SKRIPSI

RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY

Disusun Oleh :

COSMAS ERIC SEPTIAN
09.12.217

PROGRAM STUDI TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2014

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

LEMBAR PERSETUJUAN

RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Persyaratan
Guna Mencapai Gelar Sarjana Teknik

Disusun oleh :
COSMAS ERIC SEPTIAN
NIM. 0912217

Diperiksa dan Disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
M. IbrahimAshari, ST, MT	Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP.P. 1030100358	NIP.Y. 1030800417

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

M. IbrahimAshari, ST, MT
NIP.P. 1030100358

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2014

RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY

Cosmas Eric Septian, NIM 0912217
Dosen Pembimbing : M. Ibrahim Ashari, ST,MT dan
Dr. Eng. Aryuanto Soetedjo, ST,MT

Konsentrasi Teknik Elektronika, Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Nasional Malang
Jln. Raya Karanglo Km. 2 Malang
Email : cosmas.es08@gmail.com

ABSTRAK

Sistem navigasi robot dan sistem kendali merupakan salah satu topik riset yang terus berkembang seiring dengan makin banyaknya teknologi sistem embedded dengan ukuran yang semakin kecil dan kecepatan pemrosesan yang semakin cepat. untuk itu dalam perancangan sistem ini diimplementasikan konsep kendali cerdas dengan menggunakan sistem kendali logika fuzzy yang ditanamkan pada sistem embedded Raspberry Pi.

Dalam perancangan, digunakan dua buah kontroler dengan konsep master slave dimana Raspberry Pi digunakan sebagai Master dan Mikrokontroler ATMega162 digunakan sebagai kontroler Slave. Kedua kontroler berkomunikasi menggunakan sistem USART dengan format paket data menggunakan baudrate 115200 bps. Sensor jarak kanan/kiri digunakan untuk membaca jarak robot terhadap dinding sehingga akan didapat nilai penyimpangan terhadap acuan jarak. Nilai penyimpangan tersebut akan diolah oleh sistem kendali untuk menentukan aksi kendali agar posisi robot dapat mendekati nilai set point sedangkan sensor kamera digunakan untuk melakukan gerakan menghindar terhadap objek (halangan) dengan mengetahui posisi koordinat piksel horizontal objek.

Dari hasil pengujian telah dapat direalisasikan mobile robot yang dapat bergerak otomatis pada suatu trajektori dengan presentase keberhasilan sebesar 83 % dalam menelusuri jalur dengan halangan berupa objek berwarna.

Kata Kunci : *Fuzzy Logic,mobile robot, Raspberry Pi, Sistem Embedded, RaspiCam, Pengolahan Citra*

KATA PENGANTAR

Puji Syukur kehadirat Tuhan Yang Maha Kuasa atas berkat dan rahmat-Nya, sehingga kami selaku penyusun dapat menyelesaikan Laporan Skripsi ini yang berjudul "**RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY**" dapat terselesaikan.

Adapun maksud dan tujuan dari penulisan laporan ini merupakan salah satu syarat untuk dapat menyelesaikan studi dan mendapatkan gelar Sarjana Jurusan Teknik Elektro S-1, Konsentrasi Teknik Elektronika ITN Malang.

Sebagai pihak penyusun penulis menyadari tanpa adanya kemauan dan usaha serta bantuan dari berbagai pihak,maka laporan ini tidak dapat diselesaikan dengan baik. Oleh karena itu , penyusun mengucapkan terima kasih kepada yang terhormat :

1. Ir. Soeparno Djiwo, MT selaku Rektor Institut Teknologi Nasional Malang
2. Ir. Anang Subardi, MT selaku Dekan Fakultas Teknologi Industri Institut Teknologi Nasional Malang.
3. M. Ibrahim Ashari, ST,MT selaku Pembimbing Satu Skripsi dan Ketua Jurusan Teknik Elektro S-1 Institut Teknologi Nasional Malang
4. Dr. Aryuanto Soetedjo, ST, MT selaku Dosen Pembimbing Dua Skripsi.
5. Sahabat-sahabat dan rekan-rekan yang tidak dapat disebutkan satu persatu, yang telah membantu baik dari segi teknis maupun dukungan moral dalam terselesaikanya skripsi ini.

Usaha telah kami lakukan semaksimal mungkin, namun jika ada kekurangan dan kesalahan dalam penyusunan, kami mohon saran dan kritik yang sifatnya membangun. Begitu juga sangat kami perlukan untuk menambah kesempurnaan laporan ini dan dapat bermanfaat bagi rekan-rekan mahasiswa pada khususnya dan pembaca pada umumnya.

Malang, Agustus 2014

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB I
PENDAHULUAN

1.1 Latar belakang

Perkembangan teknologi robot telah berkembang dengan pesat baik dari segi teknologi maupun penggunaanya dalam berbagai bidang. Sebagai contoh penggunaan teknologi otomatisasi dalam bidang Industri maupun dalam bidang transportasi (*Autonomous vehicle*). Salah satu hal yang menjadi perhatian akhir-akhir ini adalah bagaimana merancang suatu metode kendali dalam robot yang memiliki tingkat kecerdasan dan kehandalan serta kepresisian yang tinggi. Metode yang bisa digunakan adalah memasukkan kendali cerdas ke dalam suatu sistem kontroler robot.

Mobile Robot adalah suatu robot yang memiliki aktuator baik motor dengan konsep beroda maupun aktuator motor yang disusun menyerupai kaki sehingga dapat bergerak dari suatu tempat ke tempat yang lain untuk melakukan tugas tertentu. Salah satu fokus dalam merancang dan membuat suatu robot mobile adalah sistem navigasi. Perancangan navigasi suatu robot mobile memiliki beberapa acuan yang digunakan yaitu : Kontur (*bentuk*), warna dari lingkungan dan beberapa parameter lain yang terkadang tidak dapat terprediksi sebelumnya. Untuk itu dibutuhkan suatu sistem kendali cerdas yang dapat digunakan untuk mengantisipasi *unpredictable environment* (*lingkungan yang tidak terprediksi*) dengan memberikan konsep kecerdasan buatan (*Artificial Intelegence*).

Metode logika fuzzy memberikan solusi terhadap adanya beberapa input parameter yang tidak menentu. Nilai input parameter yang masuk dipetakan dalam proses fuzzyfikasi dengan derajat keanggotaan (*grade of memberhip*) masing-masing. Nilai yang telah dipetakan kemudian akan dievaluasi dengan *rule* untuk menentukan nilai output pada kontroler yang diubah ke dalam nilai PWM Motor kanan dan kiri.

Pada beberapa rancangan navigasi mobile robot yang ada sekarang ini hanya memiliki satu jenis sensor sehingga robot tidak berjalan seperti yang diharapkan atau hanya dapat berjalan pada lingkungan tertentu saja. oleh karena itu dibutuhkan sistem *fusion* sensor yang merupakan kombinasi dari dua atau

lebih sensor untuk menghadapi beberapa kondisi lingkungan yang tidak terprediksi sebelumnya.

Dari beberapa kondisi tersebut di atas, penulis ingin merancang suatu mobile robot yang dapat melakukan navigasi (*pergerakan*) pada lingkungan yang tak terprediksi dengan metode *logika fuzzy*. Untuk itu pada perancangan sistem, akan dibuat sistem kombinasi dari sesor ultrasonic dan sensor kamera. Harapannya sistem ini dapat dikembangkan dan diterapkan pada sistem keamanan berkendara untuk mengurangi dampak tubrukan (*collision*) atau digunakan untuk konsep mobil otomatis masa depan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diutarakan diatas, maka dapat disimpulkan permasalahan yang dituangkan dalam karya ilmiah ini, yaitu :

1. Bagaimana merancang suatu robot yang dapat melakukan navigasi pada suatu lingkungan yang tidak diketahui secara otomatis.
2. Bagaimana merancang kombinasi sensor ultrasonic dan sensor kamera untuk dijadikan input parameter kendali fuzzy.
3. Bagaimana menerapkan suatu kendali fuzzy kedalam Sistem Embedded Raspberry Pi
4. Bagaimana menerapkan komunikasi Raspberry Pi dengan Atmega162 dengan Konsep Master-Slave

1.3 Tujuan

Perancangan dan pembuatan mobile robot dengan kendali fuzzy ini bertujuan untuk merancang suatu sistem kendali cerdas yang dapat mengenali parameter-parameter lingkungan, sebagai acuan dalam melakukan navigasi secara otomatis. Sehingga nantinya sistem ini dapat membantu dalam pengembangan riset laboratorium Robotika ITN Malang.

1.4 Batasan Masalah

Agar perancangan dan pembuatan mobile robot ini dapat sesuai dengan tujuan yang diharapkan dan tetap fokus pada konsep awal, maka diperlukan beberapa batasan-batasan diantaranya adalah :

1. Parameter lingkungan untuk bernavigasi adalah dinding yang memiliki permukaan rata dan objek (obstacle) berwarna.
2. Sensor kamera yang digunakan adalah modul kamera yang dibuat khusus untuk raspberry pi dengan koneksi CSI (*Camera Serial Interface*).
3. Raspberry yang digunakan memiliki kecepatan prosessor 700 Mhz
4. Tidak membahas secara detail tentang pengolahan citra digital.
5. Konstruksi mekanik menggunakan konsep DDMR (*Differential Drive Mobile Robot*) dan hanya dapat dijalankan pada jalur yang rata.
6. Tidak membahas efek dinamik pada robot

1.5 Metodologi Masalah

Metode yang digunakan dalam penyususan skripsi ini adalah:

1. Kajian Literatur

   Pengumpulan data dan informasi yang dilakukan dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang berhubungan dengan permasalahan pada perancangan alat.
2. Perancangan Mekanik

   Pembuatan design dan pencarian bahan untuk pembuatan mekanik serta melakukan pengujian beban pada motor *gearbox*
3. Perancangan Sistem Elektronika

   Pembuatan design rangakaian elektronika seperti : rangkaian driver motor L298, perancangan dan pembuatan rangkain regulator 5 volt dan 11 volt, perancangan rangkaian minimum sistem Atmega162, perancangan logic level Konverter.
4. Pembuatan Hardware

   Pembuatan rangkaian dari hasil perencanaan sistem yang meliputi :
   1) Pembuatan rangkaian Driver Motor L298
   2) Pembuatan rangkaian minimum sistem ATMega162

3) Pembuatan sistem komunikasi antara Mikrokontroler Atmega162 dan rangkaian level konverter (*CMOS to TTL*).
4) Proses pengkabelan dan penempatan keseluruhan rangkaian pada rangka robot.

5. Pembuatan Algoritma Program

   Program yang dibuat menggunakan bahasa pemrograman C dengan menggunakan metode kendali logika fuzzy yang akan ditanamkan Raspberry Pi sedangkan untuk program pengolahan citra diupayakan menggunakan library yang sudah ada sehingga dapat mempersingkat waktu.

6. Pengujian Sistem

   Proses ujicoba rangkaian dan keseluran sistem untuk mengetahui adanya kesalahan agar sistem sesuai dengan konsep yang telah dirancang sebelumnya.

7. Pelaporan hasil pengujian dan kesimpulan.

1.6 Sistematika Penulisan

Untuk mempermudah dan memahami pembahasan penulisan skripsi ini, sistematika penulisan disusun sebagai berikut:

BAB I : PENDAHULUAN

Berisi tentang latar belakang rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Membahas tentang dasar teori mengenai permasalahan yang berhubungan dengan penelitian.

BAB III : PERANCANGAN DAN ANALISA

Bab ini membahas tentang perancangan dan analisa sensor ultrasonic, analisa sistem kamera embedded raspberry pi, analisa level converter, analisa rangkaian power supply dan mikrokontroler Atmega162.

BAB VI : PEMBUATAN DAN PENGUJIAN

Berisi tentang pembahasan langkah-langkah pembuatan alat serta pengujian terhadap alat tersebut.

BAB V : PENUTUP

Berisi tentang semua kesimpulan yang berhubungan dengan penulisan skripsi, dan saran yang digunakan sebagai pertimbangan dalam pengembangan program selanjutnya.

BAB II
DASAR TEORI

2.1 Logika Fuzzy

Istilah Fuzzy Set pertama kali diperkenalkan oleh Prof. Lotfi Zadeh pada tahun 1965 dalam papernya yang fenomenal . Dalam Paper tersebut dipaparkan ide dasar fuzzy set yang meliputi *inclusion*, *union*, *intersection*, *complement*, *relation*, dan *convexity*. Ide tersebut terus dimatangkan oleh zadeh dalam beberapa *paper*nya yang lain.

Fuzzy logic digunakan untuk menyatakan hukum operasional dari suatu sistem dengan menggunakan ungkapan (*variable Linguistic*), bukan dengan persamaan matematis. Banyak sistem yang terlalu kompleks untuk dimodelkan secara akurat, meskipun dengan persamaan matematis yang kompleks. Dalam kasus seperti itu, ungkapan bahasa yang digunakan dalam Fuzzy logic dapat membantu mendefinisikan karakteristik operasional sistem dengan lebih baik. Ungkapan bahasa untuk karakteristik sistem biasanya dinyatakan dalam bentuk implikasi logika, misalnya aturan *if-then* atau jika-maka.

Dalam teori himpunan fuzzy tidak hanya memiliki dua kemungkinan (*true-false*) layaknya dalam logika boolean namun logika fuzzy menerapkan derajat keanggotaan untuk menentukan sifat keanggotaanya. Derajat keanggotaan berkisar antara 0–1. Fungsi yang menetapkan nilai ini dinamakan fungsi keanggotaan yang disertakan dalam himpunan fuzzy.

2.1.1 Struktur dasar Pengendalian Logika Fuzzy

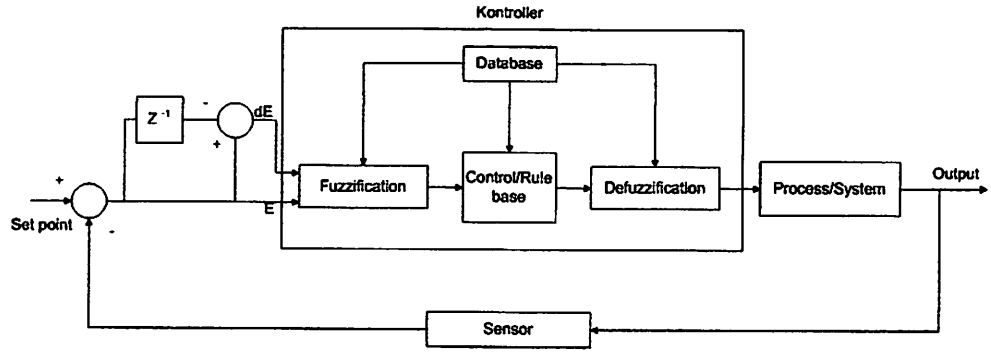
Metode—metode perancangan pengendalian klasik (misalnya : *Nyquist*, *Bode*, *root locus*) pada umumnya didasarkan pada asumsi bahwa proses yang dikendalikan adalah *linier* dan *stasioner*. Sedangkan kebanyakan proses yang ada adalah sistem yang kompleks, non linier dan mudah dipengaruhi oleh faktor-faktor gangguan sekitar. Proses-proses tersebut pada kenyataannya bisa dikendalikan secara manual dengan hasil yang cukup baik. Operator kendali tersebut biasanya adalah tenaga terampil yang mengandalkan pengalaman praktis tanpa dilatarbelakangi teori-teori pengandalian yang rumit. Hal ini karena operator tersebut mengendalikan proses dengan dasar logika yang juga non-linier dan

kompleks yang dibangun oleh pengalaman panjang sehingga pengendalian yang dilakukannya sepenuhnya bersifat intuitif.

Untuk merancang sistem pengendalian otomatis bagi proses-proses tersebut yang mampu menerjemahkan pengetahuan dan aturan-aturan fuzzy, maka diperlukan teori logika fuzzy sebagai salah satu alternatif. Secara umum pengendali logika fuzzy memiliki kemampuan sebagai berikut :

1. Beroperasi tanpa campur tangan manusia secara langsung, tetapi memiliki efektifitas yang sama dengan pengedali manusia.
2. Mempu menangani sistem-sistem yang kompleks, non-linier dan tidak stasioner.
3. Memenuhi spesifikasi operasional dan kriteria kinerja.
4. Strukturnya sederhana, kuat dan beroperasi *real-time*

Pengendali yang ditunjukkan pada Gambar 2.1 merupakan pengendali fuzzy PD (*Proportional Derrivative*). Dengan pengolahan dua sinyal masukan ke pengendali yaitu sinyal kesalahan atau error dan sinyal perubahan kesalahan atau delta error. Sinyal E didapatkan dari pengurangan keluaran proses terhadap set point, sedangkan sinyal dE didapatkan dari pengurangan sinyal error saat ini *E(k)* dengan sinyal error yang terjadi sebelumnya *E(k-1)*. Kedua sinyal tersebut diolah oleh pengendali logika fuzzy . Dengan demikian terdapat dua masukan ke pengendali dan satu keluaran dari pengendali tersebut. Jadi aturan yang akan digunakan nantinya kan meliputi tiga komponen yaitu dua komponen *antecedent* dan satu komponen *consequent*.



Gambar 2.1 Struktur dasar sistem kontrol Logika Fuzzy[1]

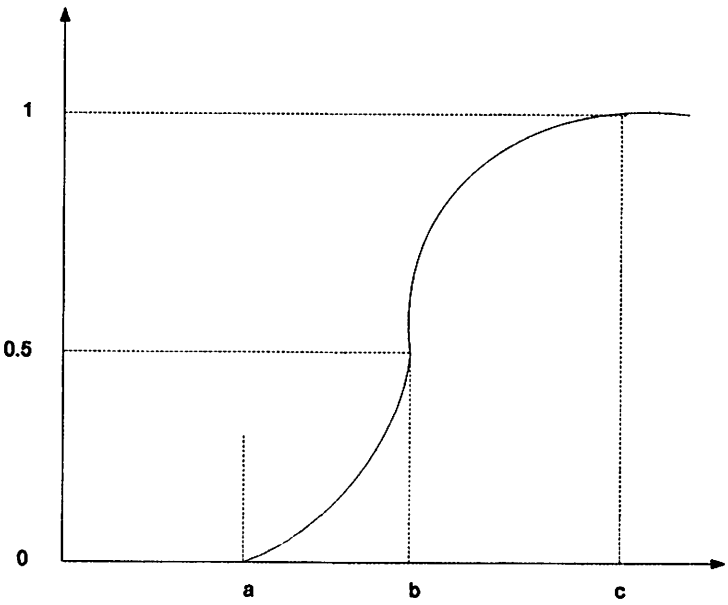
Penjelasan dari masing-masing gambar tersebut adalah :

1. *Fuzzyfikasi* yaitu suatu proses untuk mendapatkan besarnya derajat keanggotaan masukan yang berupa suatu variabel numerik non-fuzzy (elemen himpunan) dalam suatu himpunan fuzzy. Penentuan keanggotaan suatu himpunan fuzzy tidak dibatasi oleh aturan-aturan tertentu. Ada tiga macam fungsi keanggotaan yang dinyatakan dalam fungsi keanggotaan S, π, dan T (Triangular).
    1. S –Function

       Definisi S-function adalah sebagai berikut :

$$S(u;a,b,c)=\begin{cases}0 & u<a\\ 2\left(\dfrac{u-a}{c-a}\right) & a\le u\le b\\ 1-2\left(\dfrac{u-a}{c-a}\right) & b\le u\le c\\ 0 & u>c\end{cases} \quad \text{.....(2-1)}$$

       Bentuk diagramatik S-function ditunjukkan pada Grafik 2.1 Berikut :



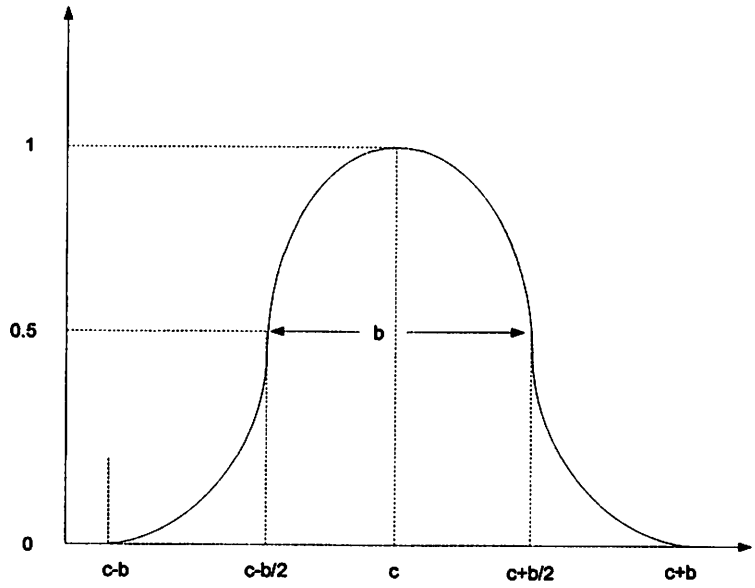
Grafik 2.1 Grafik S-function

2. Π-function

Definisi π function adalah sebagai berikut :

$$\pi(u;b,c)=\begin{cases} S(u;c-b,c-\frac{b}{2},c & u\le c \\ 1-S(u;c,c+\frac{b}{2},c+b & u\le c \end{cases} \quad \text{.....(2-2)}$$

Bentuk Diagramatik π-function ditunjukkan pada Grafik 2.2 berikut :



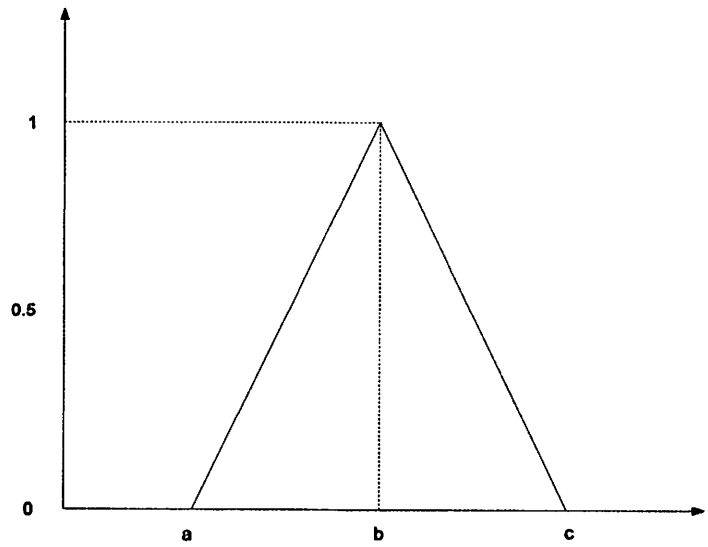
Grafik 2.2 Grafik π-function

3. Triangular function

Derajat keanggotaan *Triangular-function* didefinisikan sebagai berikut :

$$T(u;a,b,c)=\begin{cases} 0 & u<a \\ \frac{u-a}{b-a} & a\le u\le b \\ \frac{c-u}{c-b} & b\le u\le c \\ 0 & u>c \end{cases} \quad \text{.....(2-3)}$$

Bentuk diagramatik triangular-function ditunjukkan pada Grafik 2.3 Berikut :



Grafik 2.3 Diagram T-function

2. Penyusunan aturan /rule pengendalian

Pada umumnya aturan-aturan fuzzy dinyatakan dalam bentuk "if-then" yang merupakan inti dari relasi fuzzy. Relasi fuzzy dinyatakan dengan R juga disebut implikasi fuzzy. Relasi fuzzy dalam pengetahuan dasar dapat didefinisikan sebagai himpunan pada implikasi fuzzy.

