DESAIN DAN IMPLEMENTASI LOGIKA FUZZY SEBAGAI SISTEM NAVIGASI WALL FOLLOWING PADA MOBILE ROBOT KRCI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun Oleh:

RAYI YANU TARA

NIM 0510630086

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2010

DESAIN DAN IMPLEMENTASI LOGIKA FUZZY SEBAGAI SISTEM NAVIGASI WALL FOLLOWING PADA MOBILE ROBOT KRCI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

Memperoleh gelar Sarjana Teknik

Disusun Oleh:

RAYI YANU TARA

NIM 0510630086

Mengetahui dan menyetujui

Dosen pembimbing:

Adharul Muttaqin, ST., MT.	**Ir. Nanang Sulistyanto**
NIP. 19760121 200501 1 001	NIP. 19700113 199403 1 002

DESAIN DAN IMPLEMENTASI LOGIKA FUZZY SEBAGAI SISTEM NAVIGASI WALL FOLLOWING PADA MOBILE ROBOT KRCI

Disusun Oleh:
RAYI YANU TARA
NIM 0510630086

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 26 Januari 2010

DOSEN PENGUJI

Dr. Agung Darmawansyah, ST, MT.
NIP. 19721218 199903 1 002

Ir. Ponco Siwindarto, M.Eng.Sc.
NIP. 19590304 198903 1 001

Panca Mudjirahardjo, ST, MT.
NIP. 19700329 200012 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

Rudy Yuwono, ST, MSc.
NIP. 19710615 199802 1 003

ABSTRAK

Rayi Yanu Tara, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Januari 2010, *Desain dan Implementasi Logika Fuzzy sebagai Sistem Navigasi Wall Following pada Mobile Robot KRCI*, Dosen Pembimbing: Adharul Muttaqin, ST., MT. dan Ir. Nanang Sulistiyanto.

Pada Kontes Robot Cerdas Indonesia, robot yang diperlombakan harus mampu bergerak menyusuri arena perlombaan yang berbentuk labirin-labirin untuk melakukan tugasnya. Pada Divisi Expert Single, susunan ruang dan dinding lintasan pada arena perlombaan selalu berubah-ubah secara acak. Perubahan tersebut membuat robot harus mampu beradaptasi saat menyusuri arena lomba.

Berdasarkan kondisi tersebut dilakukan Desain dan Implementasi Logika Fuzzy sebagai Sistem Navigasi Wall Following pada Mobile Robot KRCI yang memodelkan perilaku mengikuti dinding dengan menggunakan logika fuzzy untuk diterapkan pada mobile robot yang mengikuti Kontes Robot Cerdas Indonesia. Sistem yang dibuat akan menghasilkan keputusan pergerakan robot berdasarkan jarak robot terhadap dinding yang diikuti. Jarak robot terhadap dinding diukur menggunakan sensor ultrasonik. Pengaturan sensor ultrasonik akan menggunakan sebuah mikrokontroler terpisah yaitu Atmel ATMega8. Proses utama yaitu logika fuzzy terdapat pada mikrokontroler utama yaitu Atmel ATMega32. Komunikasi antar mikrokontroler menggunakan metode UART dengan baudrate 1 Mbps.

Hasil pengujian menunjukan bahwa sensor rotary encoder mampu bekerja hingga frekuensi 1074 pulsa per detik. Hasil pengujian menunjukan sistem komunikasi UART antar mikrokontroler telah berhasil dengan mengirimkan paket data berisi jarak pembacaan seluruh sensor. Melalui hasil pengujian diketahui bahwa setiap proses pembuatan keputusan memerlukan waktu sebesar 0,060184 detik. Pengujian keseluruhan sistem pada model arena perlombaan dan mengikuti dua sisi dinding yang berbeda menunjukan bahwa sistem dapat memberikan keputusan gerak pada robot sesuai kondisi arena perlombaan.

Kata kunci: logika fuzzy, robot, navigasi, *wall following*

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 Logika Fuzzy sebagai Sistem Navigasi Wall Following pada Mobile Robot KRCI" 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:

- Orangtua dan kakak penulis yang selalu memberikan kasih sayang, dukungan dan semangat,
- Bapak Rudy Yuwono, ST., MSc. sebagai Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Ir. M. Julius St, MS. sebagai Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Adharul Muttaqin, ST., MT. sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, ide, saran serta motivasi yang telah diberikan,
- Bapak Ir. Nanang Sulistiyanto sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, saran dan kritik yang telah diberikan baik selama pengerjaan skripsi ini maupun selama menjadi pembimbing dalam mengikuti kompetisi KRCI 2008 dan 2009,
- Staff Recording Jurusan Teknik Elektro,
- Teman - teman Streamline angkatan 2005,
- Teman - teman Laboratorium Elka dan SisDig,
- Rekan seperjuangan dalam skripsi, Fido, Nino, Eko, Riezqy "mbatu", "om" Arnez, Rizal "tero", Pras "ebes", terima kasih atas segala bantuan yang telah diberikan,
- Seluruh anggota tim robotika KRCI dan KRI TEUB periode 2008-2010, terima kasih atas seluruh bantuanya.

- 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 2010

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kontes Robot Cerdas Indonesia (KRCI) merupakan salah satu kompetesi robotika tingkat nasional yang diadakan secara teratur setiap tahun oleh Direktorat Jenderal Pendidikan Tinggi. Pertandingan ini dibagi menjadi beberapa divisi yakni Divisi Senior Beroda, Senior Berkaki, Expert Single, dan Expert Battle. Masing-masing divisi mempunyai aturan, tugas, dan arena yang berbeda.

Robot yang mengikuti Kontes Robot Cerdas Indonesia akan diletakan pada sebuah arena yang merupakan simulasi rumah yang terdiri dari ruang-ruang. Pada beberapa divisi, arena pertandingan yang digunakan dapat berubah-ubah sesuai dengan undian. Robot harus mampu beradaptasi dan melaksanakan tugasnya sesuai dengan kondisi arena pertandingan dengan cara bergerak menyusuri arena. Salah satu metode untuk menyusuri arena adalah dengan mengikuti sisi dinding pada arena (wall following). Agar dapat mengambil keputusan sendiri, sebuah mobile robot yang cerdas tentunya harus mampu mengenali keadaan lingkungan dimana robot tersebut beroperasi. Misal mobile robot yang dirancang harus memiliki kemampuan mendeteksi keberadaan dinding arena yang menjadi lintasan robot. Untuk tujuan tersebut maka sebuah mobile robot harus dilengkapi dengan sensor yang dapat mengambil data yang kemudian di olah oleh mikrokontroler.

Salah satu sensor yang banyak digunakan untuk hal di atas adalah sensor ultrasonik. Akan tetapi ada beberapa keterbatasan dalam menggunakan sensor ini, diantaranya adalah sensitifitas dari sensor sangat tergantung dari besar sudut yang dibentuk oleh sensor dengan objek dan jika semakin jauh jarak objek yang terdeteksi, jika sudut yang dibentuk terlalu besar maka sinyal tidak akan terpantul ke penerima sehingga dimungkinkan posisi objek tersebut semakin tidak diketahui secara pasti. Selain itu semakin jauh jarak objek yang terdeteksi, maka posisi objek tersebut semakin tidak diketahui secara pasti. Penggunaan sensor ultrasonik lebih dari satu buah akan menimbulkan interferensi sehingga terjadi kesalahan pengukuran.

Banyaknya parameter dan kompleksitas sistem robot membuat kita tidak dapat secara langsung mengintepretasikan data jarak dan kemiringan posisi yang dihasilkan

sensor secara langsung tanpa pengolahan awal. Pengolahan parameter-parameter ini akan menjadi lebih sulit jika menggunakan suatu rumusan eksak matematis yang rumit.

Salah satu solusi untuk permasalahan di atas adalah dengan menggunakan algoritma fuzzy yang menggunakan rumusan secara linguistik dan rule-based untuk memodelkan perilaku mengikuti dinding pada robot. Algoritma fuzzy dapat digunakan pada berbagai macam sistem dengan hanya menggunakan variabel-variabel masukan dan keluaran sistem tanpa harus mengetahui fungsi alih sistem tersebut.

1.2. Rumusan Masalah

Berdasarkan masalah yang telah dijelaskan pada latar belakang, dapat dibuat rumusan sebagai berikut:

1). Bagaimana cara mengakses sensor ultrasonik dan sensor putaran (*rotary encoder*).
2). Bagaimana merancang sistem akses sensor ultrasonik pada mikrokontroler AVR ATMega8.
3). Bagaimana membuat dan mengolah sensor putaran (*rotary encoder*) yang terpasang pada kedua roda agar arah dan kecepatan putaran motor penggerak dapat dikendalikan .
4). Bagaimana merancang dan menerapkan logika *fuzzy* sebagai kendali robot dalam model navigasi *wall following*.
5). Bagaimana penerapan logika fuzzy yang telah dibuat pada *mobile robot* berbasis mikrokontroler AVR ATMega 32.

1.3. Batasan Masalah

Dalam perancangan untuk skripsi ini permasalahan dibatasi oleh hal-hal sebagai berikut :

1). Sensor ultrasonik yang dipakai adalah sensor ultrasonik tipe PING))) untuk mengukur jarak.
2). Sensor putaran (*rotary encoder*) yang digunakan memiliki jenis *incremental encoder*.
3). Model lintasan/lorong yang digunakan sesuai aturan dalam Kontes robot Cerdas Indonesia (KRCI) Divisi Expert Single.
4). Implementasi desain menggunakan mikrokontroler lebih dari satu buah (*multi-microcontroler*).

1.4. Tujuan

Tujuan skripsi ini adalah dapat diciptakan sistem navigasi *wall following* menggunakan metode logika fuzzy untuk diterapkan pada *mobile robot* yang akan mengikuti kompetisi KRCI.

1.5. Sistematika Pembahasan

Skripsi ini terdiri dari enam bab dengan sistematika pembahasan sebagai berikut:

BAB I Pendahuluan

Membahas latar belakang, rumusan masalah, batasan masalah, tujuan dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat, yang meliputi Kontes Robot Cerdas Indonesia, sensor putaran (*rotary encoder*), sensor ultrasonik, logika fuzzy, mikrokontroler AVR, modul pengendali motor DC *brushed* dan *mobile robot*.

BAB III Metodologi Penulisan

Membahas metode penelitian dan perencanaan alat.

BAB IV Perencanaan dan Pembuatan Alat

Membahas penentuan tata letak dan metode akses sensor ultrasonik dan sensor *rotary encoder*. Selanjutnya membahas tentang perancangan metode *wall following* menggunakan logika *fuzzy*. Setelah itu, bagaimana cara menerapkannya ke dalam mikrokontroler dan bagaimana menerapkannya dalam alat secara keseluruhan.

BAB V Pengujian dan Analisis

Membahas hasil pengujian sistem terhadap alat yang telah direalisasikan.

BAB VI Kesimpulan dan Saran

Membahas kesimpulan perancangan ini dan saran-saran yang diperlukan untuk melakukan pengembangan aplikasi selanjutnya.

BAB II

TINJAUAN PUSTAKA

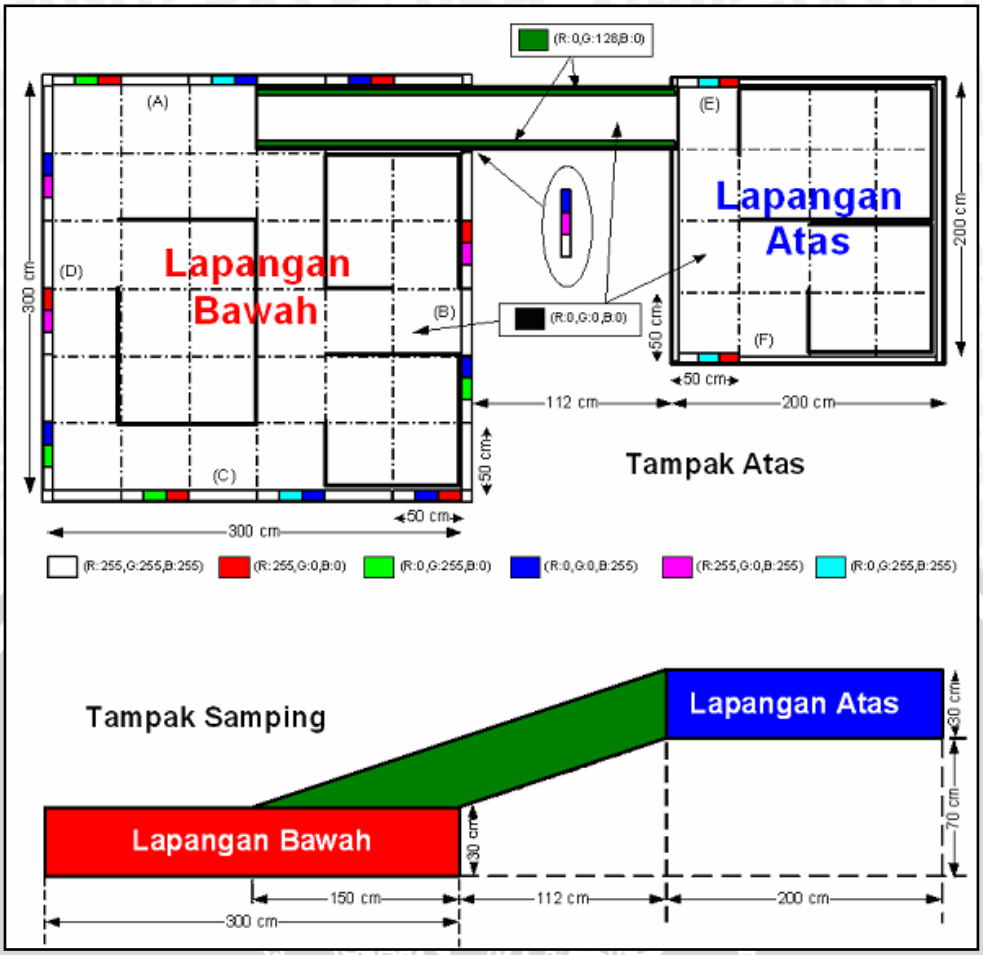
Untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar-dasar perencanaan dari sistem yang dibuat, maka perlu adanya penjelasan dan uraian mengenai teori penunjang yang digunakan dalam penulisan tugas akhir ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah:

- Kontes Robot Cerdas Indonesia
- Logika *Fuzzy*
- Sensor Ultrasonik
- Sensor *Rotary Encoder*
- Motor DC
- Mobile Robot sistem Diferensial
- Mikrokontroler AVR ATMega8 dan ATMega32
- Modul Pengendali Motor DC

2.1. Kontes Robot Cerdas Indonesia (KRCI)

Kontes Robot Cerdas Indonesia merupakan pertandingan robot tingkat nasional yang diadakan oleh Direktorat Jendral Pendidikan Tinggi (Dirjen DIKTI). Pertandingan ini klasifikasikan ke dalam beberapa divisi yakni Divisi Senior Beroda, Senior Berkaki, Expert Single, dan Expert Batle. Secara garis besar, seluruh divisi memiliki tugas yang sama yaitu bergerak dari posisi *home* menelusuri arena untuk memadamkan api kemudian kembali lagi ke posisi *home*. Pada divisi Expert Single dan Expert Battle terdapat tugas tambahan yaitu menyelamatkan sebuah boneka.

Robot yang digunakan pada Kontes Robot Cerdas Indonesia ini memiliki panjang dan lebar maksimum 30 cm x 30 cm dan tinggi maksimum 30 cm. Pada seluruh divisi selain divisi Expert Battle, arena lomba memiliki konfigurasi acak baik ruangan, posisi *home*, lokasi *hanging object* (cermin dan *sound damper*), lokasi lilin, lokasi *furniture*, lokasi boneka dan tangga (Expert Single) dan *uneven floor*. Gambar 2.1 menunjukan salah satu konfigurasi arena lomba pada divisi Expert Single. Lebar lorong lapangan sebesar 50 cm. Dalam Gambar 2.1 ditunjukan bahwa lintasan robot pada arena lomba dapat berupa lorong atau hanya sebuah dinding pada salah satu sisi lintasan.



Gambar 2.1. Arena Kontes Robot Cerdas Indonesia Divisi Expert Single

Sumber: DIKTI, 2008.

2.2. Logika Fuzzy

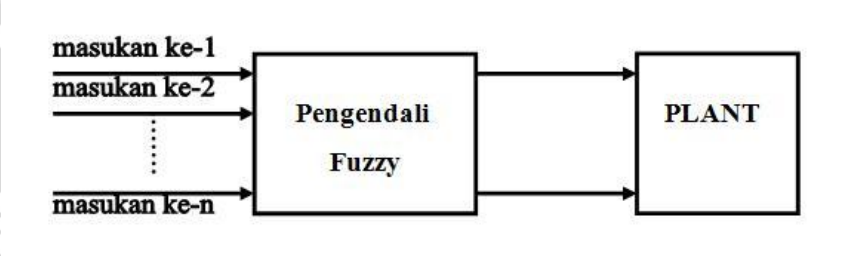
Fuzzy secara harfiah berarti samar, sedangkan kebalikannya dalam hal ini adalah C*risp* yang secara harfiah berarti tegas. Dalam kehidupan sehari-hari nilai samar lebih akrab daripada nilai tegas. Temperatur/suhu tertentu biasa dinyatakan sebagai panas, agak panas, atau sangat dingin daripada dinyatakan dalam nilai terukur tertentu.

Tahun 1965 L.A. Zadeh memodifikasi teori himpunan yang disebut himpunan samar (*Fuzzy Set*). Himpunan *fuzzy* didasarkan pada gagasan untuk memperluas jangkauan fungsi karakteristik sehingga fungsi tersebut akan mencakup bilangan real pada interval [0,1]. Nilai keanggotaannya menunjukkan bahwa suatu nilai dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga nilai yang terletak diantaranya. Dengan kata lain nilai kebenaran suatu hal tidak hanya bernilai benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar dan masih ada nilai-nilai yang terletak diantaranya.

Sejak tahun 1985 pengendalian berbasis logika *fuzzy* mengalami perkembangan pesat, terutama dalam hubungannya dengan penyelesaian masalah kendali yang bersifat tak linier, sulit dimodelkan, berubah karakteristiknya terhadap waktu (*time varying*) dan kompleks.

2.2.1. Struktur Dasar Kontrol Logika Fuzzy

Dalam sistem pengendalian dengan logika *fuzzy* dilibatkan suatu blok pengendali yang menerima satu atau lebih masukan dan mengumpankan satu atau lebih keluaran ke plant atau blok lain sebagaimana ditunjukan dalam Gambar 2.2.



Gambar 2.2. Pengendali Fuzzy

Sumber : Coughanowr,1991

Komponen utama penyusun *Fuzzy Logic Controller* adalah unit fuzzifikasi, *fuzzy inference*, basis pengetahuan dan unit defuzzifikasi.

Basis pengetahuan terdiri dari dua jenis (Yan : 94) :

- Basis data

  Mendefinisikan parameter *fuzzy* sebagai bagian dari himpunan *fuzzy* dengan menentukan batas-batas fungsi keanggotaan pada semesta pembicaraan untuk tiap-tiap variabel.
- Basis aturan

  Memetakan nilai masukan *fuzzy* menjadi nilai keluaran *fuzzy*.

2.2.2. Fungsi Keanggotaan

Fungsi keanggotaan menotasikan nilai kebenaran anggota-anggota himpunan *fuzzy*. Interval nilai yang digunakan untuk menentukan fungsi keanggotaan, yaitu nol dan satu. Tiap fungsi keanggotaan memetakan elemen himpunan *crisp* ke semesta himpunan *fuzzy*.

Suatu himpunan *fuzzy* A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, μ_A yang harganya berada dalam interval [0,1]. (Kuswadi, 2000:27). Secara matematika hal ini dinyatakan dengan :

$$\mu_A : U \rightarrow [0,1]$$

Berikut ini beberapa macam keanggotaan yang sering digunakan antara lain fungsi keanggotaan S, π, T (*triangular*).

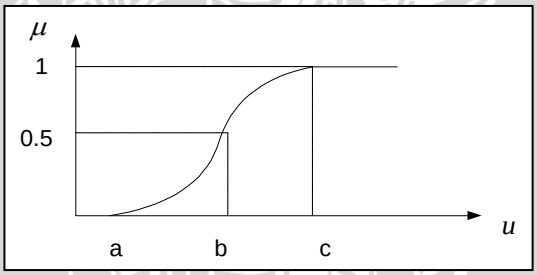
1. Fungsi keanggotaan bentuk S

   Definisi fungsi-S adalah sebagai berikut :

$$S(u,a,b,c) = \begin{cases} 0 & u < a \\ 2\left(\dfrac{u-a}{c-a}\right)^2 & a \le u \le b \\ 1-2\left(\dfrac{u-a}{c-a}\right)^2 & b \le u \le c \\ 1 & u > c \end{cases} \qquad (2.2)$$

$$b = \frac{(a+c)}{2}$$

   Bentuk fungsi keanggotaan bentuk S ditunjukkan dalam Gambar 2.3.



Gambar 2.3. Fungsi Keanggotaan Bentuk S

Sumber : Yan,1994 : 18

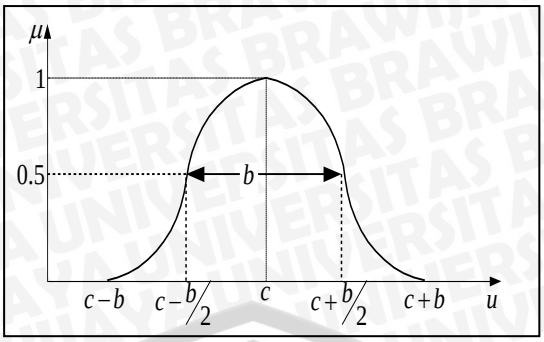
Fungsi keanggotaan bentuk S ini digunakan bila diinginkan himpunan *fuzzy* mempunyai nilai kebenaran mendekati nol dan satu lebih banyak.

2. Fungsi keanggotaan bentuk π

   Definisi fungsi-π adalah sebagai berikut:

$$\pi(u;a,b,c) = \begin{cases} S(u;c-b,c-\dfrac{b}{2},c) & u \le c \\ 1-S(u;c,c+\dfrac{b}{2},c+b) & u \ge c \end{cases} \qquad (2.3)$$

   Bentuk fungsi keanggotaan bentuk π ditunjukkan dalam Gambar 2.4.



Gambar 2.4. Fungsi Keanggotaan Bentuk π

Sumber : Yan, 1994 : 19

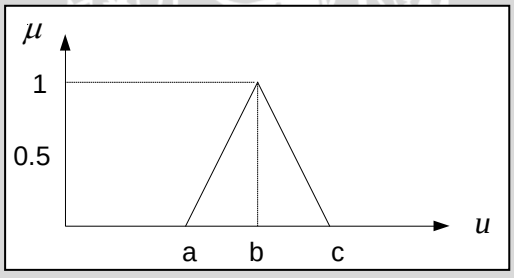
Fungsi keanggotaan bentuk π digunakan jika diinginkan elemen himpunan *fuzzy* memiliki nilai kebenaran mendekati nol lebih banyak.

3. Fungsi keanggotaan bentuk Triangular

Definisi fungsi triangular sebagai berikut:

$$T(u;a,b,c) = \begin{cases} 0 & u < a \\ \dfrac{u-a}{b-a} & a \le u \le b \\ \dfrac{c-u}{c-b} & b \le u \le c \\ 0 & u > c \end{cases} \qquad (2.4)$$

Fungsi keanggotaan bentuk Tringular ditunjukkan dalam Gambar 2.5



Gambar 2.5. Fungsi Keanggotaan Bentuk Triangular (T)

Sumber : Yan, 1994 : 19

Fungsi keanggotaan bentuk triangular ini digunakan bila diinginkan himpunan *fuzzy* mempunyai nilai proporsional terhadap nol maupun satu.

2.2.3. Kendali Logika Fuzzy

Kendali logika *fuzzy* adalah sistem berbasis aturan (*rule based system*) yang didalamnya terdapat himpunan aturan fuzzy yang mempresentasikan mekanisme pengambilan keputusan. Aturan yang dibuat digunakan untuk memetakan variabel input ke variabel output dengan pernyataan *If - Then.*

Kontroler ini akan menggunakan data tertentu *(crisp)* dari sejumlah sensor kemudian mengubahnya menjadi bentuk linguistik atau fungsi keanggotaan melalui proses fuzzifikasi. Lalu dengan aturan *fuzzy*, *inference engine* yang akan menentukan hasil keluaran *fuzzy*. Setelah itu hasil ini akan diubah kembali menjadi bentuk numerik melalui proses defuzzifikasi.

2.2.3.1. Fuzzifikasi

Proses fuzzifikasi merupakan proses untuk mengubah variabel non *fuzzy* (variabel numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali logika *fuzzy* harus diubah terlebih dahulu ke dalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun, maka dari nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy* pula. Proses ini disebut fuzzifikasi. (Yan,1994:49). Proses fuzzifikasi diekspresikan sebagai berikut:

$x = \textit{fuzzifier}\ (x_0)$

dengan:

x_0 = nilai *crisp* variabel masukan

x = himpunan *fuzzy* variabel yang terdefinisi

fuzzifier = operator fuzzifikasi yang memetakan himpunan *crisp* ke himpunan *fuzzy*

Pedoman memilih fungsi keanggotaan untuk proses fuzzifikasi, menurut Jun Yan, menggunakan :

1. Himpunan *fuzzy* dengan distribusi simetris.
2. Gunakan himpunan *fuzzy* dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*).
3. Mengatur himpunan *fuzzy* agar saling menumpuk.
4. Menggunakan fungsi keanggotaan bentuk segitiga atau trapesium.

2.2.3.2. Kaidah Atur Fuzzy (Fuzzy Rule)

Fuzzy rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel-variabel linguistik dan berbasis pengetahuan seorang operator ahli. Pernyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis *fuzzy*, aturan pengendalian *fuzzy* berbentuk aturan " IF – THEN". Untuk sebuah sistem MISO *(Multi Input Single Output)* basis aturan pengendalian fuzzy berbentuk seperti berikut ini.

