {137, 111, 29, 29, 0},  //[1,1]
        {137, 0, 55, 29, 0},    //[1,2]
        {133, 0, 81, 29, 0},    //[1,3]
        {107, 137, 107, 29, 0}, //[1,4]
        {81, 111, 133, 29, 0},  //[1,5]
        {55, 63, 137, 29, 3},   //[1,6]
        {29, 137, 137, 29, 3},  //[1,7]
        {3, 137, 137, 29, 3}    //[1,8]
    },
    {
        {17, 85, 3, 55, 0},     //[2,0]
        {0, 85, 29, 55, 0},     //[2,1]
        {18, 37, 7, 7, 4},      //[2,2]
        {0, 37, 33, 7, 4},      //[2,3]
        {14, 137, 6, 55, 0},    //[2,4]
        {0, 85, 32, 55, 0},     //[2,5]
        {55, 37, 10, 55, 3},    //[2,6]
        {29, 137, 36, 55, 3},   //[2,7]
        {3, 137, 62, 55, 3}     //[2,8]
    },
    {
        {17, 59, 3, 81, 0},     //[3,0]
        {0, 59, 29, 81, 0},     //[3,1]
        {66, 11, 7, 33, 4},     //[3,2]
        {40, 11, 33, 33, 4},    //[3,3]
        {14, 133, 59, 81, 0},   //[3,4]
        {0, 59, 85, 81, 0},     //[3,5]
```

```
    {55, 11, 10, 81, 3},    //[3,6]
    {29, 137, 36, 81, 3},   //[3,7]
    {3, 137, 62, 81, 3}     //[3,8]
},
{
    {137, 33, 3, 107, 0},   //[4,0]
    {137, 33, 29, 107, 0},  //[4,1]
    {137, 33, 55, 0, 0},    //[4,2]
    {133, 107, 81, 0, 0},   //[4,3]
    {107, 107, 107, 107, 0},//[4,4]
    {81, 33, 133, 107, 0},  //[4,5]
    {55, 33, 137, 0, 0},    //[4,6]
    {29, 107, 137, 107, 0}, //[4,7]
    {3, 107, 137, 107, 0}   //[4,8]
},
{
    {137, 7, 3, 133, 0},    //[5,0]
    {137, 7, 29, 133, 0},   //[5,1]
    {137, 7, 55, 13, 0},    //[5,2]
    {133, 81, 81, 13, 0},   //[5,3]
    {107, 81, 107, 133, 0}, //[5,4]
    {81, 7, 133, 133, 0},   //[5,5]
    {55, 7, 137, 13, 0},    //[5,6]
    {29, 81, 137, 133, 0},  //[5,7]
    {3, 81, 137, 133, 0}    //[5,8]
},
{
    {58, 55, 3, 0, 1},      //[6,0]
    {32, 55, 29, 0, 1},     //[6,1]
    {6, 55, 55, 0, 1},      //[6,2]
    {28, 55, 0, 39, 0},     //[6,3]
    {2, 55, 18, 137, 0},    //[6,4]
    {81, 55, 0, 0, 2},      //[6,5]
    {55, 55, 22, 0, 2},     //[6,6]
    {29, 55, 48, 137, 2},   //[6,7]
    {3, 55, 74, 137, 2}     //[6,8]
},
{
    {106, 29, 3, 17, 1},    //[7,0]
    {80, 29, 29, 17, 1},    //[7,1]
    {54, 29, 55, 17, 1},    //[7,2]
    {28, 29, 81, 65, 0},    //[7,3]
    {2, 29, 107, 137, 0},   //[7,4]
    {81, 29, 0, 17, 2},     //[7,5]
    {55, 29, 22, 17, 2},    //[7,6]
    {29, 29, 48, 137, 2},   //[7,7]
    {3, 29, 74, 137, 2}     //[7,8]
},
{
    {106, 3, 3, 43, 1},     //[8,0]
    {80, 3, 29, 43, 1},     //[8,1]
    {54, 3, 55, 43, 1},     //[8,2]
    {28, 3, 81, 91, 0},     //[8,3]
    {2, 3, 107, 137, 0},    //[8,4]
    {81, 3, 0, 43, 2},      //[8,5]
    {55, 3, 22, 43, 2},     //[8,6]
    {29, 3, 48, 137, 2},    //[8,7]
    {3, 3, 74, 137, 2}      //[8,8]
}
};

unsigned char petaxy_00[2][5] =
{
    {137, 137, 3, 3, 0},
    {118, 137, 3, 3, 0}
};

unsigned char petaxy_01[4][5] =
{
    {137, 137, 29, 3, 0},
    {92, 137, 29, 3, 0},
    {137, 17, 29, 3, 0},
    {92, 17, 29, 3, 0}
};

unsigned char petaxy_03[3][5] =
{
    {133, 17, 81, 3, 0},
    {133, 89, 81, 3, 0},
    {137, 137, 81, 3, 0}
};

unsigned char petaxy_04[2][5] =
{
    {107, 89, 107, 3, 0},
    {107, 137, 107, 3, 0}
};

unsigned char petaxy_05[3][5] =
{
    {81, 137, 133, 3, 0},
    {81, 17, 133, 3, 0},
    {81, 89, 133, 3, 0}
};

unsigned char petaxy_07[2][5] =
{
    {29, 89, 137, 3, 0},
    {29, 137, 137, 3, 0}
};

unsigned char petaxy_10[2][5] =
{
    {137, 111, 3, 29, 0},
    {17, 111, 3, 29, 0}
};

unsigned char petaxy_11[3][5] =
```

```
{
    {137, 111, 29, 29, 0},
    {137, 0, 29, 29, 0},
    {17, 111, 29, 29, 0}
};

unsigned char petaxy_13[2][5] =
{
    {133, 0, 81, 29, 0},
    {133, 137, 81, 29, 0}
};

unsigned char petaxy_14[6][5] =
{
    {107, 137, 107, 29, 0},
    {14, 137, 6, 29, 0},
    {107, 63, 107, 29, 0},
    {14, 63, 6, 29, 0},
    {107, 111, 107, 29, 0},
    {14, 111, 6, 29, 0}
};

unsigned char petaxy_15[3][5] =
{
    {81, 111, 133, 29, 0},
    {81, 63, 133, 29, 0},
    {0, 111, 32, 29, 0}
};

unsigned char petaxy_16[2][5] =
{
    {55, 63, 137, 29, 3},
    {55, 63, 10, 29, 3}
};

unsigned char petaxy_17[4][5] =
{
    {29, 137, 137, 29, 3},
    {29, 63, 137, 29, 3},
    {29, 137, 36, 29, 3},
    {29, 63, 36, 29, 3}
};

unsigned char petaxy_18[2][5] =
{
    {3, 137, 137, 29, 3},
    {3, 137, 62, 29, 0}
};

unsigned char petaxy_22[2][5] =
{
    {18, 37, 7, 7, 4},
    {66, 37, 7, 7, 4}
};

unsigned char petaxy_23[2][5] =
{
    {0, 37, 33, 7, 4},
    {40, 37, 33, 7, 4}
};

unsigned char petaxy_24[2][5] =
{
    {14, 137, 6, 55, 0},
    {14, 137, 59, 55, 0}
};

unsigned char petaxy_25[2][5] =
{
    {0, 85, 32, 55, 0},
    {0, 85, 85, 55, 0}
};

unsigned char petaxy_27[2][5] =
{
    {29, 137, 36, 55, 3},
    {29, 37, 36, 55, 3}
};

unsigned char petaxy_32[2][5] =
{
    {66, 11, 7, 33, 4},
    {18, 11, 7, 33, 4},
};

unsigned char petaxy_33[2][5] =
{
    {40, 11, 33, 33, 4},
    {40, 133, 33, 81, 4}
};

unsigned char petaxy_37[2][5] =
{
    {29, 137, 36, 81, 3},
    {29, 11, 36, 81, 3}
};

unsigned char petaxy_40[3][5] =
{
    {137, 33, 3, 107, 0},
    {118, 33, 3, 107, 0},
    {17, 33, 3, 107, 0}
};

unsigned char petaxy_41[4][5] =
{
    {137, 33, 29, 107, 0},
    {118, 33, 29, 107, 0},
```

```
    {0, 33, 29, 107, 0},
    {137, 33, 29, 0, 0}
};

unsigned char petaxy_43[3][5] =
{
    {133, 107, 81, 0, 0},
    {133, 33, 81, 0, 0},
    {133, 107, 81, 107, 0}
};

unsigned char petaxy_44[6][5] =
{
    {107, 107, 107, 107, 0},
    {107, 107, 107, 59, 0},
    {107, 33, 107, 107, 0},
    {14, 107, 59, 107, 0},
    {14, 107, 59, 59, 0},
    {14, 33, 59, 107, 0}
};

unsigned char petaxy_45[3][5] =
{
    {81, 33, 133, 107, 0},
    {0, 33, 85, 107, 0},
    {81, 33, 133, 0, 0}
};

unsigned char petaxy_47[3][5] =
{
    {29, 107, 137, 107, 0},
    {29, 107, 36, 107, 0},
    {29, 33, 137, 0, 0}
};

unsigned char petaxy_48[2][5] =
{
    {3, 107, 137, 107, 0},
    {3, 107, 62, 107, 0}
};

unsigned char petaxy_50[3][5] =
{
    {137, 7, 3, 133, 0},
    {106, 7, 3, 133, 0},
    {118, 7, 3, 133, 0}
};

unsigned char petaxy_51[4][5] =
{
    {137, 7, 29, 133, 0},
    {92, 7, 3, 133, 0},
    {137, 7, 29, 13, 0},
    {92, 7, 3, 133, 0}
};

unsigned char petaxy_53[3][5] =
{
    {133, 81, 81, 13, 0},
    {133, 7, 81, 13, 0},
    {133, 81, 81, 133, 0}
};

unsigned char petaxy_54[3][5] =
{
    {107, 81, 107, 133, 0},
    {107, 7, 107, 133, 0},
    {107, 81, 107, 93, 0}
};

unsigned char petaxy_55[2][5] =
{
    {81, 7, 133, 133, 0},
    {81, 7, 133, 13, 0}
};

unsigned char petaxy_57[4][5] =
{
    {29, 81, 137, 133, 0},
    {29, 7, 137, 133, 0},
    {29, 81, 137, 13, 0},
    {29, 7, 137, 13, 0}
};

unsigned char petaxy_63[6][5] =
{
    {28, 55, 0, 39, 0},
    {28, 55, 81, 39, 0},
    {28, 55, 0, 111, 0},
    {28, 55, 81, 111, 0},
    {28, 55, 0, 137, 0},
    {28, 55, 81, 137, 0}
};

unsigned char petaxy_64[8][5] =
{
    {2, 55, 18, 137, 0},
    {2, 55, 107, 137, 0},
    {2, 55, 18, 39, 0},
    {2, 55, 107, 39, 0},
    {2, 55, 18, 87, 0},
    {2, 55, 107, 87, 0},
    {2, 55, 18, 111, 0},
    {2, 55, 107, 111, 0}
};