Aturan Dasar fuzzy adalah dalam bentuk umum :

$R^{(l)}$: IF x_1 is F_1^l AND .... AND.... X_n is F_n^l , THEN y is G^l

Dimana F_1^l dan G^l adalah himpunan fuzzy masing-masing di $U_i \subset R$ dan $\underline{x} = (x^1,....,x^n)^T \in U_1 X....X U_n$ dan $y \in V$ adalah variabel linguistik. Untuk mendapatkan aturan IF-THEN di atas ada dua cara utama yaitu : menanyakan ke operator manusia yang dengan cara manual telah mampu mengendalikan sistem tersebut. Cara ini dikenal dengan istilah human expert, dan dengan menggunakan algoritma pelatihan berdasarkan data-data masukan dan keluaran. Cara pertama tersebut merupakan cara langsung untuk mendapatkan aturan, tetapi operatornya mungkin akan sulit untuk mengatakan seluruh aturannya karena seringkali terjadi bahwa operator mengendalikan sistem atas dasar perasaan semata dan refleks yang sulit dijelaskan. Karena keterbatasan-keterbatasan tersebut maka

banyak perekayasa menawarkan ide untuk menggunakan data keluaran dan masukan sebagai dasar penyusunan aturan secara otomatis.

3. Defuzzyfikasi

Defuzzifikasi didefiniskan sebagai proses pengubahan besaran fuzzy (variabel linguistik) yang disajikan dalam bentuk himpunan-himpunan fuzzy keluaran dengan fungsi keanggotaannya untuk mendapatkan kembali bentuk tegasnya (*crisp*). Dalam sistem kontrol secara umum terdapat hubungan sebab-akibat yang spesifik antara masukan dan keluaran sistem tersebut. Karakteristik hubungan inilah yang membedakan atara sistem yang satu dengan sistem yang lain. Pengendali yang menggunakan logika fuzzy juga membutuhkan spesifikasi hubungan antara masukan dan keluaran yang secara umum dinyatakan dengan :

IF (A_1) THEN (B_1)

.....

IF(A_n) THEN (B_n)

A_1,...., A_n adalah *antecedent* yaitu masukan yang defuzzifikasi, sedangkan B_1,....B_n adalah *consequent*, yaitu aksi pengendalian (keluaran). Hubungan antara *antecedent* dan *consequent* disebut aturan (rule), dan anatara satu rule dengan yang lain tidak terdapat hubungan sebab-akibat. Proses untuk mendapatkan aksi keluaran dari suatu masukan dengan mengikuti aturan-aturan yang telah ditetapkan disebut *inference* atau *reasoning* (pengambilan keputusan).

Keputusan yang dihasilkan dari proses penalaran ini masih dalam bentuk fuzzy yaitu berupa derajat keanggotaan keluaran. Hasil ini harus diubah kembali menjadi variabel numerik non-fuzzy melalui proses defuzzifikasi. Dua metode defuzzifikasi yang umum digunakan adalah :

1. Maximum of Mean

Metode ini didefinisikan sebagai :

$$V_o = \sum_{l=1}^{J} \frac{v_l}{J} \qquad \text{(2-4)}$$

$$v_j = \max_{v \in V} \mu_v(v) \quad \text{.....(2-5)}$$

v_o = nilai keluaran

J = jumlah harga maksimum

v_i = nilai keluaran maksimum ke-j

$\mu_v(v)$ = derajat keanggotaan elemen-elemen pada fuzzy set v

V = semesta pembicaraan (*universe of discourse*)

2. *Center of Area*

Metode ini sering disebut juga sebagai metode COG (*center of Gravity*) yang didefinisikan sebagai berikut :

$$v_o = \frac{\sum_{k=1}^{m} v_k \mu_v(v_k)}{\sum_{k=1}^{m} \mu_v(v_k)} \quad \text{.....(2-6)}$$

v_o = nilai keluaran

m = tingkat kuantisasi

v_k = elemen ke-k

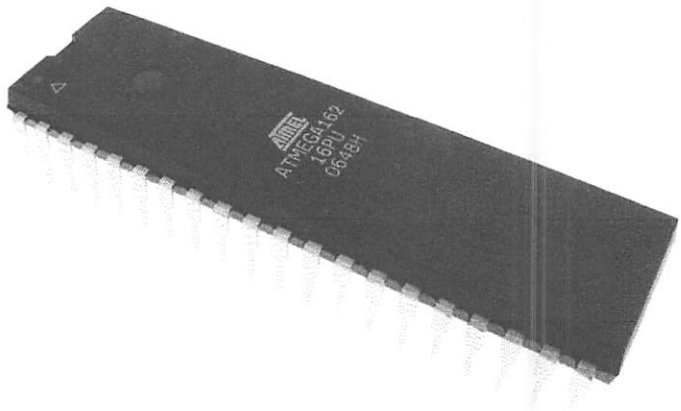
$\mu(v_k)$ = derajat keanggotaan elemen-elemen pada fuzzy set v

V = semesta pembicaraan (*Universe of Discourse*)

2.2 Mikrokontroler ATMega162

Secara definitif mikrokontroler adalah sebuah *chip* prosessor yang telah dilengkapi dengan beberapa fitur yang ada pada komputer personal seperti RAM, Clock Internal, *Interrupt* Timer, Memori *Flash*, I/O serta EEPROM Internal. Mikrokontroler dapat digunakan untuk beberapa aplikasi *embedded* yang dapat kita temui pada aplikasi rumah tangga, Industri maupun robotika. Salah satu mikrokontroler yang banyak dipakai adalah mikrokontroler tipe AVR. AVR adalah mikrokontroler tipe RISC (*Reduce in Set Computing*) 8 bit dengan mengadopsi arsitektur Harvard dimana arsitektur ini memiliki konstruksi dimana memori program dan memori data diletakkan secara terpisah. AVR memiliki keunggulan bila dibandingkan dengan mikrokontroler yang lain, diantaranya

adalah memiliki kecepatan eksekusi program yang lebih cepat karena sebagaian besar instruksi dieksekusi dalam 1 siklus clock, lebih cepat dibandingkan dengan mikrokontroler MCS51 yang memiliki arsitektur CISC (*Complex Instruction in Set Compute*) dimana mikrokontroler MCS51 membutuhkan 12 siklus clock untuk mengeksekusi 1 instruksi program.



Gambar 2.2 *Chip* Mikrokontroler ATMega162

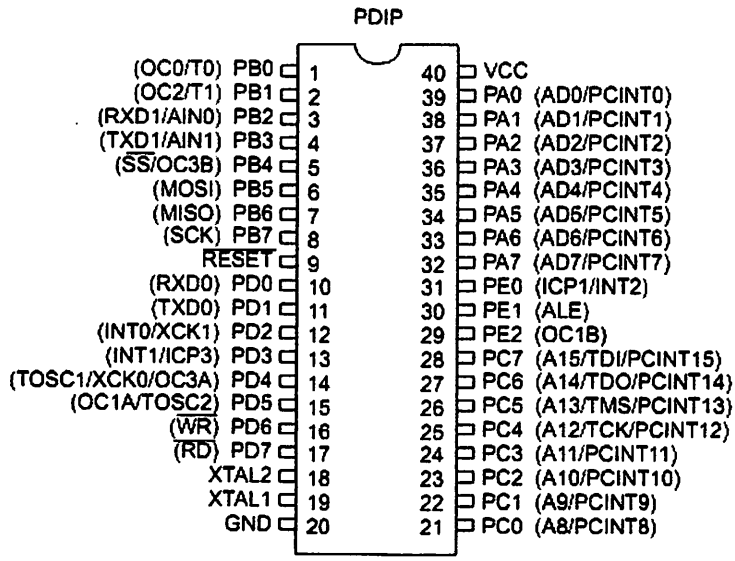
(Sumber: : http://www.nikushop.com/images/stories/virtuemart/product/ATMega162.jpg)

Setiap tipe mikrokontroler memiliki fitur dan memori yang berbeda. Secara garis besar mikrokontroler tipe ATMega162 memiliki fitur sebagai berikut [3] :

1. Arsitektur 8 bit dengan konsumsi daya yang rendah (*Low Power*).
2. Memiliki 5 I/O yang terdiri dari (PORTA, PORTB, PORTC, PORTD dan PORTE)
3. 16 KByte Memori *Flash* untuk penyimpanan program
4. 512 Bytes EEPROM (*Electriccaly Eraseable Programable Read Only Memory*)
5. 1 Kbyte SRAM (*Static RAM*) Internal
6. Dua Timer 8-bit dengan *prescaller* terpisah dan mode *Compare*
7. Dua timer 16-bit dengan *prescaler* terpisah, mode *Compare* dan mode *capture.*
8. 6 PWM (*Pulse Width Modulation*) Channel

9. Dua buah USART (*Universal Synchronous Asynchronous Receiver Transmitter*)
10. Master/Slave SPI (*Serial Peripheral Interfaces*).
11. Koneksi I2C (*Inter Integrated Circuit*)

2.2.1 Konfigurasi Pinout Atmega 162



Gambar 2.3 Konfigurasi Pinout Mikrokontroler ATMega162

Konfigurasi Pinout mikrokontroler ATMega162 kemasan DIP (*Dual in Line Package*) ditunjukkan pada Gambar 2.3. dari gambar tersebut dapat dilihat bahwa masing-masing I/O memiliki 2 fungsi yaitu sebagai General Purpose Bi-directional I/O dan fungsi khusus yang diatur pada register. Kelima port (PORTA, PORTB, PORTC, PORTD dan PORTE) dapat digunakan untuk fungsi input output bi-directional dengan melakukan konfigurasi register DDRxn, PORTxn, dan PINxn. Huruf x mewakili nama huruf dari port yang digunakan sedangkan huruf 'n' mewakili bit yang akan digunakan. Register DDR (*data direction register*) merupakan register yang digunakan untuk menetukan arah (*direction*) dari port apakah digunakan sebagai input atau output. Bila Px bernilai 1 maka Px akan berfungsi sebagai pin output, sedangkan jika diberi nilai 0 maka Px akan difungsikan sebagai inputan. Pada saat Px difungsikan sebagai Pin input maka register PORTxn diset 1 untuk mengaktifkan internal pullup yang artinya pada kondisi normal tanpa ada input aktif ke Px, maka Px akan bernilai high yang artinya agar Px aktif maka harus diberi input low (*aktif low*). Untuk mematikan *internal pullup* maka register PORTxn harus diberi nilai 0. Bila PORTxn diberi nilai 1 pada saat PORTxn dikonfigurasi sebagai output (DDRxn = 1) maka pin

port akan mengeluarkan nilai *high* (TTL 5 V). bila PORTxn diberi nilai 0 pada saat PORTxn dikonfigurasi sebagai output (DDRxn = 0) maka pin akan mengeluarkan logika *low* (0 – 0.7 V). Saat mengubah kondisi port dari kondisi *tri-state* (DDxn=0, PORTxn=0) ke kondisi output *high* (DDxn=1, PORTxn=1) maka harus ada kondisi peralihan apakah itu kondisi *pull-up enabled* (DDxn=0, PORTxn=1) atau kondisi output *low* (DDxn=1, PORTxn=0).

Biasanya kondisi *pull-up enabled* dapat diterima sepenuhnya, selama lingkungan impedansi tinggi tidak memperhatikan perbedaan antara sebuah *strong high driver* dengan sebuah *pullup*. Jika ini bukan menjadi suatu masalah, maka bit PUD pada register SFIOR (*Special Function Input Output Register*) dapat diset 1 untuk mematikan semua *pull-up* dalam sebuah port. Peralihan dari kondisi input dengan *pull-up* ke kondisi *output* low juga menimbulkan masalah yang sama. Kita harus menggunakan kondisi tri-state (DDxn=0, PORTxn=0) atau kondisi *output* high (DDxn=1, PORTxn=0) sebagai kondisi transisi.

Table 2.1 Konfigurasi Port Mikrokontroler

DDxn	PORTxn	PUD (in SFIOR)	I/O	Pull-up	Keterangan
0	0	X	In	No	Tri-state (Hi-Z)
0	1	0	In	Yes	Pxn will source current if ext.pulled low
0	1	1	In	No	Tri-state (Hi-Z)
1	0	X	Out	No	Output Low (*Sink*)
1	1	X	Out	No	Output High (*Source*)

Selain bisa digunakan untuk *bi-directional* I/O, 5 port pada mikrokontroler ATMega162 juga memiliki fungsi khusus yang dapat diaktifkan melalui register. Berikut pejelasan dari masing-masing port :

1. PORTA (A0 – A7)

   Pada port ini memiliki fungsi sebagai interface memori SRAM Eksternal dan fungsi untuk rutin Interrupt yang akan terjadi jika terjadi perubahan state (*pin change Interrupt*) pada pin PORTA.

Tabel 2.2 Fungsi Alternatif PORTA

Port Pin	Fungsi Alternatif
PA7	AD7 (Eksternal Memory Interfaces address dan data bit 7) PCINT7 *(Pin Change Interrupt 7)*
PA6	AD6 (Eksternal Memory Interfaces address dan data bit 6) PCINT6 (*Pin Change Interrupt 6*)
PA5	AD5 (Eksternal Memory Interfaces address dan data bit 5) PCINT5 *(Pin Change Interrupt 5*)
PA4	AD4 (Eksternal Memory Interfaces address dan data bit 4) PCINT4 *(Pin Change Interrupt 4)*
PA3	AD3 (Eksternal Memory Interfaces address dan data bit 3) PCINT3 (*Pin Change Interrupt 3*)
PA2	AD2 (Eksternal Memory Interfaces address dan data bit 2) PCINT2 *(Pin Change Interrupt 2*)
PA1	AD1 (Eksternal Memory Interfaces address dan data bit 1) PCINT1 *(Pin Change Interrupt 1)*
PA0	AD0 (Eksternal Memory Interfaces address dan data bit 0) PCINT0 (*Pin Change Interrupt 0*)

2. PORTB (B0 – B7)

   Port B merupakan port yang dapat digunakan sebagai bi-directional input/output dengan fitur internal pullup. Selain itu Port B juga memiliki fitur alternatif. Fungsi alternatif pada Port B dijelaskan pada Tabel 2.3 berikut :

Tabel 2.3 Fungsi Alternatif PORTB

Port Pin	Fungsi Alternatif
PB7	SCK (*SPI Bus Serial Clock*)
PB6	MISO (*SPI Bus Master Input/Slave Output*)
PB5	MOSI (*SPI Bus Master Output/Slave Input*)
PB4	SS (*SPI Slave Select Input*) OC3B (*Timer/Counter3 Output Compare Match Output*)
PB3	AIN1 (*Analog Comparator Negative Input*) TXD1 (*USART1 Output Pin*)
PB2	AIN0 (*Analog Comparator Positive Input*) OC2 (*Timer/Counter2 Output Compare Match Output*)
PB1	T1 (*Timer/Counter1 Eksternal Counter Input*) OC2 (*Timer/Counter2 Output Compare Match Output*)
PB0	T0 (*Timer/Counter0 External Counter Input*) OC0 (*Timer/Counter0 Output Compare Match Output*) clkI/O (*Divided System Clock*)

- *SCK* – Port B, Bit 7

  SCK : Merupakan Pin yang digunakan untuk Master Clock Output, jika *SPI* dikofigurasikan sebagai *SPI-Slave*, maka pin ini dikonfigurasikan sebagai Input melalui *register DDRB7*
- *MISO* – Port B,Bit 6

  MISO : *Master Data Input*, merupakan output pin untuk data *Slave* pada *channel SPI*. Saat *SPI* difungsikan sebagai *Master*, pin ini dikonfigurasikan sebagai Input melalui register *DDRB6*.
- *MOSI* – Port B, Bit 5

  MOSI : *Master data Output*, merupakan *Input data Slave* untuk pada *Channel SPI*, Saat *SPI* difungsikan sebagai *Slave*, pin ini dikonfigurasikan sebagai input melalui Register *DDRB5*
- *SS/OC3B* – Port B, Bit 4

SS : Slave Select Input, Saat *SPI* difungsikan sebagai *Slave*, pin ini dikonfigurasikan sebagai input melalui register *DDRB4*. *OC3B* pin juga digunakan sebagai output pin untuk *PWM*

- AIN1/TXD1 – Port B, Bit 3

  AIN1, Pin *Analog Komparator Input Negatif*

  TXD1, Pin yang digunakan untuk *transmit* data (*TX*) dari USART1.
- AIN0/RXD1 – Port B, Bit 2

  AIN0, Pin Analog Komparator Input Positif

  RXD1, Pin untuk menerima data (Receiver/RX) USART1
- *T1/OC2* – Port B, Bit 1

  T1, Timer/Counter1 pin yang difungsikan untuk mode counter

  OC2, *Output Compare Match* : pin ini dapat digunakan sebagai eksternal output untuk *Timer/Counter2* pada mode *Compare Match.*
- *T0/OC0* – Port B, Bit 0

  T0, Timer/Counter0 pin yang difungsika untuk mode counter.

  OC0, *Output Compare Match* : pin ini dapat digunakan sebagai eksternal output untuk *Timer/Counter0* pada mode *compare match.*

3. PORTC (C0 – C7)

   Port C merupakan port yang dapat diguakan sebagai bi-directional input/output dengan fitur internal pullup. Selain itu Port B juga memiliki fitur alternatif. Fungsi Alternatif pada *PORTC* dapat dilihat pada Tabel 2.4 Berikut :

Tabel 2.4 Fungsi Khusus pada PORTC ATMega162

Port Pin	Fungsi Alternatif
PC7	A15 (*External memory Interface address bit 15*) TDI (*JTAG Test Data Input*) PCINT15 (*Pin Change Interrupt* 15)
PC6	A14 (*External memory Interface address bit 14*) TD0 (*JTAG Test Data Output*) PCINT14 (*Pin Change Interrupt* 14)
PC5	A13 (*External memory Interface address bit 13*) TMS (*JTAG Test Mode Select*) PCINT13 (*Pin Change Interrupt* 13)
PC4	A12 (*External memory Interface address bit 12*) TCK (*JTAG Test Clock*) PCINT12 (*Pin Change Interrupt* 12)
PC3	A11 (*External memory Interface address bit 11*) PCINT11 (*Pin Change Interrupt* 11)
PC2	A10 (*External memory Interface address bit 10*) PCINT10 (*Pin Change Interrupt* 10)
PC1	A9 (*External memory Interface address bit 9*) PCINT9 (*Pin Change Interrupt* 9)
PC0	A8 (*External memory Interface address bit 8*) PCINT8 (*Pin Change Interrupt* 8)

- *A15/TDI/PCINT15* – Port C, Bit 7

  A15, Eksternal memori interfaces address bit 15

  TDI, JTAG Test Data In : Input data Serial yang akan digeser pada *Instruction Register* atau *data Register.*

  PCINT15, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.

- *A14/TD0/PCINT14* – Port C, Bit 6

  A14, Pin untuk *Eksternal Memori Interfaces address bit* 14

  TDO, Pin JTAG Test Data Output untuk output data serial dari *Instruction register* atau *data register.*

  PCINT14, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.

- *A13/TMS/PCINT13* – Port C, Bit 5

  A13, Pin untuk *interfaces Eksternal memori* pada alamat Bit 13.

  TMS, pin *JTAG* untuk *Test Mode Select.*

  PCINT11, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.
- *A12/TCK/PCINT12* – Port C, Bit 4

  A12, Pin untuk *interfaces Eksternal memori* pada alamat Bit 12.

  TCK, pin *JTAG* untuk *Test Clock.*

  PCINT12, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.
- *A11/PCINT11* – Port C, Bit 3

  A11, Pin untuk *interfaces Eksternal memori* pada alamat Bit 11.

  PCINT11, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.
- *A10/PCINT10* – Port C, Bit 2

  A10, Pin untuk *interfaces Eksternal memori* pada alamat Bit 10.

  PCINT10, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.
- *A9/PCINT9* – Port C, Bit 1

  A9, Pin untuk *interfaces Eksternal memori* pada alamat Bit 9.

  PCINT9, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.
- *A8/PCINT8* – Port C, Bit 0

  A8, Pin untuk *interfaces Eksternal memori* pada alamat Bit 8.

  PCINT8, Pin ini difungsikan untuk mengaktifkan *interrupt* ketika terjadi perubahan status pin.