Rule 1 IF X_1 is A_{11} AND … AND x_m is A_{1m} THEN Y is B_1

Rule 2 IF X_1 is A_{21} AND … AND x_m is A_{2m} THEN Y is B_2

. .

. .

. .

Rule n IF X_n is A_{n1} AND … AND x_m is A_{nm} THEN Y is B_n

Dengan X_j merupakan variabel masukan sistem , A_{ij} merupakan *fuzzy set* untuk X_j , Y merupakan variabel keluaran sistem, B_i merupakan *fuzzy set* untuk Y , AND adalah operator *fuzzy*.

2.2.3.3. Metode Inferensi

Metode inferensi merupakan proses untuk mendapatkan keluaran dari suatu kondisi masukan dengan mengikuti aturan-aturan yang telah ditetapkan. Keputusan yang didapatkan pada proses ini masih dalam bentuk *fuzzy* yaitu derajat keanggotaan keluaran.

Ada beberapa cara untuk mengidentifikasi aturan mana yang akan dipakai dengan menggunakan nilai keanggotaan masukan. Menurut Jun Yan, diantara bermacam-macam metode inferensi *fuzzy* ada dua metode yang paling sering digunakan pada kendali logika *fuzzy* yaitu :

1. Metode Inferensi Max – Min

   Pada metode Max – Min aturan operasi minimum Mamdani digunakan untuk implikasi *fuzzy*. Persamaan aturan minimum adalah

$$\mu_{C'} = \bigcup_{1}^{n} \alpha_i \wedge \mu ci \qquad (2.5)$$

   dengan $\alpha_i = \mu_{Ai}(x_0) \wedge \mu_{Bi}(y_0)$

   Sebagai contoh , terdapat dua basis kaidah atur *fuzzy*, yaitu :

R_1 : Jika x adalah A_1 dan y adalah B_1 maka z adalah C_1

R_2 : Jika x adalah A_2 dan y adalah B_2 maka z adalah C_2

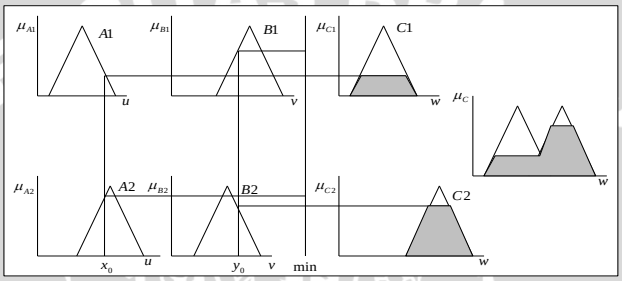
Pada metode penalaran Max-Min fungsi keanggotaan konsekuen dinyatakan dengan

$$\mu c_{1'}(W) = \mu_{c'1} \vee \mu_{c'2} = [\alpha_1 \wedge \mu_{c1}(w)] \vee [\alpha_2 \wedge \mu_{c2}(w)] \quad (2.6)$$

dimana $\alpha_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$ (2.7)

$$\alpha_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0) \quad (2.8)$$

Lebih jelas metode ini dideskripsikan dalam Gambar 2.6.



Gambar 2.6. Inferensi *Fuzzy* dengan Metode MAX-MIN

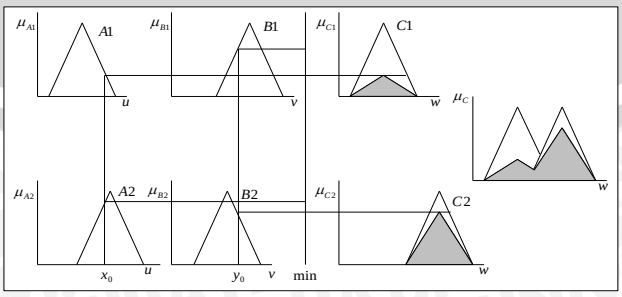
Sumber : Yan, 1994 : 55

2. Metode Inferensi Max – Dot

Pada metode inferensi Max – Dot, aturan operasi Larsen digunakan untuk fungsi implikasi *fuzzy*. Metode Max-Dot fungsi keanggotaan konsekuen C dinyatakan dengan:

$$\mu_c(\omega) = (\alpha_1 \bullet \mu_{c1}(\varpi)) \vee (\alpha_2 \bullet \mu_{c2}(\varpi)) \quad (2.9)$$

Metode penalaran Max-Dot ditunjukan dalam Gambar 2.7.



Gambar 2.7. Inferensi *Fuzzy* dengan Metode MAX-DOT

Sumber : Yan, 1994 : 55

2.2.3.4. Defuzzifikasi

Defuzzifikasi adalah proses untuk mendapatkan nilai numerik dari data fuzzy yang dihasilkan dari proses inferensi. (Yan, 1994 : 55). Proses defuzzifikasi dinyatakan sebagai berikut :

$$y_0 = defuzzifier(y) \tag{2.10}$$

dengan:

y : aksi kontrol *fuzzy.*

y_0 : aksi kontrol *crisp.*

defuzzifier : operator *defuzzifikasi*

Dua metode defuzzifikasi yang umum digunakan adalah :

1. Metode COA (*Center Of Area*)

   Metode ini didefinisikan sebagai berikut:

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i} \tag{2.11}$$

   dengan:

   U = Keluaran

   w_i = Bobot nilai benar w_i

   u_i = Nilai linguistik pada fungsi keanggotaan keluaran

   n = Banyak derajat keanggotaan

2. Metode MOM (*Mean of Maximum*).

   Metode ini didefinisikan sebagai berikut:

$$Z = \frac{\sum_{i=1}^{n} z_i}{n} \tag{2.12}$$

   dengan:

   n = Jumlah proposisi / domain

   Z = Keluaran

   z_i = Domain yang memiliki derajat keanggotaan terbesar / maksimum.

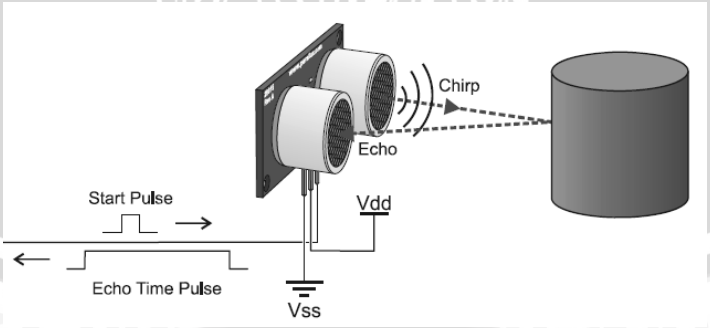
2.3. Sensor Ultrasonik PING)))

Sensor ultrasonik yang dipakai dalam skripsi ini adalah sensor PING))) produksi dari Parallax yang berupa modul siap pakai lengkap dengan pengirim dan penerima. Sensor ultrasonik digunakan untuk mendeteksi jarak dengan frekuensi 40 kHz. Sinyal data sensor PING))) ini akan masuk ke kaki mikrokontroler.

Gambar 2.8. Sensor ultrasonik PING)))

Sumber: Parallax, 2008

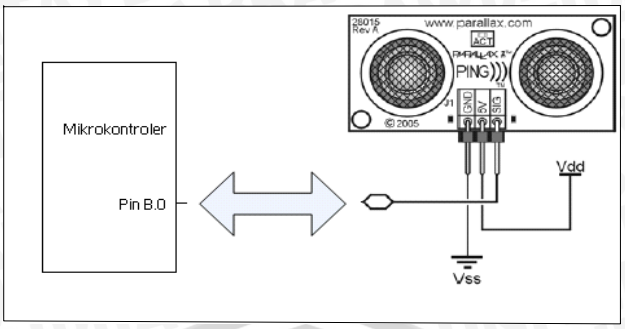
Modul PING))) mengukur jarak objek dengan cara memancarkan gelombang ultrasonik (40kHz) selama t_{BURST} (200 μs) kemudian menunggu pantulannya. Modul PING))) memancarkan gelombang ultrasonik sesuai dengan masukan kontrol dari pin SIG. Gelombang ultrasonik ini melalui udara dengan kecepatan kurang lebih 344 meter per detik, mengenai objek dan memantul kembali ke modul PING))). Modul PING))) akan mengeluarkan pulsa *high* pada pin SIG setelah 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 X 344m/s) / 2] meter. Ilustrasi cara kerja modul PING))) ditunjukkan dalam Gambar 2.9.



Gambar 2.9. Ilustrasi Cara Kerja Modul PING)))

Sumber: Parallax, 2008

Sensor 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 2.10.



Gambar 2.10. Komunikasi Mikrokontroler dengan PING)))

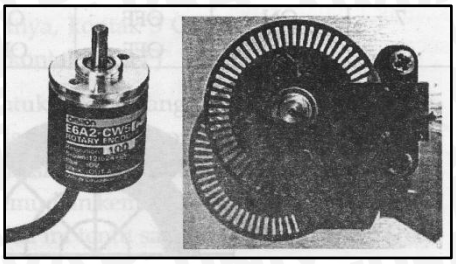
Sumber: Parallax, 2008

Sensor PING))) ini memiliki beberapa kelebihan diantaranya bahwa sensor ini memiliki tingkat akurat yang tinggi, sensor ini mampu mendeteksi benda didepannya walaupun sudutnya tidak tegak lurus dan sensor ini mampu mendeteksi benda walaupun benda tersebut berukuran kecil.

2.4. Sensor Rotary Encoder

Sensor putaran (*rotary encoder*) digunakan untuk mendeteksi banyak putaran pada motor DC. Putaran motor pada aplikasi ini dihitung setiap selang waktu tertentu. Dengan mengetahui putaran motor penggerak robot maka dapat ditentukan pergerakan, kecepatan, percepatan atau sudut putaran. Salah satu sensor *rotary* yang sering digunakan adalah jenis optik. Sensor ini terdiri dari *Light Emitting Diode* (LED) inframerah dan *phototransistor*. Secara prinsip sensor mendeteksi halangan diantara LED inframerah dan *phototransistor*. Apabila tidak terhalang maka cahaya dari LED akan mengenai transistor dan transistor akan aktif. Apabila cahaya LED terhalang maka transistor tidak aktif.

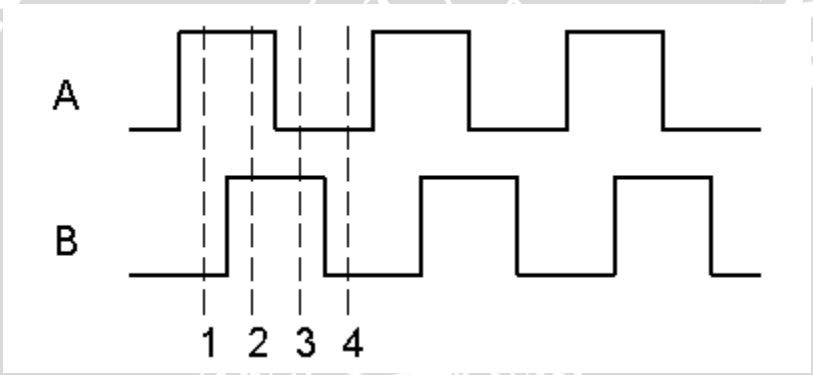
Penghalang cahaya pada sensor optocoupler ini dapat terjadi dengan menggunakan sebuah piringan yang terhubung dengan poros motor. Pada piringan tersebut perlu diberikan lubang-lubang yang akan dilewati sinar inframerah pada posisi-posisi tertentu. Untuk menambah akurasi perhitungan dapat dilakukan dengan menambah jumlah lubang yang ada pada piringan atau dengan menggunakan roda gigi percepatan. Adapun ilustrasi sensor rotari seperti ditunjukkan dalam Gambar 2.11.



Gambar 2.11. Sensor rotari

Sumber: Sigit, 2007:44.

Dengan menggunakan dua buah *optocoupler* dengan posisi bergeser 90° maka didapatkan sinyal keluaran dari sensor berupa sinyal *quadrature*. Sinyal *quadrature* terdiri dari dua sinyal periodik dengan beda fasa sebesar 90° sehingga dapat dideteksi arah putaran pada piringan berlubang. Ilustrasi sinyal *quadrature* ditunjukan dalam Gambar 2.12.



Gambar 2.12. Ilustrasi Sinyal Quadrature

Sumber: Borenstein, 1996: 14.

2.5. Motor DC Brushed

Prinsip kerja motor DC *brushed* sesuai dengan hukum kemagnetan Lorenz, yaitu membangkitkan fluksi magnet pada suatu konduktor berarus dalam medan magnet sehingga timbul ggl induksi.

Setiap arus yang mengalir melalui sebuah konduktor akan menimbulkan medan magnet. Arah medan magnet dapat ditentukan dengan kaidah tangan kiri.

Kaidah tangan kiri untuk motor menunjukkan arah arus yang mengalir didalam sebuah konduktor yang berada dalam medan magnet. Jari tengah menunjukkan arah arus yang mengalir pada konduktor, jari telunjuk menunjukkan arah medan magnet dan ibu jari menunjukkan arah gaya putar. Adapun besarnya gaya yang bekerja pada konduktor tersebut dapat dirumuskan dengan :

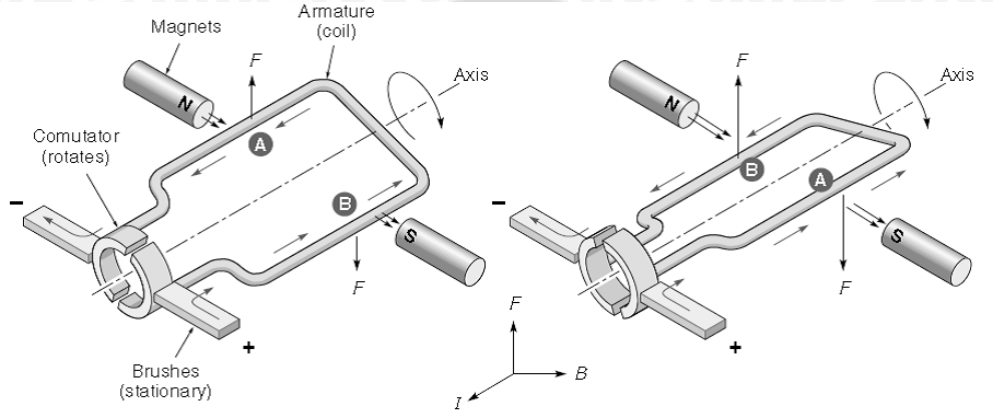
F = B.I.L (Newton) (2.13)

Dimana :

B = kerapatan fluks magnet (weber)

L = panjang konduktor (meter)

I = arus listrik (ampere)



Gambar 2.13. Ilustrasi Motor DC *Brushed*

Sumber : Kilian, 2002

Gambar 2.13 adalah ilustrasi cara kerja motor DC yang mempunyai satu lilit kawat a–b berada di dalam medan magnet. Lilitan ini dapat berputar dengan bebas, lilitan ini biasa disebut dengan jangkar (*armature*).

Pada jangkar diberikan arus yang berasal dari sumber yang terhubung dengan sikat (*brushes*). Sikat-sikat ini terpasang pada sebuah cincin yang terbelah dua, yang disebut cincin belah (*comutator*). Adapun tujuan dari konstruksi ini adalah agar lilitan kawat dapat berputar apabila ada arus listrik yang melewatinya.

Pada kawat yang berada di kanan arus mengalir dari depan ke belakang. Pada kawat yang berada di bagian kiri, arus mengalir dari belakang ke depan kawat a dan b secara bergantian berada di kiri dan kanan. Karena itu arah arus di a dan arah arus di b selalu bersifat bolak-balik. Pembalikan arah arus itu terjadi pada saat lilitan kawat melintasi posisi vertikal.

Bagian *comutator* berfungsi sebagai penyearah mekanik. Fluksi magnet yang ditimbulkan magnet permanen disebut medan magnet motor. Dalam gambar 2.13 arah fluk magnetik adalah dari kiri ke kanan. Adapun gaya yang bekerja pada penghantar b adalah ke atas, sementara gaya yang bekerja pada penghantar a adalah ke bawah . Gaya-gaya yang bekerja sama kuatnya, sehingga terdapat kopel yang bekerja pada kawat sehingga lilitan jangkar dapat berputar. Setelah berputar 180° arah arus berbalik, pada

saat itu penghantar a dan penghantar b bertukar tempat. Akibatnya arah gerak putaran tidak berubah.

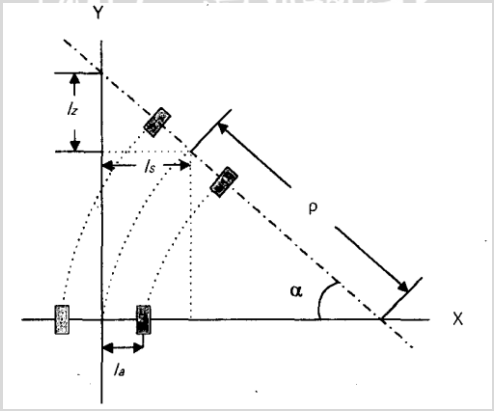
2.6. Robot Mobil Sistem Diferensial

Robot mobil sistem diferensial merupakan robot mobil yang bergerak dengan menggunakan dua buah roda yang dapat digerakkan secara terpisah. Roda diletakkan pada kedua sisi robot. Robot dapat merubah arah pergerakannya dengan mengatur putaran pada kedua roda. Oleh karena itu pada robot ini tidak diperlukan pengendali arah gerakan tambahan.

Posisi robot terhadap dapat ditentukan dengan memantau putaran roda menggunakan sensor rotari. Apabila besarnya putaran roda sisi kanan dan kiri robot tidak sama, maka akan menghasilkan gerak melingkar. Gambar 2.14 menunjukkan ilustrasi gerak melingkar pada robot mobil diferensial. Nilai jarak tempuh roda kanan (S_R) dan kiri (S_L) dapat diketahui dengan menggunakan perhitungan data sensor rotari. Besarnya sudut (α) dapat ditentukan dengan menggunakan Persamaan 2.14. Nilai jari-jari ρ dari gerak melingkar dapat ditentukan dengan menggunakan Persamaan 2.15.

$$\alpha = \frac{\Delta S_L - \Delta S_R}{2.I_a} \quad (2.14)$$

$$\rho = \frac{I_a(S_L + S_R)}{S_L - S_R} \quad (2.15)$$



Gambar 2.14. Pergerakan robot dengan putaran roda berbeda

Sumber: Furnio Harashima,1999:931

2.7. Mikrokontroler Atmel AVR ATMega32

Secara umum, mikrokontroler berfungsi sama dengan komputer. Bedanya adalah mikrokontroler memiliki desain dalam sebuah *single chip* (IC). Mikrokontroler terdapat di hampir semua peralatan elektronik di sekeliling kita, didalam tape, TV, radio, telepon genggam (*Hand Phone*). Mikrokontroler memiliki kemampuan yang diperlukan untuk

membuat keputusan berdasarkan sinyal dari luar dengan kata lain mikrokontroler merupakan otak dari sebuah perangkat elektronik.

ATMega32 merupakan salah satu mikrokontroler buatan ATMEL keluarga ATMega yang mempunyai 32 kbyte Flash PEROM (*Flash Programmable and Erasable Read Only Memory*), 2 kbyte SRAM, 32 pin I/O (4 buah port I/O bit) yang mana tiap pin tersebut dapat diprogram secara paralel dan tersendiri, mempunyai dua buah *timer/counter* 8 bit dan satu buah *timer/counter* 16 bit,mempunyai 8 bit 10 channel ADC, mempunyai *watchdog timer*.

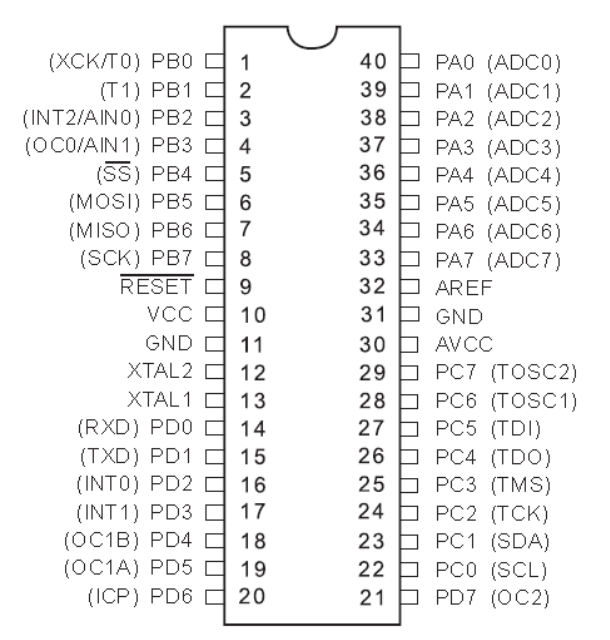
Pada dasarnya mikrokontroler adalah terdiri atas mikroprosesor, *timer*, dan *counter*, perangkat I/O dan internal memori. Mikrokontroler termasuk perangkat yang sudah didesain dalam bentuk chip tunggal. Mikrokontroler dikemas dalam satu chip (*single chip*). Mikrokontroler didesain dengan instruksi-instruksi lebih luas dan 8 bit instruksi yang digunakan membaca data instruksi dari *internal* memori ke ALU.

Sebagai suatu sistem kontrol mikrokontroler ATMega32 bila dibandingkan dengan mikroprosesor memiliki kemampuan dan segi ekonomis yang bisa diandalkan karena dalam mikrokontroler sudah terdapat RAM dan ROM sedangkan mikroprosesor didalamnya tidak terdapat keduanya. Secara umum konfigurasi yang dimiliki mikrokontroler ATMega32 adalah sebagai berikut :

- Sebuah CPU 8 bit dengan menggunakan teknologi dari Atmel.
- Memiliki memori baca-tulis sebesar 2 kbyte SRAM.
- Jalur dua arah (*bidirectional*) yang digunakan sebagai saluran masukan atau keluaran yang dikontrol oleh *register* DDR.
- Sebuah komunikasi serial USART yang dapat diprogram.
- Sebuah *master/slave* serial SPI yang dapat diprogram.
- Sebuah *Two Wire Serial Interface*.
- Dua buah *timer/counter* 8 bit dan sebuah *timer/counter* 16 bit.
- *Watcdog Timer* yang dapat diprogram.
- *Analog to Digital Converter (ADC)* 10-bit dan *Analog comparator* di dalam chip.
- Osilator internal dan rangkaian pewaktu.
- Flash PEROM yang besarnya 32 kbyte untuk memori program
- Kemampuan melaksanakan operasi perkalian, pembagian, dan operasi Boolean.
- Mampu beroperasi sampai 16 MHz.

ATMega32 mikrokontroler mempunyai kompabalitas instruksi dan konfigurasi pin dengan mikrokontroler keluarga AVR seri ATMega16 dan ATMega8535.

Masing-masing kaki dalam mikrokontroler ATMega32 mempunyai fungsi tersendiri. Dengan mengetahui fungsi masing-masing kaki mikrokontroler ATMega32, perancangan aplikasi mikrokontroler ATMega32 akan lebih mudah. ATMega32 mempunyai 40 pin, susunan masing-masing pin ditunjukan dalam Gambar 2.15.



Gambar 2.15. Konfigurasi Pin ATMega32

Sumber: ATMEL, 2007a

Fungsi kaki-kaki ATMega32adalah :

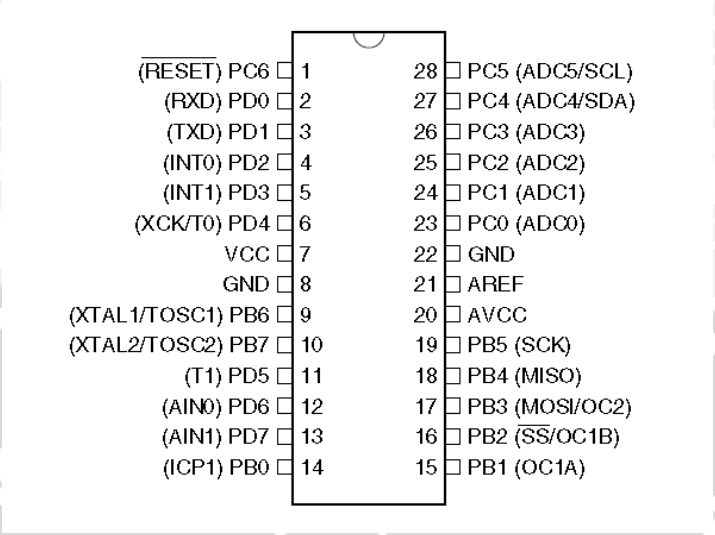
- *Port* A (Pin A0..7), merupakan saluran masukan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* A adalah sebagai ADC (*input* ADC *channel* 0..7).
- *Port* B (Pin B0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus Port B diantaranya adalah : Port B0 (T0 (*timer/counter0* eksternal *counter input*) & XCK (USART eksternal *clock input/output*), Port B1 (T1 (*timer/counter* eksternel *counter input*)), Port B2 (AIN0 (*Analog comparator positive input*) & INT2 (*External interrupt 2 input*), Port B3 (AIN1 (*Analog comparator negative input*) & (OC0 (Timer/counter0 *output compare match output*)), Port B4 (SS (SPI *slave select input*)), Port B5 (MOSI (SPI bus master output/slave input)), Port B6 (MISO (SPI bus master input/slave output)), Port B7 (SCK (SPI bus serial *clock*)).