unsigned char petaxy_67[2][5] =
{
```

```
    {29, 55, 48, 137, 2},
    {29, 55, 137, 137, 2}
};

unsigned char petaxy_68[2][5] =
{
    {3, 55, 74, 137, 2},
    {3, 55, 137, 137, 2}
};

unsigned char petaxy_70[2][5] =
{
    {106, 29, 3, 17, 1},
    {58, 29, 3, 17, 1}
};

unsigned char petaxy_71[2][5] =
{
    {80, 29, 29, 17, 1},
    {32, 29, 29, 17, 1}
};

unsigned char petaxy_72[2][5] =
{
    {54, 29, 55, 17, 1},
    {6, 29, 55, 17, 1}
};

unsigned char petaxy_73[5][5] =
{
    {28, 29, 81, 65, 0},
    {28, 29, 81, 17, 0},
    {28, 29, 81, 137, 0},
    {28, 29, 0, 65, 0},
    {28, 29, 0, 137, 0}
};

unsigned char petaxy_74[4][5] =
{
    {2, 29, 107, 137, 0},
    {2, 29, 18, 137, 0},
    {2, 29, 107, 65, 0},
    {2, 29, 18, 65, 0}
};

unsigned char petaxy_77[2][5] =
{
    {29, 29, 48, 137, 2},
    {29, 29, 48, 17, 2}
};

unsigned char petaxy_83[3][5] =
{
    {28, 3, 81, 91, 0},
    {28, 3, 81, 43, 0},
    {28, 3, 81, 137, 0}
};

unsigned char petaxy_84[2][5] =
{
    {2, 3, 107, 137, 0},
    {2, 3, 107, 91, 0}
};

unsigned char petaxy_87[2][5] =
{
    {29, 3, 48, 137, 2},
    {29, 3, 48, 43, 2}
};

//===== MODE PENGECEKAN =====//
void cek_kompas();
void buzz(int n);
void uart_speed_setting(char mode);
void cek_orientasi(void);
void cek_ping();
void cek_koreksi();
void cek_sudut();

//===== SENSING =====//
unsigned char Akses_CMPS03(char alamat);
void DATA_CMPS03();
void kalibrasi_arena();

//===== PERGERAKAN =====//
void koreksi(char arah);
void rotasi_CW();
void rotasi_CCW();
void putar_rotary(long int derajat);
void send_umc(int left,int right);

//===== INTERRUPT =====//
interrupt [EXT_INT2] void ext_int2_isr(void);
interrupt [EXT_INT3] void ext_int3_isr(void);
interrupt [TIM1_OVF] void timer1_ovf_isr(void);

//===== MAPPING STRATEGY =====//
void start_detection();
void position_detection();
void node2node(int x_1, int y_1);
void petaxy(int x, int y, int z);

//===== DISPLAY =====//
void display_awal(void);
void display_peta();

//===== INISIALISASI =====//
void RENATO(void);
```

```
void inisialisasi_system(void);

//===== PERCOBAAN =====//
unsigned char mx_now[3],mx_up[3],mx_right[3],mx_low[3],mx_left[3];
unsigned char my_now[2],my_up[2],my_right[2],my_low[2],my_left[2];

//===== GLOBAL VARIABLE =====//
int RS;
unsigned int data_cmp;
unsigned char Msg[20],Msg1[20];
unsigned char Ping_1[9],Ping_2[9],Ping_3[9],Ping_4[9];
int lama;
int mode;
unsigned int rotary_kanan=0;
unsigned int rotary_kiri=0;
unsigned char sonar_complete;
int jarak_A,jarak_B,jarak_C,jarak_D;
bit L_inc=0,R_inc=0;

void main(void)
{
    inisialisasi_system();
    uart_speed_setting(0);
    delay_ms(200);
    display_awal();
    while(PINF.1==1){};
    lcd_clear();
    delay_ms(250);
    #asm("sei");
    mode=(int)read_adc(0);
    mode/=35;

    switch(mode)
        {
                case 1:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(2);RENATO();break;
                case 2:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(3);lcd_gotoxy(0,0);lcd
_putsf("= KALIBRASI ARENA  =");kalibrasi_arena();break;
                case 3:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(4);sudut =
0;while(1){start_detection();};break;
                case 4:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(5);putar_rotary(90);de
lay_ms(1000);putar_rotary(-90);break;
                case 5:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(6);while(1){cek_ping()
;};break;
                case 6:
putchar(0);delay_ms(50);putchar(8);delay_ms(250);buzz(7);while(1){cek_kompas
();};break;
        }
    while(1);
}

//===== MODE PENGECEKAN =====
void buzz(int n)
{
    int p;
    if(n<=4){
        n-=1;
        for(p=0;p<n;p++){
            delay_ms(50);
            PORTE.6=1;
            delay_ms(50);
            PORTE.6=0;
        }
    }
    else{
        n-=4;
        PORTE.6=1;
        delay_ms(200);
        PORTE.6=0;
        n--;
        while(n!=0){
            delay_ms(50);
            PORTE.6=1;
            delay_ms(50);
            PORTE.6=0;
            n--;
        }
    }
}

void cek_kompas()
{
    DATA_CMPS03();
    lcd_gotoxy(4,1);
    lcd_putsf("SUDUT = ");
    sprintf(Msg,"%3d",data_cmp);
    lcd_gotoxy(12,1);
    lcd_puts(Msg);
    lcd_gotoxy(15,1);
    lcd_putsf("\xDF");
}

void cek_koreksi()
{
    if((koreksi_0_l<sudut_error) && (sudut_error<koreksi_0_h)){
        lcd_gotoxy(4,0);lcd_putsf("KOREKSI NORTH");
        koreksi(NORTH);}
    else            if((koreksi_90_l<sudut_error)           &&
(sudut_error<koreksi_90_h)){
        lcd_gotoxy(4,0);lcd_putsf("KOREKSI EAST");
        koreksi(EAST);}
    else            if((koreksi_180_l<sudut_error)          &&
(sudut_error<koreksi_180_h)){
        lcd_gotoxy(4,0);lcd_putsf("KOREKSI SOUTH");
```

```
                koreksi(SOUTH);}
            else             if((koreksi_270_l<sudut_error)             &&
(sudut_error<koreksi_270_h)){
                lcd_gotoxy(4,0);lcd_putsf("KOREKSI WEST");
                koreksi(WEST);}
        }

        void cek_orientasi(void)
        {
            delay_ms(500);
            DATA_CMPS03();
            sudut = data_cmp;
            sudut_error = sudut;
            cek_sudut();

            if((arah_0_l<=sudut) && (sudut<=arah_0_h)){
                heading=NORTH;
                lcd_gotoxy(8,0);lcd_putsf("ARAH 0");}
            else if((arah_90_l<=sudut) && (sudut<=arah_90_h)){
                heading=EAST;
                lcd_gotoxy(8,0);lcd_putsf("ARAH 90");}
            else if((arah_180_l<=sudut) && (sudut<=arah_180_h)){
                heading=SOUTH;
                lcd_gotoxy(7,0);lcd_putsf("ARAH 180");}
            else if((arah_270_l<=sudut) && (sudut<=arah_270_h)){
                heading=WEST;
                lcd_gotoxy(8,0);lcd_putsf("ARAH 270");}
            else{
                lcd_gotoxy(8,1);lcd_putsf("KOREKSI");
                cek_koreksi();}
            delay_ms(500);
            lcd_clear();
        }

        void cek_ping()
        {
            sprintf(Ping_1,"PingA=%3d",jarak_A);
            lcd_gotoxy(0,0);lcd_puts(Ping_1);
            sprintf(Ping_2,"PingB=%3d",jarak_B);
            lcd_gotoxy(10,0);lcd_puts(Ping_2);
            sprintf(Ping_3,"PingC=%3d",jarak_C);
            lcd_gotoxy(0,2);lcd_puts(Ping_3);
            sprintf(Ping_4,"PingD=%3d",jarak_D);
            lcd_gotoxy(10,2);lcd_puts(Ping_4);
        }

        void uart_speed_setting(char mode)
        {
            if(mode) {UBRR0H=0x00;UBRR0L=0x67;}       /// 1 mode PC debug
            else     {UBRR0H=0x00;UBRR0L=0x00;}       /// 0 mode PING data
500Kbps={UBRRH=0x00;UBRRL=0x00;}
        }

        //===== SENSING =====//                                    sudut target

unsigned char Akses_CMPS03(char alamat)
{
    unsigned char Value;
    i2c_start();
    i2c_write(COMPASS_WRITE);
    i2c_write(alamat);
    i2c_start();
    i2c_write(COMPASS_READ);
    Value=i2c_read(0);
    i2c_stop();
    return Value;
}

void DATA_CMPS03()
{
     unsigned int A, B;
     A = Akses_CMPS03(2);
     B = Akses_CMPS03(3);
     A<<=8;
     A|=B;
     data_cmp=(unsigned int)(A/10);//Data
}

void kalibrasi_arena()
{
    arah_0 = 283;
    arah_90 = 5;
    arah_180 = 95;
    arah_270 = 180;
}

//===== PERGERAKAN =====//
void koreksi(char arah)
{
    if(arah==NORTH){
        sudut_koreksi=arah_0;
        heading=NORTH;}
    else if(arah==EAST){
        sudut_koreksi=arah_90;
        heading=EAST;}
    else if(arah==SOUTH){
        sudut_koreksi=arah_180;
        heading=SOUTH;}
    else if(arah==WEST){
        sudut_koreksi=arah_270;
        heading=WEST;}

    if(sudut_koreksi<0){
        sudut_koreksi+=360;}

    delay_ms(250);
    DATA_CMPS03();
    if(abs(sudut_koreksi-data_cmp)<=180){                    //sudut:
```

```
                if(sudut_koreksi>data_cmp){rotasi_CW();}        //data_cmp:
sudut sekarang (awal)
                else{rotasi_CCW();}
            }
            else{
                if(sudut_koreksi>data_cmp){rotasi_CCW();}
                else{rotasi_CW();}
            }
            send_umc(0,0);
            delay_ms(500);
            lcd_clear();