4. PORTD (D0 – D7)

   Port D adalah *bi-directional* Port yang dapat digunakan untuk Input maupun output dengan fitur internal *pullup*. Port D juga memiliki fungsi khusus yang dapat dilihat pada tabel 2.5 di bawah ini :

Tabel 2.5 Fungsi Alternatif PORTD

Port Pin	Fungsi Alternatif
PD7	RD (*Read Strobe to External memory*)
PD6	WR (*Write Strobe to external memory*)
PD5	TOSC2 (Timer Oscillator Pin 2) OC1A (*Timer/Counter1 Output Compare A Match Output*)
PD4	TOSC1 (*Timer Oscillator Pin 1*) XCK0 (*USART0 External Clock Input/Output*) OC3A (*Timer/Counter3 Output Compare A Match Output*)
PD3	INT1 (External Interrupt 1 input) ICP3 (*Timer/Counter3 Input Capture Pin*)
PD2	INT0 (*External Interrupt 0 Input*) XCK1 (*USART1 External Clock Input/Output*)
PD1	TXD0 (*USART0 Output Pin*)
PD0	RXD0 (*USART0 Input Pin*)

- RD –Port D, Bit 7

  RD adalah pin yang digunakan untuk melakukan interface dengan penyimpanan data eksternal sebagai pin control untuk baca Data
- WR – Port D, Bit 6

  WR adalah pin yang digunakan untuk melakukan interface dengan penyimpanan data eksternal sebagai pin control untuk penulisan data.
- TOSC2/OC1A – Port D, Bit 5

  TOSC2, Pin yang digunakan oleh Timer2 untuk mengaktifkan *asynchronous* clock dari Timer/Counter2

  OC1A, (*Output Compare Match A*), pin yang digunakan sebagai output dari Timer/Counter1 pada mode Compare A Output.
- TOSC1/XCK0/OC3A – Port D, Bit 4

  TOSC1, Pin yang digunakan oleh Timer1 untuk mengaktifkan asynchronous clock dari Timer/Counter1

  XCK0, Pin yang digunakan sebagai sumber Clock Eksternal pada USART0 jika menggunakan *Sychronous* Mode.

OC3A (*Output Compare Match A*), pin yang digunakan untuk output pada saat *mode Compare* diaktifkan.

- INT1/ICP3 – Port D, Bit 3

  INT1 (*External Interrupt Source 0*), pin eksternal interupt 1

  ICP3 (*Input Capture Pin*) pin yang berfungsi untuk melakukan capture pulsa untuk timer/Counter3
- INT0/XCK1 – Port D, Bit 2

  INT0, (*External Interupt Source 1*), pin eksternal interupt 0

  XCK1, pin yang digunakan sebagai eksternal *clock* USART1 pada saat USART1 dikofigurasikan pada mode Synchronous.
- TXD0 – Port D, Bit 1

  TXD0 (*Transmit Data*) pin yang digunakan untuk mengirimkan data serial jika USART0 diaktifkan.
- RXD0 – Port D, Bit 0

  RXD0 (*Receiver Data*) pin yang digunakan untuk menerima data serial jika USART0 diaktifkan.

5. PORTE (E0 – E2)

   Port E adalah port yang dapat digunakan sebagai *Bi-directional* Input/Output yang dapat pula digunakan untuk fungsi khusus yaitu :

Tabel 2.6 Fungsi Khusus pada Port E

Port Pin	Fungsi Khusus
PE2	OC1B (*Timer/Counter1 Output Compare B Match Output*)
PE1	ALE (*Address Latch Enable ke Eksternal memory*)
PE0	ICP1 (*Timer/Counter1 Input Capture Pin*) INT2 (*External Interupt 2 Input*)

- OC1B – Port E, Bit 2

  OC1B (*Output Compare Match B Output*) pin yang digunakan untuk Ouput pada saat mode Compare B diaktifkan.

- ALE – Port E, Bit 1

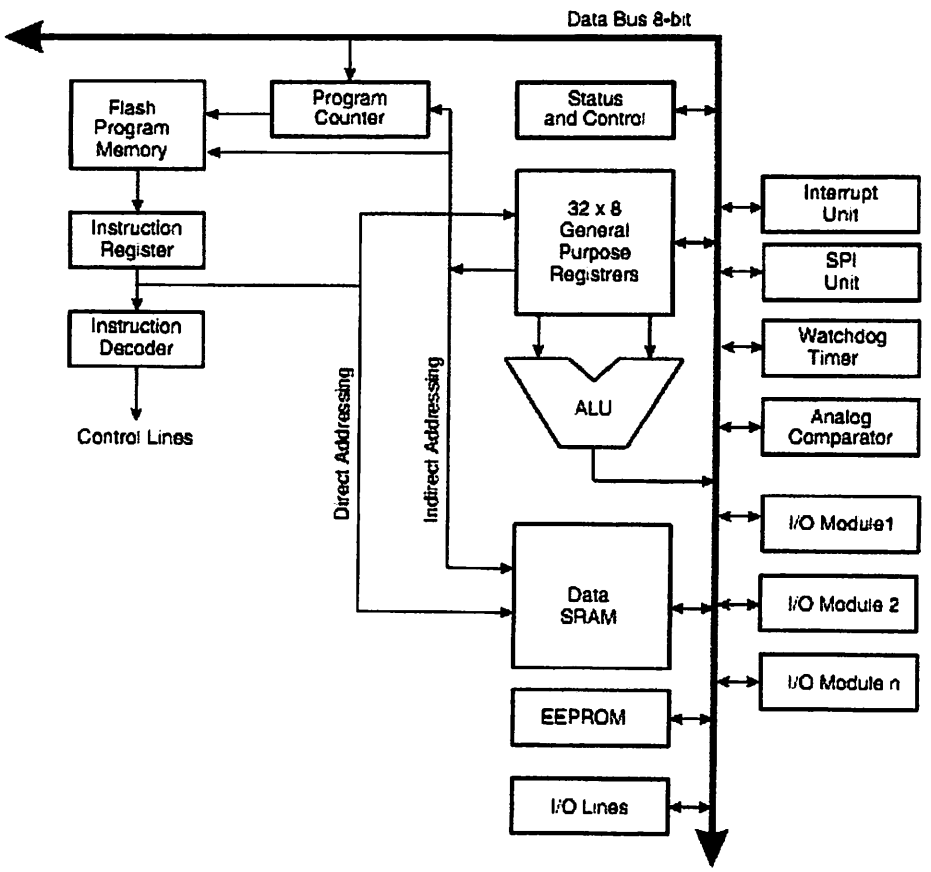
  ALE (*Address Latch Enable*) pin untuk mengaktifkan latch pada mode memori eksternal.

- ICP1/INT2 – Port E, Bit 0

  ICP1 (*Input Capture Pin*) pin yang digunakan untuk melakukan capture lebar pulsa untuk Timer/Counter1

  INT2 (*External Interupt source 2*) pin Eksternal interupt 2

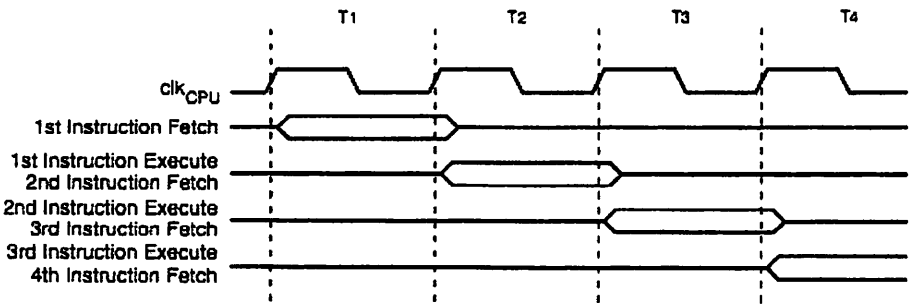
2.2.2 Arsitektur ATMega162



Gambar 2.4 Blok Diagram Arsitektur ATMega162[3]

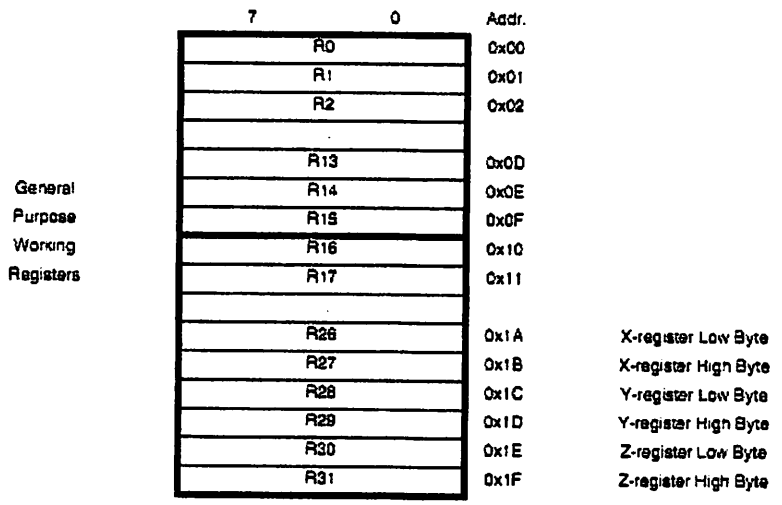
Dari blok diagram diatas, Mikrokontroler AVR menggunakan arsitektur *Harvard* dengan memisahkan antara memori dan *bus* untuk program dan data yang bertujuan untuk memaksimalkan kemampuan dan kecepatan. Instruksi dalam memori program dieksekusi dengan metode *Pipelining Single Level.* Dimana ketika satu instruksi dieksekusi, instruksi berikutnya diambil dari memori program. Konsep ini mengakibatkan instruksi dieksekusi setiap *clock cycle.* CPU terdiri dari 32x8-bit *General Purpose Register* yang dapat diakses dengan cepat dalam satu *clock cycle*, yang mengakibatkan operasi Arithmetic Logic Unit (ALU) dapat dilakukan dalam satu cycle. pada operasi ALU, dua *operand* berasal dari register, kemudian operasi dieksekusi dan hasilnya disimpan kembali pada register dalam satu *clock cycle.* Operasi aritmatik dan logic pada ALU akan

mengubah bit-bit yang terdapat pada *Status Register* (SREG). Proses pengambilan instruksi dan pengeksekusian instruksi berjalan secara *parallel*, dapat dilihat pada gambar di bawah ini [4]:



Gambar 2.5 Proses Pengambilan Instruksi Dan Pengeksekusian Instruksi Secara Paralel

2.2.3 General Purpose Register pada AVR



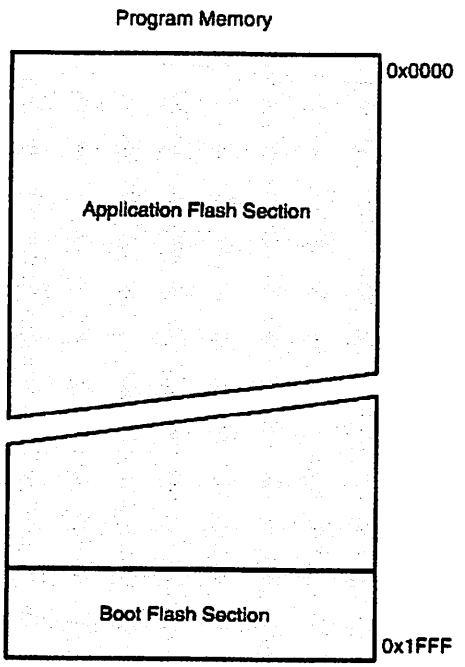
Gambar 2.6 General Purpose Register Mikrokontroler AVR[3]

Pada Gambar 2.6 diatas ditunjukkan struktur dari 32 Register *general Purpose* yang berada dalam CPU, masing-masing register ditentukan juga dalam alamat memori data, dipetakan dalam 32 lokasi pertama data user. Walaupun tidak secara fisik diimplementasikan sebagai lokasi SRAM, namun pengaturan ini

memberikan fleksibilitas dalam mengakses register, sperti register pointer X,Y, dan Z dapat diset menuju index dari register file manapun.

2.2.4 Memori Program

Arsitektur yang digunakan pada AVR membagi memori dalam dua bagian yaitu : memori data dan memori program. Selain itu Atmega 162 juga memiliki memori EEPROM untuk penyimpanan yang *non-volatile* (tidak akan terhapus jika power dimatikan). Pada ATMega162 semua program aplikasi disimpan pada memori *flash* sebesar 16Kbyte. Dari memori *flash* tersebut dibagi lagi menjadi dua bagian yaitu *boot region* dan *aplikasi. boot region* dimaksudkan untuk area penyimpanan *bootloader. bootloader* adalah program kecil yang akan dieksekusi pada saat pertama kali mikrokontroler *start-up*. Pengaturan pemakaian *boot region* dapat diatur pada saat konfigurasi *fuse bit*. jika fuse *BOOTRST* diaktifkan (*Enabled=0*) maka ketika reset atau *start-up*, mikrokontroler akan langsung menuju alamat dari *bootloader*. Besar boot region dapat diatur dengan menggunakan fuse BOOTSZ0 dan BOOTSZ1. Kapasitas maksimal dari *boot region* adalah 1024 *Word* (1 *Kbyte*).



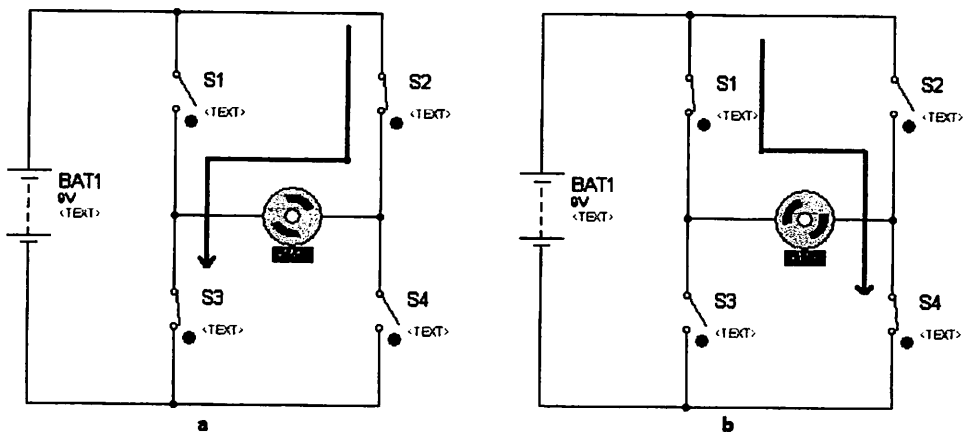
Gambar 2.7 Peta Memori ATMega162[3]

2.3 Driver Motor L298 (*H-Bridge*)

Penggunaan driver motor diperlukan sebagai penguat arus yang keluar dari mikrokontroler hal itu dikarenakan arus yang keluar dari port mikrokontroler saat kondisi high IOH = 20 mA sehingga dengan adanya Driver motor L298 maka arus kecil yang keluar dari mikrokontroler dapat menggerakkkan motor DC. IC L298 merupakan driver H-Bridge yang didesain untuk menhasilkan drive 2 arah dengan arus kontinyu sampai dengan 2 A pada level tegangan 4.8 Volt sampai dengan 46 Volt. Tiap H-Bridge dilengkapi dengan sensor arus beban yang dapat digunakan sebagai umpan balik ke pengendali (kontroler). IC Driver L298 Ini mampu men-drive beban-beban induktif seperti misalnya relay, selenoida, motor DC, motor Stepper, dan berbagai macam beban yang lain. Pada IC driver L298 telah dilengkapi dengan fitur PWM yang digunakan untuk pengendali kecepatan.

2.3.1 Konsep H-Bridge

Arah Putaran motor pada prinsipnya terdiri dari dua macam arah gerakan yaitu CW (*Counter Wise*) atau CCW (*Counter Clock Wise*). CW Adalah gerakan motor searah dengan jarum jam sedangkan CCW adalah gerakan motor berlawanan arah dengan putaran jarum jam. Untuk itulah agar kontroler dapat menggerakakkan motor ini secara searah dan berlawanan arah maka dibutuhkan konsep H-bridge. Konsep H-bridge pada dasarnya adalah menggunakan teknik *switch* yang bekerja bergantian untuk mengganti polaritas dari motor.

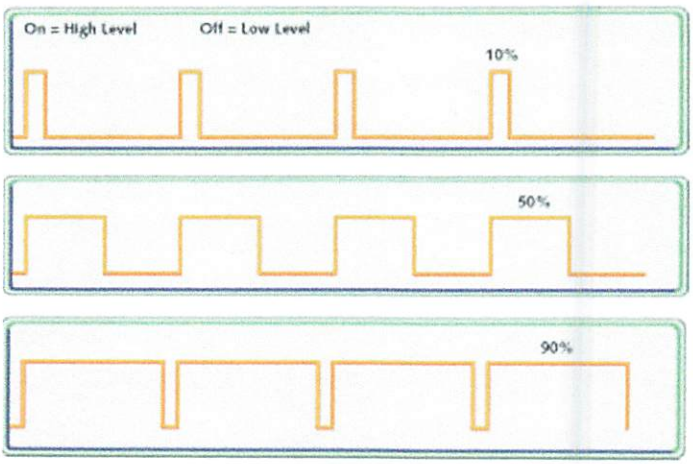


Gambar 2.8 Konfigurasi H-bridge

Pada Gambar 2.8 diatas menunjukkan skema rangkaian dari H-bridge driver. Secara sederhana konsep *H-bridge* adalah pergantian polaritas dengan menggunakan *switch* atau komponen *switching* seperti komponen transistor (BJT) atau FET (*Field Effect Transistor*). Pada Gambar 2.8 (a) saklar S2 dan S3 pada kondisi tertutup sehingga ada arus yang mengalir dari polaritas Positif ke negatif sehingga menyebabkan motor berputar CCW (berlawanan arah jarum jam). Sedangkan pada Gambar 2.8 (b) saklar S1 dan S4 dalam kondisi tertutup sehingga ada arus yang mengalir dan motor berputar berkebalikan dengan kondisi (b). hal itu dikarenakan terjadi perubahan polaritas pada kedua kutub motor DC tersebut. Hal yang tidak boleh terjadi adalah ketika posisi saklar S1 dan S3 atau posisi S2 dan S4 menutup secara bersamaan yang akan mengakibatkan terjadinya hubung singkat yang dapat menyebabkan kerusakan rangkaian secara keseluruhan. Untuk itu agar hal tersebut tidak terjadi maka pada rangkaian riilnya nanti perlu ditambahkan komponen inverter (*IC NOT 7400*). Penambahan diletakkan pada posisi S1 dan S2 serta S2 dan S4 sehingga *state* keduanya berkebalikan.

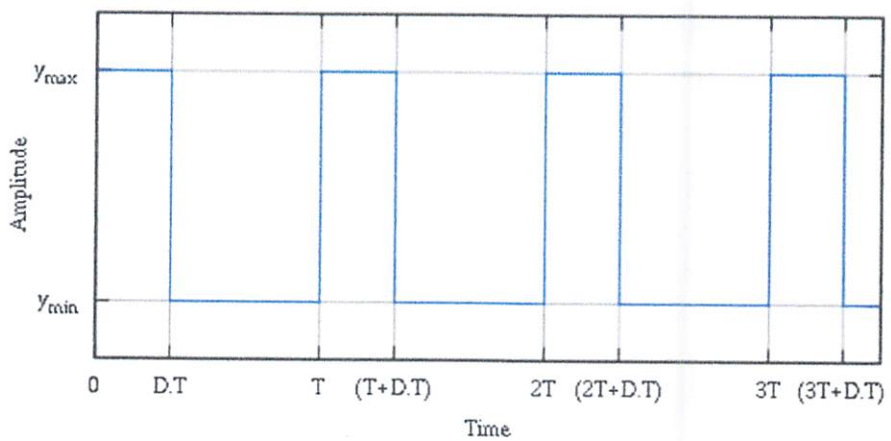
2.3.2 PWM (*Pulse Width Modulation*)

Pulse Width Modulation adalah suatu modulasi lebar pulsa yang merupakan rasio antara pulsa *high* dan pulsa *low*. Dari perbandingan tersebut dapat dinyatakan dalam nilai *Duty Cycle*. Nilai *Duty Cycle* dinyatakan dalam nilai presentase keadaan logika high (pulsa) dalam suatu periode sinyal. Satu siklus diawali dengan transisi *low* ke *high* dari sinyal dan berakhir pada transisi berikutnya. Selama satu siklus, jika waktu sinyal pada keadaan *high* sama dengan *low* maka dapat dikatakan bahwa sinyal memiliki *duty cycle* sebesar 50 %[4].