- *Port* C (Pin C0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari Port C diantaranya adalah : Port C0 (SCL (*Two-Wire serial bus clock line*)), Port C1 (SDA (*Two-Wire* serial bus data input/output line)), Port C6 (TOSC1 (*Timer Oscilator* pin1)), Port C7 (TOSC2 (*Timer oscillator* pin2)).
- *Port* D (Pin D0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari Port D diantaranya adalah : Port D0 (RXD (USART *input pin*)), Port D1 (TXD (USART *output pin*)), Port D2 (INT0 (Eksternal *interupt 0 input*)), Port D3 (INT1 (Eksternal *interupt 1 input*)), Port D4 (OC1B (*Timer/counter* 1 *output compare* B *match output*)), Port D5 (OC1A (*Timer/counter* 1 *output compare* A *match output*)), Port D6 (ICP (*Timer/counter input capture pin*), Port D7 (OC2 (*timer/counter 2 compare match output*).
- Pin 9 RESET, merupakan saluran dua masukan untuk mereset mikrokontroler dengan cara memberi masukan logika rendah.
- Pin 10 VCC, merupakan saluran masukan untuk catu daya positif sebesar 5 volt DC.
- Pin 11 GND, merupakan Ground dari seluruh rangkaian.
- Pin 12 dan 13 (XTAL2 dan XTAL1), merupakan saluran untuk mengatur pewaktuan sistem. Untuk pewaktuan dapat menggunakan pewaktuan internal maupun eksternal.
- Pin 32 AREF, merupakan pin referensi analog untuk masukan ADC.
- Pin 33 GND, merupakan *ground* dari ADC.
- Pin 30 AVCC, merupakan catu untuk perangkat ADC.

2.8. Mikrokontroler Atmel AVR ATMega8

Mikrokontroler ATMega8 secara garis besar tidak memiliki perbedaan yang jauh dengan mikrokontroler ATMega32. Dibandingkan ATMega32, mikrokontroler ATMega8 memiliki kapasitas yang lebih kecil yaitu 8 kbyte Flash PEROM (*Flash Programmable and Erasable Read Only Memory*), 512 byte SRAM, 23 pin I/O yang mana tiap pin tersebut dapat diprogram secara paralel dan tersendiri, mempunyai dua buah *timer/counter* 8 bit dan satu buah *timer/counter* 16 bit,mempunyai 8 bit 10 *channel* ADC, mempunyai *watchdog timer*.

Masing-masing kaki dalam mikrokontroler ATMega8 mempunyai fungsi tersendiri. Mikrokontroler ATMega8 mempunyai 28 pin, susunan masing-masing pin ditunjukan dalam Gambar 2.16.



Gambar 2.16. Konfigurasi Pin ATMega8

Sumber: ATMEL, 2007b

Fungsi kaki-kaki ATMega8 adalah :

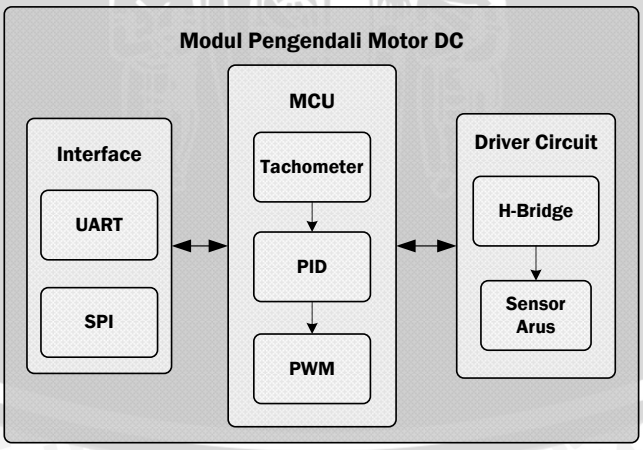
- *Port* B (Pin B0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus Port B diantaranya adalah : Port B0 (ICP (*Timer/counter input capture pin*)), Port B1 (OC1A (*Timer/counter* 1 *output compare* A *match output*)), Port B2 (OC1B (*Timer/counter* 1 *output compare* B *match output*)) & (SS (SPI *slave select input*)), Port B3 (OC2 (*timer/counter 2 compare match output*) & (MOSI (SPI bus master output/slave input)), Port B4 (MISO (SPI bus master input/slave output)), (SS (SPI *slave select input*)) , Port B5 (SCK (SPI *bus serial clock*)), Port B6 (XTAL1), Port B7 (XTAL2).
- *Port* C (Pin C0..6), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus *Port* C adalah sebagai ADC (*input* ADC *channel* 0..5). Fungsi khusus lain dari Port C diantaranya adalah : Port C5 (SCL (*Two-Wire serial bus clock line*)), Port C4 (SDA (*Two-Wire serial bus data input/output line*)), Port C6 (RESET pin).
- *Port* D (Pin D0..7), merupakan saluran masukkan/keluaran dua arah dan juga mempunyai fungsi khusus. Fungsi khusus dari Port D diantaranya adalah : Port D0 (RXD (USART *input pin*)), Port D1 (TXD (USART *output pin*)), Port D2

(INT0 (Eksternal *interupt 0 input*)), Port D3 (INT1 (Eksternal *interupt 1 input*)), Port D4 (XCK (USART eksternal *clock input/output*)), Port D5 (T1 (*timer/counter* eksternal *counter input*)), Port D6 (AIN0 (*Analog comparator negative input*), Port D7 (AIN1 (*Analog comparator negative input*).

- Pin 7 VCC, merupakan saluran masukan untuk catu daya positif sebesar 5 volt DC.
- Pin 8 GND, merupakan *ground* dari seluruh rangkaian.
- Pin 21 AREF, merupakan pin analog referensi untuk masukan ADC.
- Pin 22 GND, merupakan ground dari ADC.
- Pin 20 AVCC, merupakan catu daya untuk perangkat ADC.

2.9. Modul Pengendali Motor DC

Modul pengendali motor DC merupakan hasil riset yang dilakukan oleh tim robotika Teknik Elektro Universitas Brawijaya pada tahun 2008. Modul ini telah diuji dan digunakan oleh tim robotika Teknik Elektro Universitas Brawijaya pada robot N4HL yang mewakili Universitas Brawijaya dalam kompetisi Kontes Robot Cerdas Indonesia 2009. Secara garis besar, fungsi modul pengendali motor ini adalah untuk mengendalikan arah dan kecepatan putaran motor DC *brushed* sesuai instruksi kendali dari mikrokontroler pengguna. Blok sistem dari modul pengendali motor DC *brushed* ditunjukkan pada Gambar 2.17.



Gambar 2.17. Blok Sistem Modul Pengendali Motor DC *Brushed*

Bagian utama dari modul kontrol kendali motor DC terdiri atas sebuah mikrokontroler dan *driver* H-Bridge. Mikrokontroler bertugas mengendalikan motor DC menggunakan teknik kendali PID dengan umpan balik dari sensor *rotary encoder* untuk mendeteksi arah dan kecepatan motor DC yang sebenarnya.

Untuk konfigurasi driver motor DC digunakan rangkaian H-Bridge yang sanggup menangani beban sampai 25 W. Modul ini sanggup bekerja dengan tegangan maksimal 50 volt dan tegangan 5 volt untuk acuan logika tinggi. Kapasitas arus maksimum yang dapat dilewatkan pada modul ini sebesar 4 ampere.

Untuk menggunakan modul ini, terdapat dua tipe antarmuka yang digunakan yaitu komunikasi serial asinkron (UART - *Universal Asynchronous Receiver Transmitter*) dan komunikasi serial sinkron (SPI - *Serial Peripheral Interface*). Komunikasi serial UART dengan *baud rate* 9600 bps dan format transmisi 8-N-1 digunakan untuk konfigurasi parameter kontrol pada modul melalui komputer. Setiap instruksi untuk konfigurasi modul disusun dalam paket data selebar 5 *byte*. Semua konfigurasi yang telah ditentukan akan tetap tersimpan dalam EEPROM pada modul sehingga tidak perlu melakukan konfigurasi ulang untuk setiap penggunaan.

Sistem komunikasi serial SPI digunakan untuk memberikan instruksi kecepatan dan arah putaran yang diinginkan. Kemampuan komunikasi SPI pada modul dibatasi hingga frekuensi transmisi sebesar 4 MHz. Data yang ditransmisikan pada komunikasi SPI menggunakan data selebar 1 *byte* dengan format tipe data karakter bertanda (*signed character*). Data yang dikirimkan berupa prosentase kecepatan terhadap kecepatan nilai maksimum yaitu antara -100 hingga 100. Untuk memberikan instruksi berhenti dengan melakukan rem (*braking*) sehingga poros motor DC terkunci, dilakukan dengan mengirimkan data bernilai 120.

BAB III
METODOLOGI PENELITIAN

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat di bab pendahuluan maka diperlukan metode untuk menyelesaikan masalah tersebut.

3.1. Penentuan Spesifikasi Alat

Sebelum melakukan perencanaan dan perealisasian alat, maka ditentukan spesifikasi alat yang akan dibuat. Adapun spesifikasi alat yang akan direalisasikan sebagai berikut:

a. Sensor ultrasonik disusun melingkar dibagian depan agar mobile robot dapat mengukur jarak pada bagian depan, samping kanan dan samping kiri.
b. Mikrokontroler Atmel ATMega8 yang bekerja sebagai pengolah data dan antarmuka sensor ultrasonik.
c. Sensor putaran (*rotary encoder*) diletakan pada kedua buah roda untuk mengukur arah dan kecepatan putaran roda robot.
d. Mobile robot dapat mengikuti dan menjaga jarak terhadap salah satu sisi dinding.
e. Posisi mobile robot diinginkan agar tepat berada diantara kedua sisi dinding.
f. Mobile robot dapat berputar ditempat ketika melalui lintasan tertutup.
g. Mikrokontroler Atmel ATMega32 sebagai pengendali utama mobile robot.
h. *Mobile robot* berbahan dasar mika *acrylic* dan menggunakan dua buah motor DC sebagai aktuator.

3.2. Perancangan dan Perealisasian Alat

3.2.1. Perancangan Perangkat Keras dan Realisasi Tiap Blok

a. Pembuatan blok diagram lengkap sistem
b. Penentuan dan perhitungan komponen yang akan digunakan
c. Desain papan cetak (PCB) menggunakan *software* Eagle Layout Editor
d. Merakit perangkat keras masing-masing blok

3.2.2. Perancangan dan Penyusunan Perangkat Lunak

Setelah kita mengetahui seperti apa perangkat keras yang dirancang, maka kita membutuhkan perangkat lunak untuk mengendalikan dan mengatur kerja dari alat ini.

Desain dan parameter yang telah dirancang kemudian diterapkan kedalam mikrokontroler ATMega32 dan ATMega8 dengan menggunakan bahasa C dan *compiler* CodeVision AVR.

3.3. Pengujian Alat

Untuk memastikan bahwa sistem ini berjalan sesuai yang direncanakan maka perlu dilakukan pengujian alat meliputi perangkat keras *(hardware)* yang dilakukan baik per blok rangkaian maupun keseluruhan sistem.

3.3.1. Pengujian Perangkat Keras

Pengujian perangkat keras dilakukan dengan tujuan untuk menyesuaikan nilai tegangan dan arus yang diijinkan bekerja dalam komponen berdasarkan data sekunder komponen yang diambil dari buku data komponen elektronika maupun dari datasheet.

3.3.2. Pengujian Keseluruhan Sistem

Pengujian sistem secara keseluruhan dilakukan dengan tujuan untuk mengetahui unjuk kerja alat setelah perangkat keras dan perangkat lunak diintegrasikan bersama. Setelah didapatkan hasil dari pengujian dilakukan pengambilan kesimpulan. Jika hasil yang diperoleh telah sesuai dengan spesifikasi yang direncanakan maka alat tersebut telah memenuhi harapan dan memerlukan pengembangan untuk penyempurnaannya.

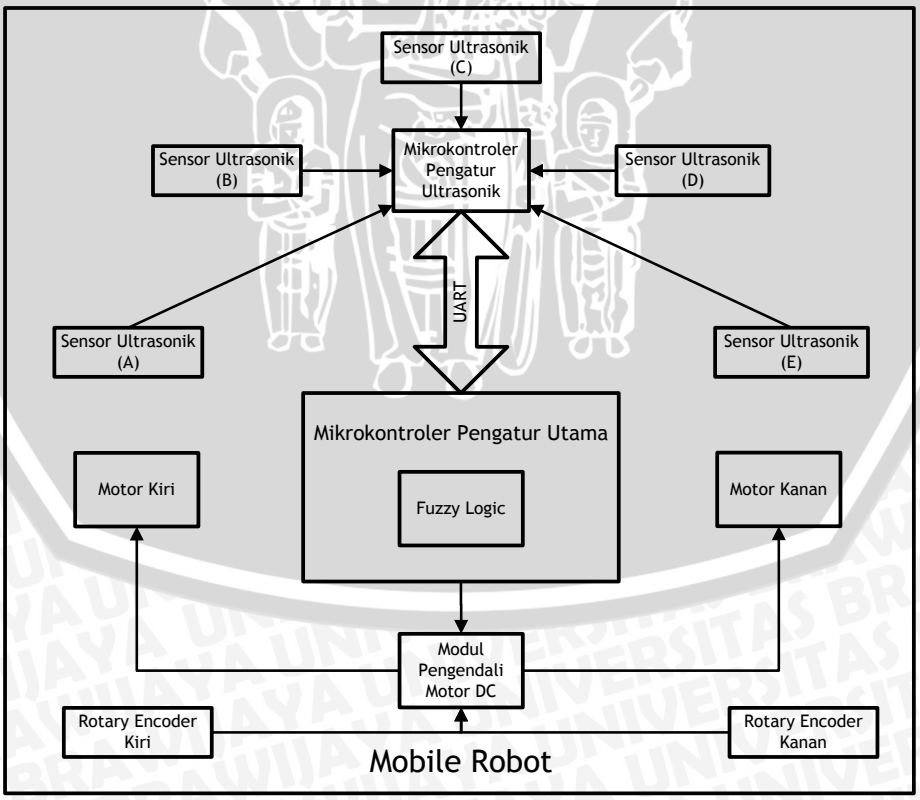
BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Perancangan alat ini dilakukan secara bertahap dalam bentuk blok sehingga akan memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. Perancangan ini terdiri dari:

- Perancangan sistem.
- Perancangan perangkat keras (perancangan sensor ultrasonik, sensor *rotary encoder*, modul pengendali motor DC, rangkaian mikrokontroler pengatur utama, rangkaian mikrokontroler pengatur ultrasonik dan bentuk mekanik robot).
- Perancangan sistem logika *fuzzy*.
- Perancangan perangkat lunak (perancangan sistem pengaturan sensor ultrasonik dan perancangan algoritma perangkat lunak).

4.1. Perancangan Sistem

Blok diagram sistem yang dirancang ditunjukkan dalam Gambar 4.1.



Gambar 4.1. Blok Diagram Sistem

Robot memiliki empat buah roda, yakni dua buah roda penggerak utama sisi kiri dan dua buah roda penggerak sisi kanan. Pada setiap sisi, roda dihubungkan dengan motor DC dan sensor rotari. Penggunaan empat buah roda membuat posisi robot tidak terguling meski dimensi badan robot lebih luas dari bagian roda yang menginjak lantai lintasan.

Pergerakan dari robot ini dibantu dengan lima buah sensor ultrasonik yang berfungsi sebagai sensor pengukur jarak antara mobile robot dengan lingkungannya yaitu dinding lintasan. Kelima buah sensor ultrasonik diletakkan di bagian depan, dua di samping kiri dan dua di samping kanan robot. Mikrokontroler ATMega8 yang bekerja mengatur sensor ultrasonik, mengolah data sensor dan mengirim data jarak ke mikrokontroler pengatur utama yaitu ATMega32 menggunakan protokol serial UART (*Universal Asynchronous Receiver Transmitter).*

Mikrokontroler pengatur utama yang berfungsi sebagai *fuzzy logic controller* menerima transmisi paket data jarak dari mikrokontroler pengatur ultrasonik. Paket data yang diterima kemudian diperiksa susunan datanya untuk mengetahui validitas data. Data yang telah sesuai akan disimpan dan digunakan utnuk perhitungan dalam algoritma fuzzy. Jarak yang disimpan dalam mikrokontroler dimasukkan sesuai dengan jangkauan yang ada pada masing-masing fungsi keanggotaan, kemudian dilakukan proses fuzzifikasi sehingga dihasilkan derajat keanggotaan masing-masing nilai masukan.

Proses selanjutnya adalah rule evaluation, dimana nilai masukan yang telah mempunyai derajat keanggotaan dimasukkan dalam kaidah atur yang direncanakan. Pada proses ini metode penalaran kontroler logika *fuzzy* yang digunakan adalah metode MAX-MIN. Setelah itu dilakukan proses defuzzifikasi, yaitu dengan menggunakan metode COA *(Center of Area).* Nilai yang dihasilkan dari proses defuzzifikasi berupa besarnya persentase yang menunjukan kecepatan putaran dan arah putar motor. Hasil defuzzifikasi kemudian dikirimkan ke modul pengendali motor DC.

Pada modul pengendali motor, data yang diterima dikonversi menjadi sinyal kendali arah dan dilakukan pengaturan kecepatan pada kedua motor penggerak robot. Pengukuran kecepatan dan arah putar robot dilakukan dengan menggunakan sensor rotary encoder yang terpasang pada roda penggerak kanan dan roda penggerak kiri. Motor kanan dan motor kiri berfungsi sebagai kemudi yang menggerakkan robot ketika harus maju, berputar, belok kanan atau belok kiri.

4.2. Perancangan Perangkat Keras

4.2.1. Sensor Ultrasonik

Sensor ultrasonik berfungsi sebagai sensor pengukur jarak antara mobile robot dengan semua benda yang berada pada lingkungannya seperti dinding lintasan. Keluaran dari sensor ultrasonik ini digunakan sebagai masukan dari proses *fuzzy*. Sistem pengukuran jarak dengan gelombang ultrasonik memanfaatkan frekuensi 40 kHz dengan amplitudo yang dapat diatur, frekuensi ini dipancarkan oleh rangkaian *transmitter* kemudian dipantulkan oleh objek yang menghalanginya. Sinyal pantulan inilah yang nantinya ditangkap oleh rangkaian penerima untuk diukur seberapa lama sinyal pantulan itu kembali dan dikonversi menjadi data jarak. Sensor ini mempunyai 3 buah kaki atau pin yakni pin VCC, pin GND dan pin SIG yang digunakan untuk mengirimkan data. Komunikasi mikrokontroler dengan Ping))) ditunjukkan dalam Gambar 4.2.

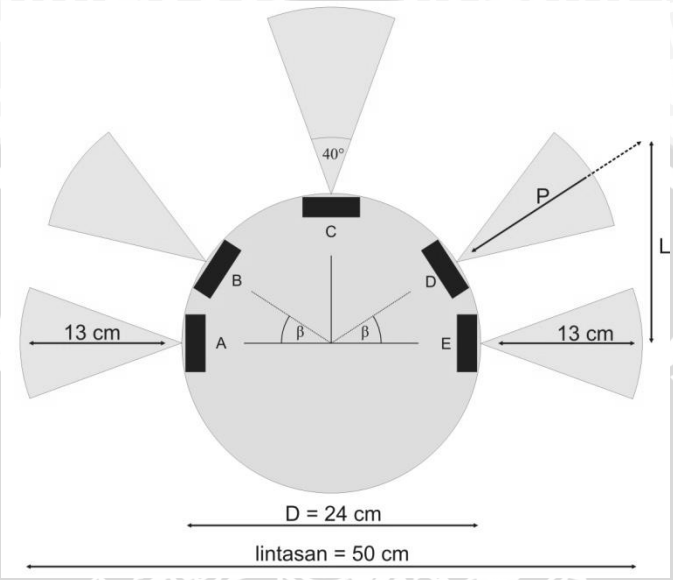
Gambar 4.2. Komunikasi Mikrokontroler dengan Ping)))

Sumber: www.parallax.com

Dalam perancangan ini, digunakan lima buah sensor ultrasonik yang diletakkan di bagian samping kiri tengah (A), samping kiri (B), bagian depan (C), samping kanan (D) dan samping kanan tengah (E) robot. Sensor A, B, C, D dan E dihubungkan dengan mikrokontroler ATMega8 sebagai mikrokontroler pengatur ultrasonik pada Pin B.0, Pin B.1, Pin B.2, Pin B.3 dan Pin B.4.

Sensor ultrasonik disusun berbentuk setengah lingkaran pada badan robot bagian depan. Sensor ultrasonik A,C dan E disusun tegak lurus dimana sensor A diletakan pada sisi kiri, sensor C diletakan pada bagian depan dan sensor E diletakan pada sisi kanan. Sedangkan sensor B dan D diletakan dengan sudut sebesar β terhadap sensor

pasanganya yaitu sensor A untuk sensor B dan sensor E untuk sensor D. Fungsi utama dari sensor samping depan yaitu sensor ultrasonik B dan D adalah untuk mendeteksi keberadaan dinding yang diikuti selanjutnya, karena itu sensor tersebut digunakan untuk melakukan prediksi (*forecasting*) adanya dinding atau tikungan. Tata letak sensor pada robot ditunjukan pada Gambar 4.3.



Gambar 4.3. Tata letak sensor ultrasonik pada robot

Dengan menentukan jangkauan total (L) sensor D dan E maka dapat dihitung besarnya sudut (β) antara sensor ultrasonik D dan E. Besar jangkauan total (L) yang digunakan adalah 2/3 dari diameter robot yaitu sebesar 16 cm. Nilai 2/3 dipilih karena panjang L diukur sejajar dari titik tengah robot, sehingga didapatkan jangkauan sensor D dan B lebih jauh 4 cm terhadap bagian depan badan robot. Besarnya sudut antara sensor samping (β) dapat dihitung menggunakan persamaan trigonometri pada Persamaan (4.1)

$$\tan\beta = \frac{L}{\left(\frac{D}{2}\right)+13\ cm} \tag{4.1}$$

$$\beta = \tan^{-1}\left(\frac{16}{25}\right)$$

$$\beta = 32{,}6°$$

Selanjutnya dapat dihitung jarak sensor samping depan (P) yaitu sensor B dan D ketika terdapat dinding yang diikuti. Besarnya jarak sensor samping depan (P) ditunjukan pada Persamaan (4.2)

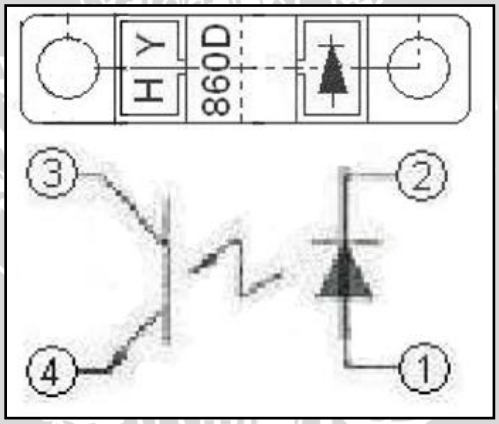
$$\cos\beta = \frac{\left(\frac{D}{2}\right)+13\ cm}{P+\left(\frac{D}{2}\right)} \quad (4.2)$$

$$P = \frac{25}{\cos 32{,}6°} - \left(\frac{24}{2}\right)$$

$$P = 17{,}68\ cm$$

4.2.2. Sensor Rotary Encoder

Sensor putaran (*rotary encoder*) yang digunakan adalah sensor putaran berbasiskan optik menggunakan komponen *optoswitch* dan piringan berpola. Sensor *rotary encoder* diletakan pada roda kanan dan roda kiri robot. Komponen *optoswitch* yang digunakan adalah tipe HY860D produksi Hongkong Hingyip Electronic Co.Ltd. Komponen ini merupakan kombinasi LED inframerah sebagai pemancar cahaya dan *phototransistor* NPN sebagai penerima cahaya. Gambar 4.4 menunjukkan konfigurasi pin dari *optoswitch*.



Gambar 4.4. Konfigurasi pin komponen optoswitch

Sumber: Hingyip, 2009: 1

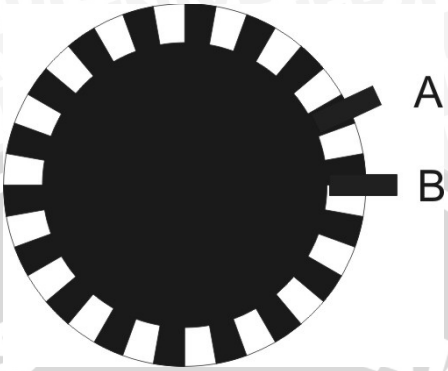
Resolusi sensor yang digunakan pada perancangan ini sebesar 70 pulsa untuk setiap putaran penuh. Dengan mengetahui diameter roda robot (D) sebesar 8 cm maka dapat dihitung besar jarak terukur (L) untuk setiap satu sinyal pulsa pada sensor *rotary* yang dirancang menggunakan persamaan berikut.

$$L = \frac{\pi \times D}{n} \quad (4.3)$$

$$L = \frac{\pi \times 8\ cm}{70}$$

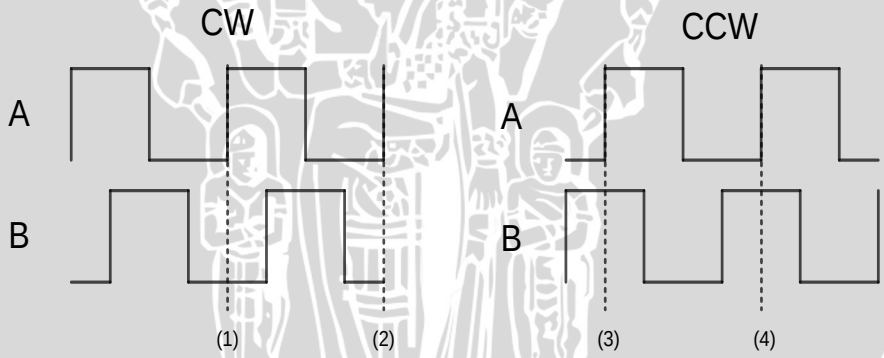
$$L = 0{,}359\ cm$$

Sensor ini dirancang agar memiliki keluaran sinyal *quadrature* dengan menyusun sedemikian rupa posisi dua buah komponen *optoswitch*. Sifat sinyal *incremental* didapatkan dengan membuat pola berlubang pada sisi luar piringan berpola seperti ditunjukan pada Gambar 4.5.