        }

        void putar_rotary(long int derajat)
        {
         int temp2;
         int temp1;
         int temp3;
         //rotary_mode(1);                 //ext.int. = ON
         temp1=rotary_kanan;
         temp3=rotary_kiri;
         send_umc(brake);
         delay_ms(200);
         rotary_kiri=rotary_kanan=0;
         TCNT0=0;
         TCNT2=0;
         #asm("nop")
         #asm("nop")
         if(derajat<0)                  //sudut putar ke : kiri
            {
             temp2 = (long int)(derajat*100);
             temp2  /=  (long  int)65;                //sedikit   berbeda  jk
menggunakan rot. kanan, krn selisih 1 pulsa tertinggal trhdp rot kiri
             temp2 = (long int)abs(temp2);
             send_umc(-10,10);
             while(rotary_kanan<temp2){};
            }
         else if(derajat>0)             //sudut putar ke : kanan
            {
             temp2 = (long int)(derajat*100);
             temp2 /= (long int)60;           //1 pulsa = 4,5 derajat
             temp2 = (long int)abs(temp2);
             send_umc(10,-10);
             while(rotary_kiri<temp2){};
            }
         send_umc(brake);
         delay_ms(100);
         rotary_kanan = temp1;
         rotary_kiri = temp3;
         //rotary_mode(0);                 //ext.int. = ON
        }

        void rotasi_CW()
        {
            send_umc(4,-4);
            while(abs(sudut_koreksi-data_cmp)>2){
                DATA_CMPS03();}
        }

        void rotasi_CCW()
        {
            send_umc(-4,4);
            while(abs(sudut_koreksi-data_cmp)>2){
                DATA_CMPS03();}
        }

        void send_umc(int left,int right)
        {
            #asm("cli")
            #asm("nop")
            ss_Right=0;spi((signed char)right);ss_Right=1;
            #asm("nop")
            #asm("nop")
            ss_Left=0;spi((signed char)left);ss_Left=1;
            #asm("nop")
            #asm("sei")
            if(left>0) L_inc=1;
            else L_inc=0;
            if(right>0) R_inc=1;
            else R_inc=0;
        }

        //===== INTERRUPT =====//
        interrupt [EXT_INT2] void ext_int2_isr(void){
          if(L_inc) {rotary_kiri++;}
          else {rotary_kiri--;}
        }

        interrupt [EXT_INT3] void ext_int3_isr(void){
          if(R_inc) {rotary_kanan++;}
          else {rotary_kanan--;}
        }

        interrupt [USART0_RXC] void usart0_rx_isr(void){
        unsigned char status,data;
        status=UCSR0A;
        data=UDR0;
        if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0){
            rx_buffer0[rx_wr_index0]=data;
            if (data==0xFF || ++rx_wr_index0 == RX_BUFFER_SIZE0){
                    rx_wr_index0=0;
                    if(rx_buffer0[0]==0xEE && rx_buffer0[5]==0xFF){
                        rx_buffer_overflow0=1;

jarak_A=(int)rx_buffer0[1];jarak_B=(int)rx_buffer0[3];jarak_C=(int)rx_buffer
0[2];
                                jarak_D=(int)rx_buffer0[4];
```

```
                PORTE.2=~PORTE.2;
                sonar_complete=1;
            }
    }
}
}

interrupt [TIM1_OVF] void timer1_ovf_isr(void)        // 20 ms
{}

//===== MAPPING STRATEGY =====//
void start_detection()
{
    int rerata_1,rerata_2,rerata_3,rerata_4;
    float delta[10],terkecil;
    delay_ms(100);
    cek_orientasi();
    for(d = 0; d<10; d++){
        for(e = 0; e<2; e++){
            data_peta[d][e]=0;}
    }
    a=b=c=d=0;
    delay_ms(100);
    if (heading==NORTH){
        data[0]=jarak_B;
        data[1]=jarak_C;
        data[2]=jarak_D;
        data[3]=jarak_A;}
    else if (heading==EAST){
        data[0]=jarak_A;
        data[1]=jarak_B;
        data[2]=jarak_C;
        data[3]=jarak_D;}
    else if (heading==SOUTH){
        data[0]=jarak_D;
        data[1]=jarak_A;
        data[2]=jarak_B;
        data[3]=jarak_C;}
    else if (heading==WEST){
        data[0]=jarak_C;
        data[1]=jarak_D;
        data[2]=jarak_A;
        data[3]=jarak_B;}

    for (a = 0; a<9; a++){
        for (b = 0; b<9; b++){
            petaxy(a,b,0);}
    }

    if(d<2){
        x_now=data_peta[0][0];
        y_now=data_peta[0][1];}
    else{
        f=d;
        for(d=0; d<f; d++){
            a=data_peta[d][0];
            b=data_peta[d][1];
            rerata_1=(abs)(data[0]-peta[a][b][0]);
            rerata_2=(abs)(data[1]-peta[a][b][1]);
            rerata_3=(abs)(data[2]-peta[a][b][2]);
            rerata_4=(abs)(data[3]-peta[a][b][3]);
            delta[d]=(rerata_1+rerata_2+rerata_3+rerata_4)/4;
            }
        terkecil=delta[0];
        for(d=1; d<f; d++){
            if(delta[d]<terkecil){
                terkecil=delta[d];
                g=d;}
        }
        x_now=data_peta[g][0];
        y_now=data_peta[g][1];
    }

    data[4]=peta[x_now][y_now][4];
    x_prev=x_now-1;
    x_next=x_now+1;
    y_prev=y_now-1;
    y_next=y_now+1;
    display_peta();
}

void position_detection()
{
    a=b=c=d=0;
    peta_ok=0;
    //cek_orientasi();
    lcd_clear();

    x_prev=x_now-1;
    x_next=x_now+1;
    y_prev=y_now-1;
    y_next=y_now+1;

    while(peta_ok==0)
    {
        if (heading==NORTH){
            data[0]=jarak_B;
            data[1]=jarak_C;
            data[2]=jarak_D;
            data[3]=jarak_A;}
        else if (heading==EAST){
            data[0]=jarak_A;
            data[1]=jarak_B;
            data[2]=jarak_C;
            data[3]=jarak_D;}
        else if (heading==SOUTH){
            data[0]=jarak_D;
            data[1]=jarak_A;
```

```
            data[2]=jarak_B;
            data[3]=jarak_C;}
        else if (heading==WEST){
            data[0]=jarak_C;
            data[1]=jarak_D;
            data[2]=jarak_A;
            data[3]=jarak_B;}
        petaxy(x_now,y_next,1);
        petaxy(x_next,y_now,1);
        petaxy(x_now,y_prev,1);
        petaxy(x_prev,y_now,1);
        send_umc(5,4);
    }
    send_umc(120,120);

    if((x_now==x_des)&&(y_now==y_des)){
        peta_cocok=1;}
    else{
        peta_cocok=0;}

    data[4]=peta[x_now][y_now][4];
    x_prev=x_now-1;
    x_next=x_now+1;
    y_prev=y_now-1;
    y_next=y_now+1;
    peta_ok=0;
    display_peta();
}

void node2node(int x_1, int y_1)
{
    x_des=x_1;
    y_des=y_1;
    peta_cocok=0;

    sprintf(Msg1,"%3d",x_des);
    lcd_gotoxy(0,0);
    lcd_puts(Msg1);
    sprintf(Msg1,"%3d",y_des);
    lcd_gotoxy(0,1);
    lcd_puts(Msg1);
    delay_ms(1000);
    lcd_clear();

    lcd_gotoxy(0,0);lcd_putsf("==   NODE 2 NODE  ==");

    if(x_des<x_now){
        koor_x=POSITIF;
        lcd_gotoxy(0,1);
        lcd_putsf("===  X POSITIF   ===");}
    else if(x_des>x_now){
        koor_x=NEGATIF;
        lcd_gotoxy(0,1);
        lcd_putsf("===  X NEGATIF   ===");}
    else{
        koor_x=NOL;
        lcd_gotoxy(0,1);
        lcd_putsf("=====  X OK   ======");}

    if(y_des<y_now){
        koor_y=POSITIF;
        lcd_gotoxy(0,2);
        lcd_putsf("===  Y POSITIF   ===");}
    else if(y_des>y_now){
        koor_y=NEGATIF;
        lcd_gotoxy(0,2);
        lcd_putsf("===  Y NEGATIF   ===");}
    else{
        koor_y=NOL;
        lcd_gotoxy(0,2);
        lcd_putsf("=====  Y OK   ======");}

    delay_ms(1000);
    lcd_clear();
    delay_ms(100);
    cek_orientasi();
    if((koor_x==NOL)&&(koor_y==POSITIF)){
        lcd_gotoxy(0,0);
        lcd_putsf("=====  SOUTH  ======");
        delay_ms(1000);
        koreksi(SOUTH);}
    else if((koor_x==NOL)&&(koor_y==NEGATIF)){
        lcd_gotoxy(0,0);
        lcd_putsf("=====  NORTH  ======");
        delay_ms(1000);
        koreksi(NORTH);}
    else if((koor_x==POSITIF)&&(koor_y==NOL)){
        lcd_gotoxy(0,0);
        lcd_putsf("======  WEST  ======");
        delay_ms(1000);
        koreksi(WEST);}
    else{
        lcd_gotoxy(0,0);
        lcd_putsf("======  EAST  ======");
        delay_ms(1000);
        koreksi(EAST);}
    lcd_clear();
    while(peta_cocok==0){
        if((x_now==x_des)&&(y_now==y_des)){
            lcd_gotoxy(0,1);
            lcd_putsf("Ok");
            delay_ms(500);
            lcd_clear();
            peta_cocok=1;}
        else{
            lcd_gotoxy(0,1);
            lcd_putsf("Maju");
            delay_ms(500);
```

```
                lcd_clear();
                position_detection();}
        }
    }

    void petaxy(int x, int y, int z)
    {
        ok=0;
        if((x==0)&&(y==0)){
            for(c = 0; c<2; c++){
                if(((petaxy_00[c][0]-
13)<=data[0])&&(data[0]<(petaxy_00[c][0]+13))){
                    if(((petaxy_00[c][1]-
13)<=data[1])&&(data[1]<(petaxy_00[c][1]+13))){
                        if(((petaxy_00[c][2]-
13)<=data[2])&&(data[2]<(petaxy_00[c][2]+13))){
                            if(((petaxy_00[c][3]-
13)<=data[3])&&(data[3]<(petaxy_00[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_00[c][i];};

        }}}}}}
        else if((x==0)&&(y==1)){
            for(c = 0; c<4; c++){
                if(((petaxy_01[c][0]-
13)<=data[0])&&(data[0]<(petaxy_01[c][0]+13))){
                    if(((petaxy_01[c][1]-
13)<=data[1])&&(data[1]<(petaxy_01[c][1]+13))){
                        if(((petaxy_01[c][2]-
13)<=data[2])&&(data[2]<(petaxy_01[c][2]+13))){
                            if(((petaxy_01[c][3]-
13)<=data[3])&&(data[3]<(petaxy_01[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_01[c][i];};

        }}}}}}
        else if((x==0)&&(y==3)){
            for(c = 0; c<3; c++){
                if(((petaxy_03[c][0]-
13)<=data[0])&&(data[0]<(petaxy_03[c][0]+13))){
                    if(((petaxy_03[c][1]-
13)<=data[1])&&(data[1]<(petaxy_03[c][1]+13))){
                        if(((petaxy_03[c][2]-
13)<=data[2])&&(data[2]<(petaxy_03[c][2]+13))){
                            if(((petaxy_03[c][3]-
13)<=data[3])&&(data[3]<(petaxy_03[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_03[c][i];};