Gambar 2.9 Variasi presentase Duty Cycle
(*sumber : http://www.embedded.com/electronics-blogs/beginner-s-corner/4023833/Introduction-to-Pulse-Width-Modulation*)

Jika kita menganggap bentuk gelombang kotak *f(t)* dengan nilai batas bawah y_{min} dan batas atas y_{max} dan duty Cycle D seperti dilihat pada Gambar 2.10



Gambar 2.10 Gelombang Kotak dengan nilai y_{min} dan y_{max} serta nilai Duty Cycle

Nilai rata-rata dari bentuk gelombang di atas dapat dicari dengan menggunakan persamaan berikut :

$$y = \frac{1}{T}\int_0^T f(t)\,dt \quad \text{..... (2.7)}$$

jika *f(t)* adalah gelombang kotak, maka nilai y_{max} adalah dari $0 < t < D \cdot T$ dan nilai y_{min} dari $D \cdot T < t < T$. dari persamaan di atas didapat :

$$y = \frac{1}{T}\left(\int_0^T y_{max}\, dt + \int_0^T y_{min}\, dt\right)$$

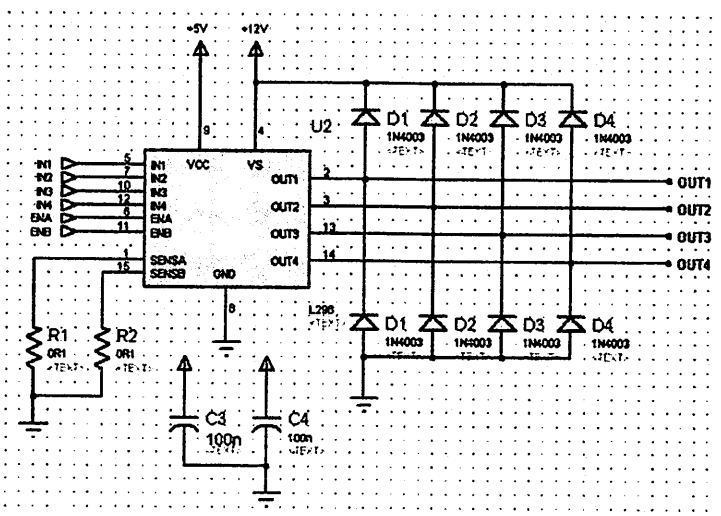
$$= DT_{max} + T(1-D)y_{min}$$

$$= D \cdot y_{max} + (1-D)\, y_{min} \quad \text{.....(2.8)}$$

Persamaan 2.8 dapat disederhanakan dalam berbagai kasus di mana $y_{min} = 0$ sehingga kita mendapat bentuk persamaan akhir $y = D \cdot y_{max}$. Dari persamaan ini jelas bahwa nilai rata-rata dari sinyal (y) secara langsung bergantung kepada *duty cycle D.*

2.3.3 Rangkaian Driver Motor L298

IC driver yang berada di pasaran ada dua jenis yaitu seri L298 dan L298D. Huruf D menandakan bahwa di dalam IC tersebut telah terdapat dioda proteksi sehingga tidak perlu menambahkan komponen dioda dalam rangkaiannya. Rangkaian driver L298 tanpa dioda proteksi internal ditunjukkan pada Gambar 2.11 berikut :



Gambar 2.11 Rangkaian Skematik Driver Motor L298

Pada gambar rangkaian diatas terdapat dua supply tegangan yaitu tegangan *logic* dan tegangan motor. Tegangan *logic* berkisar antara (4.5 – 5 Volt DC) sedangkan untuk supply motor berkisar antara 2,5 Volt s/d 46 Volt. Karena IC

yang digunakan adalah seri L298 tanpa dioda proteksi maka pada rangkaian Gambar 2.11 harus diberi dioda proteksi eksternal. Tujuan dari pemasangan dioda ini adalah untuk menahan arus balik yang keluar dari motor. Pada IC L298 terdapat 6 pin kontrol seperti ditunjukkan pada Tabel 2.7 di bawah.

Tabel 2.7 Fungsi Pin Kontrol IC L298

Nama	I/O	Fungsi
Input 1	I	Pin *Direction* untuk menentukan polaritas Out 1
Input 2	I	Pin *Direction* untuk menentukan polaritas Out 2
Input 3	I	Pin *Direction* untuk menentukan polaritas Out 3
Input 4	I	Pin *Direction* untuk menentukan polaritas Out 4
Enable A	I	Pin PWM untuk pasangan Out 1 dan Out 2
Enable B	I	Pin PWM untuk pasangan Out 3 dan Out 4

Masukan *Enable* A dan *Enable* B merupakan input PWM dimana semakin besar nilai presentase *duty cycle* maka kecepatan motor akan semakin cepat.

Tabel kebenaran untuk pin kontrol ditunjukkan pada tabel berikut :

Tabel 2.8 Tabel Kebenaran Input 1 dan Input 2 L298

INPUT			OUTPUT		Fungsi
Enable A	Input 1	Input 2	Out 1	Out 2	
H	H	L	V Motor	GND	Maju
H	L	H	GND	V Motor	Mundur
H	L	L	GND	GND	Stop/Brake
H	H	H	V Motor	V Motor	Stop/Brake
L	X	X	Z	Z	Free Running Motor Stop

Tabel 2.9 Tabel Kebenaran Input 3 dan Input 4 L298

INPUT			OUTPUT		Fungsi
Enable B	Input 3	Input 4	Out 3	Out 4	
H	H	L	V Motor	GND	Maju
H	L	H	GND	V Motor	Mundur
H	L	L	GND	GND	Stop/Brake
H	H	H	V Motor	V Motor	Stop/Brake
L	X	X	Z	Z	Free Running Motor Stop

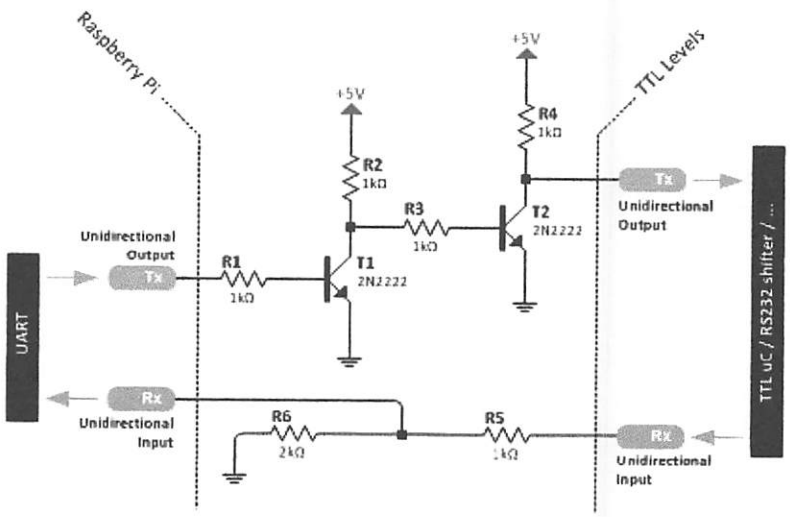
Keterangan :

H = High *Z = High Impedance/Tri State*

L = Low *X = don't care*

2.4 *Level Logic Converter*

Rangkaian *level logic converter* merupakan suatu rangkaian yang mengkonversi level tegangan CMOS ke level tegangan TTL dan sebaliknya dari level tegangan TTL ke CMOS. Level tegangan CMOS berada pada level 0V – 0,7V untuk logika 0 (*low*) dan 3.3V – 5V untuk logika 1(*high*) sedangkan pada level TTL logika 0 berada pada range tegangan 0V-0,7V dan 4.5V-5V untuk logika 1 (*high*). Level tegangan memiliki masalah berarti dalam komunikasi data, seperti contoh dalam suatu komunikasi data serial, level tegangan sangat menentukan kevalidan data yang diterima maupun yang dikirim, terlebih bila perangkat yang berkomunikasi tidak memiliki fitur *TTL 5 Volt tolerant*. TTL *tolerant* adalah suatu kemampuan perangkat yang menggunakan level logic CMOS untuk membaca data TTL dengan aman. Dalam hal ini penulis mengambil contoh perangkat Raspberry Pi dimana pin RX (*receiver*) pada komunikasi USART tidak memiliki pengaman tegangan yang berarti kesalahan dalam pemberian level tegangan dapat langsung merusak *chip* prosesor. Untuk itu rangkaian ini sangat diperlukan untuk memperpanjang umur suatu perangkat CMOS.



Gambar 2.12 Rangkaian Logic Level Konverter

(***Sumber** : http://blog.sunyday.net/?p=36)*

Rangkaian diatas dibagi menjadi dua bagian yaitu rangkaian transistor untuk *switching* dan rangkaian *attenuasi* atau rangkaian pembagi tegangan. Pada rangkaian *switching* transistor difungsikan sebagai saklar yang akan mengubah inputan TX_{in} yang mempunyai level 0 – 3.3Volt menjadi level tegangan TTL 5 Volt. Prinsip kerjanya adalah ketika input TX_{in} high maka arus akan mengalir ke kaki basis T1 sebesar : $i_b = \frac{V-V_{be}}{Rb}$. Ketika $I_b \geq \frac{I_c}{Hfe}$ maka T1 akan saturasi dan elektron akan mengalir menuju kolektor karena resistansi Colector-emittor = 0. Nilai arus yang mengalir melalui R3 menjadi bermuatan negatif dan akan menyebabkan T2 menjadi cut-off dengan nilai resistansi collector-emittor = tak terhingga sehingga arus pada R4 akan langsung menuju kaki TX_{out} dengan arus sebesar $I_{out} = \frac{V}{R_4}$ dan nilai tegangan sebesar 5 volt yang berarti level tegangannya telah berubah dari 3.3 (CMOS) menjadi 5 volt (TTL). Begitu juga sebaliknya ketika nilai TX_{in} = 0 volt (*low*) maka artinya $I_b < \frac{I_c}{Hfe}$ dan T1 akan *cut-off*. Nilai resistansi collector-emittor pada T1 akan menjadi tak terhingga dan arus akan mengalir dari R2 ke R3 menuju basis T2 sebesar $I_b = \frac{V-V_{be}}{R2+R3}$. ketika $I_b \geq \frac{I_c}{Hfe}$ maka T2 akan *saturasi* dan nilai resistansi collector-emittor T2 bernilai 0 dan akan ada aliran elektron yang bermuatan negatif menuju kolektor. TX_{out} akan bernilai 0 volt yang berarti masuk dalam level logika low.

Sedangkan prinsip rangkaian *attenuasi* (pembagi tegangan) adalah melemahkan tegangan yang masuk sesuai dengan nilai perbandingan resistansi. Pada Gambar 2.12 diatas nilai resistor yang digunakan untuk rangkaian pembagi tegangan adalah R6 dan R5. Nilai tegangan output (v_o) dapat dihitung dengan persamaan :

$$V_o = \frac{R_6}{R_5 + R_6} \quad \text{.....(2-9)}$$

2.5 Sensor Ultrasonik Ping[5]

Suara Ultrasonik adalah suatu sebutan untuk frekuensi suara diatas frekuensi suara yang dapat didengar oleh manusia. Seperti diketahui bahwa telinga manusia dapat mendengar suara dalam range frekuensi suara antara 20 Hz sampai dengan 20 kHz. Lebih dari frekuensi tersebut hanya hewan tertentu saja yang dapat mendengarnya seperti contoh adalah lumba-lumba. Lumba-lumba memanfaatkan suara ultrasonik untuk mengindera benda-benda di laut. Prinsip ini kemudian ditiru oleh sistem pengindera pada teknologi kapal selam yaitu dengan cara mengirimkan sebuah suara dan menghitung lamanya pantulan suara tersebut maka dapat diketahui jarak kapal selam dengan suatu objek tersebut. Mula-mula suara dibunyikan, kemudian dihitung lama waktu sampai terdengar suara pantulan. Jarak dapat dihitung dengan cara mengelaikan kecepatan suara merambat di udara dengan waktu pantulan. Kemudian hasilnya dibagi dengan 2. Pembagian dengan dua dikarenakan terdapat dua elemen yang digunakan yaitu elemen untuk mengeluarakan suara (*tx*) dan elemen untuk mendengar pantulan suara (*rx*).

Ping *Ultrasonik Range Finder* adalah suatu modul pengukur jarak dengan menggunakan suara ultrasonik buatan Paralax Inc. yang didesain khusus untuk teknologi robotika. Sensor Ping memiliki ukuran yang cukup kecil yaitu 2,1cm x 4,5cm. sensor ini dapat mengukur jarak antara 3 cm sampai dengan 200 cm. keluaran sinyal sensor ping adalah berupa pulsa yang lebarnya merepresentasikan jarak. Lebar pulsanya bervariasi dari 115 uS sampai dengan 18,5 ms.

Gambar 2.13 Modul Sensor Ping Paralax *Ultrasonic Range Finder*

Ultrasonik mengubah sinyal 40KHz menjadi suara sementara mikrofon ultrasonik berfungsi untuk mendeteksi suara hasil pantulan dari objek. Pada modul ping terdapat 3 pin yang digunakan untuk jalur *power supply* (+5v), Ground dan Signal. Pin Signal dapat langsung dihubungkan dengan mikrokontroler tanpa tambahan komponen apapun.

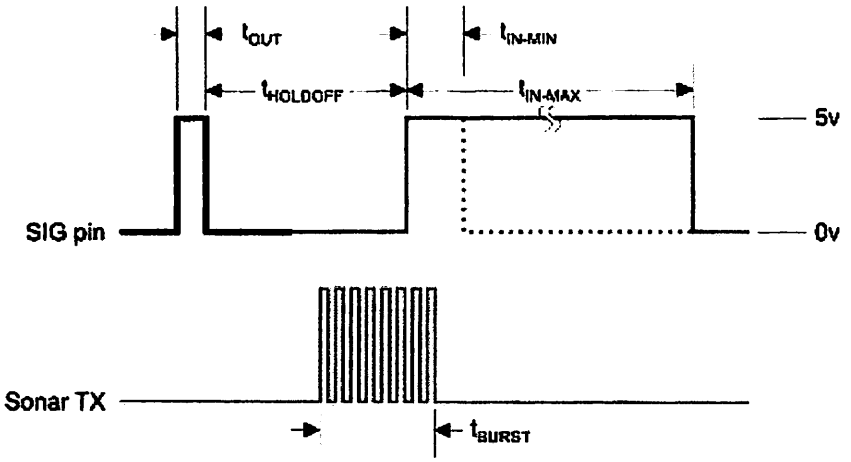
Ping mendeteksi objek dengan cara mengirimkan suara ultrasonik dan kemudian mendengarkan pantulan suara tersebt. Ping hanya akan mengirimkan suara ultrasonik ketika ada pulsa *trigger* dari mikrokontroler (Pulsa *High* selama 5 us). Suara ultrasonik dengan frekuensi sebesar 40 KHz akan dipancarkan selama 200 us. Suara ini akan merambat di udara dengan kecepatan 344.424 m/detik (atau 1 cm setiap 29.093 us), mengenai objek untuk kemudian terpantul kembali ke ping. Selama menunggu pantulan, ping akan menghasilkan sebuah pulsa. Pulsa ini akan berhenti (*low*) ketika suara pantulan terdeteksi oleh ping. Oleh karena itulah lebar pulsa tersebut akan merepresentasikan jarak antara ping dengan objek. Selanjutnya mikrokontroler cukup mengukur lebar pulsa tersebut dan mengkonversinya dalam bentuk jarak dengan perhitungan sebagai berikut :

$$Jarak = \frac{\left(\frac{Lebar\ Pulsa}{29.084\ \mu s}\right)}{2} (dalam\ cm) \quad \text{.....(2-10)}$$

Atau

$$Jarak = \frac{(Lebar\ Pulsa \times 0.03442)}{2} (dalam\ cm) \quad \text{.....(2-11)}$$

Karena 1/29.034 = 0.03442



HOST	t_{OUT}	2 uS (min), 5 uS typical
PING	$t_{HOLDOFF}$	750 uS
	t_{BURST}	200 uS @ 40 kHz
	t_{IN-MIN}	115 uS
	t_{IN-MAX}	18.5 mS

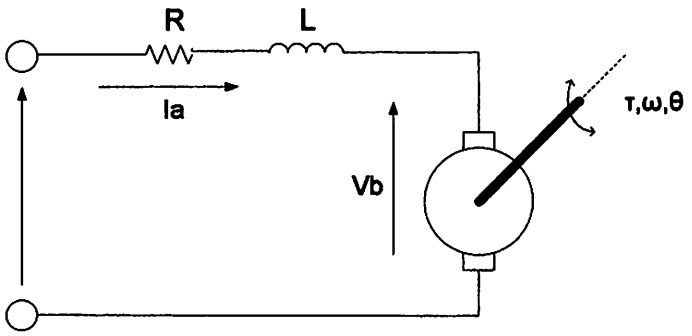
Gambar 2.14 Bentuk Pulsa Ping Parallax *Ultrasonic Range Finder*

Ada beberapa hal yang berkaitan dengan objek yang dapat dideteksi yaitu Sensor ping tidak dapat memantulkan suara dengan baik jika objek adalah suatu benda yang menyerap suara atau objek yang permukaannya tidak rata. Seperti contoh busa, kain, dan benda yang memiliki permukaan yang sejenis.

2.6 Motor Dc Magnet Permanen

Motor DC (*Direct Current*) adalah suatu peralatan elektromekanik dasar yang berfungsi sebagai pengubah tenaga listrik menjadi tenaga mekanik yang desain awalnya diperkenalkan oleh Michael Faraday lebih dari seabad yang lalu.

Rangkaian Ekivalen dari sebuah Motor DC magnet permanen dapat ditunjukkan dalam Gambar 2.15 berikut :



Gambar 2.15 Rangkaian Ekivalen Motor DC Magnet permanen

Keterangan :

V_a = Tegangan armatur

I_a = Arus Motor

R = Resistansi armatur

L = Induktansi lilitan armatur

V_b = tegangan induksi balik, emf (*elektro motor force*)

τ = torsi motor

ω = kecepatan putar motor

θ = sudut putaran poros motor

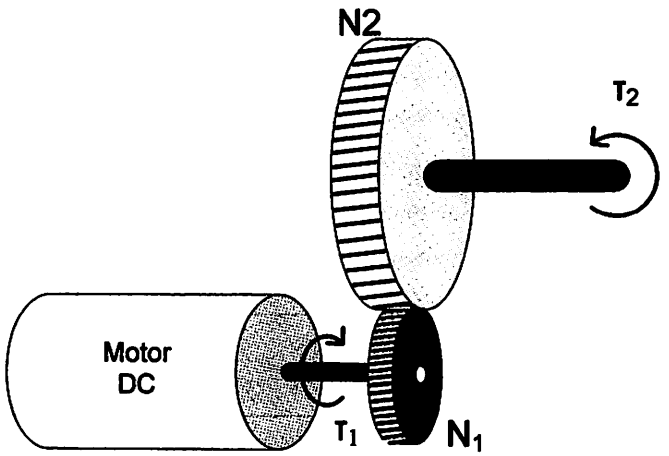
persamaan tegangan V_a adalah ,

$$V_a = L\frac{di_a}{dt} + Ri_a + K_b\omega \quad \text{.....(2-12)}$$

Dengan K_b adalah konstanta yang diukur dari tegangan yang dihasilkan oleh motor ketika berputar setiap satuan kecepatan (Volt.det/rad). *Magnitude* dan polaritas K_b adalah fungsi dari kecepatan angular, ω dan arah putaran poros motor.

Hal mendasar yang perlu diperhatikan dalam desain mekanik robot adalah perhitungan kebutuhan torsi untuk menggerakkan sendi atau roda. Motor sebagai penggerak utama (*prime-mover*) yang paling sering dipakai umumnya akan bekerja optimal (torsi dan kecepatan putar paling ideal) pada putaran yang relatif tinggi yang hal ini tidak sesuai bila porosnya dihubungkan langsung ke sendi gerak atau roda. Sebab kebanyakan gerak yang diperlukan pada sisi anggota badan robot adalah relatif pelan namun bertenaga. Untuk itu diperlukan cara-cara

transmisi daya motor (atau aktuator secara umum) secara tepat. Salah satu metoda yang paling umum adalah menggunakan sistem *gear*.



Gambar 2.16 Penggunaan Transmisi *Gear* Hubungan Langsung

Pada Gambar 2.16, N_1 adalah jumlah gigi pada gear poros motor, N_2 adalah jumlah gigi pada poros output, τ_1 adalah torsi pada poros motor, dan τ_2 adalah torsi pada poros output. Torsi output dapat dihitung dengan menggunakan persamaan :

$$\tau_2 = \frac{N_2}{N_1}\tau_1 \quad \text{.....(2-13)}$$

Sedangkan putaran output dapat dihitung sebagai :

$$Putaran_{OUT} = \frac{N_1}{N_2} Putaran_{Motor} \quad \text{.....(2-14)}$$

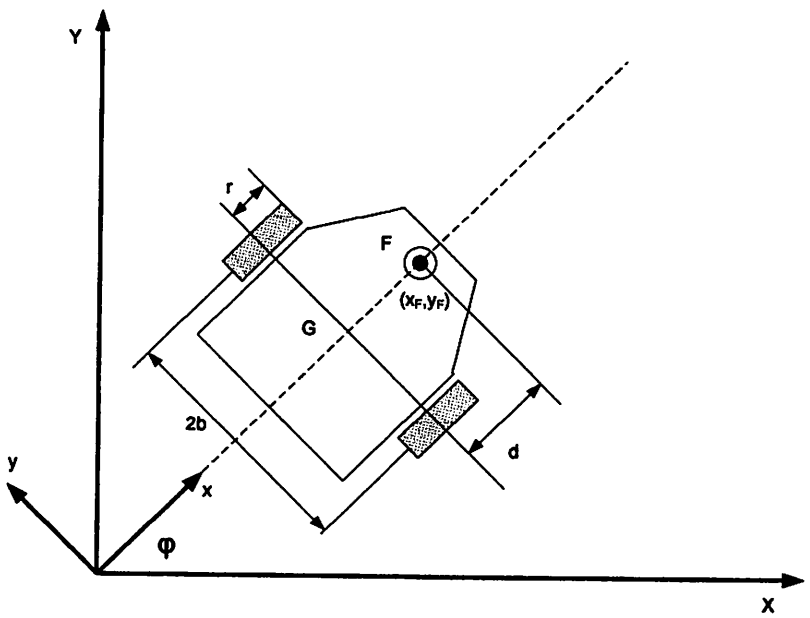
Arah putaran poros pada transmisi gear hubungan langsung seperti pada Gambar 2.16 adalah selalu berlawanan untuk tiap sambungan serial. Untuk mendapatkan arah putaran yang sama seperti pada poros motor maka *gear* harus disusun dengan jumlah yang ganjil.

Transmisi *gear* hubungan langsung ini terkenal mudah instalasinya, namun memiliki kelemahan utama yaitu jeda gerakan ketika dikemudikan dalam

arah yang berlawanan. Hal ini dikenal sebagai *backlash*. Kelemahan lain yang sering muncul adalah masalah friksi antar gir dan friksi poros. Namun demikian, tipe gir ini adalah yang paling banyak dipakai karena untuk mendapatkan rasio gir yang bear (transmisi dicapai dengan susunan gir yang relatif banyak) arsitekturnya dapat dibuat ringkas dalam satu rumah.