Gambar 4.5. Letak optocoupler pada piringan berpola

Sinyal *quadrature* yang dikeluarkan *rotary encoder* digunakan untuk menentukan arah putaran roda pada robot. Dengan menganalisa bentuk sinyal saat putaran roda searah jarum jam dan berlawanan jarum jam yang ditunjukan pada Gambar 4.6 maka selanjutnya dapat dibuat rangkaian antarmuka sensor *rotary encoder*.



Gambar 4.6. Sinyal quadrature CW dan CCW

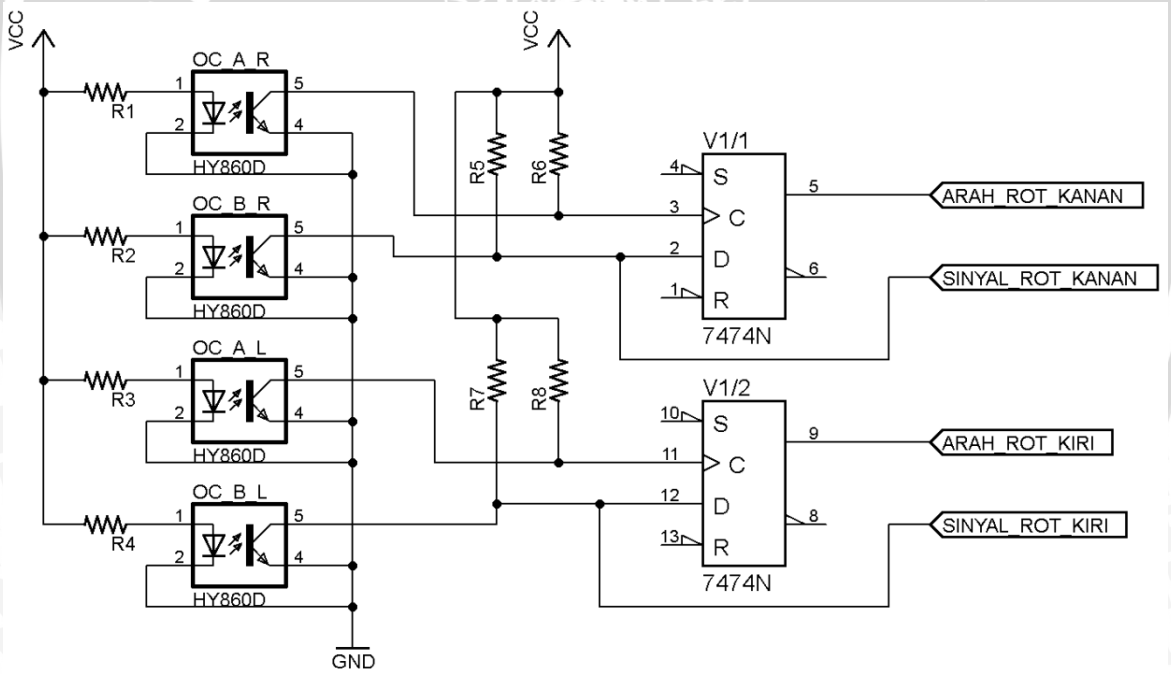
Pada Gambar 4.6, perbedaan sinyal antara putaran searah jarum jam (CW) dan putaran berlawanan arah jarum jam (CCW) dapat dilihat pada garis(1) dan garis(3). Pada putaran searah jarum jam (CW), sinyal A mendahului sinyal B sehingga pada awal tepi naik dari sinyal A, logika sinyal B adalah rendah seperti yang ditunjukan pada garis(1) dan garis(2). Sebaliknya untuk putaran berlawanan arah jarum jam (CCW), sinyal A tertinggal dari sinyal B sehingga pada awal tepi naik dari sinyal A, logika sinyal B adalah tinggi seperti yang ditunjukan pada garis(3) dan garis(4). Perbedaan tingkat logika sinyal B untuk setiap tepi naik sinyal A dapat digunakan untuk menentukan arah putaran. Rangkaian penahan (*latch*) dengan menggunakan *flip-flop* D

tepi naik dapat digunakan untuk mendeteksi perbedaan sinyal tersebut. Tabel kebenaran dari rangkaian *latch* yang akan digunakan ditunjukan pada Tabel 4.1

Tabel 4.1 Tabel kebenaran rangkaian latch

Sinyal B (D)	Sinyal A (CLK)	Arah (Q)
0	0→1	CW (0)
0	1→0	Arah Sebelumnya
1	0→1	CCW (1)
1	1→0	Arah Sebelumnya

Berdasarkan tabel kebenaran rangkaian *latch* yang dibuat, selanjutnya dirancang rangkaian sensor *rotary encoder* yang terdiri atas dua bagian utama yaitu rangkaian *flip-flop* D sebagai penahan (*latch*) sinyal keluaran *rotary encoder* untuk mendeteksai arah putar dan rangkaian *optoswitch* untuk mendeteksi pola berlubang pada piringan berpola. Rangkaian sensor *rotary encoder* ditunjukan pada Gambar 4.7.



Gambar 4.7. Rangkaian Sensor *Rotary Encoder*

Pada rangkaian yang dirancang, digunakan tegangan sumber (V_{CC}) sebesar 5 V. Berdasarkan datasheet *optoswitch* HY860D, arus bias maju (I_F) yang diperlukan untuk menyalakan LED inframerah adalah sebesar 20 mA dengan tegangan jatuh (V_F) maksimal pada LED sebesar 1,6 V. Sehingga nilai minimum resistansi R_1 dapat ditentukan dengan menggunakan persamaan berikut.

$$R_{1(\min)} = \frac{V_{CC} - V_F}{I_F} \quad (4.4)$$
$$R_{1(\min)} = \frac{5\ V - 1{,}6\ V}{20\ mA}$$
$$R_{1(\min)} = 170\ \Omega$$

Merujuk pada nilai resistansi komponen resistor yang dijual dipasaran maka dipilih nilai R_1 yang digunakan pada rangkaian sebesar 180Ω, maka besar V_F dapat diketahui dengan menggunakan persamaan berikut.

$$V_F = V_{CC} - R_1 . I_F \quad (4.5)$$
$$V_F = 5\ V - 180.20.10^{-3}\ V$$
$$V_F = 1{,}4\ V$$

Tegangan jatuh (V_F) yang dihasilkan sebesar 1,4 V. Nilai tersebut berada dibawah nilai maksimal yaitu 1,6 V, sehingga didapatkan nilai komponen R_1, R_2, R_3 dan R_4 sebesar 180 Ω.

Pada bagian penerima optoswitch bekerja pada dua kondisi yakni kondisi saat cahaya LED terhalang dan tidak terhalang. Berdasarkan *datasheet optoswitch* HY860D, pada saat I_F bernilai 20 mA dan I_C bernilai 0,2 mA maka tegangan saturasi *collector-emitter* (VCE_{SAT}) maksimal bernilai 0,4 V. Sehingga nilai resistansi minimum R_5 dapat ditentukan dengan menggunakan persamaan sebagai berikut.

$$R_{5(\min)} = \frac{V_{CC} - VCE_{SAT\ (maks)}}{I_C} \quad (4.6)$$
$$R_{5(\min)} = \frac{5\ V - 0{,}4\ V}{0{,}2\ mA}$$
$$R_{5(\min)} = 23\ k\Omega$$

Dengan memperhatikan besaran nilai resistansi komponen yang dijual dipasaran maka digunakan nilai R_5 dalam rangkaian sebesar 24kΩ. Sehingga nilai VCE_{SAT} pada rangkaian dapat dihitung dengan menggunakan persamaan berikut.

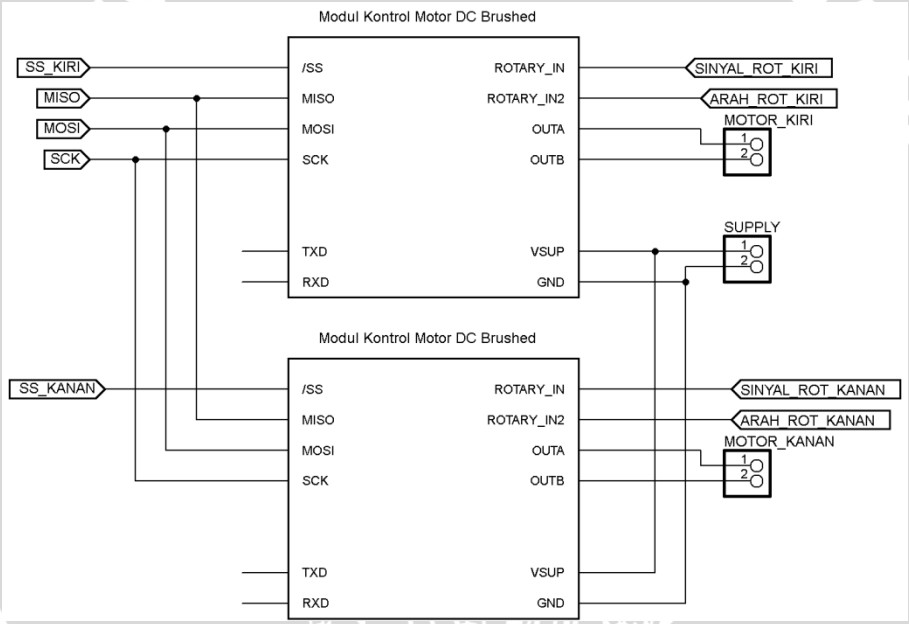
$$VCE_{SAT} = V_{CC} - R_5 . I_C \quad (4.7)$$
$$VCE_{SAT} = 5\ V - 24.10^3 . 0{,}2 . 10^{-3}\ V$$
$$VCE_{SAT} = 0{,}2\ V$$

Berdasarkan perhitungan, VCE_{SAT} yang dihasilkan sebesar 0,2 V. Nilai tersebut berada dibawah nilai maksimum, sehingga nilai resistansi sebesar 24kΩ dapat digunakan pada komponen R_5, R_6, R_7 dan R_8. Nilai tegangan VCE_{SAT} tersebut juga

memenuhi sebagai tingkat logika rendah pada komponen *flip-flop* 74LS74 dimana besar batas tegangan logika rendah (V_{IL}) maksimum yaitu 0,8 V. Sedangkan untuk batas tegangan logika tinggi (V_{IH}) minimum sebesar 2 V, dapat dipenuhi saat *phototransistor* berada dalam kondisi *cut-off*.

4.2.3. Modul Pengendali Motor DC

Modul pengendali motor DC ini digunakan untuk mengendalikan arah dan putaran motor DC *brushed* yang menjadi penggerak pada robot. Modul ini dihubungkan dengan rangkaian mikrokontroler pengatur utama menggunakan komunikasi serial SPI dengan frekuensi 4 MHz. Keluaran sensor *rotary encoder* juga dihubungkan pada modul untuk umpan balik sistem kendali. Antarmuka modul pengendali motor DC pada robot ditunjukan pada Gambar 4.8.



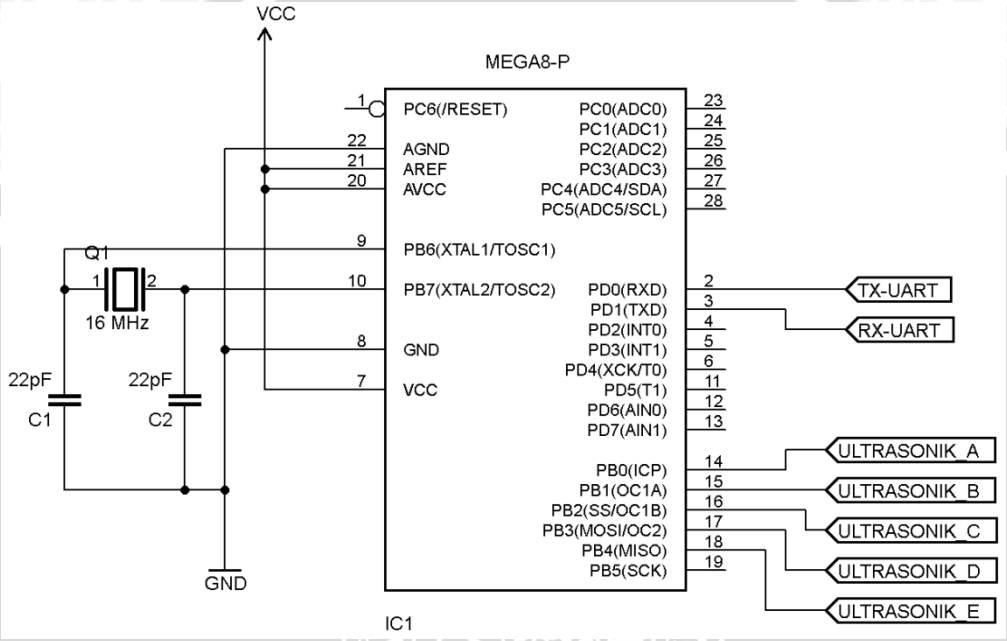
Gambar 4.8. Antarmuka Modul Pengendali Motor DC

Sebelum digunakan, dilakukan konfigurasi pada setiap modul terlebih dahulu. Konfigurasi cukup dilakukan sekali, data akan tetap tersimpan pada memori modul pengendali hingga nilainya diubah. Konfigurasi yang dilakukan adalah menentukan resolusi sensor *rotary encoder* dan kecepatan putaran motor maksimum. Resolusi sensor *rotary encoder* disesuaikan dengan jenis *rotary encoder* yang digunakan yaitu 70 ppr (*pulse per revolution*). Kecepatan putaran motor maksimum ditentukan sebesar 120 rpm (*revolution per minute*) berdasarkan label spesifikasi yang tertulis pada motor DC yang digunakan untuk sumber 12 V.

Proses konfigurasi dilakukan dengan menghubungkan antarmuka serial UART (pin TXD dan RXD) pada modul dengan port serial pada komputer. Untuk penentuan parameter kontrol, digunakan mode *auto-tuning* yang tersedia pada modul.

4.2.4. Rangkaian Mikrokontroler Pengatur Ultrasonik

Pada robot ini juga digunakan mikrokontroler ATMega8 sebagai pengolah dalam proses pengaturan sensor ultrasonik. Konfigurasi kaki I/O dari mikrokontroler ATMega8 ditunjukkan dalam Gambar 4.9.



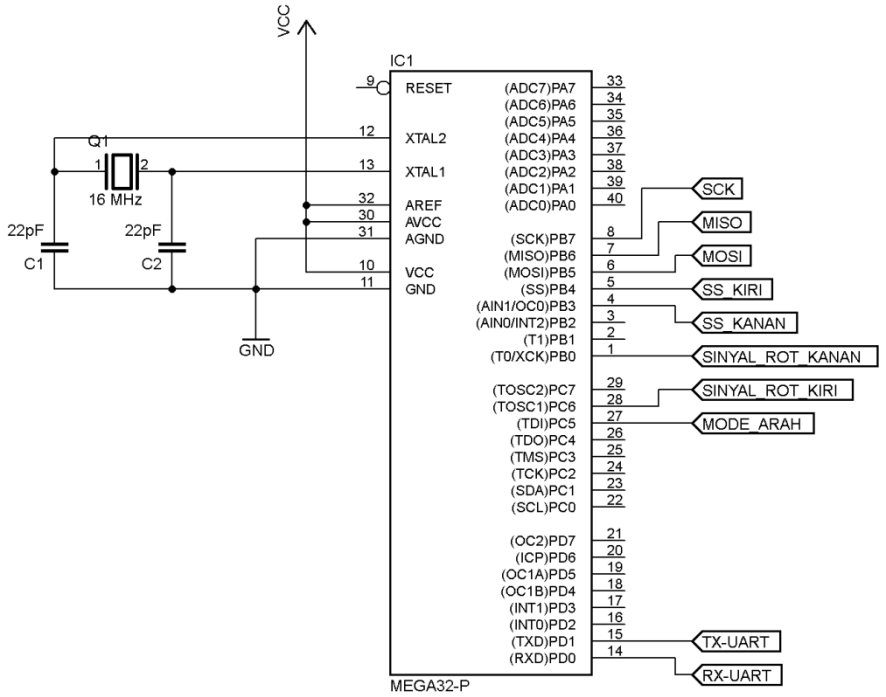
Gambar 4.9. Minimum Sistem Mikrokontroler ATMega 8

Mikrokontroler ATMega8 mempunyai 3 port, dengan 21 jalur I/O yang dapat diprogram menjadi masukan atau keluaran, pada perancangan ini pin 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.4 = digunakan untuk antarmuka sensor ultrasonik (E)
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

4.2.5. Rangkaian Mikrokontroler Pengatur Utama

Pada robot ini digunakan mikrokontroler ATMega32 sebagai pengolah utama dalam melakukan proses logika *fuzzy*. Konfigurasi kaki I/O dari mikrokontroler ATMega 32 ditunjukkan dalam Gambar 4.10.



Gambar 4.10. Minimum Sistem Mikrokontroler ATMega 32

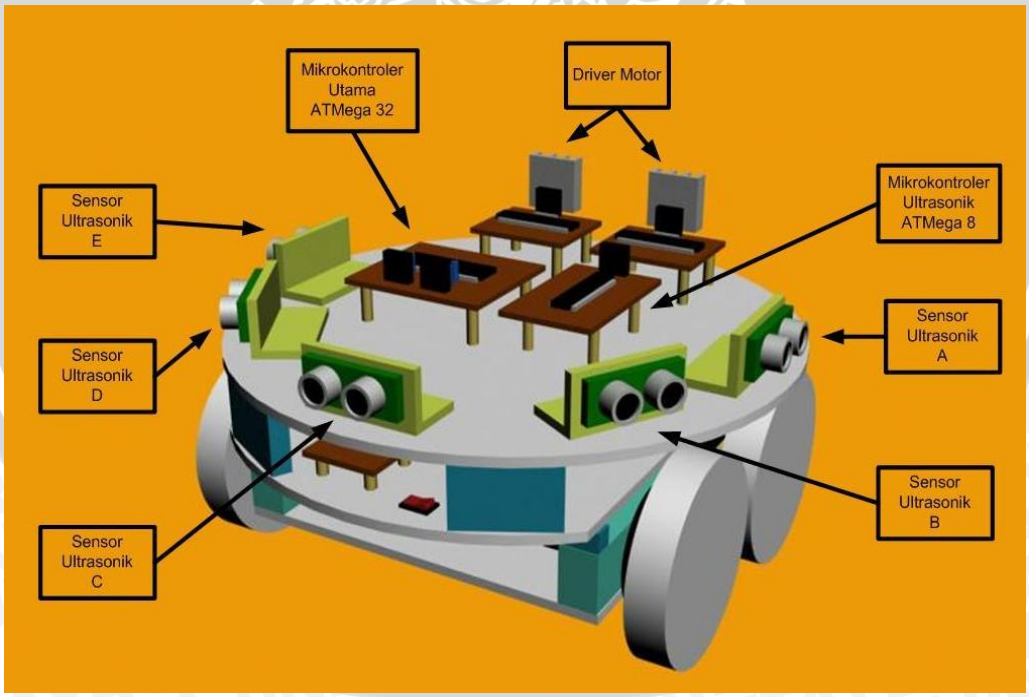
Mikrokontroler ATMega32 mempunyai 4 port, 32 jalur yang dapat diprogram menjadi masukan atau keluaran, pada perancangan ini pin-pin yang digunakan adalah:

Pin D.0 = Rx untuk transmisi UART
Pin D.1 = Tx untuk transmisi UART
Pin B.0 = digunakan sebagai masukan sensor putaran roda kanan
Pin C.6 = digunakan sebagai masukan sensor putaran roda kiri
Pin C.5 = digunakan sebagai masukan mode program
Pin B.3 = digunakan sebagai pemilih modul pengendali kanan (*Slave Select-SPI*)
Pin B.4 = digunakan sebagai pemilih modul pengendali kiri (*Slave Select-SPI*)
Pin B.5 = digunakan sebagai antarmuka dengan modul pengendali (*MOSI-SPI*)
Pin B.6 = digunakan sebagai antarmuka dengan modul pengendali (*MISO-SPI*)
Pin B.7 = digunakan sebagai antarmuka dengan modul pengendali (*SCK-SPI*)
XTAL1 = digunakan sebagai masukan dari rangkaian osilator kristal
XTAL2 = digunakan sebagai masukan dari rangkaian osilator kristal

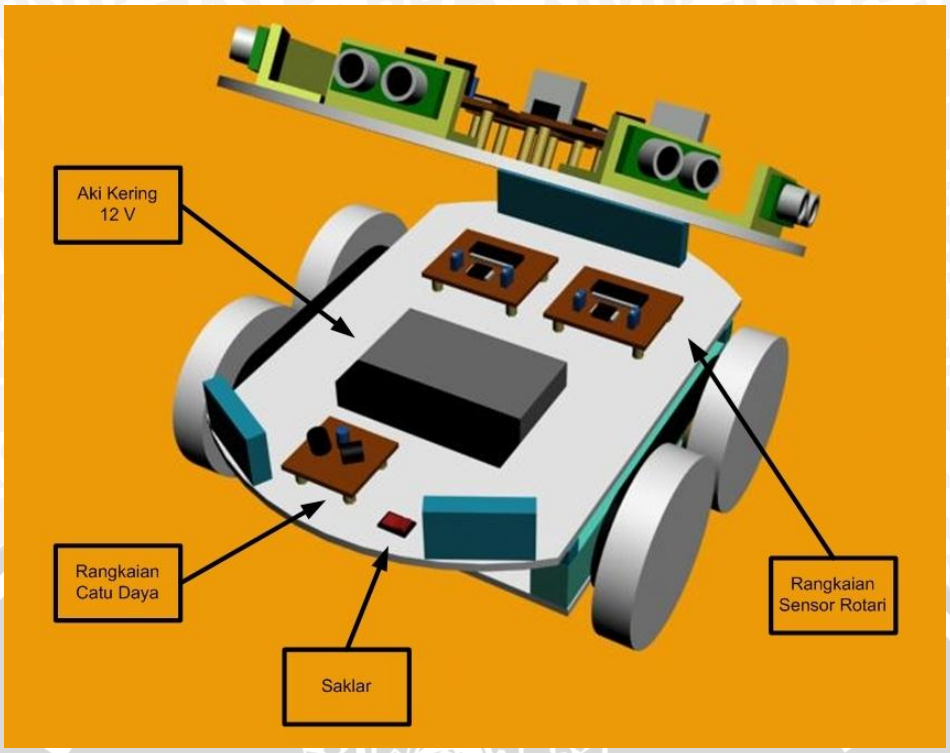
4.2.6. Bentuk Mekanik Robot

Badan robot terbuat dari bahan mika *acrylic* dan berbentuk lingkaran dengan diameter 24 cm. Badan robot dibuat dengan bentuk lingkaran dengan alasan agar robot tidak tertahan pada posisinya ketika membentur lintasan. Susunan badan robot terdiri atas 3 lapisan yaitu lapisan atas sebagai tempat rangkaian mikrokontroler, modul pengendali motor dan sensor ultrasonik, kemudian lapisan tengah untuk tempat baterai sebagai sumber daya dan rangkaian sensor *rotary encoder*. Pada bagian bawah terdapat motor DC *brushed* sebagai penggerak dan sensor *rotary encoder* pada setiap sisi roda.

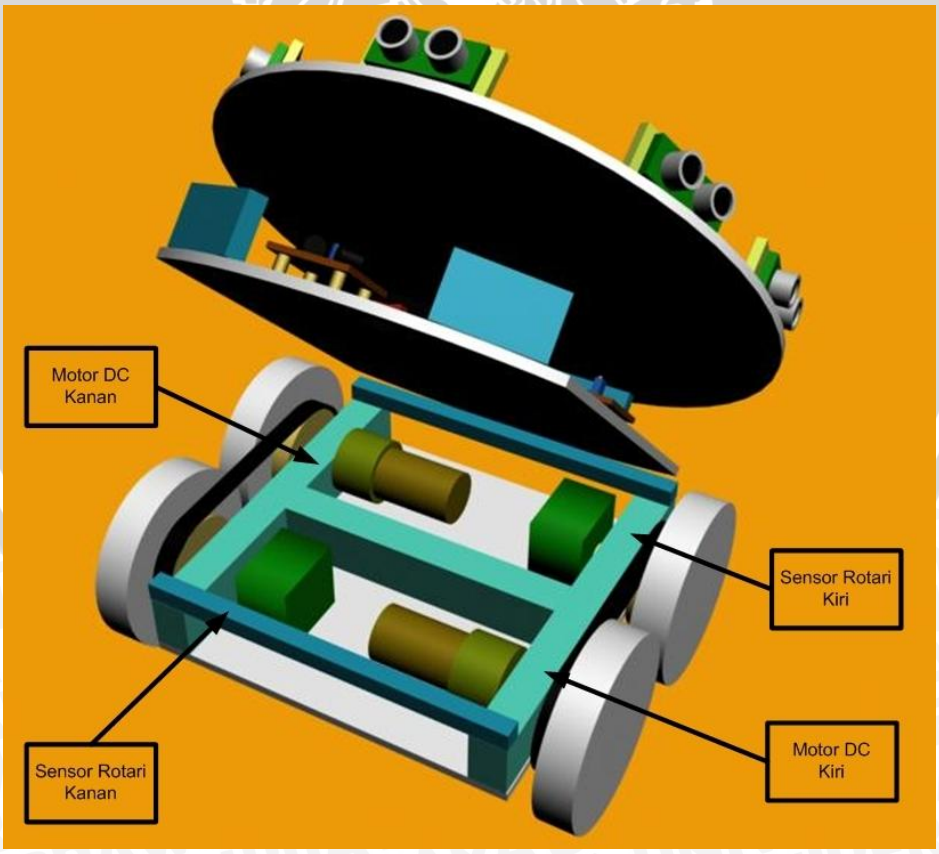
Rangka utama berbahan dasar alumunium diletakan pada lapisan bawah robot. Motor DC dan poros roda yang terhubung pada sensor *rotary encoder* juga terletak pada rangka utama. Roda robot berjumlah 4 buah dimana untuk setiap sisi terdapat dua buah roda. Sistem transmisi penggerak antara roda pada sisi yang sama menggunakan sistem belt. Bentuk mekanik robot dalam tiga persepektif berbeda ditunjukan pada Gambar 4.11 , Gambar 4.12 dan Gambar 4.13.