        }}}}}}
        else if((x==0)&&(y==4)){
            for(c = 0; c<2; c++){
                if(((petaxy_04[c][0]-
13)<=data[0])&&(data[0]<(petaxy_04[c][0]+13))){
                    if(((petaxy_04[c][1]-
13)<=data[1])&&(data[1]<(petaxy_04[c][1]+13))){
                        if(((petaxy_04[c][2]-
13)<=data[2])&&(data[2]<(petaxy_04[c][2]+13))){
                            if(((petaxy_04[c][3]-
13)<=data[3])&&(data[3]<(petaxy_04[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_04[c][i];};

        }}}}}}
        else if((x==0)&&(y==5)){
            for(c = 0; c<3; c++){
                if(((petaxy_05[c][0]-
13)<=data[0])&&(data[0]<(petaxy_05[c][0]+13))){
                    if(((petaxy_05[c][1]-
13)<=data[1])&&(data[1]<(petaxy_05[c][1]+13))){
                        if(((petaxy_05[c][2]-
13)<=data[2])&&(data[2]<(petaxy_05[c][2]+13))){
                            if(((petaxy_05[c][3]-
13)<=data[3])&&(data[3]<(petaxy_05[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_05[c][i];};

        }}}}}}
        else if((x==0)&&(y==7)){
            for(c = 0; c<2; c++){
                if(((petaxy_07[c][0]-
13)<=data[0])&&(data[0]<(petaxy_07[c][0]+13))){
                    if(((petaxy_07[c][1]-
13)<=data[1])&&(data[1]<(petaxy_07[c][1]+13))){
                        if(((petaxy_07[c][2]-
13)<=data[2])&&(data[2]<(petaxy_07[c][2]+13))){
                            if(((petaxy_07[c][3]-
13)<=data[3])&&(data[3]<(petaxy_07[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_07[c][i];};

        }}}}}}
        else if((x==1)&&(y==0)){
            for(c = 0; c<2; c++){
                if(((petaxy_10[c][0]-
13)<=data[0])&&(data[0]<(petaxy_10[c][0]+13))){
                    if(((petaxy_10[c][1]-
13)<=data[1])&&(data[1]<(petaxy_10[c][1]+13))){
                        if(((petaxy_10[c][2]-
13)<=data[2])&&(data[2]<(petaxy_10[c][2]+13))){
                            if(((petaxy_10[c][3]-
13)<=data[3])&&(data[3]<(petaxy_10[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_10[c][i];};

        }}}}}}
```

```
            else if((x==1)&&(y==1)){
                for(c = 0; c<3; c++){
                    if(((petaxy_11[c][0]-
13)<=data[0])&&(data[0]<(petaxy_11[c][0]+13))){
                        if(((petaxy_11[c][1]-
13)<=data[1])&&(data[1]<(petaxy_11[c][1]+13))){
                            if(((petaxy_11[c][2]-
13)<=data[2])&&(data[2]<(petaxy_11[c][2]+13))){
                                if(((petaxy_11[c][3]-
13)<=data[3])&&(data[3]<(petaxy_11[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_11[c][i];};
            }}}}}}
            else if((x==1)&&(y==3)){
                for(c = 0; c<2; c++){
                    if(((petaxy_13[c][0]-
13)<=data[0])&&(data[0]<(petaxy_13[c][0]+13))){
                        if(((petaxy_13[c][1]-
13)<=data[1])&&(data[1]<(petaxy_13[c][1]+13))){
                            if(((petaxy_13[c][2]-
13)<=data[2])&&(data[2]<(petaxy_13[c][2]+13))){
                                if(((petaxy_13[c][3]-
13)<=data[3])&&(data[3]<(petaxy_13[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_13[c][i];};
            }}}}}}
            else if((x==1)&&(y==4)){
                for(c = 0; c<6; c++){
                    if(((petaxy_14[c][0]-
13)<=data[0])&&(data[0]<(petaxy_14[c][0]+13))){
                        if(((petaxy_14[c][1]-
13)<=data[1])&&(data[1]<(petaxy_14[c][1]+13))){
                            if(((petaxy_14[c][2]-
13)<=data[2])&&(data[2]<(petaxy_14[c][2]+13))){
                                if(((petaxy_14[c][3]-
13)<=data[3])&&(data[3]<(petaxy_14[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_14[c][i];};
            }}}}}}
            else if((x==1)&&(y==5)){
                for(c = 0; c<3; c++){
                    if(((petaxy_15[c][0]-
13)<=data[0])&&(data[0]<(petaxy_15[c][0]+13))){
                        if(((petaxy_15[c][1]-
13)<=data[1])&&(data[1]<(petaxy_15[c][1]+13))){
                            if(((petaxy_15[c][2]-
13)<=data[2])&&(data[2]<(petaxy_15[c][2]+13))){
                                if(((petaxy_15[c][3]-
13)<=data[3])&&(data[3]<(petaxy_15[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_15[c][i];};
            }}}}}}
            else if((x==1)&&(y==6)){
                for(c = 0; c<2; c++){
                    if(((petaxy_16[c][0]-
13)<=data[0])&&(data[0]<(petaxy_16[c][0]+13))){
                        if(((petaxy_16[c][1]-
13)<=data[1])&&(data[1]<(petaxy_16[c][1]+13))){
                            if(((petaxy_16[c][2]-
13)<=data[2])&&(data[2]<(petaxy_16[c][2]+13))){
                                if(((petaxy_16[c][3]-
13)<=data[3])&&(data[3]<(petaxy_16[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_16[c][i];};
            }}}}}}
            else if((x==1)&&(y==7)){
                for(c = 0; c<4; c++){
                    if(((petaxy_17[c][0]-
13)<=data[0])&&(data[0]<(petaxy_17[c][0]+13))){
                        if(((petaxy_17[c][1]-
13)<=data[1])&&(data[1]<(petaxy_17[c][1]+13))){
                            if(((petaxy_17[c][2]-
13)<=data[2])&&(data[2]<(petaxy_17[c][2]+13))){
                                if(((petaxy_17[c][3]-
13)<=data[3])&&(data[3]<(petaxy_17[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_17[c][i];};
            }}}}}}
            else if((x==1)&&(y==8)){
                for(c = 0; c<2; c++){
                    if(((petaxy_18[c][0]-
13)<=data[0])&&(data[0]<(petaxy_18[c][0]+13))){
                        if(((petaxy_18[c][1]-
13)<=data[1])&&(data[1]<(petaxy_18[c][1]+13))){
                            if(((petaxy_18[c][2]-
13)<=data[2])&&(data[2]<(petaxy_18[c][2]+13))){
                                if(((petaxy_18[c][3]-
13)<=data[3])&&(data[3]<(petaxy_18[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_18[c][i];};
            }}}}}}
            else if((x==2)&&(y==2)){
                for(c = 0; c<2; c++){
                    if(((petaxy_22[c][0]-
13)<=data[0])&&(data[0]<(petaxy_22[c][0]+13))){
                        if(((petaxy_22[c][1]-
13)<=data[1])&&(data[1]<(petaxy_22[c][1]+13))){
                            if(((petaxy_22[c][2]-
13)<=data[2])&&(data[2]<(petaxy_22[c][2]+13))){
                                if(((petaxy_22[c][3]-
13)<=data[3])&&(data[3]<(petaxy_22[c][3]+13))){
```

```
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_22[c][i];};
            }}}}}}
            else if((x==2)&&(y==3)){
                for(c = 0; c<2; c++){
                    if(((petaxy_23[c][0]-
13)<=data[0])&&(data[0]<(petaxy_23[c][0]+13))){
                        if(((petaxy_23[c][1]-
13)<=data[1])&&(data[1]<(petaxy_23[c][1]+13))){
                            if(((petaxy_23[c][2]-
13)<=data[2])&&(data[2]<(petaxy_23[c][2]+13))){
                                if(((petaxy_23[c][3]-
13)<=data[3])&&(data[3]<(petaxy_23[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_23[c][i];};
            }}}}}}
            else if((x==2)&&(y==4)){
                for(c = 0; c<2; c++){
                    if(((petaxy_24[c][0]-
13)<=data[0])&&(data[0]<(petaxy_24[c][0]+13))){
                        if(((petaxy_24[c][1]-
13)<=data[1])&&(data[1]<(petaxy_24[c][1]+13))){
                            if(((petaxy_24[c][2]-
13)<=data[2])&&(data[2]<(petaxy_24[c][2]+13))){
                                if(((petaxy_24[c][3]-
13)<=data[3])&&(data[3]<(petaxy_24[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_24[c][i];};
            }}}}}}
            else if((x==2)&&(y==5)){
                for(c = 0; c<2; c++){
                    if(((petaxy_25[c][0]-
13)<=data[0])&&(data[0]<(petaxy_25[c][0]+13))){
                        if(((petaxy_25[c][1]-
13)<=data[1])&&(data[1]<(petaxy_25[c][1]+13))){
                            if(((petaxy_25[c][2]-
13)<=data[2])&&(data[2]<(petaxy_25[c][2]+13))){
                                if(((petaxy_25[c][3]-
13)<=data[3])&&(data[3]<(petaxy_25[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_25[c][i];};
            }}}}}}
            else if((x==2)&&(y==7)){
                for(c = 0; c<2; c++){
                    if(((petaxy_27[c][0]-
13)<=data[0])&&(data[0]<(petaxy_27[c][0]+13))){
                        if(((petaxy_27[c][1]-
13)<=data[1])&&(data[1]<(petaxy_27[c][1]+13))){
                            if(((petaxy_27[c][2]-
13)<=data[2])&&(data[2]<(petaxy_27[c][2]+13))){
                                if(((petaxy_27[c][3]-
13)<=data[3])&&(data[3]<(petaxy_27[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_27[c][i];};
            }}}}}}
            else if((x==3)&&(y==2)){
                for(c = 0; c<2; c++){
                    if(((petaxy_32[c][0]-
13)<=data[0])&&(data[0]<(petaxy_32[c][0]+13))){
                        if(((petaxy_32[c][1]-
13)<=data[1])&&(data[1]<(petaxy_32[c][1]+13))){
                            if(((petaxy_32[c][2]-
13)<=data[2])&&(data[2]<(petaxy_32[c][2]+13))){
                                if(((petaxy_32[c][3]-
13)<=data[3])&&(data[3]<(petaxy_32[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_32[c][i];};
            }}}}}}
            else if((x==3)&&(y==3)){
                for(c = 0; c<2; c++){
                    if(((petaxy_33[c][0]-
13)<=data[0])&&(data[0]<(petaxy_33[c][0]+13))){
                        if(((petaxy_33[c][1]-
13)<=data[1])&&(data[1]<(petaxy_33[c][1]+13))){
                            if(((petaxy_33[c][2]-
13)<=data[2])&&(data[2]<(petaxy_33[c][2]+13))){
                                if(((petaxy_33[c][3]-
13)<=data[3])&&(data[3]<(petaxy_33[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_33[c][i];};
            }}}}}}
            else if((x==3)&&(y==7)){
                for(c = 0; c<2; c++){
                    if(((petaxy_37[c][0]-
13)<=data[0])&&(data[0]<(petaxy_37[c][0]+13))){
                        if(((petaxy_37[c][1]-
13)<=data[1])&&(data[1]<(petaxy_37[c][1]+13))){
                            if(((petaxy_37[c][2]-
13)<=data[2])&&(data[2]<(petaxy_37[c][2]+13))){
                                if(((petaxy_37[c][3]-
13)<=data[3])&&(data[3]<(petaxy_37[c][3]+13))){
                                ok=1;
                                for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_37[c][i];};
            }}}}}}
            else if((x==4)&&(y==0)){
                for(c = 0; c<3; c++){
                    if(((petaxy_40[c][0]-
13)<=data[0])&&(data[0]<(petaxy_40[c][0]+13))){
                        if(((petaxy_40[c][1]-
13)<=data[1])&&(data[1]<(petaxy_40[c][1]+13))){
```