2.7 DDMR (*Differential Drive Mobile Robot*)

Mobile robot adalah robot yang dapat bergerak dari suatu titik ke titik yang lain baik dengan menggunakan roda atau motor yang disusun menyerupai kaki. Salah satu mobile jenis mobile robot yang sering digunakan adalah jenis DDMR (*Differentially Driven Mobile Robot*) seperti yang ditunjukkan pada Gambar 2.17 berikut :



Gambar 2.17 DDMR Pada Medan 2D Kartesian

Robot diasumsikan berada dalam kawasan 2D pada koordinat Catesian XY. Parameter-parameter dalam gambar adalah :

φ = Sudut arah hadap robot

2b = lebar robot yang diukur dari garis tengah roda ke roda

R = jari-jari roda (roda kiri dan kanan adalah sama dan sebangun)

D = jarak antara titik tengah antara 2 roda, G dengan titik acuan F

(x,y) = koordinat acuan di tubuh robot terhadap sumbu XY

Dalam kajian kinematik ini robot diasumsikan bergerak relatif pelan dan roda tidak slip terhadap permukaan jalan. Maka komponen x dan y dapat diekspresikan dalam suatu persamaan *nonholonomic* sebagai berikut :

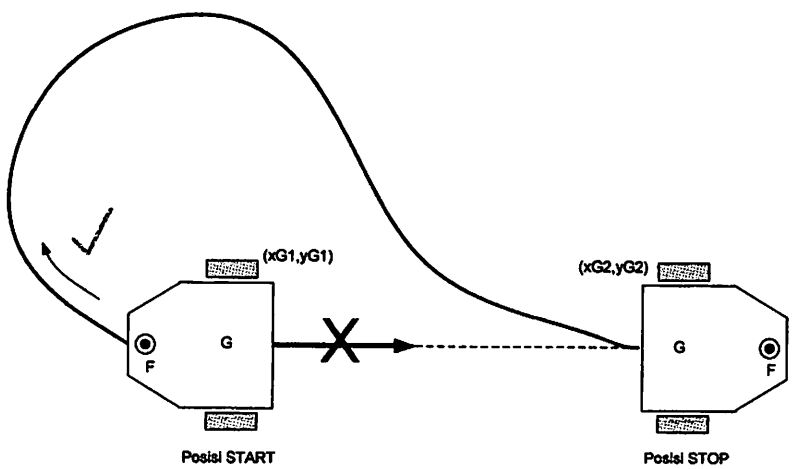
$$\dot{x}_G \sin\varphi - \dot{y}_G \cos\varphi = 0 \quad \text{.....(2-15)}$$

Untuk titik F sebagai acuan analisa, persamaan di atas dapat ditulis :

$$\dot{x}_F \sin\varphi - \dot{y}_F \cos\varphi + \dot{\varphi} = 0 \quad \text{.....(2-16)}$$

Masalah klasik dalam kontrol kinematik DDMR ini adalah bahwa ia memiliki dua aktuator, namun parameter kontrolnya lebih dari dua, yaitu x untuk gerakan ke arah X (1 DOF) dan y untuk arah Y (1DOF) yang diukur relatif terhadap perpindahan titik G dan gerakan sudut hadap φ yang diukur dari garis hubung titik G dan F terhadap sumbu X (1DOF). Inilah ciri khas dari sistem *nonholonomic.*

Dari Persamaan (2.15) nampak bahwa derajat kebebasan dalam kontrol kinematiknya berjumlah tiga yaitu : (x,y, φ). Karena ketiga parameter ini perlu dikontrol secara simultan untuk mendapatkan gerakan *nonholonomic.*



Gambar 2.18 Contoh Manuver DDMR

Perpindahan kedudukan robot dari START ke STOP bila dipandang pada titik G adalah perpindahan dari koordinat (x_{G1},y_{G1}) ke (x_{G2},y_{G2}) secara translasi. Namun hal ini tidak dapat dilakukan sebab robot harus dikontrol agar bergerak maju, sehingga ia harus membuat manuver belok membentuk lingkaran terlebih hingga pada posisi yang memungkinkan untuk mengarahkannya ke koordinat (x_{g2},y_{G2}). Oleh karena itu diperlukan titik acuan F yang berada di luar garis yang menghubungkan kedua roda agar sudut hadap dapat dihitung.

Bentuk umum persamaan kinematik untuk DDMR ini dapat dinyatakan dalam persamaan kecepatan sebagai berikut.

$$\begin{pmatrix} \dot{x}_F \\ \dot{y}_F \\ \dot{\theta}_F \end{pmatrix} = T_{NH} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix} \text{ atau } \dot{q}(t) = T_{NH}(q)\dot{\theta}(t) \quad \text{.....(2-17)}$$

T_{NH} adalah matriks transformasi *nonholonomic*, $\dot{\theta}_L$ dan $\dot{\theta}_R$ adalah kecepatan radial roda kiri dan kanan, dan q adalah sistem koordinat umum robot.

$$q = [x_F, y_F, \varphi]^T \text{ atau } q = \begin{bmatrix} x_F \\ y_F \\ \varphi \end{bmatrix} \quad \text{.....(2-18)}$$

Jika T_{NH} diuraikan dari persamaan 2.2 dengan memperhatikan Gambar 2.17 maka dapat ditentukan,

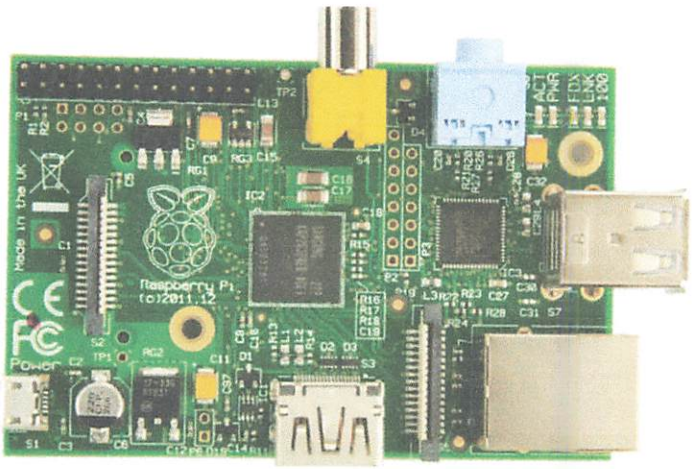
$$T_{NH}(q) = \begin{pmatrix} \frac{r}{2}\cos\varphi + \frac{d \cdot r}{2b}\sin\varphi & \frac{r}{2}\cos\varphi - \frac{d \cdot r}{2b}\sin\varphi \\ \frac{r}{2}\sin\varphi - \frac{d \cdot r}{2b}\cos\varphi & \frac{r}{2}\sin\varphi + \frac{d \cdot r}{2b}\cos\varphi \\ -\frac{r}{2b} & \frac{r}{2b} \end{pmatrix} \quad \text{.....(2-19)}$$

Kinematik inversnya dapat ditulis

$$\dot{\theta}(t) = T_{NH}^{-1}(q)\dot{q}(t) \quad \text{.....(2-20)}$$

2.8 Raspberry pi[7]

Raspberry pi adalah sebuah mini computer atau komputer kecil seukuran kartu kredit yang dapat digunakan untuk pembelajaran ataupun untuk aplikasi di bidang kontrol. Seperti layaknya komputer konvensional, Raspberry Pi telah dilengkapi dengan perangkar keras seperti CPU, prosessor, RAM, GPU, I/O USB Port dan ethernet port.



Gambar 2.19 Raspberry Pi Board Model B

Pada saat ini ada dua model yang telah berada di pasaran yaitu model A dan model B. model A adalah versi terdahulu dari model B sehingga perangkat keras yang ada pun masih terbatas jika dibandingkan dengan model B.
Perbedaan spesifikasi model A dan model B dapat dilihat pada Tabel 2.10 berikut:

Tabel 2.10 Perbedaan Spesifikasi Model A dan Model B[2]

Hardware	Model A	Model B
SoC (System on Chip)	Broadcom BCM2835(CPU+GPU)	
CPU	700Mhz ARM11 ARM1176JZF-S core	
GPU	Broadcom VideoCore IV, OpenGL ES 2.0, OpenVG 1080p30 H.64	
Momori (SDRAM)iB	256 MB	512 MB
USB 2.0 ports	1 buah	2 buah
Video Output	Composite video/Composite RCA, HDMI	
Audio Output	Konektor TRS/3.5 mm Jack, HDMI	
Media penyimpanan	SD/MMC/SDIO	
Ethernet Port	Tidak ada	10/100 Ethernet RJ45
I/O	GPIO,SPI(Serial Peripheral Interface),I2C,I2S,UART	

Kebutuhan power supply untuk model A dan model B berbeda, hal itu dikarenakan terdapat perbedaan hardware yang terpasang pada board. Nilai nominal tegangan supply untuk raspberry adalah 5 volt (*model A dan B*) sedangkan arus supply untuk model A adalah 500-700 mA sedangkan untuk model B berkisar antara 700-1500 mA. Nilai arus supply ini tidak mutlak karena nilai arus yang dibutuhkan adalah tergantung dari pemakaian peripheral yang terpasang. Untuk menghindari kerusakan yang terjadi karena kesalahan dalam pemberian supply, pada board Raspberry Pi telah terpasang rangkaian pengaman seperti dioda proteksi, rangkaian tegangan *clamp*, dan *fuse* yang dapat direset (*resetable fuse*).

Untuk dapat beroperasi seperti layaknya komputer maka Raspberry harus diinstall OS (Operating System) terlebih dahulu. Untuk saat ini ada beberapa OS yang dapat diinstal pada raspberry pi yaitu : Raspbian OS, Arch Linux ARM, Debian ARM, Fedora Remix, Raspbmc, RISC OS bahkan Android. Sistem operasi yang digunakan semuanya berbasis Linux dan belum mendukung untuk sistem operasi berbasis Windows atau Apple. Sedangkan untuk bahasa pemrograman yang didukung oleh Raspberry Pi diantaranya adalah : Python, C/C++, C# (Mono Develop), Java, Erlang, Pascal (*Lazarus*), PHP, Javascript (*Node JS*).

GPIO (*General Purpose Input Output*) merupakan suatu antarmuka port yang dapat digunakan untuk hubungan dengan hardware lain (*arduino, mikrokontroler, driver dll*) melalui media komunikasi serial atau paralel.

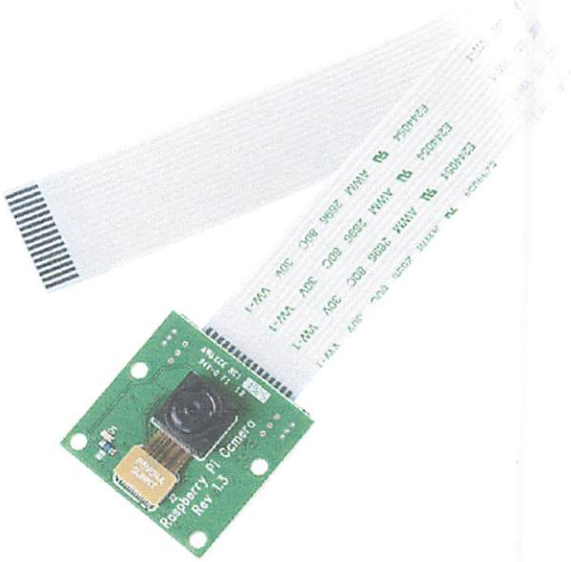
Gambar 2.20 GPIO Pada Raspberry Pi

(***Sumber*** : *http://elinux.org/RPi_Low-level_peripherals*)

Jumlah pin pada GPIO raspberry Pi ada 26 buah dengan fungsi khusus pada masing-masing pin. Setiap pin dapat dikonfigurasi menjadi input atau output secara software. Hal yang penting adalah bahwa level tegangan pada GPIO Raspberry Pi adalah level CMOS (3.3 volt) dengan koneksi langsung ke prosessor BCM2835 tanpa proteksi tegangan. Sehingga apabila kita salah memberikan level tegangan maka dapat langsung merusak *chip*. Untuk itu agar koneksi dengan GPIO aman maka diperlukan rangkaian *buffer* dengan proteksi tegangan.

2.9 Modul Kamera Raspberry Pi[8]

Modul kamera Raspberry adalah suatu perangkat kamera yang khusus didesain untuk digunakan pada Board Raspberry Pi melalui konektor CSI (*Camera Serial Interface*). Modul kamera ini memiliki dimensi 25mm x 20mm x 9 mm dan berat 3 g, sehingga cocok digunakan untuk aplikasi *embedded*.



Gambar 2.21 Modul Kamera Raspberry Pi

(*sumber* : https://www.sparkfun.com/products/11868)

Spesifikasi modul Kamera :

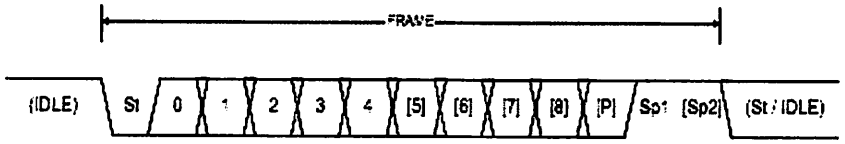
1. Tipe Kamera OmniVision OV5647 Color CMOS QSXGA (5 Megapixel)
2. Ukuran Sensor 3.67 x 2.74 mm
3. Jumlah Piksel= 2592 x 1944
4. Ukuran piksel = 1.4 x 1.4 µm
5. Lensa : f=3.6 mm, f/2.9
6. *Angle of View* (Jangkauan sudut) : 54 x 41 derajat
7. *Field of View* (Jangkauan Pandangan) : 2.0 x 1.33 m pada jarak 2 m
8. Persamaan Lensa SLR (*full frame*) : 35 mm
9. Fokus : 1 m sampai dengan tak terbatas
10. Video : 1080p pada 30 fps dengan codec H.264 (AVC)
11. Kecepatan sampai dengan 90 fps pada mode VGA

Hal yang perlu diperhatikan adalah Modul Kamera Raspberry pi merupakan perangkat yang sensitif terhadap ESD (*Electrostatic Discharge*) yaitu perpindahan muatan elektrostatic antara dua objek yang terjadi saat kedua objek berbeda potensial. Hal ini terjadi jika tubuh kita terlalu banyak muatan positif sehingga bila kita memegang suatu perangkat yang sensitif maka akan terjadi

peristiwa *static discharge* yang bisa merusak *device*. Untuk mencegah terjadinya ESD maka sebelum kita memegang modul kamera sebaiknya tubuh kita melakukan *grounding* terlebih dahulu dengan memegang permukaan lantai.

2.10 Komunikasi USART (*Universal Synchronous Asynchronous receiver Transmitter*)

Transmisi data serial dibedakan menjadi dua macam, yaitu komunikasi data seri sinkron dan komunikasi data asinkron, perbedaan ini tergantung dari clock pendorong data. Dalam komunikasi data seri sinkron, clock untuk *shift register* ikut dikirimkan bersama dengan data serial. Sebaliknya dalam komunikasi data serial asinkron , clock pendorong *shift register* tidak ikut dikirim, rangkaian penerima data harus dilengkapi dengan rangkaian yang mampu membangkitkan clock yang bisa dipakai untuk mendorong *shift register* penerima. Untuk keperluan tersebut terlebih dahulu ditentukan bahwa saat tidak ada pengiriman data, keadaan saluran adalah "1", saat akan mulai mengirim data 1 byte saluran dibuat menjadi "0" dulu selama 1 periode clock pendorong, dalam 8 periode clock berikutnya dikirim data bit 0, bit 1 dan seterusnya sampai bit 8, dan pada periode clock yang ke 10 saluran dikembalikan menjadi "1" dengan demikian, data 8 bit yang dikirim diawali dengan bit start yang bernilai "0" dan diakhiri dengan bit stop yang bernilai "1", seperti terlihat pada gambar.



Gambar 2.22 Frame data Komunikasi Serial Asinkron

Kemasan data ini dimaksud agar rangkaian penerima bisa membangkitkan clock yang frekuensinya sama dengan clock pengirim dan fasanya disinkronkan pada awal penerimaan data 1 *byte*.

2.10.1 Konverter RS232 Serial to TTL

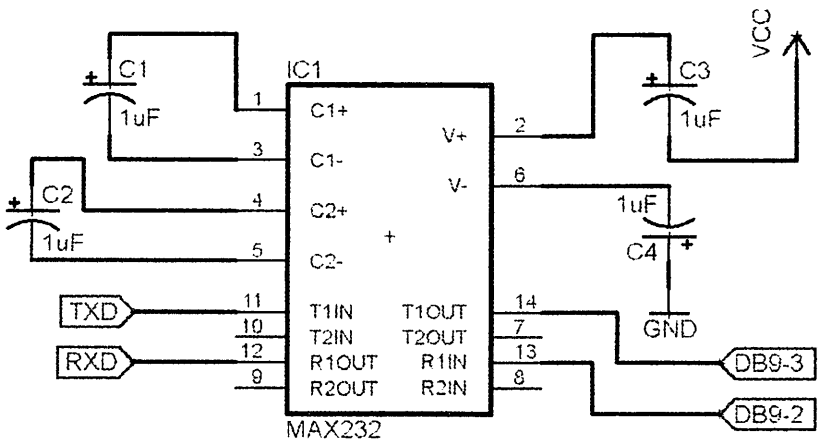
Pada komputer biasanya terdapat sebuah port untuk komunikasi serial. Beberapa contoh penerapan komunikasi serial adalah mouse, scanner dan siistem akuisisi data yang terhubung ke port serial COM1/COM2. Bagian yang penting dari komunikasi serial adalah konektor DB9 dan RS232. DB9 adalah konektor yang digunakan untuk menghubungkan hardware dengan komputer.

Kegunaan konverter RS232 adalah sebagai driver yang akan mengkonversi tegangan dari hardware agar sesuai dengan tegangan pada komputer sehingga data dapat dibaca.

Tabel 2.11 Perbandingan level Tegangan RS232 dan TTL

Interface	Level High	Level Low
Serial RS232	*-15V s/d -3V*	*+3V s/d +15V*
TTL	*+2V s/d +5V*	*0V s/d +0.8V*

Rangkaian interface menerjemahkan level tegangan RS232 ke level tegangan TTL dan sebaliknya. Rangkaian Interface tersebut menggunakan IC MAX232.



Gambar 2.23 Rangkaian Konverter RS232

BAB III
PERANCANGAN DAN ANALISA SISTEM

3.1 Pendahuluan

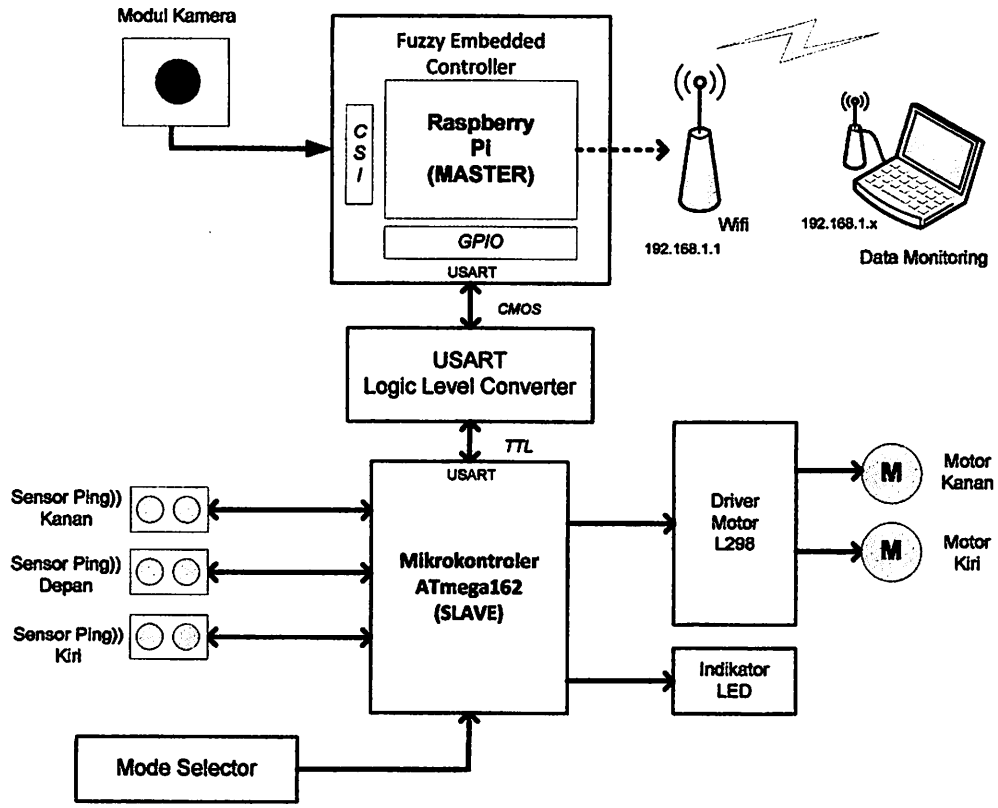
Pada bab ini akan membahas mengenai perencanaan sistem, prinsip kerja dan perancangan perangkat keras (*hardware*) serta perangkat lunak (*software*) yang berkaitan dengan kontrol robot yaitu mengenai design sistem kendali fuzzy yang diimplementasikan pada sistem *embedded raspberry pi.* Pada perancangan ini akan diimplementasikan konsep dan teori dasar yang telah dibahas sebelumnya, sehingga tujuan dari perencanaan dapat tercapai dengan baik. Untuk itu pembahasan difokuskan pada desain yang direncanakan pada diagram blok sistem.

3.2 Perancangan Sistem

Sistem yang akan dirancang akan dibagi menjadi dua bagian utama yaitu perancangan perangkat keras (*hardware*) dan perangkat lunak (*software*) robot. Perancangan sistem perangkat keras meliputi bagian Input, kontroler, output dan rangkaian supply. Pada bagian input terdiri dari rangkaian sensor yang dipakai yaitu 1 modul kamera raspberry pi dan 3 buah modul sensor jarak berbasis ultrasonik dan rangkaian switch (*push button/ tactile switch*) untuk menentukan mode dan yang diinputkan oleh user.

Bagian kontroler terdiri dari modul SBC (*Single Board Computer*) Raspberry Pi yang dikombinasikan dengan mikrokontroler ATMega162 dengan konsep Master Slave dimana Raspberry Pi dikonfigurasikan sebagai Master dan mikrokontroler ATMega162 sebagai slave. Raspberry Pi dipilih sebagai Kontroler Master karena memiliki *resource* yang cukup besar dan memiliki kecepatan pengolahan yang cepat jika dibandingkan dengan mikrokontroler konvensional. Sedangkan untuk bagian output terdiri dari driver motor yang akan memberikan daya ke motor untuk dapat menggerakkan robot maju,mundur,dan belok dengan nilai kecepatan yang bisa diubah yaitu dengan mengatur nilai PWM (*Pulse Width Modulation*).