Gambar 4.11. Mekanik Robot tampak depan



Gambar 4.12. Mekanik Robot bagian tengah

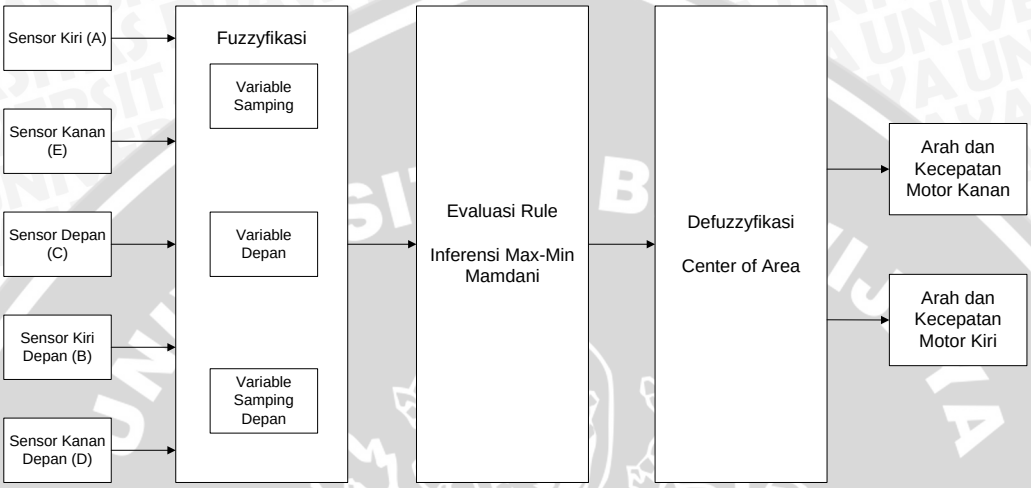


Gambar 4.13. Mekanik Robot bagian dalam

4.3. Perancangan Sistem Logika Fuzzy

4.3.1. Variabel Masukan dan Keluaran

Variabel masukan untuk sistem logika *fuzzy* ada lima yaitu jarak dari pembacaan setiap sensor ultrasonik yang berjumlah lima buah, sedangkan variabel keluaran berupa arah putar dan kecepatan putar motor kanan dan motor kiri. Blok diagram dari sistem fuzzy yang dirancang ditunjukan pada Gambar 4.14.

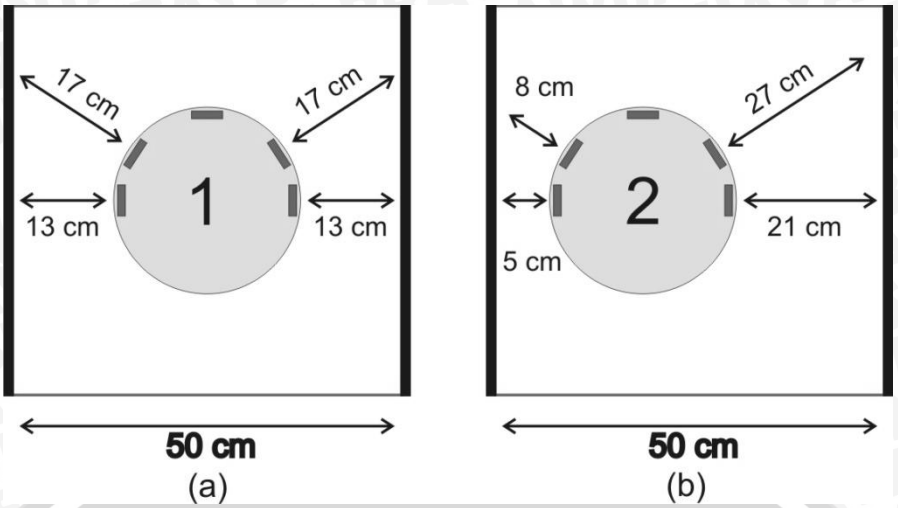


Gambar 4.14. Blok diagram sistem fuzzy

4.3.1.1. Fungsi Keanggotaan Masukan pada Sisi Samping

Fungsi keanggotaan masukan pada sisi dinding yang diikuti terdiri atas tiga label yaitu dekat, normal dan jauh. Fungsi keanggotaan pada sisi samping digunakan sesuai sisi dinding yang diikuti. Pada sisi dinding yang diikuti terdapat dua buah sensor ultrasonik yang menjadi masukan, sehingga setiap sensor memiliki tiga label fungsi keanggotaan. Pada sisi samping, masukan dari fungsi keanggotaan adalah empat buah sensor yang terletak di samping kanan sejumlah dua buah (D dan E) dan disamping kiri sejumlah dua buah (A dan B).

Penentuan nilai batas pada masing-masing fungsi keanggotaan dilakukan dengan memperhatikan jarak terbaca dari sensor ultrasonik pada saat posisi robot berada ditengah-tengah lintasan dan posisi robot berada terlalu dekat dengan salah satu sisi lintasan. Ilustrasi posisi robot dalam lintasan ditunjukan pada Gambar 4.15.



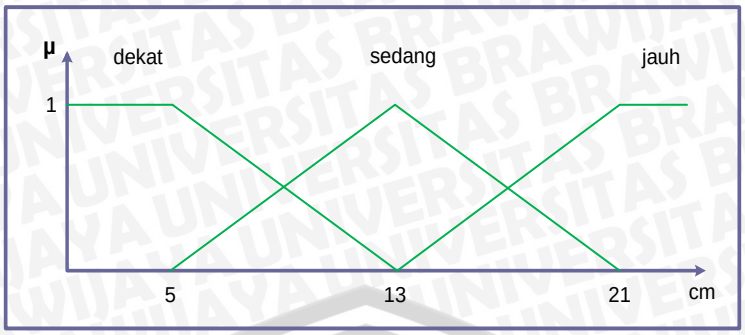
Gambar 4.15. Ilustrasi Posisi Robot Dalam Lintasan

Saat robot berada pada posisi (1), dimana robot berada pada tengah lintasan, jarak terbaca pada sensor ultrasonik samping yaitu sensor ultrasonik A dan E adalah 13 cm. Nilai tersebut menjadi batas derajat keanggotaan 1 pada label sedang yang berbentuk segitiga untuk sisi samping. Sedangkan jarak terbaca pada sensor samping depan sebesar 17 cm. Nilai tersebut menjadi batas derajat keanggotaan 1 pada label sedang yang berbentuk segitiga untuk sisi samping depan.

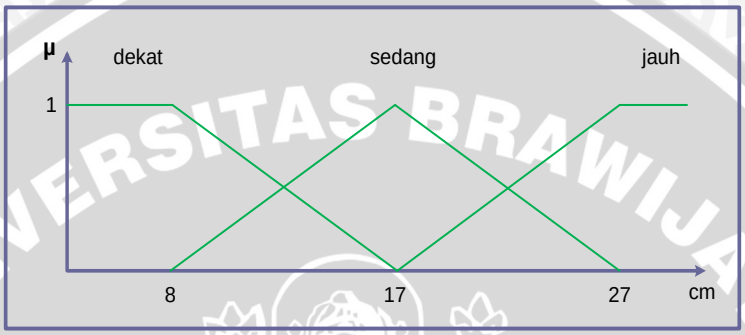
Pada posisi (2), posisi robot dekat dengan sisi dinding sebelah kiri. Pada sensor ultrasonik A terbaca jarak 5 cm, sensor ultrasonik B terbaca jarak 8 cm, sensor ultrasonik D terbaca jarak 27 cm dan sensor ultrasonik E terbaca jarak 21 cm.

Kemudian jarak 5 cm menjadi batas derajat keanggotaan 1 pada label dekat untuk sisi samping. Jarak 8 cm menjadi batas derajat keanggotaan 1 pada label dekat untuk sisi samping depan. Jarak 21 cm menjadi batas derajat keanggotaan 1 pada label jauh untuk sisi samping. Jarak 27 cm menjadi batas derajat keanggotaan 1 pada label jauh untuk sisi samping depan.

Ketika sisi dinding yang diikuti adalah sisi kanan maka sensor ultrasonik yang menjadi masukan adalah sensor ultrasonik samping kanan (E) dan sensor ultrasonik samping kanan depan (D). Sebaliknya untuk sisi dinding kiri, sensor ultrasonik yang menjadi masukan adalah sensor ultrasonik samping kiri (A) dan sensor ultrasonik samping kiri depan (B). Gambar 4.16 menunjukan fungsi keanggotaan masukan untuk sensor samping. Sedangkan untuk fungsi keanggotaan masukan sensor samping depan ditunjukan pada Gambar 4.17.



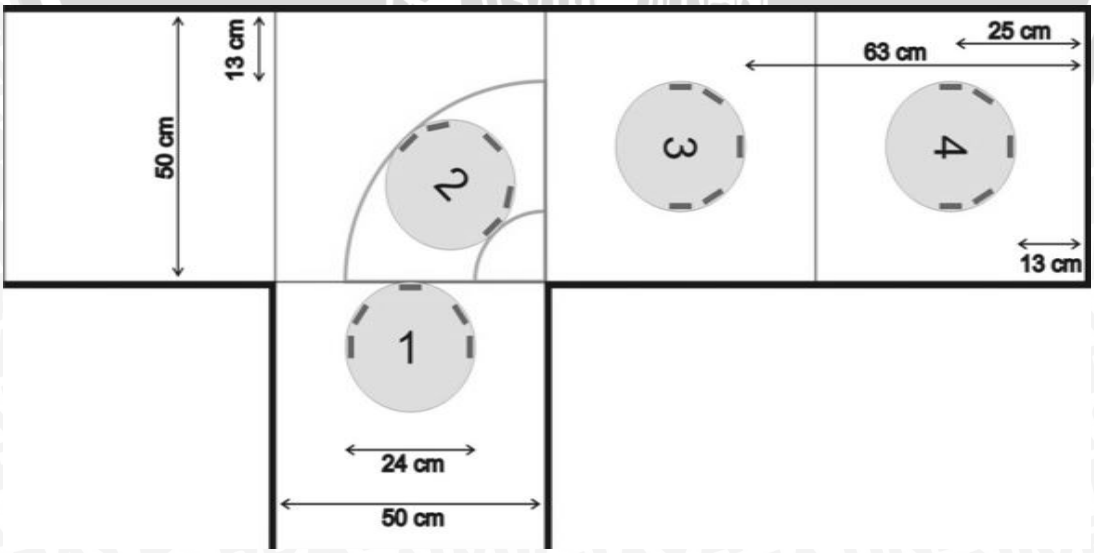
Gambar 4.16. Fungsi Keanggotaan Masukan Sensor Samping



Gambar 4.17. Fungsi Keanggotaan Masukan Sensor Samping Depan

4.3.1.2. Fungsi Keanggotaan Masukan pada Sisi Depan

Fungsi keanggotaan untuk sensor depan tidak dapat disamakan dengan fungsi keanggotaan sensor pada sisi samping walaupun saling mempengaruhi. Sensor depan digunakan untuk mendeteksi keberadaan dinding pada bagian depan robot. Ilustrasi untuk menentukan batas-batas fungsi keanggotaan pada label dekat, sedang dan jauh ditunjukan pada Gambar 4.18.



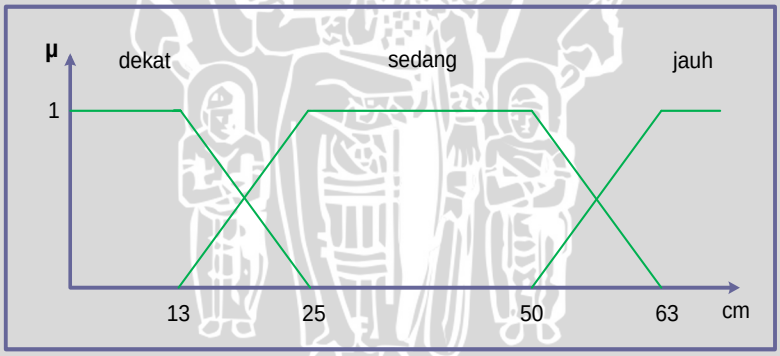
Gambar 4.18. Ilustrasi Posisi Robot Ketika Mengikuti Dinding Kanan

Ketika robot berada di posisi 1, jarak robot terhadap dinding pada sisi depan semakin dekat karena robot akan memasuki *grid* pada lintasan yang berbentuk pertigaan atau belokan. Pada saat itu sensor depan membaca jarak satu *grid* selebar 50 cm sama dengan lebar lintasan. Jarak ini dijadikan batas fungsi keanggotaan pada label sedang dengan derajat keanggotaan 1. Label sedang yang berbentuk trapesium dengan derajat keanggotaan 1 yang sebesar 25 cm yaitu ketika bagian depan robot tepat berada pada separuh *grid* lintasan. Label sedang yang digunakan berupa trapesium dengan dua batas sebagai derajat keanggotaan bernilai 1 yaitu 50 cm dan 25 cm.

Ketika robot berada di posisi 2, robot berbelok dengan radius 13 cm terhadap sisi dinding yang diikuti. Label dekat dengan derajat keanggotaan 1 ditentukan sebesar 13 cm berdasarkan posisi 2 dan posisi 4 dimana robot berada ditengah *grid* lintasan. Batas label dekat dengan derajat keanggotaan 0 sebesar 25 cm.

Pada posisi 3 dimana robot berada ditengah-tengah *grid* lintasan sebelum menuju posisi 4, jarak antara robot dengan dinding depan sejauh 63 cm. Jarak tersebut menjadi batas fungsi keanggotaan label jauh dengan derajat keanggotaan 1.

Dari data-data tersebut kemudian dirancang fungsi keanggotaan untuk masukan sensor depan yang ditunjukkan dalam Gambar 4.19.



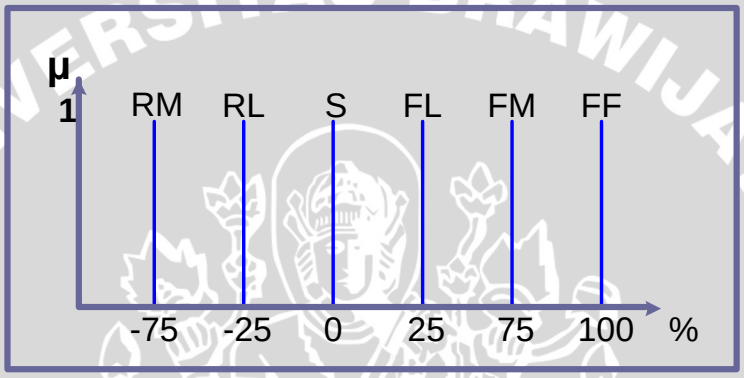
Gambar 4.19. Fungsi Keanggotaan Masukan Sensor Depan

4.3.1.3. Fungsi Keanggotaan Keluaran Kecepatan Motor Kanan dan Kiri

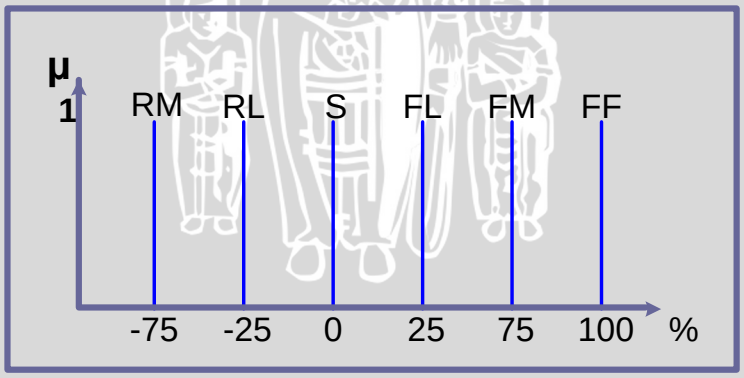
Fungsi keanggotaan keluaran motor DC merupakan representasi arah putar dan rasio kecepatan putaran terhadap putaran maksimum motor yang berbentuk *singleton*. Pemilihan bentuk fungsi keanggotaan berupa *singleton* dengan alasan untuk menghemat memori dan mempercepat eksekusi program. Fungsi keanggotaan pada motor kiri dan kanan dibuat sama karena tipe motor DC yang digunakan sama dan adanya kendali umpan-balik pada modul pengendali motor DC.

Fungsi keanggotaan keluaran motor DC kanan terdiri dari 6 label yaitu RM (*reverse medium*) sebesar -75%, RL (*reverse low*) sebesar -25%, S (*Stop*) sebesar 0%, FL (*forward low*) sebesar 25%, FM (*forward medium*) sebesar 75% dan FF (*forward full*) sebesar 100%. Fungsi keanggotaan motor DC sisi kanan ditunjukan pada Gambar 4.20.

Untuk fungsi keanggotaan keluaran motor DC kiri berbentuk singleton yang terdiri dari 6 label yaitu RM (*reverse medium*) sebesar -75%, RL (*reverse low*) sebesar -25%, S (*Stop*) sebesar 0%, FL (*forward low*) sebesar 25%, FM (*forward medium*) sebesar 75% dan FF (*forward full*) sebesar 100%. Fungsi keanggotaan motor DC pada sisi kiri ditunjukan pada Gambar 4.21.



Gambar 4.20. Fungsi Keanggotaan Output Motor DC Kanan



Gambar 4.21. Fungsi Keanggotaan Output motor DC Kiri

4.3.2. Kaidah Atur (Rule) Logika Fuzzy

Kaidah atur (*rule*) dalam logika fuzzy didasarkan pada model perilaku bergerak mengikuti dinding dan disusun dalam bentuk Jika-Maka (*If-Then*). Setelah masukan *crisp* diubah menjadi masukan *fuzzy*, selanjutnya diolah sesuai dengan kaidah aturnya. Metode penalaran *fuzzy* yang dipergunakan adalah metode MAX-MIN. Terdapat dua buah kelompok kaidah atur yang digunakan yaitu kaidah atur utama mengikuti dinding

kanan dan kaidah atur utama mengikuti dinding kiri. Kaidah atur utama untuk mengikuti dinding kanan dapat dilihat dalam Tabel 4.2. dan kaidah atur utama untuk mengikuti dinding kiri dapat dilihat dalam Tabel 4.3.

Tabel 4.2 Kaidah atur utama mengikuti dinding kanan

	C	dekat		Sedang		jauh	
D	**E**	kiri	kanan	kiri	kanan	kiri	kanan
dekat	dekat	RM	FF	ST	FM	ST	FF
sedang		RM	FF	FL	FM	FM	FF
jauh		RL	FF	FM	ST	FF	ST
dekat	sedang	RM	FM	RL	FM	RL	FF
sedang		RM	FM	FM	FM	FF	FF
jauh		RL	FM	FM	RL	FF	ST
dekat	Jauh	RM	FM	RM	FM	RM	FF
sedang		RM	FM	FM	FL	FF	FM
jauh		FF	RL	FM	RL	FF	RL

Tabel 4.3 Kaidah atur utama mengikuti dinding kiri

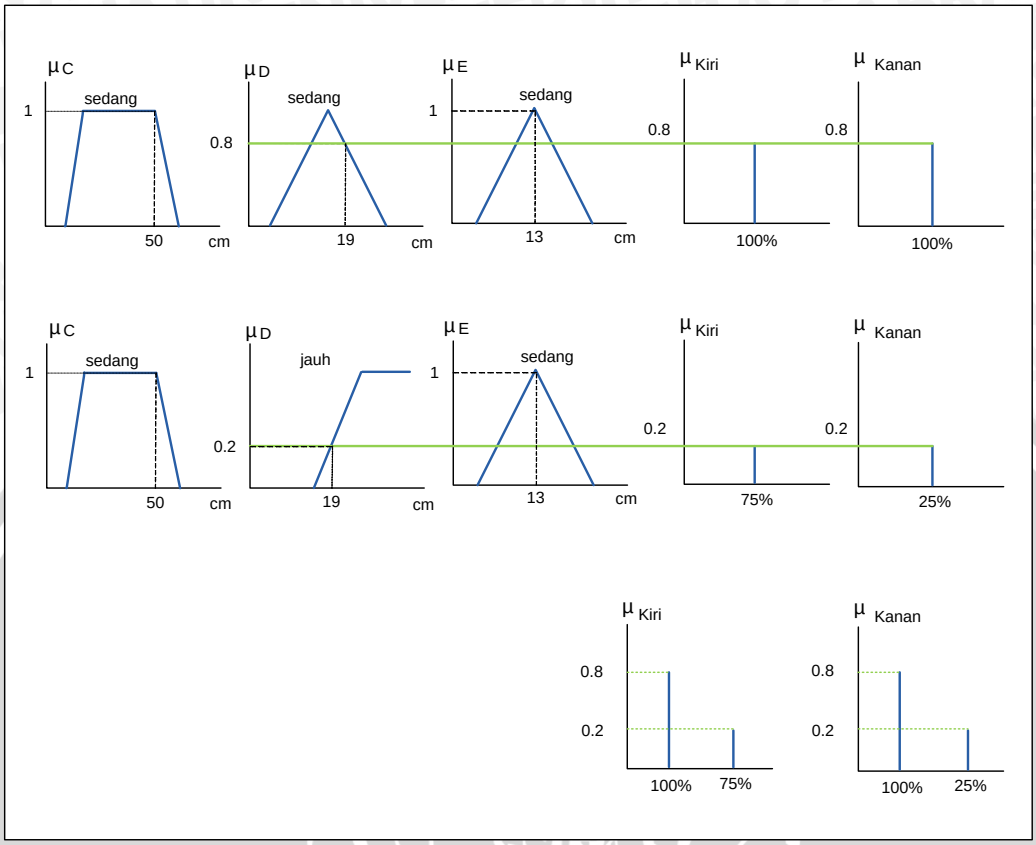
	C	dekat		sedang		jauh	
B	**A**	kiri	kanan	kiri	kanan	kiri	kanan
dekat	dekat	FF	RM	FM	ST	FF	ST
sedang		FF	RM	FM	FL	FF	FM
jauh		FF	RL	ST	FM	ST	FF
dekat	sedang	FM	RM	FM	RL	FF	RL
sedang		FM	RM	FM	FM	FF	FF
jauh		FM	RL	RL	FM	ST	FF
dekat	Jauh	FM	RM	FM	RM	FF	RM
Sedang		FM	RM	FL	FM	FM	FF
Jauh		RL	FF	RL	FM	RL	FF

Penggunaan kaidah atur utama bergantung pada sisi dinding yang akan diikuti. Jika robot mengikuti dinding kanan maka digunakan kaidah atur utama mengikuti dinding kanan, sedangkan kaidah atur utama mengikuti dinding kiri digunakan jika robot mengikuti sisi dinding sebelah kiri.

4.3.3. Metode Inferensi Max-Min

Visualisasi proses pengolahan masukan ketika sensor ultrasonik depan (C) membaca jarak 50 cm, sensor ultrasonik kanan depan (D) membaca jarak 19 cm dan sensor ultrasonik kanan (E) membaca jarak 13 cm dengan metode Max-Min

menggunakan kaidah atur mengikuti dinding kanan yang telah ditetapkan ditunjukkan dalam Gambar 4.22.



Gambar 4.22. Ilustrasi Metode Max-Min

4.3.4. Defuzzyfikasi

Defuzifikasi adalah proses untuk mengubah keluaran *fuzzy* menjadi keluaran *crisp*. Hasil defuzifikasi inilah yang digunakan untuk mengatur besarnya rasio kecepatan pada masing-masing motor DC. Metode defuzzifikasi yang digunakan adalah COA *(Center of Area).*

Motor kanan:

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i}$$

$$= \frac{0.8 \times FF + 0.2 \times FL}{0.8 + 0.2}$$

$$= \frac{0.8 \times 100 + 0.2 \times 25}{0.8 + 0.2}$$

$$= 85\%$$

Motor kiri:

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i}$$

$$= \frac{0.8 \times FF + 0.2 \times FM}{0.8 + 0.2}$$

$$= \frac{0.8 \times 100 + 0.2 \times 75}{1}$$

$$= 95\%$$

Jadi ketika robot mengikuti dinding kanan, sensor ultrasonik depan (C) membaca jarak 50 cm, sensor ultrasonik kanan depan (D) membaca jarak 19 cm dan sensor ultrasonik kanan (E) membaca jarak 13 cm maka besarnya kecepatan putaran motor kanan adalah 85% dan kecepatan putaran motor kiri adalah 95%.

4.4. Perancangan Perangkat Lunak

4.4.1. Pengaturan Sensor Ultrasonik

Penggunaan sensor ultrasonik dalam jumlah banyak memungkinkan terjadinya gangguan berupa *crosstalk* karena pantulan gelombang dari sensor satu ke sensor lainya (Borenstein, 1995:1). Untuk menghindari terjadinya *crosstalk*, digunakan metode penjadwalan (*scheduling*) dengan cara mengaktifkan sensor ultrasonik secara bergantian dengan selang waktu yang ditentukan. Selain itu, metode aktif bergantian pada sensor ultrasonik dilakukan dengan urutan aktivasi sensor yang melompat-lompat sehingga tidak ada dua atau lebih sensor yang berdekatan aktif dalam waktu yang dekat. Tabel 4.4 menunjukan urutan aktivasi sensor ultrasonik dan selang waktu aktif maksimum sensor ultrasonik.