```
                            if(((petaxy_40[c][2]-
13)<=data[2])&&(data[2]<(petaxy_40[c][2]+13))){
                                if(((petaxy_40[c][3]-
13)<=data[3])&&(data[3]<(petaxy_40[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_40[c][i];};
            }}}}}}
            else if((x==4)&&(y==1)){
                for(c = 0; c<4; c++){
                    if(((petaxy_41[c][0]-
13)<=data[0])&&(data[0]<(petaxy_41[c][0]+13))){
                        if(((petaxy_41[c][1]-
13)<=data[1])&&(data[1]<(petaxy_41[c][1]+13))){
                            if(((petaxy_41[c][2]-
13)<=data[2])&&(data[2]<(petaxy_41[c][2]+13))){
                                if(((petaxy_41[c][3]-
13)<=data[3])&&(data[3]<(petaxy_41[c][3]+13))){
                                ok=1;
                                 for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_41[c][i];};
            }}}}}}
            else if((x==4)&&(y==3)){
                for(c = 0; c<3; c++){
                    if(((petaxy_43[c][0]-
13)<=data[0])&&(data[0]<(petaxy_43[c][0]+13))){
                        if(((petaxy_43[c][1]-
13)<=data[1])&&(data[1]<(petaxy_43[c][1]+13))){
                            if(((petaxy_43[c][2]-
13)<=data[2])&&(data[2]<(petaxy_43[c][2]+13))){
                                if(((petaxy_43[c][3]-
13)<=data[3])&&(data[3]<(petaxy_43[c][3]+13))){
                                 ok=1;
                                 for(i=0;i<4;i++){
                                    peta[a][b][i]=petaxy_43[c][i];};
            }}}}}}
            else if((x==4)&&(y==4)){
                for(c = 0; c<6; c++){
                    if(((petaxy_44[c][0]-
13)<=data[0])&&(data[0]<(petaxy_44[c][0]+13))){
                        if(((petaxy_44[c][1]-
13)<=data[1])&&(data[1]<(petaxy_44[c][1]+13))){
                            if(((petaxy_44[c][2]-
13)<=data[2])&&(data[2]<(petaxy_44[c][2]+13))){
                                if(((petaxy_44[c][3]-
13)<=data[3])&&(data[3]<(petaxy_44[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_44[c][i];};
            }}}}}}
            else if((x==4)&&(y==5)){
                for(c = 0; c<3; c++){
                    if(((petaxy_45[c][0]-
13)<=data[0])&&(data[0]<(petaxy_45[c][0]+13))){
                            if(((petaxy_45[c][1]-
13)<=data[1])&&(data[1]<(petaxy_45[c][1]+13))){
                                if(((petaxy_45[c][2]-
13)<=data[2])&&(data[2]<(petaxy_45[c][2]+13))){
                                    if(((petaxy_45[c][3]-
13)<=data[3])&&(data[3]<(petaxy_45[c][3]+13))){
                                        ok=1;
                                        for(i=0;i<4;i++){
                                            peta[a][b][i]=petaxy_45[c][i];};
            }}}}}}
            else if((x==4)&&(y==7)){
                for(c = 0; c<3; c++){
                    if(((petaxy_47[c][0]-
13)<=data[0])&&(data[0]<(petaxy_47[c][0]+13))){
                        if(((petaxy_47[c][1]-
13)<=data[1])&&(data[1]<(petaxy_47[c][1]+13))){
                            if(((petaxy_47[c][2]-
13)<=data[2])&&(data[2]<(petaxy_47[c][2]+13))){
                                if(((petaxy_47[c][3]-
13)<=data[3])&&(data[3]<(petaxy_47[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_47[c][i];};
            }}}}}}
            else if((x==4)&&(y==8)){
                for(c = 0; c<2; c++){
                    if(((petaxy_48[c][0]-
13)<=data[0])&&(data[0]<(petaxy_48[c][0]+13))){
                        if(((petaxy_48[c][1]-
13)<=data[1])&&(data[1]<(petaxy_48[c][1]+13))){
                            if(((petaxy_48[c][2]-
13)<=data[2])&&(data[2]<(petaxy_48[c][2]+13))){
                                if(((petaxy_48[c][3]-
13)<=data[3])&&(data[3]<(petaxy_48[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_48[c][i];};
            }}}}}}
            else if((x==5)&&(y==0)){
                for(c = 0; c<3; c++){
                    if(((petaxy_50[c][0]-
13)<=data[0])&&(data[0]<(petaxy_50[c][0]+13))){
                        if(((petaxy_50[c][1]-
13)<=data[1])&&(data[1]<(petaxy_50[c][1]+13))){
                            if(((petaxy_50[c][2]-
13)<=data[2])&&(data[2]<(petaxy_50[c][2]+13))){
                                if(((petaxy_50[c][3]-
13)<=data[3])&&(data[3]<(petaxy_50[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_50[c][i];};
            }}}}}}
            else if((x==5)&&(y==1)){
                for(c = 0; c<4; c++){
```

```
                    if(((petaxy_51[c][0]-
13)<=data[0])&&(data[0]<(petaxy_51[c][0]+13))){
                        if(((petaxy_51[c][1]-
13)<=data[1])&&(data[1]<(petaxy_51[c][1]+13))){
                            if(((petaxy_51[c][2]-
13)<=data[2])&&(data[2]<(petaxy_51[c][2]+13))){
                                if(((petaxy_51[c][3]-
13)<=data[3])&&(data[3]<(petaxy_51[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_51[c][i];};
            }}}}}}
            else if((x==5)&&(y==3)){
                for(c = 0; c<3; c++){
                    if(((petaxy_53[c][0]-
13)<=data[0])&&(data[0]<(petaxy_53[c][0]+13))){
                        if(((petaxy_53[c][1]-
13)<=data[1])&&(data[1]<(petaxy_53[c][1]+13))){
                            if(((petaxy_53[c][2]-
13)<=data[2])&&(data[2]<(petaxy_53[c][2]+13))){
                                if(((petaxy_53[c][3]-
13)<=data[3])&&(data[3]<(petaxy_53[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_53[c][i];};
            }}}}}}
            else if((x==5)&&(y==4)){
                for(c = 0; c<3; c++){
                    if(((petaxy_54[c][0]-
13)<=data[0])&&(data[0]<(petaxy_54[c][0]+13))){
                        if(((petaxy_54[c][1]-
13)<=data[1])&&(data[1]<(petaxy_54[c][1]+13))){
                            if(((petaxy_54[c][2]-
13)<=data[2])&&(data[2]<(petaxy_54[c][2]+13))){
                                if(((petaxy_54[c][3]-
13)<=data[3])&&(data[3]<(petaxy_54[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_54[c][i];};
            }}}}}}
            else if((x==5)&&(y==5)){
                for(c = 0; c<2; c++){
                    if(((petaxy_55[c][0]-
13)<=data[0])&&(data[0]<(petaxy_55[c][0]+13))){
                        if(((petaxy_55[c][1]-
13)<=data[1])&&(data[1]<(petaxy_55[c][1]+13))){
                            if(((petaxy_55[c][2]-
13)<=data[2])&&(data[2]<(petaxy_55[c][2]+13))){
                                if(((petaxy_55[c][3]-
13)<=data[3])&&(data[3]<(petaxy_55[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_55[c][i];};
            }}}}}}
            else if((x==5)&&(y==7)){
                for(c = 0; c<4; c++){
                    if(((petaxy_57[c][0]-
13)<=data[0])&&(data[0]<(petaxy_57[c][0]+13))){
                        if(((petaxy_57[c][1]-
13)<=data[1])&&(data[1]<(petaxy_57[c][1]+13))){
                            if(((petaxy_57[c][2]-
13)<=data[2])&&(data[2]<(petaxy_57[c][2]+13))){
                                if(((petaxy_57[c][3]-
13)<=data[3])&&(data[3]<(petaxy_57[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_57[c][i];};
            }}}}}}
            else if((x==6)&&(y==3)){
                for(c = 0; c<6; c++){
                    if(((petaxy_63[c][0]-
13)<=data[0])&&(data[0]<(petaxy_63[c][0]+13))){
                        if(((petaxy_63[c][1]-
13)<=data[1])&&(data[1]<(petaxy_63[c][1]+13))){
                            if(((petaxy_63[c][2]-
13)<=data[2])&&(data[2]<(petaxy_63[c][2]+13))){
                                if(((petaxy_63[c][3]-
13)<=data[3])&&(data[3]<(petaxy_63[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_63[c][i];};
            }}}}}}
            else if((x==6)&&(y==4)){
                for(c = 0; c<8; c++){
                    if(((petaxy_64[c][0]-
13)<=data[0])&&(data[0]<(petaxy_64[c][0]+13))){
                        if(((petaxy_64[c][1]-
13)<=data[1])&&(data[1]<(petaxy_64[c][1]+13))){
                            if(((petaxy_64[c][2]-
13)<=data[2])&&(data[2]<(petaxy_64[c][2]+13))){
                                if(((petaxy_64[c][3]-
13)<=data[3])&&(data[3]<(petaxy_64[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_64[c][i];};
            }}}}}}
            else if((x==6)&&(y==7)){
                for(c = 0; c<2; c++){
                    if(((petaxy_67[c][0]-
13)<=data[0])&&(data[0]<(petaxy_67[c][0]+13))){
                        if(((petaxy_67[c][1]-
13)<=data[1])&&(data[1]<(petaxy_67[c][1]+13))){
                            if(((petaxy_67[c][2]-
13)<=data[2])&&(data[2]<(petaxy_67[c][2]+13))){
                                if(((petaxy_67[c][3]-
13)<=data[3])&&(data[3]<(petaxy_67[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
```