Diagram blok keseluruhan sistem dapat dilihat pada gambar ,



Gambar 3.1 Diagram Blok Sistem

Penjelasan Diagram Blok :

1) Input , masukan nilai aktual dari sensor dan *mode selector* dari *user*. Bagian input terdiri dari :
    a. Sensor kamera yang akan membaca ada dan tidaknya halangan di depan yang akan dipetakan menjadi koordinat horizontal halangan terhadap posisi heading robot. Pengolahan sensor kamera dilakukan oleh Raspberry Pi
    b. Sensor ultrasonik ping bagian kiri atau kanan yang akan membaca nilai jarak aktual robot terhadap dinding kiri/kanan
    c. Sensor ultrasonik depan digunakan untuk membaca jarak robot bagian depan terhadap halangan depan.

d. Mode selector digunakan untuk menentukan mode telusur dinding yang akan dijalankan apakah telusur kiri atau kanan.

2) Kontroler, yaitu bagian pengolahan dari nilai yang dibaca oleh sensor. Kontrol pada perancangan sistem ini dibagi menjadi 2 bagian yaitu kontroler master (*Raspberry Pi*) dan Kontroler slave (ATMega162). Kontroler slave digunakan untuk pembacaan 3 sensor ultrasonik, pembacaan mode yang diinputkan oleh user dan mengolah hasil *defuzzyfikasi* oleh kontroler master menjadi nilai gerakan dan kecepatan PWM Motor. Kontroler Master digunakan sebagai kontroler utama yang digunakan untuk mengolah semua input yang akan menentukan gerakan dari robot melalui sistem kendali logika fuzzy. Selain itu kontroler master digunakan juga untuk mengirimkan data nilai *error* dan hasil *defuzifikasi* melalui komunikasi wifi untuk kemudian ditampilkan pada komputer PC dengan grafik (*fungsi debug*). Komunikasi yang digunakan untuk menghubungkan kotroler master dan kontroler slave adalah menggunakan protokol komunikasi USART .

3) Output, yaitu bagian yang akan memproses hasil *defuzzyfikasi* yang dilakukan oleh kontroler menjadi suatu gerakan robot (maju, mundur dan belok) dengan nilai kecepatan dan percepatan. Bagian ouput terdiri dari beberapa bagian, yaitu :

   a. Keluaran data error dan nilai hasil *defuzzifikasi* yang dikirimkan melalaui wifi oleh Raspberry Pi
   b. Driver motor yang akan mengubah hasil *defuzzifikasi* menjadi nilai daya yang akan menggerakkan motor
   c. Display LED, digunakan untuk membantu user dalam menentukan mode yang dipilih serta indikator untuk menandai proses terjadinya komunikasi USART antara kontroler master dan slave.

3.2.1 Prinsip Kerja

Fokus sistem yang akan dirancang ini adalah perancangan sistem navigasi mobile robot dalam suatu trajectori dalam bentuk sekat-sekat dinding dimana terdapat halangan-halangan di dalam trajectori tersebut. untuk itu sistem dibuat dengan pembacaan sensor-sensor yang berada di dalam robot yaitu ultrasonik ping yang akan dibaca oleh mikrokontroler ATMega162 (*slave*) dan sensor kamera yang akan dibaca dan diolah Raspberry Pi (*Master*). Kedua kontroler tersebut berkomunikasi dengan menggunakan protokol USART dengan terlebih dahulu melewati rangkaian level converter agar keduanya dapat berkomunikasi dengan aman. Format komunikasi yang akan digunakan adalah dengan memakai sistem paket data yang berisi data-data sensor dan data mode dengan kecepatan 115200bps.

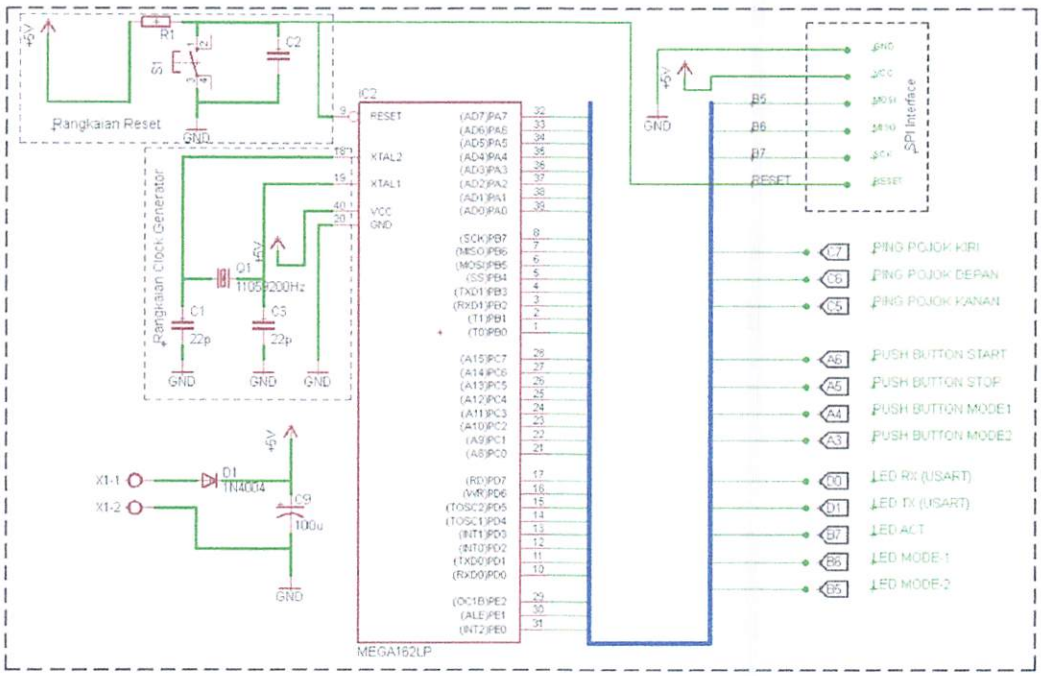
Pada saat start, robot terlebih dahulu akan membaca penekanan push button oleh *user* yang akan menentukan mode telusur dinding yang akan dijalankan yaitu mode telusur dinding kanan atau mode telusur dinding kiri. Setelah mode ditentukan maka robot dapat distart. namun karena Raspberry Pi adalah sebuah komputer yang memerlukan waktu untuk *booting* maka start robot harus menunggu sinyal bahwa Raspberry Pi (kontroler master) telah siap melakukan proses kontrol. Jika kontroler *master* telah siap melakukan pemrosesan maka robot dapat distart.

Konsep kendali yang digunakan pada perancangan sistem ini adalah dengan menggunakan dua kendali logika fuzzy yaitu kontroler fuzzy untuk menelusur dinding dan kontroler fuzzy untuk penghindar halangan. Pada kontroler fuzzy untuk penelusur dinding, masukan dibagi menjadi dua yaitu nilai error dan delta error pembacaan sensor ultrasonik kiri/kanan. Sedangkan kontroler fuzzy untuk penghindar halangan dibagi menjadi dua input yaitu jarak halangan dan posisi koordinat piksel halangan. Acuan untuk transisi dari telusur dinding ke penghindar halangan adalah posisi halangan depan dan ada tidaknya objek berwarna yang ada di depan.

3.3 Perancangan Perangkat Keras

3.3.1 Perancangan Minimum Sistem ATMega162

Minimum sistem merupakan suatu rangkaian yang digunakan sebagai pendukung kerja dari perangkat mikrokontroler. Dalam perancangannya minimum sistem memiliki 2 bagian utama yang wajib ada yaitu rangkaian reset dan rangkaian clock (jika menggunakan sumber clock eksternal) sedangkan rangkaian yang lain bersifat opsional tergantung dari fungsi mikrokontroler yang akan digunakan pada suatu sistem/alat yang akan dirancang. Pada perancangan sistem, mikrokontroler yang dipakai adalah mikrokontroler buatan ATMEL dengan tipe ATMega162 kemasan DIP (*dual In Line Package*). Mikrokontroler ini sengaja dipilih karena pertimbangan memor *flash* (memori program) yang cukup untuk menampung aplikasi yang dibuat serta memiliki fitur 2 buah USART yaitu USART0 dan USART1.



Gambar 3.2 Rangkaian Minimum Sistem ATMega162

3.3.1.1 Perancangan Rangkaian *Clock Generator*

Mikrokontroler ATMega162 telah dilengkapi dengan fitur eksternal Clock untuk menambah kecepatan operasi pada saat menjalankan program. Untuk dapat menggunakan fitur eksternal clock maka mikrokontroler harus disetting terlebih dahulu dengan cara memprogram fuse bit. hal tersebut dikarenakan pada kondisi default (pertama kali dibeli) mikrokontroler telah diset menggunakan clock internal sebesar 1 Mhz. Fuse yang berfungsi untuk mengatur nilai konfigurasi clock yang digunakan adalah fuse CKSEL(*Clock select*) bit3 s/d 1. Nilai fuse ini hanya perlu diprogram satu kali saja atau pada saat terjadi perubahan sistem clock yang digunakan. Pada keadaan awal (clock 1 Mhz) nilai CKSEL3..1

Nilai fuse CKSEL dan clock yang dapat digunakan dapat dilihat pada tabel berikut :

Tabel 3.1 Mode Operasi Pada Eksternal Clock

CKSEL3:1	**Range Frekuensi (Mhz)**	**Rekomendasi nilai kapasitor yang digunakan pada rangkaian Clock (pF)**
100	0.4 - 0.9	-
101	0.9 - 3.0	12 -22
110	3.0 - 8.0	12 -22
111	8.0 -	12 -22

Untuk itu agar dapat penambahan eksternal clock dapat berfungsi dengan baik maka nilai fuse CKSEL3..1 diset (CKSEL3=1, CKSEL2=1, CKSEL1=1) dengan nilai C_1 dan C_2 adalah sebesar 22 pF seperti yang telah direkomendasikan pada datasheet ATMega162

Nilai clock yang digunakan adalah sebesar 11,059200 Mhz dengan pertimbangan bahwa nilai clock sangat berpengaruh terhadap nilai error yang berkaitan dengan nilai baudrate komunikasi serial. nilai 11,059200 Mhz adalah nilai maksimal eksternal clock dengan nilai error baud minimal.

Nilai baudrate (kecepatan pengiriman data serial) pada AVR ditentukan oleh register UBRR (USART Baud Rate Register). Nilai UBRR dapat ditentukan dengan persamaan sebagai berikut :

$$UBRR = \left(\frac{F_{Kristal}}{16 \times baudrate} \right) - 1$$

Pada sistem, baudrate yang digunakan adalah sebesar 115200bps, maka nilai UBRR adalah :

$$UBRR = \left(\frac{11059200}{16 \times 115200}\right) - 1$$

$$UBRR = \left(\frac{11059200}{1843200}\right) - 1$$

$$UBRR = (6) - 1$$

$$UBRR = 5d \quad atau \quad 5H$$

Nilai error pada baudrate dapat terjadi bila nilai akhir UBRR bernilai pecahan (*float*) yang berarti akan terjadi pembulatan. Error baudrate dapat dihitung dengan menggunakan persamaan :

$$Error\ (\%) = \left(\left(\frac{baudrate\ pembulatan}{baudrate\ yang\ diinginkan}\right) - 1\right) \times 100\ \%$$

Sebagai perbandingan akan diberikan contoh dengan menggunakan frekuensi kristal sebesar 12 Mhz dengan baudrate sebesar 115200bps.

$$UBRR = \left(\frac{12000000}{16 \times 115200}\right) - 1$$

$$UBRR = \left(\frac{12000000}{1843200}\right) - 1$$

$$UBRR = (6.51) - 1$$

$UBRR = 5.51$ tau dibulatkan menjadi 6

Untuk mendapatkan baudrate hasil pembulatan maka rumus di atas dibalik menjadi :

$$BAUD = \left(\frac{F_{Kristal}}{(16 \times (UBRR + 1))}\right)$$

$$BAUD = \left(\frac{12000000}{(16 \times (6 + 1))}\right)$$

$$BAUD = \left(\frac{12000000}{(112)}\right)$$

$$BAUD = 107142.8 \quad bps$$

Error dapat dihitung dengan persamaan :

$$Error\ (\%) = \left(\left(\frac{baudrate\ pembulatan}{baudrate\ yang\ diinginkan} \right) - 1 \right) \times 100\ \%$$

$$Error\ (\%) = \left(\left(\frac{107143}{115200} \right) - 1 \right) \times 100\ \%$$

$$Error\ (\%) = (0.93 - 1) \times 100\ \%$$

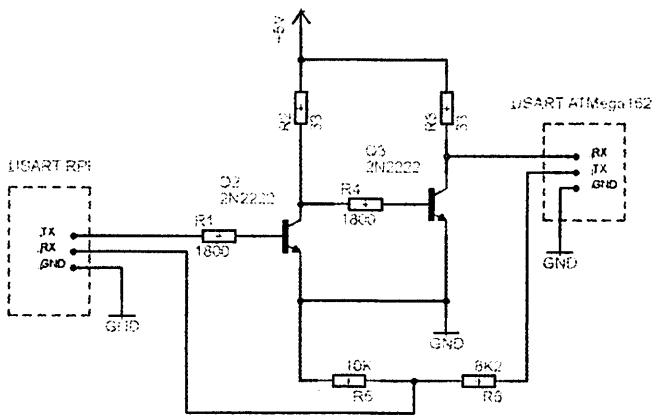
$$Error\ (\%) = 7\ \%$$

3.3.1.2 Perancangan Rangkaian Reset

Reset digunakan untuk menginisiasi semua I/O register kembali ke kondisi awal dan mengembalikan instruksi program ke alamat awal. Pin reset mikrokontroler adalah pin aktif low dimana pada datasheet kondisi reset terpenuhi jika t_{RST} (lebar pulsa low) adalah minimal selama 2,5 µs. pada pin reset mikrokontroler telah dilengkapi dengan internal pullup namun untuk aplikasi dimana lingkungan dalam kondisi penuh noise maka eksternal pullup perlu ditambahkan. Lingkungan penuh noise disini adalah lingkungan yang dinamis dan mengalami pergerakan seperti contoh : lingkungan Industri, Robot, Otomotif dll . Penambahan kapasitor digunakan untuk menghindari terjadinya debouncing yang diakibatkan oleh konstruksi mekanik dari push button.

3.3.1.3 Perancangan *Level Logic Converter.*

Level Logic converter pada perancangan sistem berfungsi sebagai pengubah level tegangan USART TTL menjadi level tegangan USART CMOS. Kontroler Slave menggunakan level tegangan TTL yaitu logika Low berada pada tegangan 0 s/d 0.7 Volt dan logika High pada level tegangan 4.5 s/d 5 Volt. Sedangkan kontroler master menggunakan level tegangan logic CMOS yaitu logika low pada level tegangan 0 s/d 0.7 V dan logika High pada level 2.2 s/d 3.3 V. selain sebagai konverter level tegangan, rangkaian ini digunakan sekaligus sebagai pengaman kontroler master dari over voltage dikarenakan menurut skematik rangkaian kontroler master (*Raspberry Pi Schematic Rev.B*), GPIO berhubungan secara langsung dengan Chip Prosessor tanpa pengaman. Rangkaian Level Logic Converter ini terdiri dari dua bagian yaitu Rangkaian Switching dengan menggunakan transistor dan rangkaian pembagi tegangan (*Attenuasi*).



Gambar 3.3 Rangkaian Level Logic Converter

Rangkaian *switching* transistor digunakan untuk menaikkan level tegangan dari level 3.3 V (CMOS) pada pin TX kontroler Master menjadi level tegangan 5 V (TTL) pada pin RX kontroler Slave. Dalam merancang rangkaian transistor yang digunakan untuk aplikasi switching maka parameter yang digunakan adalah arus basis (*Ib*). untuk dapat bekerja pada daerah saturasi, maka nilai $Ib \geq \frac{Ic}{Hfe}$ sedangkan untuk bekerja pada daerah cutoff, maka nilai $Ib < \frac{I_C}{Hfe}$. Pada perancangan dipakai dua buah transistor NPN dengan seri 2N2222 (hfe = 100, V_{BE}=0.6V, VCC=5 V, I_C=150 mA). Maka :

$$R_C = \frac{VCC}{I_C}$$

$$R_C = \frac{5}{0.15}$$

$$R_C = 33\,\Omega$$

Sedangkan nilai *Ib* dapat dicari dengan :

$$Ib = \frac{I_C}{Hfe}$$

$$Ib = \frac{0.15}{100}$$

$$Ib = 0.0015\ A\ atau\ 1.5\ mA$$

Dari persamaan $Ib = \frac{v - vbe}{Rb}$, maka :

$$Rb = \frac{v - vbe}{Ib}$$

$$Rb = \frac{3.3 - 0.6}{0.0015}$$

$$Rb = 1800\,\Omega$$

Sedangkan pada rangkaian kedua digunakan rangkaian pembagi tegangan atau *attenuasi* yang ditunjukkan pada Gambar 3.3 diatas dengan parameter nilai R5 dan R6.

Nilai output dapat ditentukan dengan mengatur parameter nilai R5 dan R6 dan nilai tegangan Input (Vin) seperti pada persamaan :

$$V_{out} = \left(\frac{R5}{R5 + R6}\right) \times Vin$$

Pada perancangan sistem nilai output yang diinginkan adalah sebesar 3.3 Volt dengan nilai Vin 5 Volt. Untuk menentukan nilai R5 dan R6 dapat ditentukan dengan cara menentukan terlebih sembarang nilai pada R5 atau R6. Misalkan disini ambil nilai 10K untuk R5 sedangkan nilai R6 dicari. Untuk itu nilai R6 dapat ditentukan dengan persamaan :

$$Vout = \left(\frac{R5}{R6 \times R5}\right) \times Vin$$

$$3.3 = \left(\frac{10000}{R6 \times 10000}\right) \times 5$$

$$3.3 = \left(\frac{50000}{R6 \times 10000}\right)$$

$$50000 = 3.3(R6 + 10000)$$

$$50000 = 3.3R6 + 33000$$

$$3.3R6 = 50000 - 33000$$

$$3.3R6 = 17000$$

$$R6 = \frac{17000}{3.3}$$

$$R6 = 5151\Omega$$

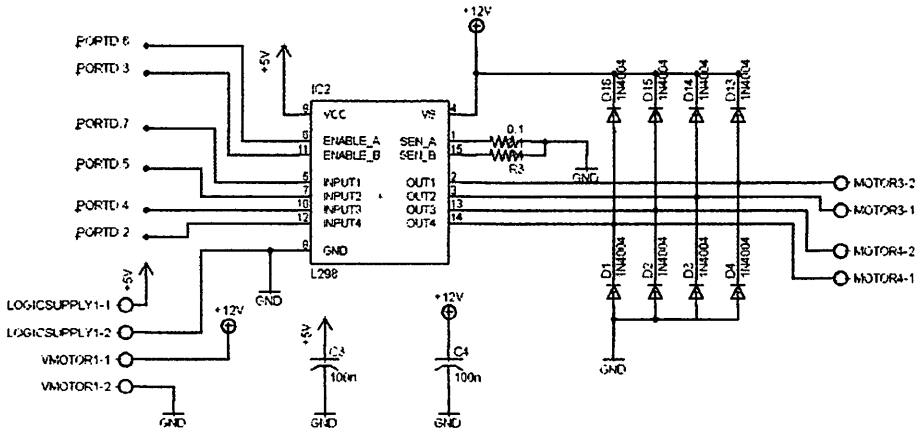
Pembulatan direkomendasikan menggunakan nilai di atas 5151 Ω yaitu sebesar 8200 Ω atau 8K2Ω (Vout = 2.7 Volt) dikarenakan pembulatan dengan nilai di bawah 5151Ω akan menyebabkan nilai output akan menjadi lebih besar dari 3.3 Volt.

3.3.2 Perancangan Rangkaian Driver Motor

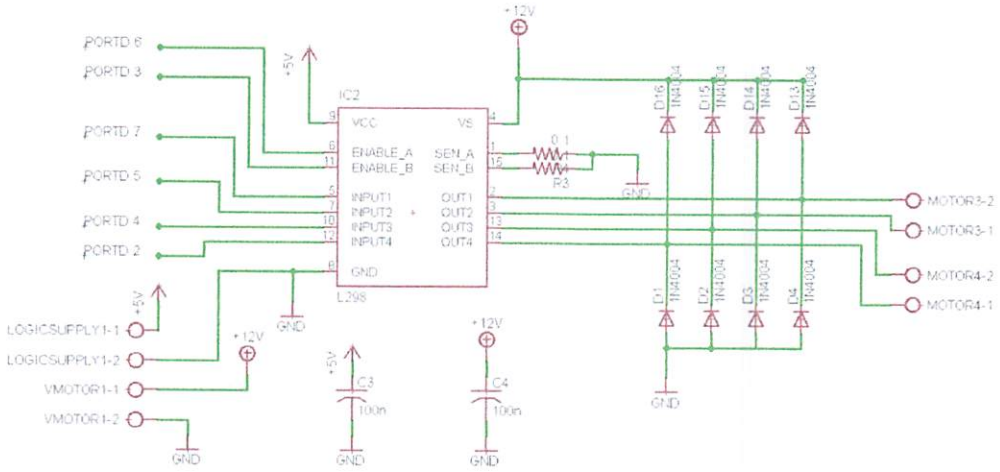
Fungsi utama dari rangkaian driver motor pada sistem ini adalah sebagai penguat arus yang keluar dari port mikrokontroler agar dapat menggerakkan motor dan sebagai driver yang dapat mengatur kecepatan putaran motor dengan fungsi PWM (*Pulse Width Modulation*). Rangkaian yang digunakan adalah driver motor menggunakan IC L298 dengan konsep H-Bridge yang dapat menggerakkan motor secara CW (*Counter Wise*) dan CCW (*Counter Clock Wise*).