Tabel 4.4 Urutan aktivasi sensor ultrasonik pada robot

Sensor	Aktif (ms)
A	12
D	12
B	12
E	12
C	12
Total	60

Dimensi arena sebesar 3 m x 3 m dan dimensi lintasan yang memiliki lebar 50 cm menyebabkan pantulan gelombang ultrasonik pada dinding lintasan sehingga menimbulkan kesalahan pengukuran jarak pada sensor ultrasonik untuk jarak lebih dari 2 meter. Lama waktu aktif maksimum pada sensor sebesar 12 ms sebanding dengan jarak terbaca sebesar 206 cm. Ketika jarak terukur lebih pendek dari 206 cm sehingga lama waktu aktif pada sensor ultrasonik kurang dari 12 ms maka sensor selanjutnya baru akan diaktifkan setelah melewati sisa waktu aktif yang ada pada sensor sebelumnya.

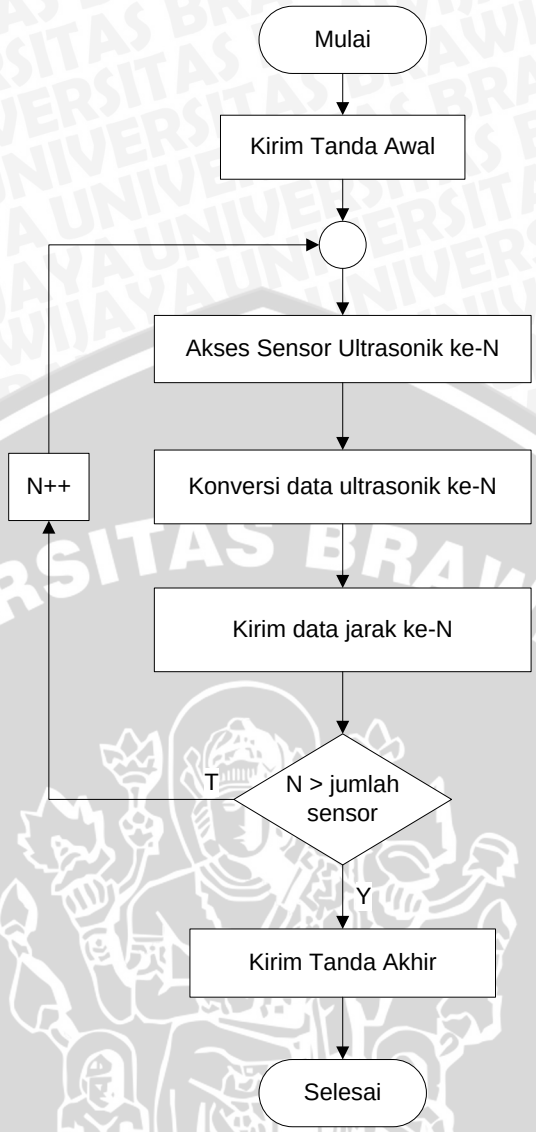
Perangkat *timer 1* pada mikrokontroler digunakan untuk melakukan perhitungan lamanya sinyal logika tinggi dari sensor ultrasonik yang merupakan lama waktu tempuh gelombang ultrasonik. Penggunann timer 1 dikarenakan resolusi timer 1 adalah 16 bit sehingga hasil pengukuran yang dilakukan lebih presisi. Perangkat *timer 2* digunakan sebagai penanda siklus lama waktu aktif maksimum untuk setiap sensor ultrasonik.

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 ultrasonik menuju mikrokontroler pengatur utama. Data yang dikirim disusun dalam bentuk sebuah paket data selebar 7 *byte* dimana *byte* pertama dan *byte* terakhir berfungsi sebagai penanda paket data yang dikirim. Urutan data yang terdapat diantara *byte* penanda sesuai dengan urutan aktivasi sensor. Ilustrasi paket data ditunjukan pada Tabel 4.5.

Tabel 4.5. Ilustrasi susunan paket data jarak

data_0	**data_1**	**data_2**	**data_3**	**data_4**	**data_5**	**data_6**
0xEE	jarak A	jarak D	jarak B	jarak E	jarak C	0xFF

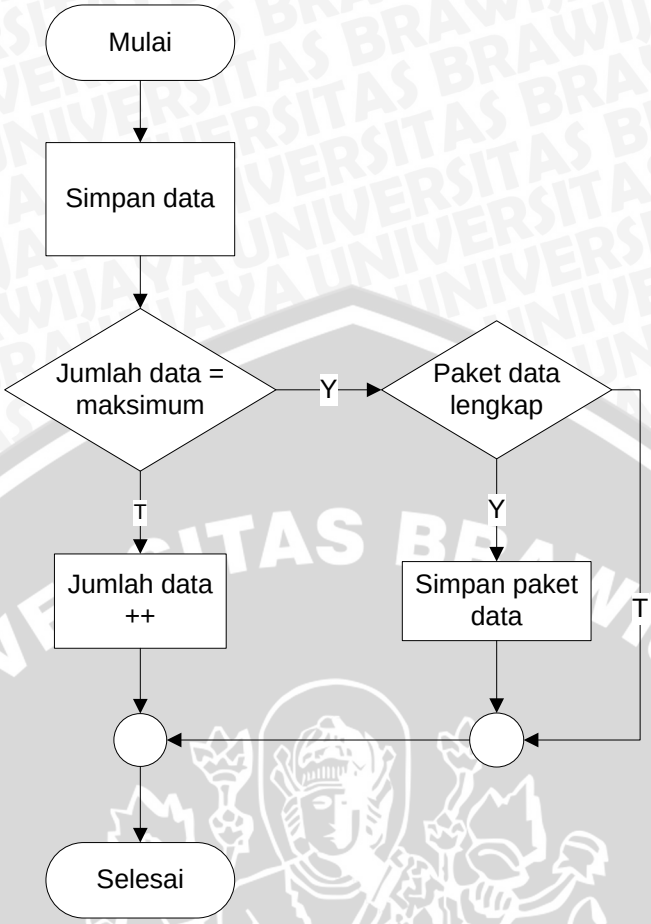
Diagram alir program pengaturan sensor ultrasonik ditunjukan pada Gambar 4.23.



Gambar 4.23. Diagram Alir Program Pengaturan Sensor Ultrasonik

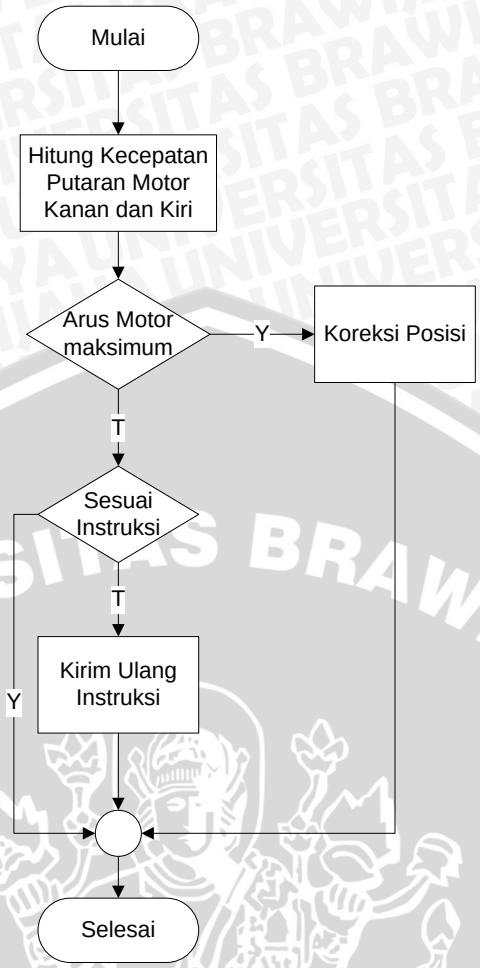
4.4.2. Perancangan Algoritma Utama

Mikrokontroler pengatur utama merupakan pusat pengolah data dan proses logika *fuzzy*. Paket data ultrasonik yang diterima mikrokontroler berupa jarak terukur dalam satuan centimeter. Proses penerimaan data ultrasonik menggunakan sistem interupsi komunikasi serial, sehingga mikrokontroler pengatur utama dapat melakukan proses lain selama data ultrasonik yang diperlukan untuk proses logika *fuzzy* belum lengkap. Diagram alir program interupsi komunikasi serial ditunjukan pada Gambar 4.24.



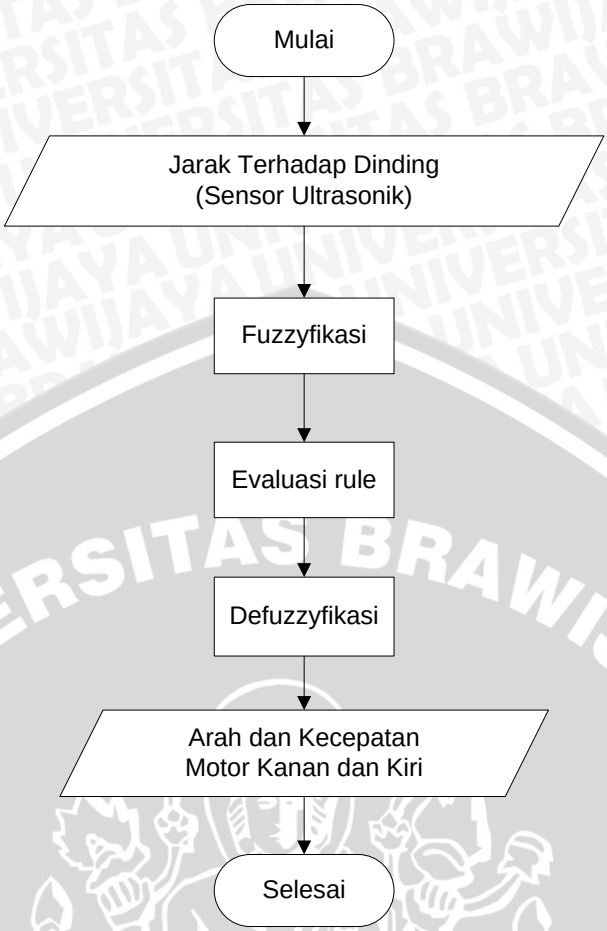
Gambar 4.24. Diagram Alir Program Interupsi komunikasi serial

Untuk menerima sinyal dari sensor *rotary encoder* digunakan perangkat *timer* pada mikrokontroler yaitu *timer 0* untuk sinyal *rotary encoder* sisi kiri dan *timer 2* untuk sinyal *rotary encoder* kanan. Perangkat *timer 1* pada mikrokontroler digunakan sebagai penanda siklus perhitungan kecepatan putaran roda robot. Saat terjadi interupsi *timer 1*, dilakukan perhitungan kecepatan putaran roda robot dan dilakukan koreksi berupa pengiriman ulang nilai arah dan kecepatan pada modul pengendali motor DC. Jika diketahui arus yang mengalir pada motor naik melewati batas kemampuan modul pengendali motor dan kecepatan motor yang terukur adalah nol maka disimpulkan robot membentur dan tertahan sehingga akan dilakukan prosedur koreksi posisi robot. Diagram alir program koreksi posisi robot ditunjukan pada Gambar 4.25.



Gambar 4.25. Diagram Alir Program Koreksi Posisi

Pada algoritma *wall following* yang dirancang, sisi dinding yang akan diikuti bergantung pada mode yang dipilih dengan mengatur saklar pada pin mode pada mikrokontroler. Selanjutnya dilakukan proses awal dari metode logika *fuzzy* yaitu fuzzyfikasi sesuai rancangan yang telah dibuat. Kemudian seluruh derajat keanggotaan yang telah didapatkan dievaluasi dengan *rule* yang telah ditentukan, dimana rule yang digunakan sesuai dengan mode yang dipilih apakah mengikuti diding kanan atau dinding kiri. Penulisan *rule* pada program disusun menggunakan tipe data larik (*array*) agar proses eksekusi lebih cepat dan menghemat alokasi memori. Proses defuzzyfikasi dilakukan dengan menghitung hasil evaluasi *rule* yang telah dilakukan. Hasil defuzzyfikasi berupa kecepatan dan arah motor penggerak selanjutnya dikirimkan pada modul kendali motor DC. Diagram alir program utama logika *fuzzy* ditunjukan pada Gambar 4.26.



Gambar 4.26. Diagram Alir Program Logika Fuzzy pada Mobile Robot

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan yang telah dilakukan. Pengujian dilakukan per blok sistem kemudian secara keseluruhan. Adapun pengujian yang dilakukan sebagai berikut:

1) Pengujian sensor ultrasonik PING)))
   a) Pengujian data sensor ultrasonik
   b) Pengujian rangkaian mikrokontroler pengatur ultrasonik (ATMega8)
2) Pengujian sensor *rotary encoder*
3) Pengujian rangkaian mikrokontroler pengatur utama (ATMega32)
4) Pengujian modul pengendali motor DC *brushed*
5) Pengujian komunikasi UART antar mikrokontroler
6) Pengujian kaidah atur logika fuzzy
7) Pengujian frekuensi kerja logika fuzzy
8) Pengujian keseluruhan sistem
   a) Pengujian robot mengikuti dinding kanan
   b) Pengujian robot mengikuti dinding kiri

5.1. Pengujian Sensor Ultrasonik

Pengujian sensor ultrasonik terdiri atas dua bagian yaitu pengujian data sensor ultrasonik dan pengujian rangkaian mikrokontroler pengatur sensor ultrasonik.

5.1.1. Pengujian Data Sensor Ultrasonik

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah pembacaan sensor sesuai dengan jarak sesungguhnya. Prosedur pengujian dilakukan dengan menghubungkan sensor ultrasonik, mikrokontroler ATMega8 dan terminal pada komputer sesuai dengan Gambar 5.1.

Gambar 5.1. Diagram blok pengujian data sensor ultrasonik

Pada pengujian ini, sebuah objek berbentuk kotak diletakkan sejajar dengan sebuah sensor ultrasonik. Jarak objek dengan sensor diubah-ubah setiap kelipatan jarak 5 cm. Perangkat *timer 1* pada mikrokontroler pengatur ultrasonik digunakan untuk menghitung lama waktu aktif sinyal jawaban dari sensor ultrasonik. Selanjutnya, data berupa lama waktu dalam milisekon (ms) dan jarak terbaca dalam centimeter (cm) dikirim menggunakan kabel serial menuju komputer. Untuk menerima data, digunakan perangkat lunak RealTerm pada komputer dengan konfigurasi *baudrate* 9600 bps, 8 bit data, tanpa paritas dan 1 stop bit.

Hasil pengujian yang diperoleh melalui beberapa kali pengambilan data ditunjukan pada Tabel 5.1.

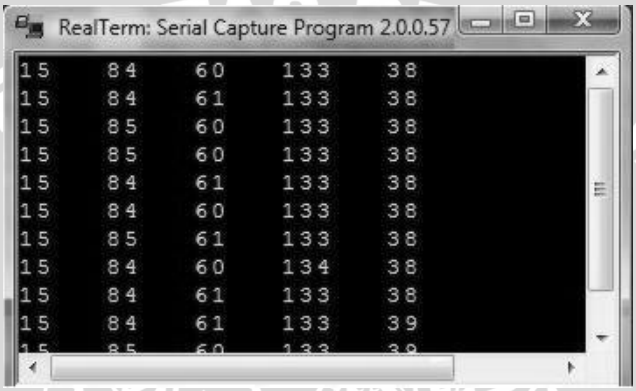
Tabel 5.1. Hasil pengujian data sensor ultrasonik

Pengujian ke-	**Jarak Sesungguhnya (cm)**	**Lama waktu (ms)**	**Jarak Terbaca (cm)**	**Kesalahan (cm)**
1	5	0,332	5,710	0,710
2	10	0,632	10,870	0,870
3	15	0,896	15,411	0,411
4	20	1,180	20,296	0,296
5	25	1,468	25,250	0,250
6	30	1,772	30,478	0,478
7	35	2,064	35,501	0,501
8	40	2,348	40,386	0,386
9	45	2,640	45,408	0,408
10	50	2,928	50,362	0,362
11	55	3,208	55,178	0,178
12	60	3,520	60,544	0,544
13	300	17,496	300,931	0,931
Kesalahan rata-rata				0,487

Berdasarkan Tabel 5.1, dapat diperoleh hasil bahwa kesalahan rata-rata yang terjadi saat pembacaan sensor ultrasonik sebesar 0,487 cm. Pada pengujian, kesalahan pembacaan yang terjadi berupa hasil pengukuran yang lebih besar pada nilai desimal dibelakang koma, sedangkan pada nilai desimal didepan koma bernilai sama dengan jarak yang diuji. Kesalahan tersebut tidak memberikan pengaruh pada kinerja sistem yang dirancang karena pada sistem hanya digunakan data jarak dengan nilai desimal didepan koma sehingga dapat disimpulkan bahwa pada sistem yang dirancang, kesalahan pengukuran yang terjadi adalah nol.

5.1.2. Pengujian Rangkaian Mikrokontroler Pengatur Ultrasonik

Pengujian ini dilakukan untuk mengetahui apakah rangkaian mikrokontroler pengatur ultrasonik berhasil melakukan proses pengaturan sensor ultrasonik sesuai perancangan dan dapat mengirimkan data hasil konversi melalui komunikasi serial UART. Untuk pengujian ini, mikrokontroler mengirimkan data pada terminal komputer menggunakan konfigurasi *baudrate* 9600 bps, 8 bit data, tanpa paritas dan 1 stop bit. Data yang dikirimkan merupakan paket data yang berisi data pembacaan sensor ultrasonik secara urut dari sensor ultrasonik A hingga sensor ultrasonik E. Hasil pengujian ditunjukan pada Gambar 5.2.



Gambar 5.2. Hasil pengujian rangkaian mikrokontroler pengatur ultrasonik

Berdasarkan Gambar 5.2, ditunjukan bahwa data yang diterima pada komputer adalah data pembacaan jarak dari sensor ultrasonik dengan susunan berurutan dari sensor A hingga sensor E. Susunan data yang diterima ditunjukan lebih jelas pada Tabel 5.2.

Tabel 5.2. Susunan data sensor ultrasonik

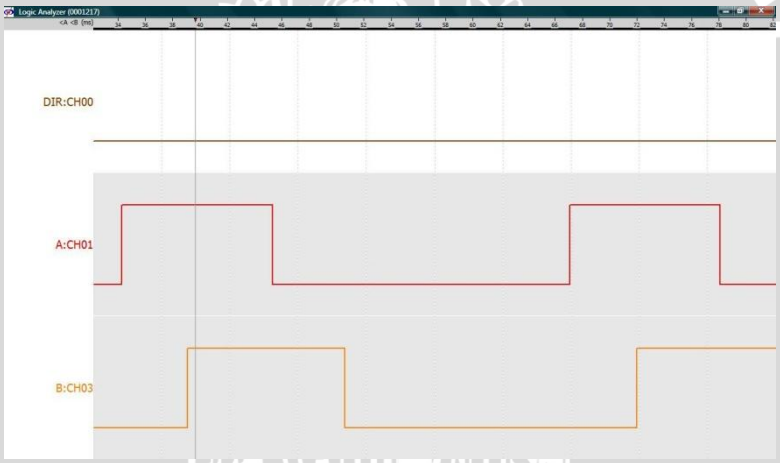
Urutan Data	Nilai (desimal)	Keterangan
1	15	Sensor A
2	84	Sensor B
3	61	Sensor C
4	133	Sensor D
5	38	Sensor E

Pada Tabel 5.2 dapat diketahui bahwa program pengaturan sensor ultrasonik pada mikrokontroler berjalan dengan benar sesuai perancangan dan perangkat UART pada mikrokontroler bekerja dengan baik saat mengirimkan data jarak seluruh sensor ultrasonik melalui kabel serial yang terhubung dengan komputer.

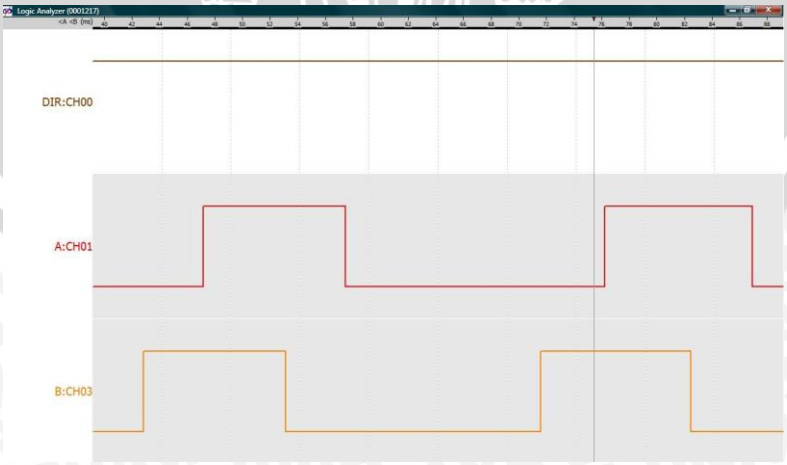
5.2. Pengujian Sensor Rotary Encoder

Pengujian ini bertujuan untuk mengetahui keberhasilan rangkaian sensor *rotary encoder* dalam mendeteksi arah putaran dan respon rangkaian sensor dalam mendeteksi putaran. Pengujian dilakukan dengan menggunakan perangkat *logic analyzer* ELAB-080 dan osiloskop untuk mendeteksi tingkat logika masukan dan keluaran rangkaian sensor.

Sinyal keluaran sensor *rotary encoder* yaitu sinyal A dihubungkan dengan CH1 dan sinyal B dihubungkan dengan CH3 pada *logic analyzer*. Keluaran dari rangkaian antarmuka sensor dihubungkan dengan CH0 pada *logic analyzer*. Selanjutnya, motor DC yang telah dihubungkan dengan rotary encoder diberi catu daya 12 V agar berputar. Pengujian dilakukan sebanyak dua kali dengan membalik polaritas catu daya motor agar arah putar motor berubah sehingga dapat diketahui kinerja rangkaian sensor *rotary encoder*. Hasil pengujian ditunjukkan pada Gambar 5.3 dan Gambar 5.4.



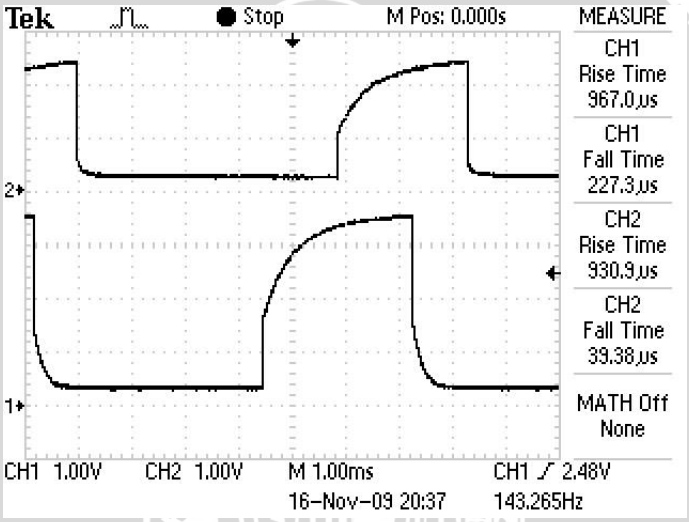
Gambar 5.3. Hasil pengujian putaran searah jarum jam



Gambar 5.4. Hasil pengujian putaran berlawanan jarum jam

Berdasarkan hasil pengujian dapat diketahui bahwa rangkaian sensor *rotary encoder* bekerja dengan baik. Sinyal keluaran sensor *rotary encoder* yaitu sinyal A dan B dapat terdeteksi sebagai sinyal *quadrature* dan arah putar dapat diketahui, dimana sinyal keluaran untuk arah putar searah jarum jam adalah logika 0 dan arah putar berlawanan jarum jam adalah logika 1.

Untuk pengujian respon rangkaian sensor *rotary encoder* digunakan osiloskop untuk melihat lama waktu perubahan tingkat logika. Prosedur pengujian dilakukan dengan menghubungkan *probe osciloscope* pada pin *optoswitch. Channel 1* pada osiloskop dihubungkan dengan *optoswitch* A dan *channel 2* pada osiloskop dihubungkan dengan *optoswitch* B. Selanjutnya motor DC yang terhubung dengan sensor *rotary encoder* diberi sumber 12 V agar berputar. Hasil pengujian yang dilakukan ditunjukan pada Gambar 5.5.



Gambar 5.5. Hasil pengujian respon sensor rotary

Berdasarkan hasil pengujian yang ditunjukan pada Gambar 5.5, dapat diketahui besarnya waktu naik (*rise time*) yaitu 967 uS untuk *optoswitch* A dan 930,9 uS untuk *optoswitch* B. Karena lamanya *rise time* merupakan lama waktu yang menunjukan respon perubahan tingkat logika, maka dapat dihitung besarnya kecepatan maksimum yang bisa diketahui menggunakan persamaan berikut.

$$freq_A = \frac{1}{967\ \mu S} \tag{5.1}$$

$$freq_A = 1034\ Hz$$

$$freq_B = \frac{1}{930{,}9\ \mu S} \tag{5.2}$$

$$freq_B = 1074{,}2\ Hz$$

Dari Persamaan 5.1 dan 5.2 diketahui bahwa sensor yang dirancang dapat bekerja pada frekuensi sebesar 1034 Hz untuk *optoswitch* A dan 1074,2 Hz untuk *optoswitch* B. Dengan menggunakan nilai frekuensi maksimum terendah yaitu 1034 Hz dapat dihitung kecepatan putaran motor maksimum yang dapat dideteksi oleh sensor rotary menggunakan persamaan berikut.

$$\omega_{maks} = \frac{F_{maks}}{70} \tag{5.3}$$

$$\omega_{maks} = \frac{1034\ Hz}{70}$$

$$\omega_{maks} = 14{,}771\ rps$$

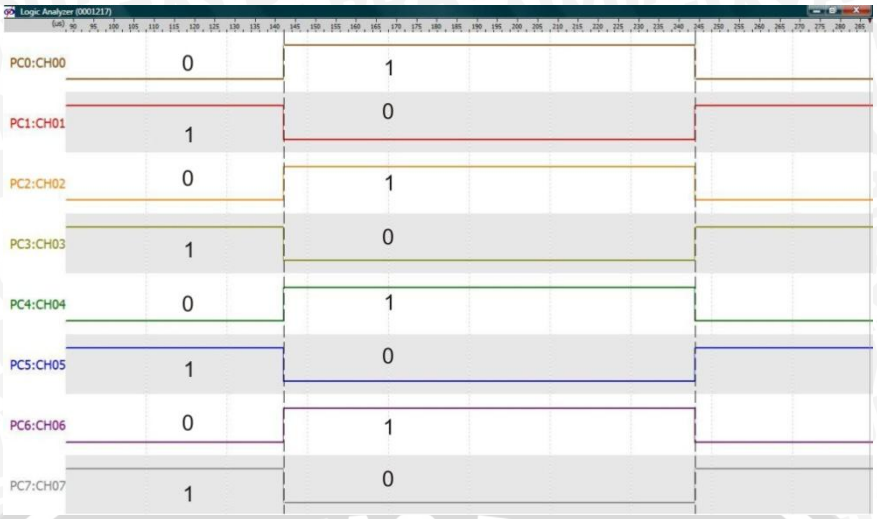
$$\omega_{maks} = 886{,}26\ rpm$$

Berdasarkan Persamaan 5.3 diketahui bahwa kecepatan putaran motor maksimum yang dapat dideteksi pada sensor yang dirancang adalah 886,26 rpm. Pada robot yang digunakan pada perancangan, kecepatan putaran maksimum motor DC penggerak robot adalah 100 rpm sehingga dapat disimpulkan bahwa sensor yang dirancang mampu mendeteksi dengan baik kecepatan roda robot yang digunakan dalam perancangan sistem.