```
                                peta[a][b][i]=petaxy_67[c][i];};
            }}}}}}
            else if((x==6)&&(y==8)){
                for(c = 0; c<2; c++){
                    if(((petaxy_68[c][0]-
13)<=data[0])&&(data[0]<(petaxy_68[c][0]+13))){
                        if(((petaxy_68[c][1]-
13)<=data[1])&&(data[1]<(petaxy_68[c][1]+13))){
                            if(((petaxy_68[c][2]-
13)<=data[2])&&(data[2]<(petaxy_68[c][2]+13))){
                                if(((petaxy_68[c][3]-
13)<=data[3])&&(data[3]<(petaxy_68[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_68[c][i];};
            }}}}}}
            else if((x==7)&&(y==0)){
                for(c = 0; c<2; c++){
                    if(((petaxy_70[c][0]-
13)<=data[0])&&(data[0]<(petaxy_70[c][0]+13))){
                        if(((petaxy_70[c][1]-
13)<=data[1])&&(data[1]<(petaxy_70[c][1]+13))){
                            if(((petaxy_70[c][2]-
13)<=data[2])&&(data[2]<(petaxy_70[c][2]+13))){
                                if(((petaxy_70[c][3]-
13)<=data[3])&&(data[3]<(petaxy_70[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_70[c][i];};
            }}}}}}
            else if((x==7)&&(y==1)){
                for(c = 0; c<2; c++){
                    if(((petaxy_71[c][0]-
13)<=data[0])&&(data[0]<(petaxy_71[c][0]+13))){
                        if(((petaxy_71[c][1]-
13)<=data[1])&&(data[1]<(petaxy_71[c][1]+13))){
                            if(((petaxy_71[c][2]-
13)<=data[2])&&(data[2]<(petaxy_71[c][2]+13))){
                                if(((petaxy_71[c][3]-
13)<=data[3])&&(data[3]<(petaxy_71[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_71[c][i];};
            }}}}}}
            else if((x==7)&&(y==2)){
                for(c = 0; c<2; c++){
                    if(((petaxy_72[c][0]-
13)<=data[0])&&(data[0]<(petaxy_72[c][0]+13))){
                        if(((petaxy_72[c][1]-
13)<=data[1])&&(data[1]<(petaxy_72[c][1]+13))){
                            if(((petaxy_72[c][2]-
13)<=data[2])&&(data[2]<(petaxy_72[c][2]+13))){
                                if(((petaxy_72[c][3]-
13)<=data[3])&&(data[3]<(petaxy_72[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_72[c][i];};
            }}}}}}
            else if((x==7)&&(y==3)){
                for(c = 0; c<5; c++){
                    if(((petaxy_73[c][0]-
13)<=data[0])&&(data[0]<(petaxy_73[c][0]+13))){
                        if(((petaxy_73[c][1]-
13)<=data[1])&&(data[1]<(petaxy_73[c][1]+13))){
                            if(((petaxy_73[c][2]-
13)<=data[2])&&(data[2]<(petaxy_73[c][2]+13))){
                                if(((petaxy_73[c][3]-
13)<=data[3])&&(data[3]<(petaxy_73[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_73[c][i];};
            }}}}}}
            else if((x==7)&&(y==4)){
                for(c = 0; c<4; c++){
                    if(((petaxy_74[c][0]-
13)<=data[0])&&(data[0]<(petaxy_74[c][0]+13))){
                        if(((petaxy_74[c][1]-
13)<=data[1])&&(data[1]<(petaxy_74[c][1]+13))){
                            if(((petaxy_74[c][2]-
13)<=data[2])&&(data[2]<(petaxy_74[c][2]+13))){
                                if(((petaxy_74[c][3]-
13)<=data[3])&&(data[3]<(petaxy_74[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_74[c][i];};
            }}}}}}
            else if((x==7)&&(y==7)){
                for(c = 0; c<2; c++){
                    if(((petaxy_77[c][0]-
13)<=data[0])&&(data[0]<(petaxy_77[c][0]+13))){
                        if(((petaxy_77[c][1]-
13)<=data[1])&&(data[1]<(petaxy_77[c][1]+13))){
                            if(((petaxy_77[c][2]-
13)<=data[2])&&(data[2]<(petaxy_77[c][2]+13))){
                                if(((petaxy_77[c][3]-
13)<=data[3])&&(data[3]<(petaxy_77[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_77[c][i];};
            }}}}}}
            else if((x==8)&&(y==3)){
                for(c = 0; c<3; c++){
                    if(((petaxy_83[c][0]-
13)<=data[0])&&(data[0]<(petaxy_83[c][0]+13))){
                        if(((petaxy_83[c][1]-
13)<=data[1])&&(data[1]<(petaxy_83[c][1]+13))){
                            if(((petaxy_83[c][2]-
13)<=data[2])&&(data[2]<(petaxy_83[c][2]+13))){
```

```
                                if(((petaxy_83[c][3]-
13)<=data[3])&&(data[3]<(petaxy_83[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_83[c][i];};
            }}}}}}
            else if((x==8)&&(y==4)){
                for(c = 0; c<2; c++){
                    if(((petaxy_84[c][0]-
13)<=data[0])&&(data[0]<(petaxy_84[c][0]+13))){
                        if(((petaxy_84[c][1]-
13)<=data[1])&&(data[1]<(petaxy_84[c][1]+13))){
                            if(((petaxy_84[c][2]-
13)<=data[2])&&(data[2]<(petaxy_84[c][2]+13))){
                                if(((petaxy_84[c][3]-
13)<=data[3])&&(data[3]<(petaxy_84[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_84[c][i];};
            }}}}}}
            else if((x==8)&&(y==7)){
                for(c = 0; c<2; c++){
                    if(((petaxy_87[c][0]-
13)<=data[0])&&(data[0]<(petaxy_87[c][0]+13))){
                        if(((petaxy_87[c][1]-
13)<=data[1])&&(data[1]<(petaxy_87[c][1]+13))){
                            if(((petaxy_87[c][2]-
13)<=data[2])&&(data[2]<(petaxy_87[c][2]+13))){
                                if(((petaxy_87[c][3]-
13)<=data[3])&&(data[3]<(petaxy_87[c][3]+13))){
                                    ok=1;
                                    for(i=0;i<4;i++){
                                        peta[a][b][i]=petaxy_87[c][i];};
            }}}}}}
            else{
                if(((peta[x][y][0]-
13)<=data[0])&&(data[0]<(peta[x][y][0]+13))){
                    if(((peta[x][y][1]-
13)<=data[1])&&(data[1]<(peta[x][y][1]+13))){
                        if(((peta[x][y][2]-
13)<=data[2])&&(data[2]<(peta[x][y][2]+13))){
                            if(((peta[x][y][3]-
13)<=data[3])&&(data[3]<(peta[x][y][3]+13))){
                                ok=1;
                }}}}}

            if(ok){
                if(z==0){
                    data_peta[d][0]=a;
                    data_peta[d][1]=b;
                    d++;}
                else{
                    peta_ok=1;
                    x_now=x;
                                        y_now=y;}}
            }

            void cek_sudut()
            {
                arah_0_l = arah_0-4;
                arah_0_h = arah_0+5;
                arah_90_l = arah_90-4;
                arah_90_h = arah_90+5;
                arah_180_l = arah_180-4;
                arah_180_h = arah_180+5;
                arah_270_l = arah_270-4;
                arah_270_h = arah_270+5;

                koreksi_0_l = arah_0-44;
                koreksi_0_h = arah_0+45;
                koreksi_90_l = arah_90-44;
                koreksi_90_h = arah_90+45;
                koreksi_180_l = arah_180-44;
                koreksi_180_h = arah_180+45;
                koreksi_270_l = arah_270-44;
                koreksi_270_h = arah_270+45;


            if(((355<arah_0)&&(arah_0<=359))||((355<arah_90)&&(arah_90<=359))||
            ((355<arah_180)&&(arah_180<=359))||((355<arah_270)&&(arah_270<=359)))
                {
                    if((0<=sudut)&&(sudut<5))
                    {
                        sudut+=360;
                    }
                }
                else
            if(((0<=arah_0)&&(arah_0<=5))||((0<=arah_90)&&(arah_90<=5))||
            ((0<=arah_180)&&(arah_180<=5))||((0<=arah_270)&&(arah_270<=5)))
                {
                    if((355<=sudut)&&(sudut<=359))
                    {
                        sudut-=360;
                    }
                }


            if(((315<arah_0)&&(arah_0<=359))||((315<arah_90)&&(arah_90<=359))||
            ((315<arah_180)&&(arah_180<=359))||((315<arah_270)&&(arah_270<=359)))
                {
                    if((0<=sudut_error)&&(sudut_error<45))
                    {
                        sudut_error+=360;
                    }
                }
```

```
            else
if(((0<=arah_0)&&(arah_0<=45))||((0<=arah_90)&&(arah_90<=45))||
((0<=arah_180)&&(arah_180<=45))||((0<=arah_270)&&(arah_270<=45)))
            {
                if((315<sudut)&&(sudut<=359))
                {
                    sudut_error-=360;
                }
            }
        }

        //===== DISPLAY =====//
        void display_awal(void)
        {
            lcd_gotoxy(0,1);
            lcd_putsf("===  BISMILLAH  ====");
            delay_ms(1000);
            lcd_clear();
            delay_ms(250);
            for (lama = 0; lama < 3; lama++)
            {
                lcd_gotoxy(0,1);lcd_putsf("====  LOADING  =====");
                delay_ms(250);
                lcd_clear();
                delay_ms(250);
            };

            lcd_gotoxy(0,0);lcd_putsf("====  SKRIPSI  =====");
            lcd_gotoxy(0,1);lcd_putsf("=NUR ISKANDAR JUANG=");
            lcd_gotoxy(0,2);lcd_putsf("===  0910630083  ===");
            lcd_gotoxy(0,3);lcd_putsf("=  TEKNIK ELEKTRO  =");
            delay_ms(2500);
            lcd_clear();
            lcd_gotoxy(0,1);
            lcd_putsf("=== SYSTEM READY ===");
            lcd_gotoxy(0,2);
            lcd_putsf(" TEKAN TOMBOL START ");
        }

        void display_peta()
        {
            if(heading==NORTH){
                if(y_next>8){
                    lcd_gotoxy(8,0);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_up,"[%d,",x_now);
                    sprintf(my_up,"%d]",y_next);
                    lcd_gotoxy(8,0);lcd_puts(mx_up);
                    lcd_gotoxy(11,0);lcd_puts(my_up);}
                if(x_next>8){
                    lcd_gotoxy(15,1);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_right,"[%d,",x_next);
                    sprintf(my_right,"%d]",y_now);
                    lcd_gotoxy(15,1);lcd_puts(mx_right);
                    lcd_gotoxy(18,1);lcd_puts(my_right);}
                if(y_prev<0){
                    lcd_gotoxy(8,2);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_low,"[%d,",x_now);
                    sprintf(my_low,"%d]",y_prev);
                    lcd_gotoxy(8,2);lcd_puts(mx_low);
                    lcd_gotoxy(11,2);lcd_puts(my_low);}
                if(x_prev<0){
                    lcd_gotoxy(0,1);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_left,"[%d,",x_prev);
                    sprintf(my_left,"%d]",y_now);
                    lcd_gotoxy(0,1);lcd_puts(mx_left);
                    lcd_gotoxy(3,1);lcd_puts(my_left);}
            }
            else if(heading==EAST){
                if(y_next>8){
                    lcd_gotoxy(0,1);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_up,"[%d,",x_now);
                    sprintf(my_up,"%d]",y_next);
                    lcd_gotoxy(0,1);lcd_puts(mx_up);
                    lcd_gotoxy(3,1);lcd_puts(my_up);}
                if(x_next>8){
                    lcd_gotoxy(8,0);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_right,"[%d,",x_next);
                    sprintf(my_right,"%d]",y_now);
                    lcd_gotoxy(8,0);lcd_puts(mx_right);
                    lcd_gotoxy(11,0);lcd_puts(my_right);}
                if(y_prev<0){
                    lcd_gotoxy(15,1);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_low,"[%d,",x_now);
                    sprintf(my_low,"%d]",y_prev);
                    lcd_gotoxy(15,1);lcd_puts(mx_low);
                    lcd_gotoxy(18,1);lcd_puts(my_low);}
                if(x_prev<0){
                    lcd_gotoxy(8,2);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_left,"[%d,",x_prev);
                    sprintf(my_left,"%d]",y_now);
                    lcd_gotoxy(8,2);lcd_puts(mx_left);
                    lcd_gotoxy(11,2);lcd_puts(my_left);}
            }
            else if(heading==SOUTH){
                if(y_next>8){
                    lcd_gotoxy(8,2);lcd_putsf("[-,-]");}
                else{
                    sprintf(mx_up,"[%d,",x_now);
                    sprintf(my_up,"%d]",y_next);
```