Pada perancangan, IC yang digunakan adalah tipe L298 tanpa dioda Internal sehingga pada rangkaian perlu ditambahkan rangkaian dioda eksternal. Fungsi dioda pada rangkaian ini adalah sebagai pengaman IC dan rangkaian yang terhubung dari terjadinya EMF (*Electro Motor Force*) atau tegangan balik yang diakibatkan oleh putaran sisa motor.

Pada Gambar 3.4 ditunjukkan rangkaian driver motor dengan IC L298 dengan antarmuka 6 pin paralel yang dihubungkan dengan mikrokontroler slave ATMega162.



Gambar 3.4 Rangkaian Driver Motor L298



Gambar 3.4 Rangkaian Driver Motor L298

Pemberian kapasitor C3 dan C4 serta Resistor pada pin SENS_A dan SENS_B merupakan rekomendasi dari datasheet IC L298. Rekomendasi nilai yang digunakan adalah 100nF pada C3 dan C4 sedangkan R_{sens} menggunakan nilai 0.5Ω sampai dengan 1Ω. Dioda pengaman direkomendasikan memiliki nilai arus 2A dengan nilai trr ≤ 200nS (*trr = reverse recovery time*).

3.3.3 Perancangan Rangkaian Push Button (*User Input*)

Rangkaian push button pada perancangan sistem ini digunakan untuk menentukan pilihan mode wall following yaitu mode *wall following* kanan atau wall following dinding kiri. Rangkaian yang digunakan adalah menggunakan push button (*tactile switch*) tanpa menggunakan resistor pullup. Hal ini dikarenakan mikrokontroler slave (ATMega162) telah dilengkapi dengan fitur internal pullup sehingga pilihan untuk penambahan eksternal bersifat opsional. Konsep yang digunakan adalah dengan menggunakan aktif low yang berarti PINA.4..7 akan menunggu terjadi transisi dari keadaan *high* menjadi *low*. Rangkaian push button ditunjukkan pada Gambar 3.5 berikut :

Switch (S1) digunakan sebagai switch untuk pengaktifan power utama yang menuju ke seluruh rangkaian utama dengan konsep *latch circuit* yang dikombinasikan dengan *power* mosfet. Rangkaian *power switch* dibahas pada sub bab berikutnya. Switch (S2) digunakan untuk pengaktifan robot ketika power utama sudah diaktifkan sedangkan S3 digunakan untuk memberhentikan robot dari kondisi running ke kondisi *standby*. Switch S4 dan S5 digunakan untuk pemilihan mode yang digunakan apakah menggunakan mode *wall following* kiri atau kanan. Mode 1 untuk *wall following* kanan dan Mode 2 digunakan untuk mode *wall following* kiri.

3.3.4 Perancangan Rangkaian Indikator LED

Indikator pada perancangan sistem ini digunakan sebagai penanda untuk memberitahukan kondisi dan mode yang sedang berlangsung pada robot. Rangkaian ini menggunakan LED berwarna merah berukuran 3mm yang terhubung ke mikrokontroler slave. Hal yang perlu diperhatikan adalah kebutuhan arus forward yang dibutuhkan oleh LED untuk dapat menyala. Jika nilai arus yang disupply ke LED terlalu besar maka led dapat terbakar sedangkan jika arus yang mengalir ke led terlalu kecil maka led hanya akan menyala redup. Untuk membatasi nilai arus yang mengalir ke led maka diperlukan resistor yang dipasang seri dengan led. Persamaan untuk menetukan nilai R adalah :

$$R_S = \frac{Vs - Vf}{I_F}$$

Dimana :

R_S = Nilai resistor yang dipasang seri dengan led
V_S = Tegangan yang mengalir ke led
V_F = Tegangan Forward dioda led
I_F = Arus nominal yang dibutuhkan oleh dioda led

Jika diketahui dari datasheet nilai V_S yang mengalir dari Port mikrokontroler adalah 5 Volt, V_F led adalah 3.5V, dan I_F led = 25 mA maka nilai R_S dapat dicari sebagai berikut :

terhubung ke mikrokontroler slave. Hal yang perlu diperhatikan adalah kebutuhan arus forward yang dibutuhkan oleh LED untuk dapat menyala. Jika nilai arus yang disupply ke LED terlalu besar maka led dapat terbakar sedangkan jika arus yang mengalir ke led terlalu kecil maka led hanya akan menyala redup. Untuk membatasi nilai arus yang mengalir ke led maka diperlukan resistor yang dipasang seri dengan led. Persamaan untuk menetukan nilai R adalah :

$$= \frac{-}{\quad}$$

Dimana :

R_S = Nilai resistor yang dipasang seri dengan led
V_S = Tegangan yang mengalir ke led
V_F = Tegangan Forward dioda led
I_F = Arus nominal yang dibutuhkan oleh dioda led

Jika diketahui dari datasheet nilai V_S yang mengalir dari Port mikrokontroler adalah 5 Volt, V_F led adalah 3.5V, dan I_F led = 25 mA maka nilai R_S dapat dicari sebagai berikut :

$$= \frac{-}{\quad}$$

$$= \frac{5 - 3.5}{25 \quad 10}$$

$$= \frac{1.5}{0.025}$$

= 60 Ω dibulatkan menjadi 100 Ω

Menurut beberapa percobaan yang telah dilakukan berhubungan dengan nilai R_S , nilai 100Ω s/d 10 KΩ pada V_S = 5 V masih dapat digunakan dan led masih menyala namun semakin besar nilai R_S maka led menjadi redup.

mode *wall following* kanan dan jika LED1 menyala maka mode yang digunakan adalah mode *wall following* kiri.

3.4 Perancangan Perangkat Lunak

Melihat pada batasan masalah yang telah dikemukakan pada awal perancangan sistem, maka perancangan perangkat lunak hanyak akan difokuskan kepada perancangan kontroler master dan kontroler slave sedangkan untuk perancangan perangkat lunak untuk pengolahan citra tidak dibahas disini.
Perancangan perangkat lunak dibagi menjadi dua bagian yaitu :

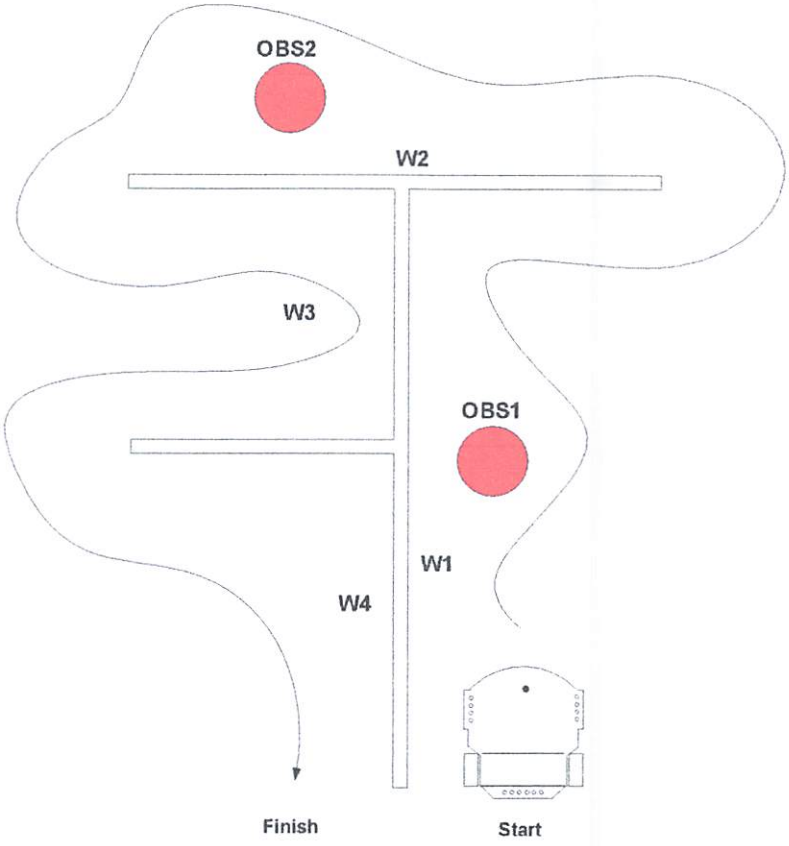
1. Perancangan kontroler Master, terdiri dari perancangan kontroler Fuzzy dan program untuk komunikasi USART dengan kontroler Slave.
2. Perancangan kontroler Slave yang terdiri dari program pembacaan sensor Ultrasonik Ping, program pembangkit PWM, program komunikasi USART ke kontroler Master dan program pembacaan input push button.

3.4.1 Perancangan Kontroler Master

Pembuatan program pada kontroler Master (*Raspberry Pi*) menggunakan bahasa pemrograman C++ dengan Compiler GNU GCC dan menggunakan library WiringPi (*www.wiringpi.com*) yang digunakan untuk mengakses GPIO (*General Purpose Input Output*) dari Raspberry Pi.

3.4.1.1 Perancangan *Path Planning*

Sebelum merancang perangkat lunak yang akan menentukan gerakan robot maka terlebih dahulu harus ditentukan perencanaan arah gerak robot dalam bentuk path planning. Path planning disini mempunyai pengertian yaitu suatu aksi atau gerakan robot dalam jalur untuk bergerak dari titik awal menuju titik akhir. Dalam perjalanan robot dari titik start menuju titik akhir terdapat halangan yang akan menghalangi gerakan robot dalam bernavigasi. Urutan pergerakan robot ditunjukkan pada Gambar 3.7 dengan urutan robot berangkat dari posisi start kemudian menelusuri dinding W1 dan menghindar halangan OBS1, robot kemudian bergerak menelusuri dinding W2 dan menghindar halangan OBS2 kemudian menuju dinding W3 dan terakhir melewati dinding W4 sebelum mencapai titik akhir (*finish*).



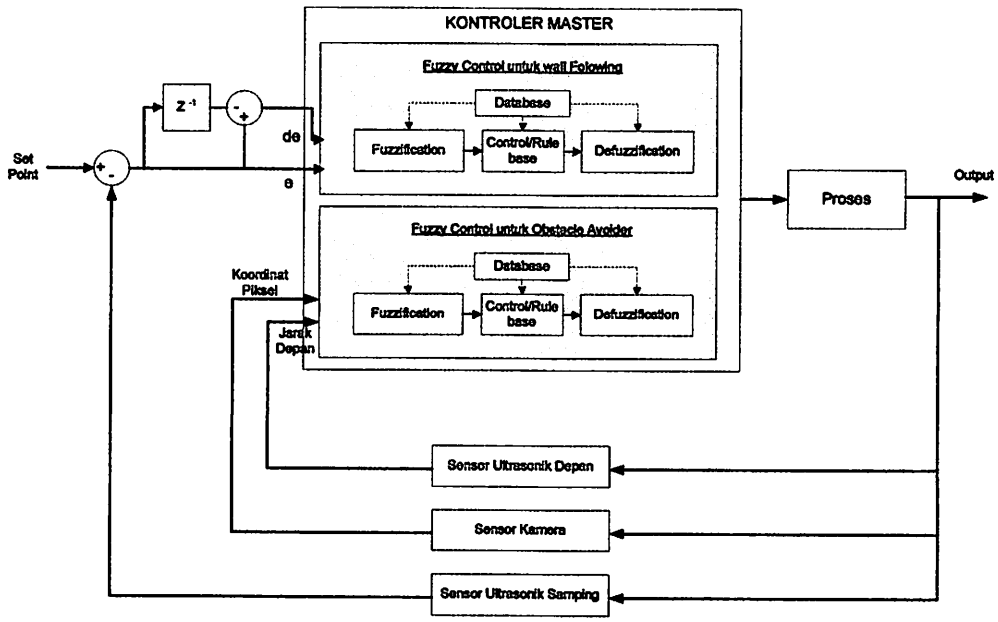
Gambar 3.7 Path Planning

Indikator keberhasilan pergerakan robot adalah kemampuan robot untuk bergerak dari titik start ke titik finish sesuai dengan tanda arah yang diberikan dan mampu menghindari halangan serta tanpa menyentuh dinding.

3.4.1.2 Perancangan Sistem Kendali Fuzzy

Pada perangkat lunak master, konsep kendali untuk penelusur dinding menggunakan konsep kendali logika fuzzy dengan menerapkan karakteristik dari PD kontroler (*Proportional Derrivative*) sehingga parameter input yang diberikan ke kontroler adalah nilai error dan delta error. Sedangkan untuk penghindar halangan, parameter input yang digunakan adalah jarak robot terhadap halangan dan posisi koordinat piksel halangan. Jarak robot terhadap halangan dibaca dengan menggunakan sensor ultrasonik sedangkan posisi koordinat piksel halangan dibaca oleh sensor kamera (Raspberry Camera).

Struktur kendali kontroler Master ditunjukkan pada gambar berikut :



Gambar 3.8 Diagram Blok Sistem kendali

Penjelasan diagram blok :

[1] Set Point, acuan nilai jarak ideal posisi robot terhadap dinding

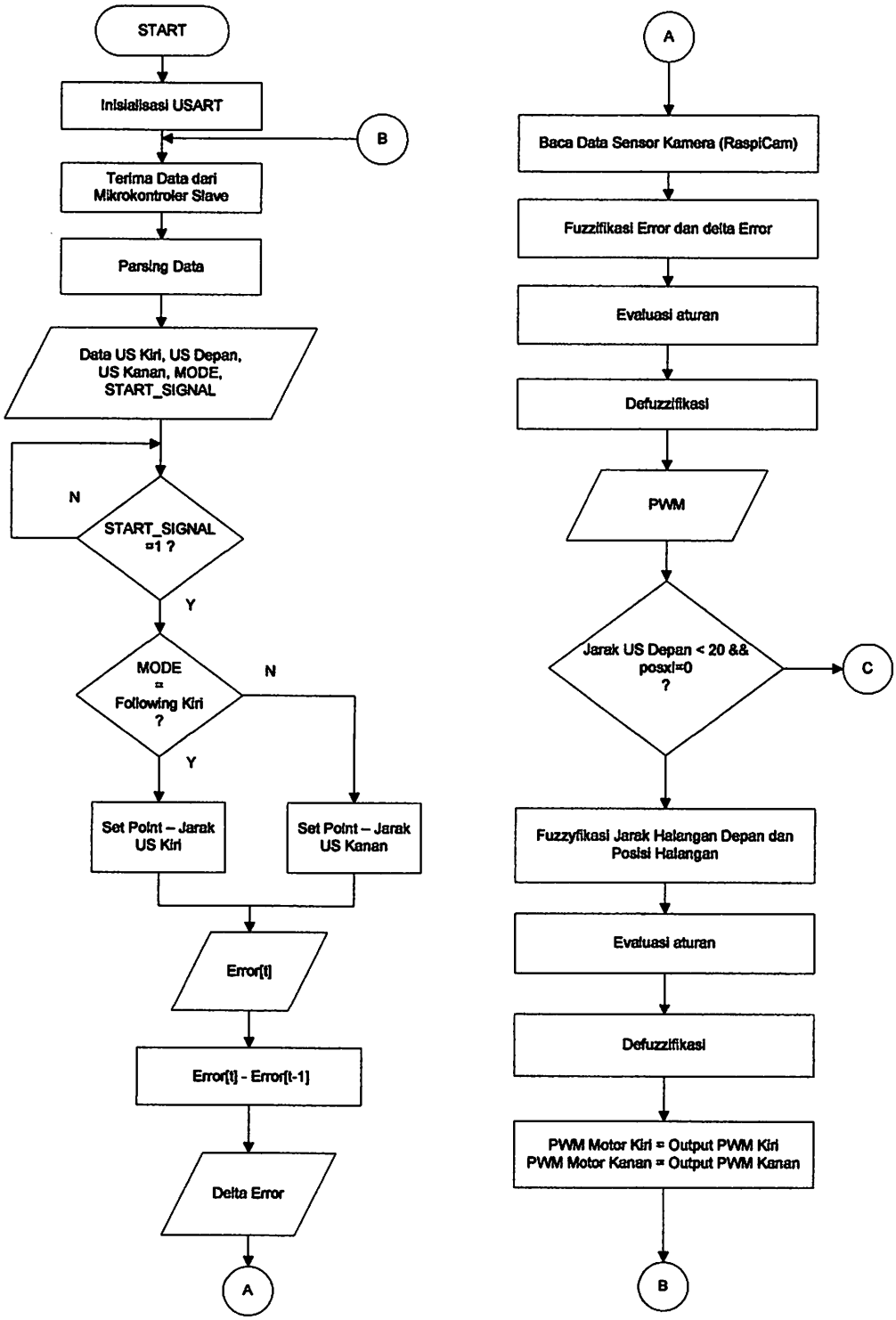
[2] E atau error adalah besar penyimpangan nilai set point dengan nilai aktual posisi robot terhadap dinding

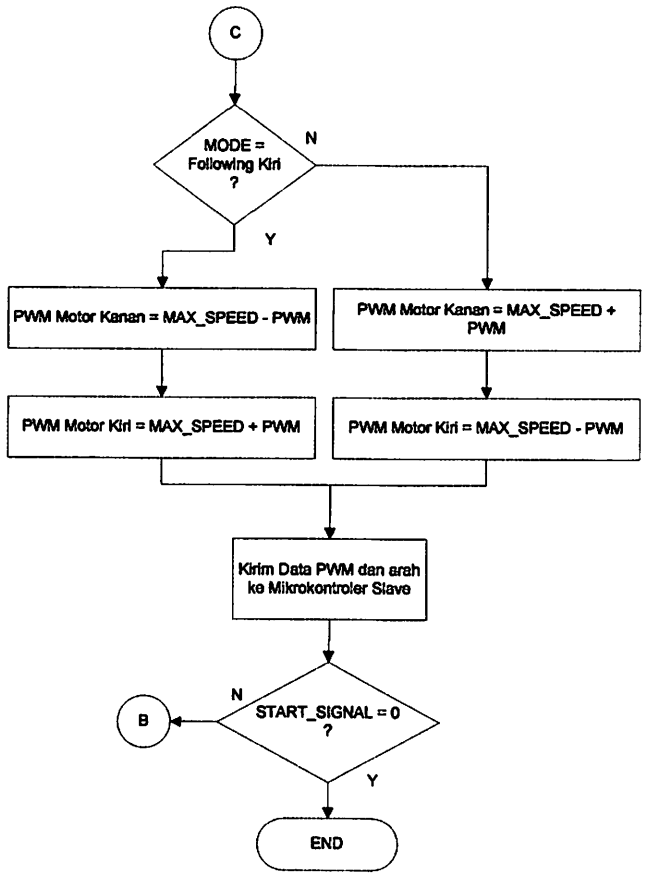
[3] dE yaitu nilai delta error yang didapat dari nilai error sekarang E[t] dikurangi dengan error sebelumnya E[t-1].

[4] Process, yaitu blok proses untuk mengubah hasil defuzzifikasi ke dalam nilai PWM Motor

[5] Jarak sensor ultrasonik Depan, yaitu nilai jarak robot terhadap halangan yang berada di depan

[6] Sensor kamera, sensor yang digunakan untuk mendeteksi warna dari objek


START
Inisialisasi USART
B
Terima Data dari Mikrokontroler Slave
Parsing Data
Data US Kiri, US Depan, US Kanan, MODE, START_SIGNAL
N
START_SIGNAL =1 ?
Y
MODE = Following Kiri ?
N
Y
Set Point – Jarak US Kiri
Set Point – Jarak US Kanan
Error[t]
Error[t] - Error[t-1]
Delta Error
A
A
Baca Data Sensor Kamera (RaspiCam)
Fuzzifikasi Error dan delta Error
Evaluasi aturan
Defuzzifikasi
PWM
Jarak US Depan < 20 && posxi=0 ?
C
Fuzzyfikasi Jarak Halangan Depan dan Posisi Halangan
Evaluasi aturan
Defuzzifikasi
PWM Motor Kiri = Output PWM Kiri
PWM Motor Kanan = Output PWM Kanan
B



Gambar 3.9 Flowchart Kontroler Master

A. Fuzzyfikasi

Model yang digunakan untuk proses fuzzyfikasi adalah dengan menggunakan model *triangular function* (fungsi keanggotaan segitiga) dimana formula untuk menentukan derajat keanggotaannya adalah :

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

Dimana :

u = Nilai input *Crisp*

a = batas minimum nilai fungsi keanggotaan segitiga

b = nilai tengah fungsi keanggotaan segitiga

c = batas maksimum nilai fungsi keanggotaan segitiga

Pada perancangan sistem terdapat 3 variabel input fuzzy yaitu nilai error, delta error dan jarak halangan depan. Error dan delta error didapat dari penyimpangan nilai jarak aktual robot terhadap nilai set point (jarak ideal robot terhadap dinding). Error yang terjadi dapat dicari dengan persamaan sebagai berikut :

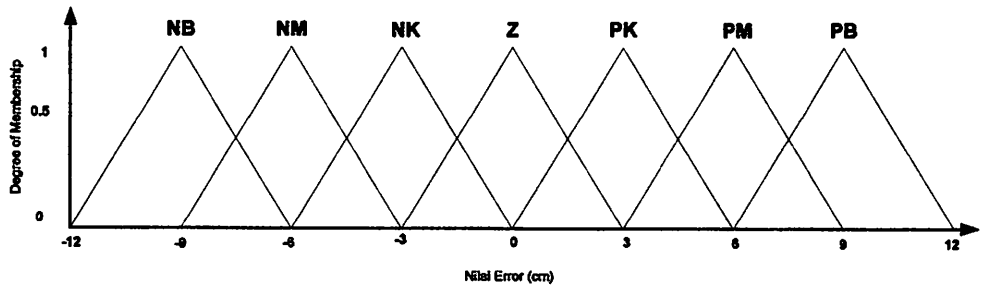
$$Error = set\ point - jarak\ aktual\ robot$$

Nilai error bergantung kepada besar nilai aktual posisi robot terhadap dinding yang artinya kemungkinan nilai error dapat bernilai positif dan negatif dengan posisi ideal berada pada nilai 0. Variabel input error dibagi menjadi 7 fungsi keanggotaan yang merepresentasikan nilai penyimpangan dalam format positif dan negatif.