5.3. Pengujian Rangkaian Mikrokontroler Pengatur Utama

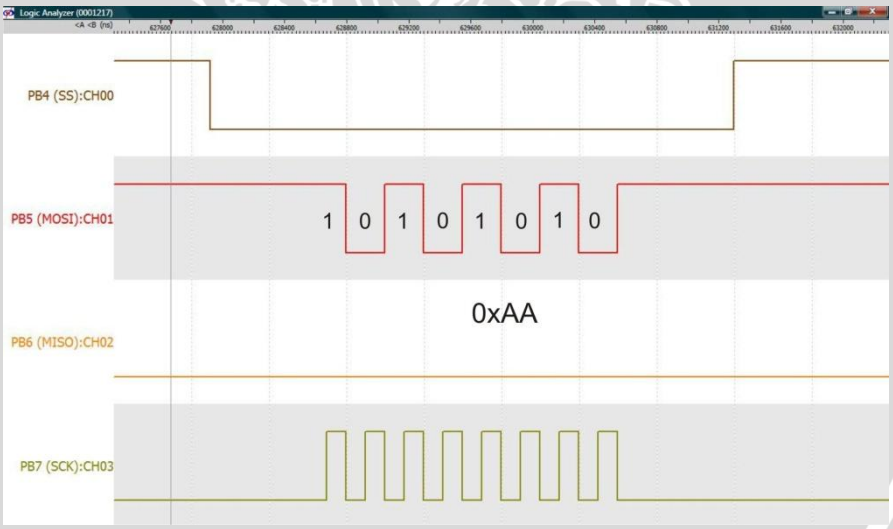
Pengujian ini dilakukan dengan tujuan untuk mengetahui kondisi dari sistem mikrokontroler baik antarmuka dan perangkat komunikasi dalam mikrokontroler yaitu SPI dan UART. Pelaksanaan pengujian menggunakan perangkat *logic analyzer* ELAB-080 dan adaptor IC 40 pin untuk mendeteksi tingkat logika pada kaki mikrokontroler.

Proses pengujian dilakukan sebanyak tiga kali, pengujian pertama yaitu dengan menuliskan nilai sebesar 0xAA pada port C selama 100 mikrosekon kemudian menuliskan nilai 0x55 pada port C selama 100 mikrosekon. Proses tersebut dilakukan secara berulang-ulang. Hasil pengujian port pada mikrokontroler ditunjukan pada Gambar 5.6.



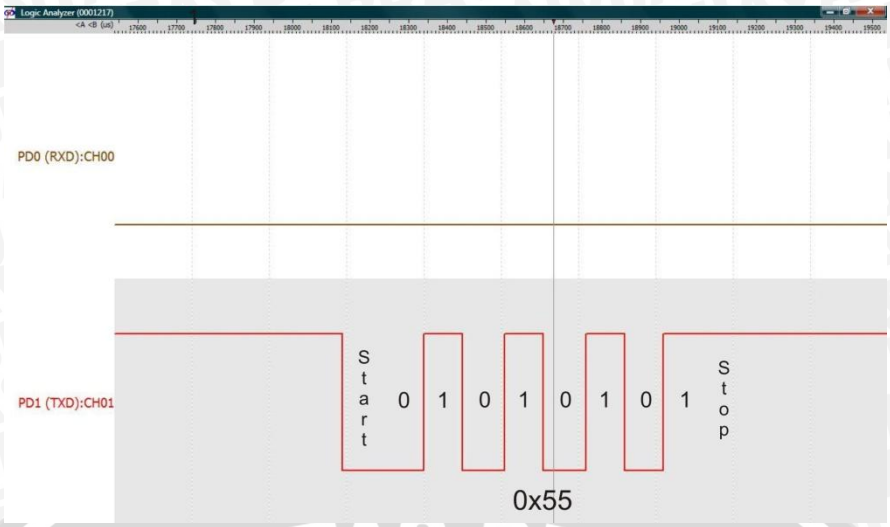
Gambar 5.6. Hasil pengujian port C mikrokontroler

Pengujian kedua adalah pengujian perangkat SPI pada mikrokontroler. Proses pengujian dilakukan dengan mengatur perangkat SPI sebagai master device dan melakukan pengiriman data senilai 0xAA. Hasil pengujian perangkat SPI ditunjukan pada Gambar 5.7.



Gambar 5.7. Hasil pengujian perangkat SPI mikrokontroler

Pengujian ketiga adalah pengujian perangkat UART pada mikrokontroler. Proses pengujian dilakukan dengan mengatur perangkat UART dengan baudrate 9600 bps dan melakukan pengiriman data senilai 0x55. Hasil pengujian perangkat UART ditunjukan pada Gambar 5.8.

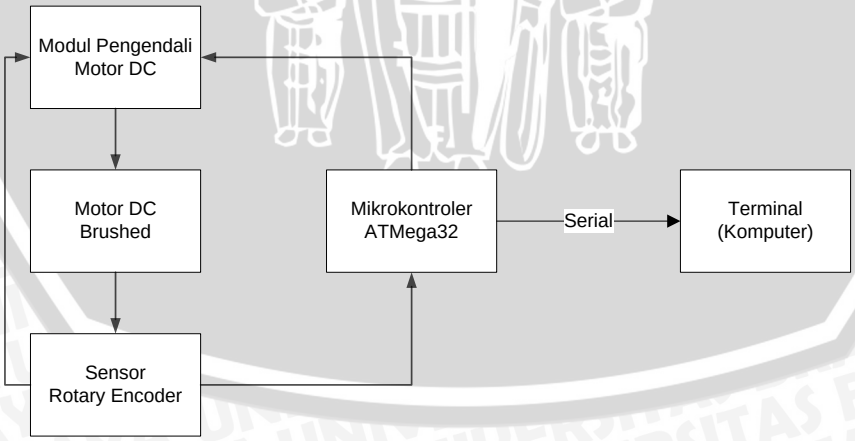


Gambar 5.8. Hasil pengujian perangkat UART mikrokontroler

Berdasarkan hasil pengujian dapat diketahui bahwa port antarmuka, perangkat komunikasi serial SPI dan perangkat komunikasi serial UART pada mikrokontroler pengatur utama dapat bekerja dengan baik.

5.4. Pengujian Modul Pengendali Motor DC Brushed

Pengujian ini bertujuan untuk mengetahui kinerja dari modul pengendali motor DC dengan membandingkan kecepatan putaran motor DC dengan nilai kecepatan yang diinginkan. Prosedur pengujian dilakukan dengan menghubungkan modul pengendali motor DC, mikrokontroler ATMega32, komputer, sensor *rotary encoder* dan motor DC sesuai Gambar 5.9.



Gambar 5.9. Diagram blok pengujian modul pengendali motor DC

Mikrokontroler akan memberikan instruksi berupa arah dan kecepatan pada modul. Kecepatan dan arah putar motor DC diukur dengan menggunakan perangkat *timer 1* dan *external interuppt 1* pada mikrokontroler. Data kemudian dikirimkan

melalui kabel serial menuju komputer. Data yang ditampilkan pada komputer adalah arah dan kecepatan yang diinginkan serta arah dan kecepatan terbaca pada motor DC. Hasil pengujian ditunjukan pada Tabel 5.3.

Tabel 5.3. Hasil pengujian modul pengendali motor DC

Pengujian ke-	Kecepatan Diinginkan (rpm)	Kecepatan Terbaca (rpm)	Kesalahan (%)
1	100	100	0,000
2	80	80	0,000
3	60	59	1,667
4	40	40	0,000
5	20	20	0,000
6	-20	-21	5,000
7	-40	-42	5,000
8	-60	-60	0,000
9	-80	-81	1,250
10	-100	-98	2,000
Kesalahan rata-rata			1,492

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 5.3, dapat diketahui bahwa modul pengendali motor DC berhasil mengendalikan kecepatan dan arah putar motor DC sesuai nilai kecepatan yang telah ditentukan. Kesalahan rata-rata yang terjadi pada pengujian sebesar 1,492% dan kesalahan terbesar yaitu 5% atau sebesar 2 rpm. Kesalahan sebesar 2 rpm sebanding dengan kecepatan putaran roda robot yang dapat dihitung melalui persamaan berikut.

$$V_{error} = 2\,rpm \times \pi \times 8\,cm \tag{5.4}$$
$$V_{error} = 50{,}265\,cm/menit$$
$$V_{error} = 0{,}8377\,cm/detik$$

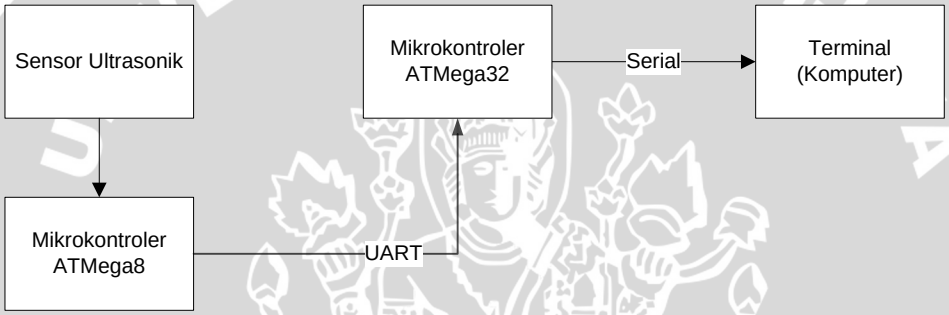
Sehingga jika diasumsikan robot bergerak dengan kecepatan putaran roda kanan dan kiri yang sama dan roda kanan mengalami kesalahan maksimum maka terjadi perbedaan jarak tempuh sebesar 0,8377 cm untuk setiap detiknya. Sehingga dapat diketahui kemiringan posisi robot yang muncul akibat kesalahan pengendalian kecepatan putaran motor melalui persamaan berikut.

$$\alpha_{error} = \frac{0{,}8377\,cm \times 180°}{24\,cm \times \pi} \tag{5.5}$$
$$\alpha_{error} = 1{,}99°$$

Dari Persamaan 5.5 dapat diketahui bahwa terdapat kemiringan sebesar 1,99° yang muncul setiap detik akibat kesalahan dalam pengaturan kecepatan motor penggerak robot. Kesalahan tersebut tidak mengganggu sistem karena pada sistem yang dirancang digunakan sensor ultrasonik dengan referensi jarak terhadap dinding sebagai masukan.

5.5. Pengujian Komunikasi UART antar Mikrokontroler

Pengujian ini dilakukan untuk mengetahui apakah sistem komunikasi UART dengan *baudrate* 1 Mbps antara mikrokontroler utama dan mikrokontroler pengatur ultrasonik berjalan dengan benar dan paket data jarak yang ditransmisikan dapat dikenali. Pada pengujian ini, sensor ultrasonik, mikrokontroler pengatur ultrasonik, mikrokontroler pengatur utama dan komputer disusun seperti pada Gambar 5.10.



Gambar 5.10. Diagram blok pengujian komunikasi UART antar mikrokontroler

Proses pengujian dilakukan dengan mengirimkan data jarak sensor ultrasonik bagian depan dan samping kanan (sensor ultrasonik C dan sensor ultrasonik E) pada komputer menggunakan kabel serial dengan *baudrate* 9600 bps. Pengujian dilakukan dengan meletakan objek didepan sensor ultrasonik dengan jarak yang berubah-ubah. Hasil pengujian ditunjukan pada Tabel 5.4.

Tabel 5.4. Hasil pengujian komunikasi UART antar mikrokontroler

Pengujian ke-	Sensor ultrasonik C		Sensor ultrasonik E	
	Jarak Uji (cm)	Jarak Terukur (cm)	Jarak Uji (cm)	Jarak Terukur (cm)
1	10	10	50	50
2	20	20	40	40
3	30	30	30	30
4	40	40	20	20
5	50	50	10	10

Berdasarkan Tabel 5.4 dapat diketahui bahwa paket data jarak yang dikirimkan mikrokontroler pengatur ultrasonik dapat diterima oleh mikrokontroler pengatur utama dan susunan data jarak dalam paket data yang diterima telah berhasil dikenali oleh mikrokontroler pengatur utama.

5.6. Pengujian Kaidah Atur Logika Fuzzy

Pengujian ini dilakukan untuk mengetahui apakah program yang dibuat telah sesuai dengan perancangan dan kaidah atur yang digunakan dalam program telah sesuai dengan perancangan. Untuk pengujian ini, robot diletakan diatas sebuah penyangga yang menyangga badan robot sehingga roda tidak menginjak tanah dan menggunakan algoritma fuzzy untuk mengikuti dinding kanan. Objek diletakan pada bagian depan, samping kanan depan dan samping kanan robot. Jarak objek terhadap sensor ultrasonik diubah sesuai pengujian. Selanjutnya dicatat arah putaran roda kanan dan roda kiri robot. Hasil pengujian ditunjukan pada Tabel 5.5.

Tabel 5.5. Hasil pengujian kaidah atur logika fuzzy mengikuti dinding kanan

Uji ke-	Jarak (cm)						Putaran motor (rpm)		
	Depan		Samping Kanan Depan		Samping Kanan		Kiri	Kanan	Keterangan
1	70	jauh	15	sedang	7	dekat	75	99	Maju kekiri
2	70	jauh	15	sedang	13	sedang	100	100	Maju penuh
3	30	sedang	15	sedang	13	sedang	74	75	Maju sedang
4	30	sedang	15	sedang	7	dekat	25	76	Maju kekiri
5	10	dekat	30	jauh	30	jauh	99	-25	Berputar kekanan
6	10	dekat	30	jauh	13	sedang	-25	74	Berputar kekiri
7	10	dekat	30	jauh	7	dekat	-25	100	Berputar kekiri
8	10	dekat	7	dekat	13	sedang	-74	75	Berputar kekiri
9	10	dekat	7	dekat	30	jauh	-75	75	Berputar kekiri

Berdasarkan hasil pengujian pada Tabel 5.5 dapat diketahui bahwa kaidah atur yang telah dirancang berhasil diimplementasikan pada sistem. Pergerakan motor menunjukan bahwa fungsi keanggotaan yang dirancang berfungsi dengan benar sehingga kaidah atur yang berhubungan menjadi aktif.

5.7. Pengujian Frekuensi Kerja Sistem Logika Fuzzy

Pengujian ini dilakukan untuk mengetahui lama waktu yang diperlukan untuk membuat sebuah keputusan pergerakan pada sistem logika *fuzzy* yang dirancang. Untuk pengujian ini, perangkat *timer 1* pada mikrokontroler pengatur utama digunakan untuk

menghitung lama waktu antara dua paket data dari mikrokontroler pengatur ultrasonik. Acuan menggunakan paket data jarak karena pada perancangan, sistem logika *fuzzy* akan dijalankan setiap ada paket data jarak yang diterima. Hasil pengujian ditunjukan pada Tabel 5.6.

Tabel 5.6. Hasil pengujian frekuensi kerja sistem

Pengujian ke-	Hitungan Timer 1	Lama waktu (ms)	Frekuensi (Hz)
1	15044	60,176	16,618
2	15046	60,184	16,616
3	15047	60,188	16,615
4	15044	60,176	16,618
5	15044	60,176	16,618
6	15047	60,188	16,615
7	15043	60,172	16,619
8	15046	60,184	16,616
9	15045	60,180	16,617
10	15047	60,188	16,615
Frekuensi rata-rata			16,616

Dari Tabel 5.6 diketahui bahwa frekuensi kerja rata-rata dari sistem adalah 16,616 Hz. Diketahui bahwa kecepatan putaran (ω) motor maksimum pada robot sebesar 100 rpm dan diameter roda robot (D) adalah 8 cm. Sehingga dapat dihitung kecepatan robot (V) saat kedua motor penggerak berputar dengan kecepatan maksimal pada persamaan berikut.

$$V = \omega . \pi . D \tag{5.6}$$

$$V = 100\ rpm\ \times\ \pi\ \times 8\ cm$$

$$V = 2513{,}274\ cm/menit$$

$$V = 41{,}88\ cm/s$$

Selanjutnya dapat dihitung pergeseran jarak (H_{maks}) robot untuk setiap satu siklus kerja sistem menggunakan persamaan berikut.

$$H_{maks} = \frac{V}{F} \tag{5.7}$$

$$H_{maks} = \frac{41{,}88\ cm/s}{16{,}616\ Hz}$$

$$H_{maks} = 2{,}52\ cm$$

Dari Persamaan 5.7 diketahui bahwa jarak maksimum yang dapat ditempuh robot untuk setiap siklus kerja sistem logika *fuzzy* yang dirancang sebesar 2,52 cm. Hal ini berarti bahwa sistem dapat memberikan respon terhadap perubahan mendadak dalam bentuk gangguan (*disturbance*) yang muncul pada jarak lebih dari 2,52 cm.

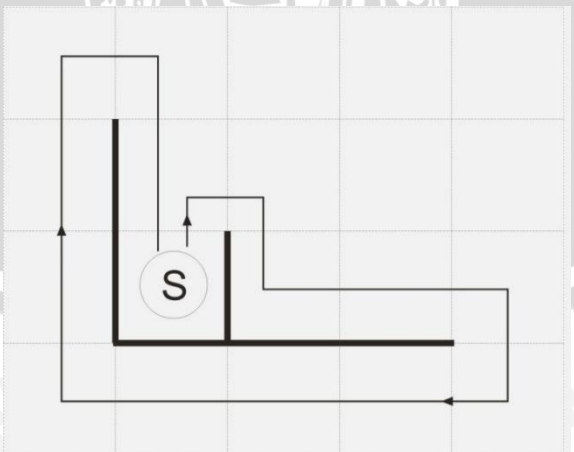
5.8. Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem terdiri atas dua bagian yaitu pengujian robot saaat mengikuti dinding kanan dan pengujian robot saat mengikuti dinding kiri pada model lapangan pengujian. Proses pengujian dilakukan menggunakan model lapangan yang memiliki dimensi standar seperti perlombaan dan tersusun atas blok-blok (*grid*) selebar 50 x 50 cm. Model lapangan yang digunakan merepresentasikan bentuk-bentuk lintasan yang dapat terjadi dalam arena perlombaan.

Prosedur pengujian dilakukan dengan meletakkan robot pada posisi awal dalam arena dan mengaktifkan robot agar bergerak menelusuri sisi lintasan dalam selang waktu maksimum 6 menit untuk setiap pengujian yang merupakan waktu maksimum pada pelaksanaan Kontes Robot Cerdas Indonesia (KRCI). Selanjutnya dihitung banyaknya benturan yang yang dilakukan robot pada dinding arena selama selang waktu robot dalam menelusuri lintasan.

5.8.1. Pengujian Robot Mengikuti Dinding Kanan

Pengujian ini dilakukan untuk mengetahui performa sistem logika *fuzzy* yang diimplementasikan pada robot dalam mengikuti sisi dinding lintasan sebelah kanan. Arena pengujian dan ilustrasi jalur pergerakan robot ditunjukan pada Gambar 5.11.



Gambar 5.11. Ilustrasi pergerakan robot pada arena pengujian

Robot diletakan pada posisi awal (S) kemudian bergerak menelusuri arena dengan mengikuti sisi dinding sebelah kanan robot secara terus menerus. Arah panah menunjukan ilustrasi rute yang dilewati robot selama proses pengujian. Hasil pengujian ditunjukan pada Tabel 5.7.

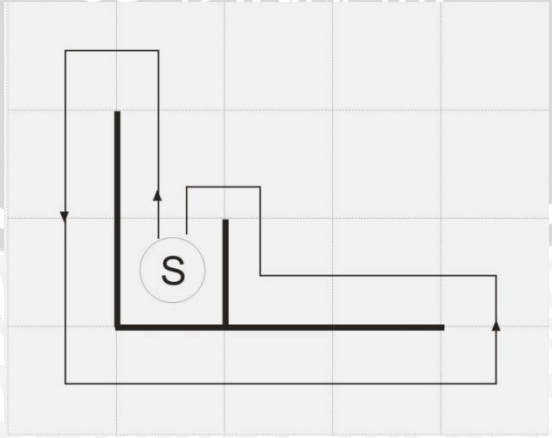
Tabel 5.7. Hasil pengujian robot mengikuti dinding kanan

Pengujian ke-	Jumlah Benturan	Lama waktu (menit:detik)	Hasil Pengujian
1	6	6:00	Berhasil mengikuti dinding
2	2	3:24	Tertahan dinding saat belok
3	4	6:00	Berhasil mengikuti dinding
4	4	6:00	Berhasil mengikuti dinding
5	4	6:00	Berhasil mengikuti dinding

Berdasarkan hasil pengujian dapat diketahui bahwa sistem logika *fuzzy* pada robot telah berhasil membuat keputusan dalam menentukan pergerakan robot selama mengikuti dinding sebelah kanan. Kegagalan robot untuk bergerak terus menerus hingga waktu maksimum seperti pada pengujian kedua dan kelima disebabkan oleh terbenturnya badan robot pada bagian tertentu pada dinding lintasan sehingga posisi robot bergeser dan mengakibatkan arah pergerakan robot berubah sehingga robot tidak dapat bergerak akibat tertahan ujung dinding lintasan.

5.8.2. Pengujian Robot Mengikuti Dinding Kiri

Pengujian ini dilakukan untuk mengetahui performa sistem logika *fuzzy* yang diimplementasikan pada robot dalam mengikuti sisi dinding sebelah kiri. Arena pengujian dan ilustrasi jalur pergerakan robot ditunjukan pada Gambar 5.12.



Gambar 5.12. Ilustrasi pergerakan robot pada arena pengujian

Robot diletakan pada posisi awal (S) kemudian bergerak menelusuri arena dengan mengikuti sisi dinding sebelah kiri robot secara terus menerus. Hasil pengujian ditunjukan pada Tabel 5.8.

Tabel 5.8. Hasil pengujian robot mengikuti dinding kanan

Pengujian ke-	Jumlah Benturan	Lama waktu (menit:detik)	Hasil Pengujian
1	5	6:00	Berhasil mengikuti dinding
2	3	6:00	Berhasil mengikuti dinding
3	3	6:00	Berhasil mengikuti dinding
4	4	6:00	Berhasil mengikuti dinding
5	2	2:48	Tertahan dinding lintasan

Berdasarkan hasil pengujian dapat diketahui bahwa sistem logika *fuzzy* pada robot telah berhasil membuat keputusan dalam menentukan pergerakan robot selama mengikuti dinding sebelah kiri. Kegagalan robot untuk bergerak terus menerus pada pengujian kelima hingga waktu maksimum disebabkan oleh terbenturnya badan robot pada bagian ujung pada dinding lintasan yang membentuk sebuah tikungan model U sehingga posisi robot bergeser yang mengakibatkan robot terhenti. Secara garis besar, performa sistem logika *fuzzy* untuk mengikuti sisi dinding kanan maupun kiri berfungsi dengan baik. Hasil pengujian keseluruhan yang dilakukan baik pengujian mengikuti dinding kanan dan pengujian mengikuti dinding kiri mendapatkan kemiripan hasil karena pada perancangan yang telah dilakukan menggunakan kaidah atur dan fungsi keanggotaan yang sama.

BAB VI
KESIMPULAN DAN SARAN

6.1. Kesimpulan

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

1. Sensor ultrasonik PING))) memiliki kesalahan rata-rata pengukuran sebesar 0,449 cm. Semakin dekat jarak objek terhadap sensor, kesalahan pengukuran semakin meningkat. Kesalahan pengukuran tidak berpengaruh pada sistem keseluruhan karena hanya nilai desimal didepan koma yang digunakan.
2. Sensor *rotary encoder* yang dirancang dapat mendeteksi arah putaran roda pada robot dan bekerja dengan baik hingga 1034 pulsa per detik. Kecepatan putaran maksimum yang mampu dideteksi sensor sebesar 886,26 putaran per menit. Kecepatan putaran motor robot maksimum yang digunakan adalah 100 putaran per menit atau 41,88 centimeter per detik dengan diameter roda robot sebesar 8 centimeter.
3. Sistem komunikasi UART dengan *baudrate* 1 Mbps antara mikrokontroler berjalan dengan baik dimana paket data yang telah disusun dan dikirim oleh mikrokontroler pengatur ultrasonik berhasil diterima dan diolah oleh mikrokontroler pengatur utama.
4. Frekuensi kerja sistem yang dirancang sebesar 16,616 Hz, mampu memberikan keputusan pergerakan bagi mobile robot setiap 60,184 milidetik.
5. Dengan menggunakan algoritma *fuzzy* yang dirancang, robot telah mampu membuat keputusan model pergerakan dalam mengikuti sisi dinding arena dengan lama waktu hingga 6 menit yang merupakan lama waktu maksimum untuk setiap pertandingan dalam Kontes Robot Cerdas Indonesia.