```
        lcd_gotoxy(8,2);lcd_puts(mx_up);
        lcd_gotoxy(11,2);lcd_puts(my_up);}
    if(x_next>8){
        lcd_gotoxy(0,1);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_right,"[%d,",x_next);
        sprintf(my_right,"%d]",y_now);
        lcd_gotoxy(0,1);lcd_puts(mx_right);
        lcd_gotoxy(3,1);lcd_puts(my_right);}
    if(y_prev<0){
        lcd_gotoxy(8,0);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_low,"[%d,",x_now);
        sprintf(my_low,"%d]",y_prev);
        lcd_gotoxy(8,0);lcd_puts(mx_low);
        lcd_gotoxy(11,0);lcd_puts(my_low);}
    if(x_prev<0){
        lcd_gotoxy(15,1);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_left,"[%d,",x_prev);
        sprintf(my_left,"%d]",y_now);
        lcd_gotoxy(15,1);lcd_puts(mx_left);
        lcd_gotoxy(18,1);lcd_puts(my_left);}
}
else if(heading==WEST){
    if(y_next>8){
        lcd_gotoxy(15,1);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_up,"[%d,",x_now);
        sprintf(my_up,"%d]",y_next);
        lcd_gotoxy(15,1);lcd_puts(mx_up);
        lcd_gotoxy(18,1);lcd_puts(my_up);}
    if(x_next>8){
        lcd_gotoxy(8,2);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_right,"[%d,",x_next);
        sprintf(my_right,"%d]",y_now);
        lcd_gotoxy(8,2);lcd_puts(mx_right);
        lcd_gotoxy(11,2);lcd_puts(my_right);}
    if(y_prev<0){
        lcd_gotoxy(0,1);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_low,"[%d,",x_now);
        sprintf(my_low,"%d]",y_prev);
        lcd_gotoxy(0,1);lcd_puts(mx_low);
        lcd_gotoxy(3,1);lcd_puts(my_low);}
    if(x_prev<0){
        lcd_gotoxy(8,0);lcd_putsf("[-,-]");}
    else{
        sprintf(mx_left,"[%d,",x_prev);
        sprintf(my_left,"%d]",y_now);
        lcd_gotoxy(8,0);lcd_puts(mx_left);
        lcd_gotoxy(11,0);lcd_puts(my_left);}
}
```

```
        sprintf(mx_now,"[%d,",x_now);
        sprintf(my_now,"%d]",y_now);
        lcd_gotoxy(8,1);lcd_puts(mx_now);
        lcd_gotoxy(11,1);lcd_puts(my_now);

        if(data[4]==1){
            RS=R1;
            lcd_gotoxy(7,3);lcd_putsf("RUANG 1");}
        else if(data[4]==2){
            RS=R2;
            lcd_gotoxy(7,3);lcd_putsf("RUANG 2");}
        else if(data[4]==3){
            RS=R3;
            lcd_gotoxy(7,3);lcd_putsf("RUANG 3");}
        else if(data[4]==4){
            RS=R4;
            lcd_gotoxy(7,3);lcd_putsf("RUANG 4");}
        else{
            RS=LR;
            lcd_gotoxy(7,3);lcd_putsf("LORONG");}
        delay_ms(500);
        lcd_clear();
    }

    //===== INISIALISASI =====
    void RENATO(void)
    {
        start_detection();
        node2node(3,4);
        send_umc(10,10);
        delay_ms(300);
        send_umc(0,0);
        delay_ms(100);
        node2node(0,4);
        send_umc(10,10);
        delay_ms(100);
        send_umc(0,0);
        delay_ms(100);
        node2node(0,0);
        send_umc(10,10);
        delay_ms(300);
        send_umc(0,0);
        delay_ms(100);
        node2node(5,0);}

    void inisialisasi_system(void)
    {
    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In

    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
```

```
        PORTA=0x00;
        DDRA=0x00;

        // Port B initialization
        //   Func7=In   Func6=In   Func5=In   Func4=Out   Func3=In   Func2=Out
Func1=Out Func0=Out
        // State7=T State6=T State5=T State4=1 State3=T State2=0 State1=0
State0=0
        PORTB=0x00;
        DDRB=0x17;

        // Port C initialization
        // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
        // State7=P State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
        PORTC=0x80;
        DDRC=0x00;

        // Port D initialization
        // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
        // State7=T State6=T State5=T State4=T State3=T State2=T State1=T
State0=T
        PORTD=0x00;
        DDRD=0x00;

        // Port E initialization
        // Func7=In Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
        // State7=T State6=0 State5=T State4=T State3=T State2=T State1=T
State0=T
        PORTE=0x00;
        DDRE=0x40;

        // Port F initialization
        // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
        // State7=T State6=T State5=T State4=T State3=T State2=T State1=P
State0=T
        PORTF=0x02;
        DDRF=0x00;

        // Port G initialization
        // Func4=In Func3=In Func2=In Func1=In Func0=In
        // State4=T State3=T State2=T State1=T State0=T
        PORTG=0x00;
        DDRG=0x00;

        // Timer/Counter 0 initialization
        // Clock source: System Clock
        // Clock value: Timer 0 Stopped
        // Mode: Normal top=0xFF
        // OC0 output: Disconnected

        ASSR=0x00;
        TCCR0=0x00;
        TCNT0=0x00;
        OCR0=0x00;

        // Timer/Counter 1 initialization
        // Clock source: System Clock
        // Clock value: Timer1 Stopped
        // Mode: Normal top=0xFFFF
        // OC1A output: Discon.
        // OC1B output: Discon.
        // OC1C output: Discon.
        // Noise Canceler: Off
        // Input Capture on Falling Edge
        // Timer1 Overflow Interrupt: Off
        // Input Capture Interrupt: Off
        // Compare A Match Interrupt: Off
        // Compare B Match Interrupt: Off
        // Compare C Match Interrupt: Off
        TCCR1A=0x00;
        TCCR1B=0x00;
        TCNT1H=0x00;
        TCNT1L=0x00;
        ICR1H=0x00;
        ICR1L=0x00;
        OCR1AH=0x00;
        OCR1AL=0x00;
        OCR1BH=0x00;
        OCR1BL=0x00;
        OCR1CH=0x00;
        OCR1CL=0x00;

        // Timer/Counter 2 initialization
        // Clock source: System Clock
        // Clock value: Timer2 Stopped
        // Mode: Normal top=0xFF
        // OC2 output: Disconnected
        TCCR2=0x00;
        TCNT2=0x00;
        OCR2=0x00;

        // Timer/Counter 3 initialization
        // Clock source: System Clock
        // Clock value: Timer3 Stopped
        // Mode: Normal top=0xFFFF
        // OC3A output: Discon.
        // OC3B output: Discon.
        // OC3C output: Discon.
        // Noise Canceler: Off
        // Input Capture on Falling Edge
        // Timer3 Overflow Interrupt: Off
        // Input Capture Interrupt: Off
        // Compare A Match Interrupt: Off
        // Compare B Match Interrupt: Off
```

```
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: On
// INT2 Mode: Falling Edge
// INT3: On
// INT3 Mode: Falling Edge
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0xA0;
EICRB=0x00;
EIMSK=0x0C;
EIFR=0x0C;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

ETIMSK=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600
UCSR0A=0x00;
UCSR0B=0x98;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x67;

// USART1 initialization
// USART1 disabled
UCSR1B=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 1000.000 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// SPI initialization
// SPI Type: Master
// SPI Clock Rate: 4000.000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x54;
SPSR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

// I2C Bus initialization
i2c_init();

// Alphanumeric LCD initialization
// Connections specified in the
// Project|Configure|C Compiler|Libraries|Alphanumeric LCD menu:
// RS - PORTA Bit 0
// RD - PORTA Bit 1
// EN - PORTA Bit 2
// D4 - PORTA Bit 4
// D5 - PORTA Bit 5
// D6 - PORTA Bit 6
// D7 - PORTA Bit 7
// Characters/line: 20
lcd_init(20);
}
```

Listing Program Mikrokontroler Pengantur Sensor Ultrasonik PING))) (ATmega8)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/15/2013
Author  : Nur Iskandar Juang
Company :
Comments:


Chip type               : ATmega8
Program type            : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega8.h>
#include <delay.h>
#include <stdio.h>
#include <sleep.h>
#define  t1_off             TCCR1B=0x00
#define  t1_on              TCCR1B=0x0B
#define  reset_t1           TCNT1=0x0000

#define  PING_A 0
#define  PING_B 1
#define  PING_C 2
#define  PING_D 3

//deklarasi sub fungsi
void inisialisasi_sistem(void);
void uart_speed_setting(void);
void baca_ping(char i);
void data_out(void);
void kirim_data_UART(unsigned char data);

// Declare your global variables here
char flag=0,schedule=0;
int eksekusi;

unsigned int jarak[4]={0,0,0,0};
char x=0;

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
    schedule=1;
}

// Timer 1 output compare B interrupt service routine
interrupt [TIM1_COMPB] void timer1_compb_isr(void)
{
// Place your code here
    flag=1;
}

void main(void)
{
    inisialisasi_sistem();
    t1_off;
    reset_t1;
    PORTD.6=1;
    PORTD.7=1;
    uart_speed_setting();
    delay_ms(200);
    #asm("sei");

    if(PIND.3==1)
    {
        while(1)
        {
            PORTD.6 = ~PORTD.6;
            PORTD.7 = 0;
            eksekusi=PING_A;   baca_ping(eksekusi);
            eksekusi=PING_C;   baca_ping(eksekusi);
            eksekusi=PING_B;   baca_ping(eksekusi);
            eksekusi=PING_D;   baca_ping(eksekusi);
            data_out();
        };
    }
    else if (PIND.3==0)
    {
        while(getchar()!=0){};
        x=0;
        while(1)
        {
            if(x==0)
            {
                PORTD.6 = ~PORTD.6;
                PORTD.7 = ~PORTD.7;
                kirim_data_UART(0xEE);
                eksekusi=PING_A;   baca_ping(eksekusi);
kirim_data_UART((unsigned char)jarak[eksekusi]);
```

```
                eksekusi=PING_C;   baca_ping(eksekusi);
kirim_data_UART((unsigned char)jarak[eksekusi]);
                eksekusi=PING_B;   baca_ping(eksekusi);
kirim_data_UART((unsigned char)jarak[eksekusi]);
                eksekusi=PING_D;   baca_ping(eksekusi);
kirim_data_UART((unsigned char)jarak[eksekusi]);
                kirim_data_UART(0xFF);
                if(UCSRA & (1<<7))
                {
                    x=UDR;              /// apakah ada instruksi berhenti
                    if(x==1)        {while(getchar()!=0){};}    ///menunggu
re-start
                    else if(x==5)   {OCR1A=0x07D0;OCR1B=0x04E2;}
///menunggu 5ms cycle 8 ms total
                    else if(x==8)   {OCR1A=0x09C4;OCR1B=0x07D0;}
///menunggu 8ms cycle 10 ms total   //OCRA=2ndint,ocrb=1stint
                    else if(x==10)  {OCR1A=0x0BB8;OCR1B=0x09C4;}
////menunggu 10ms cycle 12 ms total
                    else if(x==12)  {OCR1A=0x1193;OCR1B=0x0BB8;}
///menunggu 12ms cycle 18 ms total
                    else if(x==14)  {OCR1A=0x1193;OCR1B=0x0DAB;}
///menunggu 14ms cycle 18 ms total
                    else if(x==16)  {OCR1A=0x1193;OCR1B=0x0FA0;}
///menunggu 16ms cycle 18 ms total
                    x=0;
                }
            }
        };
    }
}

void baca_ping(char i)
{
    t1_off;
    flag=0;
    schedule=0;
    reset_t1;
    PORTB&=0x00;
    DDRB&=0x00;
    PORTC&=0x00;
    DDRC&=0x00;
    t1_on;
    switch(i)
    {
       case 0:
                DDRB.0=1;PORTB.0=1;delay_us(5);PORTB.0=0;
                DDRB.0=0;delay_us(4);
                while(PINB.0==1 && flag==0){};
                while(PINB.0==0 && flag==0){}; t1_on;reset_t1;
                while(PINB.0==1 && flag==0){}; t1_off;
                jarak[i]=(unsigned int)((344*0.5*100*TCNT1*64)/16000000);
                delay_ms(50);
                break;
        case 1:
                DDRB.1=1;PORTB.1=1;delay_us(5);PORTB.1=0;
                DDRB.1=0;delay_us(4);
                while(PINB.1==1 && flag==0){};
                while(PINB.1==0 && flag==0){}; t1_on;reset_t1;
                while(PINB.1==1 && flag==0){}; t1_off;
                jarak[i]=(unsigned int)((344*0.5*100*TCNT1*64)/16000000);
                delay_ms(50);
                break;
            case 2:
                DDRB.2=1;PORTB.2=1;delay_us(5);PORTB.2=0;
                DDRB.2=0;delay_us(4);
                while(PINB.2==1 && flag==0){};
                while(PINB.2==0 && flag==0){}; t1_on;reset_t1;
                while(PINB.2==1 && flag==0){}; t1_off;
                jarak[i]=(unsigned int)((344*0.5*100*TCNT1*64)/16000000);
                delay_ms(50);
                break;
            case 3:
                DDRB.3=1;PORTB.3=1;delay_us(5);PORTB.3=0;
                DDRB.3=0;delay_us(4);
                while(PINB.3==1 && flag==0){};
                while(PINB.3==0 && flag==0){}; t1_on;reset_t1;
                while(PINB.3==1 && flag==0){}; t1_off;
                jarak[i]=(unsigned int)((344*0.5*100*TCNT1*64)/16000000);
                delay_ms(50);
                break;
        }
        t1_on;
        if(schedule==0){idle();}
        t1_off;
    }

    void data_out(void)
    {
       printf(" Ping A=%i , B=%i , C=%i , D=%i", jarak[0], jarak[1], jarak[2],
jarak[3]);
       printf("\r");
    }

    void uart_speed_setting(void)
    {
        #asm("cli")
         if(PIND.3==0)    //operational 1Mbps
            {
             UBRRH=0x00;
             UBRRL=0x00;
            }
          else   //9600bps for debug purpose
            {
             UBRRH=0x00;
             UBRRL=0x67;
            }
        #asm("sei")
    }
```

```
void kirim_data_UART(unsigned char data)
{
while(!(UCSRA & (1<<UDRE))){};
UDR=data;
}

void inisialisasi_sistem(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization
// Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=Out Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=1 State6=1 State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0xC0;
DDRD=0xC0;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 250.000 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: On
TCCR1A=0x00;
TCCR1B=0x0B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x09;
OCR1AL=0xC4;
OCR1BH=0x07;
OCR1BL=0xD0;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x18;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=0x00;
UCSRB=0x18;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x67;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;
}
```

Listing Program Mikrokontroler Pengantur *Driver* Motor (ATmega8)

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.05.0 Professional
Automatic Program Generator
© Copyright 1998-2010 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : Pengenalan Posisi dengan Grid-Based Map
Version : 1.1
Date    : 9/12/2013
Author  : Nur Iskandar Juang
Company : Teknik Elektro Universitas Brawijaya
Comments:


Chip type               : ATmega8
Program type            : Application
AVR Core Clock frequency: 16.000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega8.h>
#include <stdio.h>
#include <delay.h>
#include <math.h>
#include <spi.h>
#include <stdlib.h>

#define system_timer_on     TCCR2=0x0F;TCNT2=0x00
#define system_timer_off    TCCR2=0x00;TCNT2=0x00
#define FWD                 PORTC=0b0001100 // alarm-reset // int.VR.0-ext.1
// cw.0-ccw.1 // run.0-brake.1 // start.0-stop.1 //
#define REV                 PORTC=0b0001000
#define BRK                 PORTC=0b0001011
#define STP                 PORTC=0b0001001
#define full                1023
#define half                512
#define normal              0
#define tuning              1
#define mode_sistem         PIND.2
#define sisi_motor          PIND.5
#define motor_R             0
#define motor_L             1



unsigned int status_mode;
unsigned int motor;
unsigned int pulsa;

struct{
            signed char  terima;
            unsigned long int speed_terima;
            }data={0,0};

// Timer 0 overflow interrupt service routine
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
// Reinitialize Timer 0 value
    TCNT0=0xFF;
// Place your code here
    pulsa++;
}

// SPI interrupt service routine
interrupt [SPI_STC] void spi_isr(void)
{
     //unsigned char data;
     data.terima=SPDR;
// Place your code here

}

// Declare your global variables here
void init_sys(void);
void data_send(void);
void check_mode(void);
void config_kontrol(void);

void main(void)
{
     init_sys();
     #asm("sei")
     pulsa=0;
     data.terima=0;
     while(status_mode==normal)             //NORMAL OPERATION
          {
               check_mode();
               PORTD.3=0;
               if(motor==motor_R)
               {
                       config_kontrol();
                       data.speed_terima=(unsigned long int)abs(data.terima);

                       OCR1A=(unsigned long int)full*data.speed_terima/100;
               }
               else if (motor==motor_L)
               {
                       config_kontrol();
                       data.speed_terima=(unsigned long int)abs(data.terima);
                       OCR1A=(unsigned long int)full*data.speed_terima/100;
               }
          };
```

```
    while(status_mode==tuning)                  // TUNING
        {
                check_mode();
                data.terima=100;
                config_kontrol();
                data.speed_terima=(unsigned long int)abs(data.terima);
                OCR1A=(unsigned long int)125;
                while(1){
                    PORTC.5=~PORTC.5;
                    printf("%d \r",pulsa);
                    delay_ms(250);
                    while(pulsa>30){BRK;OCR1A=(unsigned long int)0;};
                };
        };
}

void data_send(void)
{
    printf("%c     %d \r",data.terima,OCR1A);
    delay_ms(200);
}

void check_mode(void)
{
   if(mode_sistem==0)
        {
            PORTD.3=0;
            status_mode=normal;
        }                   // mode operasional            JUMPER OFF
   if(mode_sistem==1)
        {
            PORTD.3=1;
            status_mode=tuning;
        }                    // mode untuk tuning                JUMPER ON
   if(sisi_motor==0)
        {
            PORTD.6=1;PORTD.7=0;
            motor = motor_R;
        }               //isi pointer structure dengan alamat structure motor
kanan                JUMPER OFF
   if(sisi_motor==1)
        {
            PORTD.6=0;PORTD.7=1;
            motor = motor_L;
        }              //isi pointer structure dengan alamat structure motor
kiri                   JUMPER ON
}

void config_kontrol(void)
{

    if (data.terima>100 || data.terima<-100) {BRK;}
    else if(data.terima>0){
        if(motor==motor_R){REV;}
        else if (motor==motor_L){FWD;}
    }
    else if(data.terima<0){
        if(motor==motor_R){FWD;}
        else if (motor==motor_L){REV;}
    }
    else if(data.terima==0){STP;data.terima=0;}
}

void init_sys(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=Out Func3=Out Func2=Out Func1=Out
Func0=In
// State7=T State6=T State5=T State4=0 State3=0 State2=0 State1=0 State0=T
PORTB=0x00;
DDRB=0x1E;

// Port C initialization
// Func6=In Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State6=T State5=1 State4=0 State3=0 State2=0 State1=0 State0=0
PORTC=0x20;
DDRC=0x3F;

// Port D initialization
// Func7=Out Func6=Out Func5=In Func4=In Func3=Out Func2=In Func1=In
Func0=In
// State7=0 State6=0 State5=T State4=T State3=0 State2=T State1=T State0=T
PORTD=0x00;
DDRD=0xC8;

// Timer/Counter 0 initialization
// Clock source: T0 pin Falling Edge
TCCR0=0x06;
TCNT0=0xFF;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 16000.000 kHz
// Mode: Ph. correct PWM top=0x03FF
// OC1A output: Non-Inv.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x83;
TCCR1B=0x01;
TCNT1H=0x00;
```

```
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer2 Stopped
// Mode: Normal top=0xFF
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x01;

// USART initialization
// USART disabled
UCSRB=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI Type: Slave
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0xC4;
SPSR=0x00;

// Clear the SPI interrupt flag
#asm
    in   r30,spsr
    in   r30,spdr
#endasm

// TWI initialization
// TWI disabled
TWCR=0x00;
}
```

LAMPIRAN 3

DATASHEET