6.2. Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah:

1. Penambahan fungsi keanggotaan masukan dan modifikasi kaidah atur agar robot dapat menjauhi sisi dinding yang tidak diikuti.
2. Penggunaan sensor *rotary encoder* dengan resolusi lebih tinggi untuk meningkatkan ketelitian dalam pengukuran kecepatan putaran roda.

DAFTAR PUSTAKA

Atmel. 2007a. *8-bit AVR with 32K Bytes In-System Programmable Flash Atmega32, Atmega32L*. San Jose: Atmel.

Atmel. 2007b. *8-bit AVR with 8K Bytes In-System Programmable Flash Atmega8, Atmega8L*. San Jose: Atmel.

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

Braunl, Thomas. 2006. *Embedded Robotics Second Edition*. Jerman: Springer.

Coughanowr, D.R. 1991. *Process System–Analysis and Control*. USA: Prentice Hall.

DIKTI. 2008. *Panduan Kontes Robot Cerdas Indonesia 2009*. Jakarta: DIKTI.

Harashima, Furnio. 1999. *Kinematic Correction of Differential Drive Mobile Robot and Design for Velocity Trajectory with Acceleration Constraints on Motor Controller*. Proceeding of the 1999 IEEE/RSJ, 930-935.

Kilian. 2002. *Modern Control Technology--Components & Systems (2nd Ed.)*. Delmar.

Kuswadi, Son. 2007. *Kendali Cerdas: Teori dan Aplikasi Praktisnya*. Yogyakarta: ANDI.

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

Pitowarno, Endra. 2006. *Robotika: Desain, Kontrol dan Kecerdasan Buatan*. Yogyakarta: ANDI.

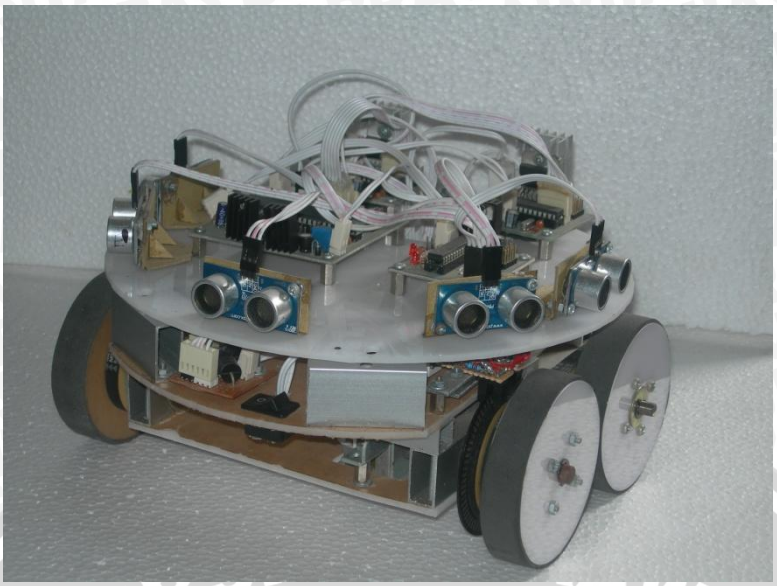
Sigit, Riyanto. 2007. *Robotika, Sensor, dan Aktuator*. Yogyakarta: Graha Ilmu.

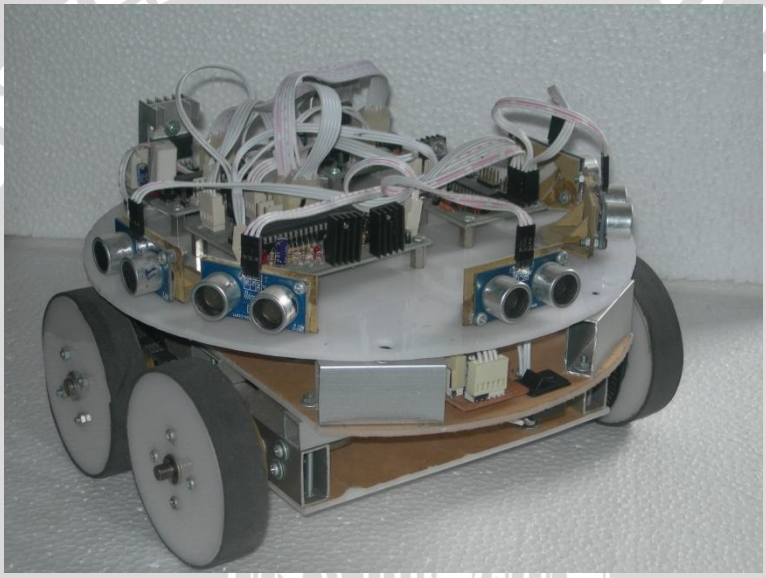
Thiang, Resmana, Wahyudi. 2007. *Aplikasi Kendali Fuzzy Logic untuk Pengaturan Kecepatan Motor Universal*. Surabaya: Universitas Kristen Petra.

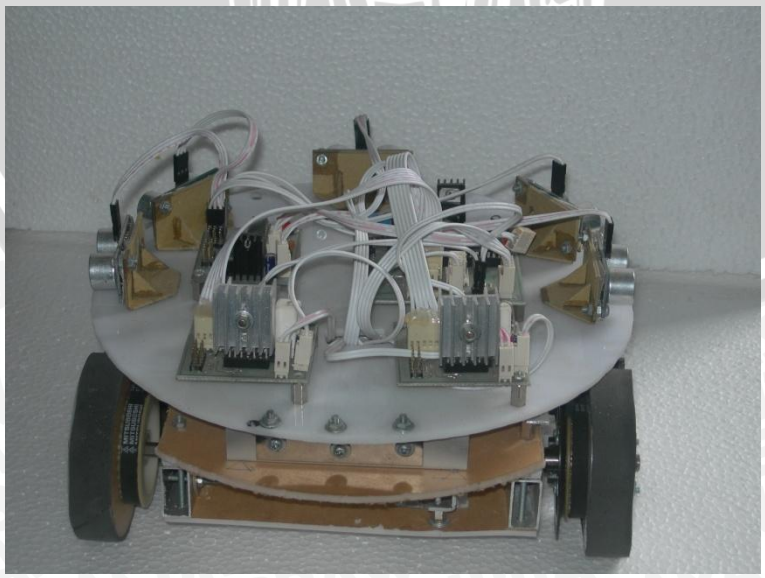
Yan,J.,Ryan,M., dan Power,J. 1993. *Using Fuzzy Logic*. NewYork: Prentice Hall.

LAMPIRAN

FOTO ALAT







LAMPIRAN

KODE PROGRAM

1. File “sonar.c” (mikrokontroler ultrasonik)

```
#include <mega8.h>
#include <stdio.h>
#include <delay.h>
#include <sleep.h>
#define     t1_off              TCCR1B=0x00
#define     t1_on               TCCR1B=0x0B
#define     reset_t1            TCNT1=0x0000
#define UDRE 5
enum sequence {
                        PING_A,
                        PING_B,
                        PING_C,
                        PING_D,
                        PING_E,
                       } eksekusi;
unsigned int timing, jarak[5];
char schedule=0, flag=0;
unsigned int iterasi;
char x=0;
/*____deklarasi___Sub rutin program_________*/
//void ambil_data(unsigned char indeks);
void inisialisasi_sistem(void);
void uart_speed_setting(void);
void baca_ping(char i);
void data_out(void);
void data_out2(void);
void kirim_data_UART(unsigned char data);

/*____Sub rutin interupsi_______*/
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{  schedule=1;}
interrupt [TIM1_COMPB] void timer1_compb_isr(void)
{  flag=1;}

///// program utama
void main(void)
{
inisialisasi_sistem();
sleep_enable();
t1_off;
reset_t1;
PORTD.6=1;PORTD.7=1;
uart_speed_setting();

delay_ms(200);
#asm("sei")
    if(PIND.3==1)
     {
       printf("debug mode \r");
       while(1)
       {
       PORTD.6 = ~PORTD.6;PORTD.7 = 0;       // indikator
       eksekusi=PING_A;   baca_ping(eksekusi);
       eksekusi=PING_C;   baca_ping(eksekusi);
       eksekusi=PING_E;   baca_ping(eksekusi);
       eksekusi=PING_B;   baca_ping(eksekusi);
       eksekusi=PING_D;   baca_ping(eksekusi);
       data_out();
       iterasi++;
```

```
      };
    }
   if(PIND.3==0)
    {
     while(getchar()!=0);x=0;     //menunggu instruksi start
     while(1)
     {
      if(x==0)
        {
         PORTD.6 = ~PORTD.6;PORTD.7 = ~PORTD.7;    ///   indikator
         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_E;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);
         flag=0;
         reset_t1;
        ////////////////////////////////////////////////////////////////////
        if(UCSRA & (1<<7))
          {
            x=UDR;           /// apakah ada instruksi berhenti
            if(x==1)    {while(getchar()!=0);}    ///menunggu re-start
            x=0;
          }
        }
      };
    }
}

void baca_ping(char i)
{
  t1_off;
  flag=0;schedule=0;
  reset_t1;
  PORTB&=0x00;
  DDRB&=0x00;      // nulling PING PORT == IN
  PORTC&=0x00;
  DDRC&=0x00;      // nulling PING PORT == IN
  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);
         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);
            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);
            timing=(unsigned int)TCNT1;
            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);
            break;
        case 4:
            DDRB.4=1;PORTB.4=1;delay_us(5);PORTB.4=0;
            DDRB.4=0;delay_us(4);
            while(PINB.4==1 && flag==0){};
            while(PINB.4==0 && flag==0){}; t1_on;reset_t1;
            while(PINB.4==1 && flag==0){}; t1_off;
            jarak[i]=(unsigned int)((344*0.5*100*TCNT1*64)/16000000);
            break;
    }
    t1_on;
    if(schedule==0){idle();}
    t1_off;
}
void uart_speed_setting(void)
{
    #asm("cli")
     if(PIND.3==0)    //UBRRL=0x01; operational 500kbps
        {            //UBRRL=0x00; operational 1Mbps
         UBRRH=0x00;
         UBRRL=0x00;
        }
      else if(PIND.3==1)  //9600bps for debug purpose
         {
         UBRRH=0x00;
         UBRRL=0x67;
         }
    #asm("sei")
}

void data_out(void)
{
   printf("%i   %i   %i   %i   %i", jarak[0], jarak[1], jarak[2], jarak[3], jarak[4]);
   printf("\r");
}

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

```
}

void inisialisasi_sistem(void)
{
   PORTB=0x00;
   DDRB=0x00;
   PORTC=0x00;
   DDRC=0x00;
   PORTD=0xC0;
   DDRD=0xC0;
   // Clock source: System Clock
   // Clock value: Timer 0 Stopped
   TCCR0=0x00;
   TCNT0=0x00;
   // 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
   // Timer 1 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;
   // Clock source: System Clock
   // Clock value: Timer 2 Stopped
   // Mode: Normal top=FFh
   // OC2 output: Disconnected
   ASSR=0x00;
   TCCR2=0x00;
   TCNT2=0x00;
   OCR2=0x00;
   // INT0: Off
   // INT1: Off
   MCUCR=0x00;
   // Timer(s)/Counter(s) Interrupt(s) initialization
   TIMSK=0x18;
   // Communication Parameters: 8 Data, 1 Stop, No Parity
   // USART Receiver: on
   // USART Transmitter: On
   // USART Mode: Asynchronous
   // USART Baud Rate:  /////115200
   UCSRA=0x00;
   UCSRB=0x18;
   UCSRC=0x86;
   UBRRH=0x00;
   UBRRL=0x67;///08;

   // Analog Comparator initialization
```

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

2. Mikrokontroler Utama

2.1. Macro

```
#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7
#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)
#define RX_BUFFER_SIZE 7
#define PING_A 0
#define PING_B 1
#define PING_C 2
#define PING_D 3
#define PING_E 4
#define sisi_kanan 1
#define sisi_kiri 2
#define MF_dekat 0
#define MF_sedang 1
#define MF_jauh 2
#define FF_v 0
#define FM_v 1
#define FL_v 2
#define ST_v 3
#define RL_v 4
#define RM_v 5
#define t1_on TCNT1=0;TCCR1B=0x03
#define t1_off TCCR1B=0x00
```

2.2. File “global_var.h”

```
unsigned char rx_buffer[RX_BUFFER_SIZE];
unsigned char rx_wr_index;
bit rx_buffer_overflow;
//////////////////////////////////////
unsigned char hk=0;
struct {
      int sonar[5];
      unsigned char sonar_complete;
      int rotary_kanan;
      int rotary_kiri;
      int arus_kanan;
      int arus_kiri;
     }sensor={{0,0,0,0,0},0,0,0,0,0};
```

2.3. File “interrupt.h”

```
interrupt [USART_RXC] void usart_rx_isr(void);
```

```
interrupt [TIM1_COMPA] void timer1_compa_isr(void);
```

2.4. File "fuzzy.h"

```
float triangle(int value,float x0, float x1,float x2);
float grade(int value,float x0, float x1);
float reverse_grade(int value,float x0,float x1);
float trapesium(int value,float x0,float x1,float x2,float x3);
void fuzzyfikasi(char sisi);
void check_rule();
void defuzzyfikasi(char sisi);

float sisi_depan[3];
float sisi_samping_depan[3];
float sisi_samping[3];
float R_out[6],L_out[6];
unsigned char rule_R[3][3][3]={
                    {               //depan dekat
                      {FF_v,FF_v,FF_v},       //samping  dekat
                      {FF_v,FF_v,FF_v},       //samping sedang
                      {FM_v,FM_v,RL_v}        //samping jauh
                    },
                    {               //depan sedang
                      {FM_v,FM_v,ST_v},       //samping  dekat
                      {FM_v,FM_v,RL_v},       //samping sedang
                      {FM_v,FL_v,RL_v}        //samping jauh
                    },
                    {               //depan jauh
                      {FF_v,FF_v,ST_v},       //samping  dekat
                      {FF_v,FF_v,RL_v},       //samping sedang
                      {FF_v,FM_v,RL_v}        //samping jauh
                    }
                  };
unsigned char rule_L[3][3][3]={
                    {               //depan dekat
                      {RM_v,RM_v,RM_v},       //samping  dekat
                      {RM_v,RM_v,RM_v},       //samping sedang
                      {RM_v,RM_v,FF_v}        //samping jauh
                    },
                    {               //depan sedang
                      {ST_v,FL_v,FM_v},       //samping  dekat
                      {RL_v,FM_v,FM_v},       //samping sedang
                      {RM_v,FM_v,FM_v}        //samping jauh
                    },
                    {               //depan jauh
                      {ST_v,FM_v,FF_v},       //samping  dekat
                      {RL_v,FF_v,FF_v},       //samping sedang
                      {RM_v,FF_v,FF_v}        //samping jauh
                    }
                  };
```

2.5. File "fuzzy.c"

```
void fuzzyfikasi(char sisi)
{
 sisi_depan[MF_dekat]     =   reverse_grade(sensor.sonar[PING_C],13,25);
 sisi_depan[MF_sedang]    =   trapesium(sensor.sonar[PING_C],13,25,50,63);
 sisi_depan[MF_jauh]      =   grade(sensor.sonar[PING_C],50,63);
    if(sisi==sisi_kanan)
    {
     sisi_samping_depan[MF_dekat]    =   reverse_grade(sensor.sonar[PING_D],8,17);
     sisi_samping_depan[MF_sedang]   =   triangle(sensor.sonar[PING_D],8,17,27);
```

```
      sisi_samping_depan[MF_jauh]    =  grade(sensor.sonar[PING_D],17,27);
      sisi_samping[MF_dekat]    =  reverse_grade(sensor.sonar[PING_E],5,13);
      sisi_samping[MF_sedang]   =  triangle(sensor.sonar[PING_E],5,13,21);
      sisi_samping[MF_jauh]     =  grade(sensor.sonar[PING_E],13,21);
     }
     else if(sisi==sisi_kiri)
     {
      sisi_samping_depan[MF_dekat]    =  reverse_grade(sensor.sonar[PING_B],8,17);
      sisi_samping_depan[MF_sedang]   =  triangle(sensor.sonar[PING_B],8,17,27);
      sisi_samping_depan[MF_jauh]     =  grade(sensor.sonar[PING_B],17,27);
      sisi_samping[MF_dekat]    =  reverse_grade(sensor.sonar[PING_A],5,13);
      sisi_samping[MF_sedang]   =  triangle(sensor.sonar[PING_A],5,13,21);
      sisi_samping[MF_jauh]     =  grade(sensor.sonar[PING_A],13,21);
     }
}

void check_rule()
{
 float temp_v=0;
 char x,y,z;
 for(x=0;x<6;x++)
      {
      R_out[x]=0;
      L_out[x]=0;
      }
 for(x=0;x<3;x++)        //depan
   {
    for(y=0;y<3;y++)     //samping
    {
     for(z=0;z<3;z++)    //samping depan
     {
      if(sisi_depan[x]>0 && sisi_samping_depan[z]>0 && sisi_samping[y]>0)
       {
         temp_v =  (float)fmin(sisi_depan[x],sisi_samping_depan[z]);
         temp_v =  (float)fmin(sisi_samping[y],temp_v);
         R_out[rule_R[x][y][z]] = (float)fmax(R_out[rule_R[x][y][z]],temp_v);
         L_out[rule_L[x][y][z]] = (float)fmax(L_out[rule_L[x][y][z]],temp_v);
       }
     }
    }
   }
}

void defuzzyfikasi(char sisi)
{
 float temp_x,temp_L,temp_R;
 if(sisi==sisi_kanan)
 {
  temp_L = (float) (L_out[FF_v] + L_out[FM_v] + L_out[FL_v] + L_out[ST_v] + L_out[RL_v] +
  L_out[RM_v]);
  temp_R = (float) (R_out[FF_v] + R_out[FM_v] + R_out[FL_v] + R_out[ST_v] + R_out[RL_v]
  + R_out[RM_v]);

  temp_x = (float) (L_out[FF_v]*100 + L_out[FM_v]*75 + L_out[FL_v]*25 + L_out[ST_v]*0 +
  L_out[RL_v]*-25 + L_out[RM_v]*-75);
  temp_L = (float) (temp_x / temp_L);

  temp_x = (float) (R_out[FF_v]*100 + R_out[FM_v]*75 + R_out[FL_v]*25 + R_out[ST_v]*0 +
  R_out[RL_v]*-25 + R_out[RM_v]*-75);
  temp_R = (float) (temp_x / temp_R);
```

```
 }
 else if(sisi==sisi_kiri)
 {
  temp_L = (float) (L_out[FF_v] + L_out[FM_v] + L_out[FL_v] + L_out[ST_v] + L_out[RL_v] +
  L_out[RM_v]);
  temp_R = (float) (R_out[FF_v] + R_out[FM_v] + R_out[FL_v] + R_out[ST_v] + R_out[RL_v]
  + R_out[RM_v]);

  temp_x = (float) (L_out[FF_v]*100 + L_out[FM_v]*75 + L_out[FL_v]*25 + L_out[ST_v]*0 +
  L_out[RL_v]*-25 + L_out[RM_v]*-75);
  temp_L = (float) (temp_x / temp_L);

  temp_x = (float) (R_out[FF_v]*100 + R_out[FM_v]*75 + R_out[FL_v]*25 + R_out[ST_v]*0 +
  R_out[RL_v]*-25 + R_out[RM_v]*-75);
  temp_R = (float) (temp_x / temp_R);

  temp_x = temp_R;
  temp_R = temp_L;
  temp_L = temp_x;
 }
 kirim_motor((int)temp_L,(int)temp_R);
}

////////////////////////////////////////////////////////////////////////////////////////
float triangle(int value, float x0, float  x1,float x2)
{
   float result = 0;
    float x;
    x = value;
    if((x <= x0) || (x >= x2))
       {result = 0;}
    else if(x == x1)
       {result = 1;}
    else if((x>x0) && (x<x1))
       {result = ((x-x0)/(x1-x0));}
    else
       {result =(((-x)+x2)/(x2-x1));}
    return result;
}
float grade(int value,float x0, float x1)
{
    float result = 0;
    float x;
    x = value;
    if(x <= x0)
       {result = 0;}
    else if(x >= x1)
       {result = 1;}
    else
       {result = ((x-x0)/(x1-x0));}
    return result;
}
float reverse_grade(int value,float x0,float  x1)
{
    float result = 0;
    float x;
    x = value;
    if(x <= x0)
       {result = 1;}
    else if(x >= x1)
```

```
    {result = 0;}
  else
    {result = (((-x)+x1)/(x1-x0));}
  return result;
}
float trapesium(int value,float x0,float x1,float x2,float x3)
{
 float result = 0;
 float x = value;

   if(x <= x0) {result = 0;}
   else if((x>=x1) && (x<=x2)){result = 1;}
   else if((x>x0) && (x<x1)){result = ((x-x0)/(x1-x0));}
   else {result = ((-x+x3)/(x3-x2));}
   return result;
}
```

2.6. File "interrupt.c"

```
interrupt [USART_RXC] void usart_rx_isr(void)       //penerima transmisi data jarak
{
char status;
unsigned char data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
  {
  rx_buffer[rx_wr_index]=data;
  if (data==0xFF || (++rx_wr_index == RX_BUFFER_SIZE))
   {
    rx_wr_index=0;
    if(rx_buffer[0]==0xEE && rx_buffer[6]==0xFF)
    {
     sensor.sonar[PING_A] = (int)rx_buffer[1];
     sensor.sonar[PING_C] = (int)rx_buffer[2];
     sensor.sonar[PING_E] = (int)rx_buffer[3];
     sensor.sonar[PING_B] = (int)rx_buffer[4];
     sensor.sonar[PING_D] = (int)rx_buffer[5];
     rx_buffer_overflow=1;
     sensor.sonar_complete=1;
     PORTC.0=~PORTC.0;
    }
   }
  }
}

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
hk=1;
sensor.rotary_kanan+=TCNT0;
sensor.rotary_kiri+=TCNT2;
}
```

2.7. File "utama.c"

```
#include <mega32.h>
#include <stdio.h>
#include <spi.h>
#include <math.h>
#include <delay.h>

#include "global_var.h"
```

```
#include "interrupt.h"
#include "fuzzy.h"

void init_sys();
void hitung_koreksi();
void kirim_motor(int kiri,int kanan);

void main(void)
{
 unsigned char arah;
 init_sys();
 delay_ms(300);
 putchar(0);
 delay_ms(50);
 putchar(12);
 t1_on;
 while (1)
     {
      arah=PINC.5;
      arah++;
      if(sensor.sonar_complete)
         {
          sensor.sonar_complete=0;
          fuzzyfikasi(arah);
          check_rule();
          defuzzyfikasi(arah);
         }
       if(hk)
         {
          hk=0;
          hitung_koreksi();
          sensor.rotary_kanan=sensor.rotary_kiri=0;
         }
     };
}


void hitung_koreksi()
{
if((sensor.arus_kanan>200 && sensor.arus_kiri>200) && (sensor.rotary_kanan==0 ||
sensor.rotary_kiri==0))
    {
     kirim_motor(-40,-40);
     while(sensor.rotary_kanan<70);
     kirim_motor(120,120);
    }
else if(sensor.arus_kanan>200 && sensor.rotary_kanan==0)
    {
     kirim_motor(-60,-20);
     while(sensor.rotary_kiri<70);
     kirim_motor(40,40);
     sensor.rotary_kanan=sensor.rotary_kiri=0;
     while(sensor.rotary_kanan<70);
     kirim_motor(120,120);
    }
else if(sensor.arus_kiri>200 && sensor.rotary_kiri==0)
    {
     kirim_motor(-20,-60);
     while(sensor.rotary_kanan<70);
     kirim_motor(40,40);
```

```
    sensor.rotary_kanan=sensor.rotary_kiri=0;
    while(sensor.rotary_kiri<70);
    kirim_motor(120,120);
   }
TCNT0=TCNT2=0;
}

void kirim_motor(int kiri,int kanan)
{
 #asm("cli")
 #asm("nop")
 PORTB.3=0;
 #asm("nop")
 sensor.arus_kiri=spi((signed char)kiri);
 PORTB.3=1;
 #asm("nop")
 #asm("nop")
 PORTB.4=0;
 #asm("nop")
 sensor.arus_kanan=spi((signed char)kanan);
 PORTB.4=1;
 #asm("sei")
}

void init_sys()
{
// 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;
// Func7=Out Func6=In Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=In
// State7=0 State6=T State5=0 State4=1 State3=1 State2=1 State1=1 State0=T
PORTB=0x1E;
DDRB=0xBE;
// 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
PORTC=0x00;
DDRC=0x07;
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=Out Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=0 State0=T
PORTD=0x00;
DDRD=0x02;
// Clock source: T0 pin Falling Edge
// Mode: CTC top=OCR0
// OC0 output: Disconnected
TCCR0=0x0E;
TCNT0=0x00;
OCR0=100;
// Clock source: System Clock
// Clock value: 250,000 kHz
// Mode: Fast PWM top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
TCCR1A=0x03;
```

```
TCCR1B=0x1B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x61;
OCR1AL=0xA8;
OCR1BH=0x00;
OCR1BL=0x00;
// Clock source: TOSC1 pin
// Clock value: PCK2
// Mode: CTC top=OCR2
// OC2 output: Disconnected
ASSR=0x08;
TCCR2=0x09;
TCNT2=0x00;
OCR2=100;
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x92;
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 1 Mbps
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x00;
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;
// SPI Type: Master
// SPI Clock Rate: 4000,000 kHz
// SPI Clock Phase: Cycle Half
// SPI Clock Polarity: Low
// SPI Data Order: MSB First
SPCR=0x50;
SPSR=0x00;
#asm("sei")
}
```

LAMPIRAN

DATA SHEET