Tabel 3.2 Range Fungsi Keanggotaan Error

Fungsi Keanggotaan Error	Range		
	a	b	c
NB (Negatif Besar)	-12	-9	-6
NM (Negatif Medium)	-9	-6	-3
NK (Negatif Kecil)	-6	-3	0
Z (Zero)	-3	0	3
PK (Positif Kecil)	0	3	6
PM (Positif Medium)	3	6	9
PB (Positif Besar)	6	9	12

Grafik membership function untuk error digambarkan sebagai berikut :



Grafik 3.1 Grafik Fungsi keanggotaan error

Keterangan :

NB = Negatif Besar
NM = Negatif Medium
NK = Negatif Kecil
Z = Zero
PK = Positif Kecil
PM = Positif Medium
PB = Positif Besar

Untuk mengurangi dampak dari jalannya robot yang tidak stabil dan cenderung bergelombang maka efek ini dapat dikurangi dengan menggunakan prinsip pemberian input delta error. Konsep delta error adalah mengukur seberapa jauh penyimpangan robot pada saat sekarang *E(t)* dengan error yang terjadi sebelumnya *E(t-1)* sehingga dapat ditentukan sinyal kontrol yang sesuai untuk dapat mengembalikan posisi pada keadaan ideal.

$$Delta\ Error = error(t) - error(t-1)$$

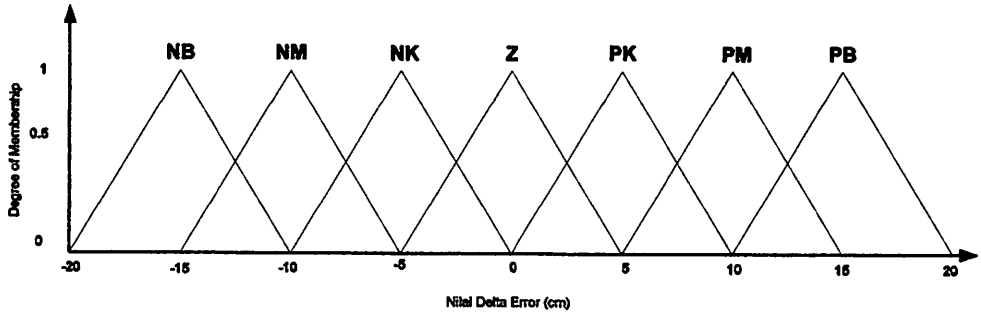
Pengaruh pemberian input delta error pada suatu sistem adalah memberikan efek redaman terhadap terjadinya *overshoot* atau *undershoot* dan memperbaiki respon *transien* karena nilai delta error akan timbul seketika terjadi perubahan error.

Nilai delta error dapat bernilai positif dan negatif dengan nilai 0 sebagai posisi ideal. Untuk itu fungsi keanggotaan delta errror dibagi menjadi 5 seperti dilihat pada Tabel 3.6

Tabel 3.3 Range Fungsi Keanggotaan Delta Error

Fungsi Keanggotaan Delta Error	**Range**		
	a	**b**	**c**
NB (Negatif Besar)	-20	-15	-10
NM (Negatif Medium)	-15	-10	-5
NK (Negatif Kecil)	-10	-5	0
Z (Zero)	-5	0	5
PK (Positif Kecil)	0	5	10
PM (Positif Medium)	5	10	15
PB (Positif Besar)	10	15	20

Grafik fungsi keanggotaan segitiga dari delta error dapat dilihat pada gambar berikut :



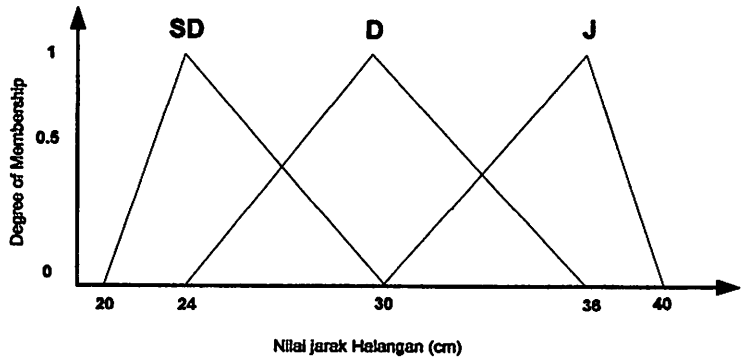
Grafik 3.2 Grafik Fungsi Keanggotaan Delta Error

Input kontroler fuzzyfikasi untuk penghindar halangan dibagi menjadi 2 yaitu masukan jarak robot terhadap halangan dan posisi koordinat piksel horizontal halangan. Fungsi keanggotaan jarak Halangan dibagi menjadi 3 yaitu :

Tabel 3.4 Range Fungsi Keanggotaan Jarak Halangan

Fungsi Keanggotaan Jarak Halangan	Range		
	a	b	c
SD (Sangat Dekat)	20	16	20
D (Dekat)	16	20	30
J (Jauh)	20	30	40

Grafik fungsi keanggotaan segitiga dari Jarak Halangan depan adalah sebagai berikut :

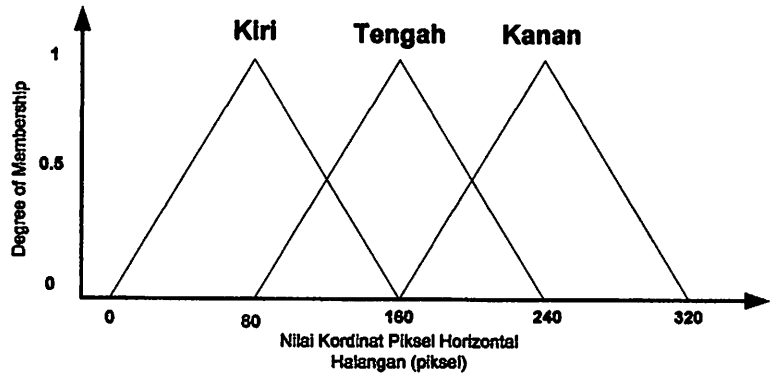


Grafik 3.3 Grafik Fungsi Keanggotaan Jarak Halangan

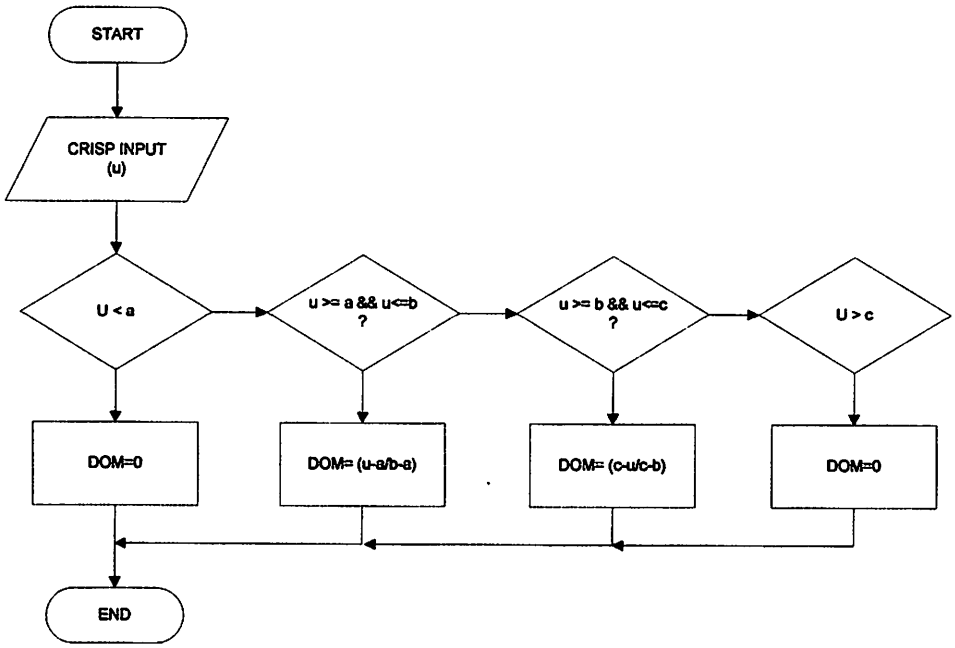
Proses deteksi menggunakan nilai koordinat piksel dari objek yang dideteksi dengan mengambil nilai piksel horizontal sehingga range nilai posisi objek dapat bernilai antara 0 sampai dengan 320 sesuai dengan resolusi kamera. Dengan demikian fungsi keanggotaan posisi objek dibagi menjadi 3 yaitu :

Tabel 3.5 Range Fungsi Keanggotaan Posisi Halangan

Fungsi Keanggotaan Posisi Halangan	Range		
	a	b	c
KI (Kiri)	0	80	160
TN (Tengah)	80	160	240
KA (Kanan)	160	240	320



Grafik 3.4 Grafik Fungsi Keanggotaan Posisi Halangan



Gambar 3.10 Flowchart Proses Fuzzyfikasi

B. Evaluasi Rule

Agar suatu sistem dapat berjalan sesuai dengan yang diharapkan maka proses evaluasi rule memegang peranan yang penting dalam sistem berbasis fuzzy logic. Konsep evaluasi rule adalah memasukkan pola pikir manusia ke dalam suatu sistem otomatis sehingga efektifitas sistem dalam menjalankan tugasnya sama dengan sistem dengan kendali manual (dioperasikan oleh manusia). Proses penentuan rule dilakukan dengan menggunakan logika penalaran berdasarkan pengetahuan dari operator (manusia) yang telah mengerti bagaimana sistem itu bekerja secara optimal.

Dalam perancangan sistem ini, tujuan utama robot adalah menjaga jarak aktual agar nilai error dapat seminimal mungkin dan presentase tabrakan (*collision*) sekecil mungkin. Cara yang umum dipakai adalah dengan menanyakan ke operator manusia yang dengan cara manual telah mampu mengendalikan sistem tersebut atau dikenal dengan isitilah *human expert experience*. Namun adakalanya dalam melakukan pengendalian, manusia cenderung melakukannya atas dasar refleks dan intuitif terutama kegiatan/aksi yang dilakukan berulang-ulang sehingga penalaran yang dilakukan oleh manusia akan sulit untuk dituangkan dalam bentuk data rule. Maka dari itu dalam perancangan sistem ini,

metode pengambilan data evalusai rule adalah mengadaptasi konsep PD Kontroler pada kontrol PID dengan melihat karakteristik dari masing-masing komponen PID.

Parameter *Proportional* (P) memiliki karakteristik untuk memberikan gain terhadap error agar sistem dapat kembali pada posisi set point dengan *rise time* yang cepat sedangkan parameter *Derivatif* (D) memiliki karakteristik dalam meredam terjadinya *overshoot/undershoot* dengan kemampuan untuk memprediksi terjadinya error. Gabungan dari kedua parameter tersebut sudah cukup dalam proses pengendalian sistem.

Pada kontroler untuk penelusur dinding menggunakan konsep MISO (*Multiple Input Single Output*) dengan masukan x_1 = error dan x_2 = delta error dan keluaran y_1 = pwm sedangkan untuk kontroler penghindar halangan menggunakan konsep MIMO (*Multiple Input Multiple Output*) dengan masukan x_1 = jarak halangan dan masukan x_2 = posisi kordinat halangan sedangkan keluaran y_1 berupa nilai PWM kiri dan keluaran y_2 adalah nilai PWM Kanan.

Metode evaluasi rule yang digunakan adalah metode mamdani dengan aturan dasar sebagai berikut :

$R^{(1)}$: IF x_1 is F_1^1 **AND** .... **AND**.... X_n is F_n^1 , THEN y_1 is G^1 AND y_2 is G^2

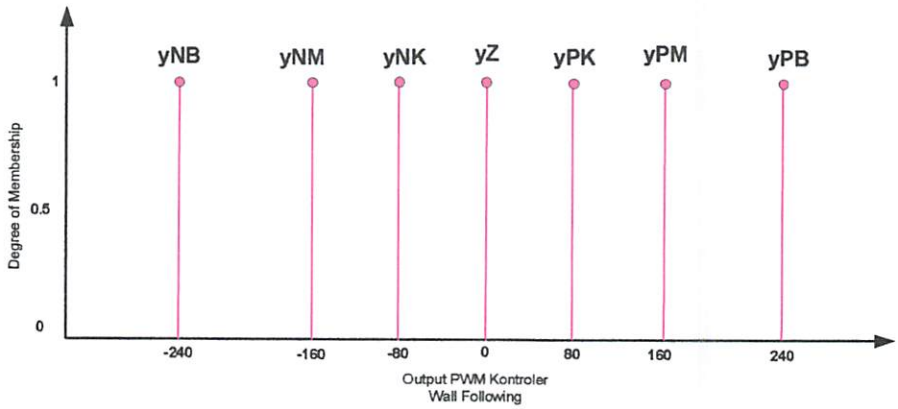
Dimana :

X_n = *Antecedent*

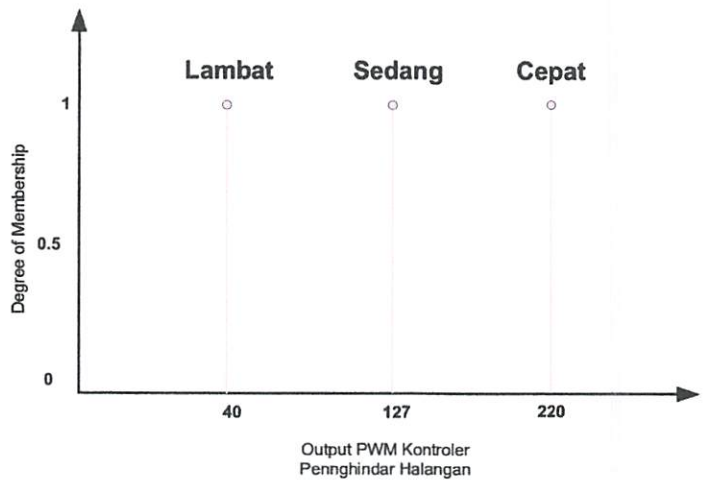
Y_n = *Consequent*

Hasil output evaluasi *rule* berupa nilai titik pusat dari fungsi keanggotaan output yang digunakan untuk proses defuzzyfikasi dengan menggunakan metode COA (*Center of Area*) atau COG(*Center of Gravity*)

Membership output PWM dapat dilihat pada gambar berikut :



Grafik 3.5 Fungsi Keanggotaan Output PWM pada kontroler fuzzy Untuk Penelusur Dinding



Grafik 3.6 Fungsi Keanggotaan Output Pada Kontroler Fuzzy untuk Penghindar Halangan

Jumlah aturan yang digunakan tergantung kepada jumlah membership tiap input fuzzy sehingga jumlah total aturan yang dipakai adalah 49 aturan untuk kontroler fuzzy penelusur dinding dan 9 aturan untuk kontroler fuzzy penghindar halangan. Relasi antara antecedent menggunakan hubungan AND sehingga nilai output didasarkan kepada nilai derajat keanggotaan antecedent minimum.

Rule yang dipakai dapat dilihat pada tabel berikut :

Tabel 3.6 Aturan Pada Kontroler fuzzy untuk Penelusur Dinding

de / e	NB	NM	NK	Z	PK	PM	PB
NB	*yNB*	*yNB*	*yNB*	*yNK*	*yNM*	*yNM*	*yZ*
NM	*yNB*	*yNB*	*yNM*	*yNM*	*yNM*	*yZ*	*yPM*
NK	*yNB*	*yNM*	*yNM*	*yNK*	*yZ*	*yPM*	*yPM*
Z	*yNM*	*yNM*	*yNK*	*yZ*	*yPK*	*yPM*	*yPM*
PK	*yNM*	*yNM*	*yZ*	*yPK*	*yPM*	*yPM*	*yPB*
PM	*yNM*	*yZ*	*yPM*	*yPM*	*yM*	*yPB*	*yPB*
PB	*yZ*	*yPM*	*yPM*	*yPM*	*yPB*	*yPB*	*yPB*

Keterangan :

Antecedent :

NB = Negatif Besar

NM = Negatif Medium

NK = Negatif Kecil

Z = Zero (Nol)

PK = Positif Kecil

PM = Positif Medium

PB = Positif Besar

Consequent :

yNB = Negatif Besar

yNM = Negatif Medium

yNK = Negatif Kecil

yZ = Zero (Nol)

yPK = Positif Kecil

yPM = Positif Medium

yPB = Positif Besar

Tabel 3.7 Aturan pada Kontroler fuzzy untuk Penghindar Halangan

Posisi / Jarak	KI	TN	KA
SD	mKI=C	mKI=C	mKI=S
	mKA=L	mKA=L	mKA=L
D	mKI=C	mKI=S	mKI=S
	mKA=L	mKA=L	mKA=L
J	mKI=C	mKI=S	mKI=S
	mKA=S	mKA=L	mKA=L

Keterangan Tabel

Antecedent :

KI = Posisi Halangan Kiri

TN = Posisi Halangan Tengah

KA = Posisi Halangan Kanan

SD = Jarak Halangan Sangat Dekat

D = Jarak Halangan Dekat

J = Jarak Halangan Jauh

Consequent :

mKI = Motor Kiri

mKA = Motor Kanan

L = Kecepatan PWM Lambat

S = Kecepatan PWM Sedang

C = Kecepatan PWM Cepat

C. Defuzzyfikasi

Proses defuzzyfikasi adalah proses yang akan dilakukan setelah proses evaluasi rule dimana data variabel linguistik akan diubah kembali menjadi bentuk tegasnya untuk dapat diolah menjadi nilai gerak robot dalam bentuk pwm dan set point pwm. Dalam perancangan sistem, metode defuzzyfikasi menggunakan metode Weighted Area atau COA (*Center of Area*) dimana setiap membership output akan diambil nilai tengahnya untuk dikalikan dengan derajat keanggotaan

variabel input minimum dan dibagi dengan jumlah derajat keanggotaan setiap rule. Persamaan yang digunakan adalah :

$$z^* = \frac{\sum_{k=1}^{m} y_k \mu_v(y_k)}{\sum_{k=1}^{m} \mu_v(y_k)} \quad \text{.....(3.14)}$$

Dimana :

Z^* = nilai keluaran

M = jumlah rule

y_k = elemen ke - k

$\mu_Z(y_n)$ = derajat keanggotaan elemen-elemen pada fuzzy set *y*.

y = semesta pembicaraan

Metode COA/COG dipilih dikarenakan metode ini tidak membutuhkan perhitungan yang terlalu rumit sehingga memudahkan dalam pengimplementasian pada sistem embedded.

3.4.2 Perancangan Kontroler Slave (ATMega162)

Minimum sistem berbasis ATMega162 pada perancangan sistem ini difungsikan sebagai kontroler slave yang akan menerima data dari kontroler Slave melalui komunikasi serial (USART). Kontroler Slave memiliki 3 fungsi yaitu :

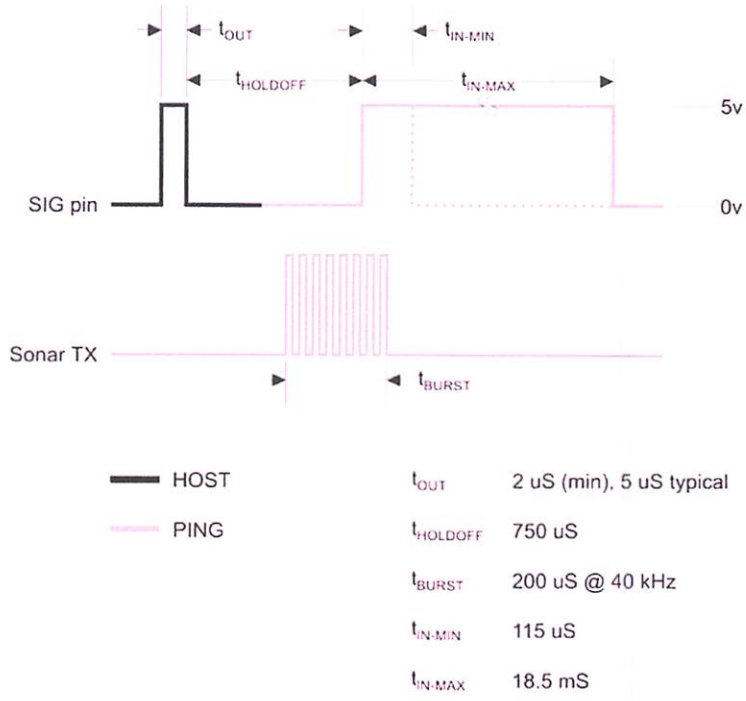
1. Membaca data sensor ultrasonik.
2. Membaca input push button untuk menentukan mode dan sinyal start/stop yang akan digunakan untuk mengaktifkan robot.
3. Membangkitkan sinyal PWM dengan menggunakan fitur Timer 8 bit.

Pembuatan program kontroler Slave menggunakan bahasa C dengan bantuan Software CodeVisionAVR 2.05.3. Program ini menggunakan konsep *Rapid Application Development* dimana telah dilengkapi dengan fitur-fitur dan library bantuan yang lebih memudahkan programmer untuk membuat program dengan cepat dan efisien.

3.4.2.1 Pembacaan Data Jarak Sensor Ultrasonik

Pada dasarnya prinsip kerja sensor ultrasonik ping adalah mengirimkan gelombang suara ultrasonik dan menerima pantulannya sehingga jarak dapat dicari dengan menghitung lama waktu pantulan suara diterima pada bagian receiver. Ping hanya akan mengirimkan suara ultrasonik ketika ada pulsa trigger dari mikrokontroler (pulsa *high* selama 5 us). Suara ultrasonik dengan frekuensi sebesar 40 Khz akan dipancarkan selama 200uS. Suara ini akan merambat di udara dengan kecepatan 344.424m/s (atau 1 cm setiap 29.034uS). selama menunggu pantulan, ping akan menghasilkan sebuah pulsa. Pulsa ini akan berhenti (*low*) ketika suara pantulan terdeteksi oleh ping. Selanjutnya mikrokontroler akan mengukur lama waktu dari sinyal dpancarkan sampai sinyal diterima kemudia mengkonversinya dalam bentuk jarak dengan perhitungan sebagai berikut :

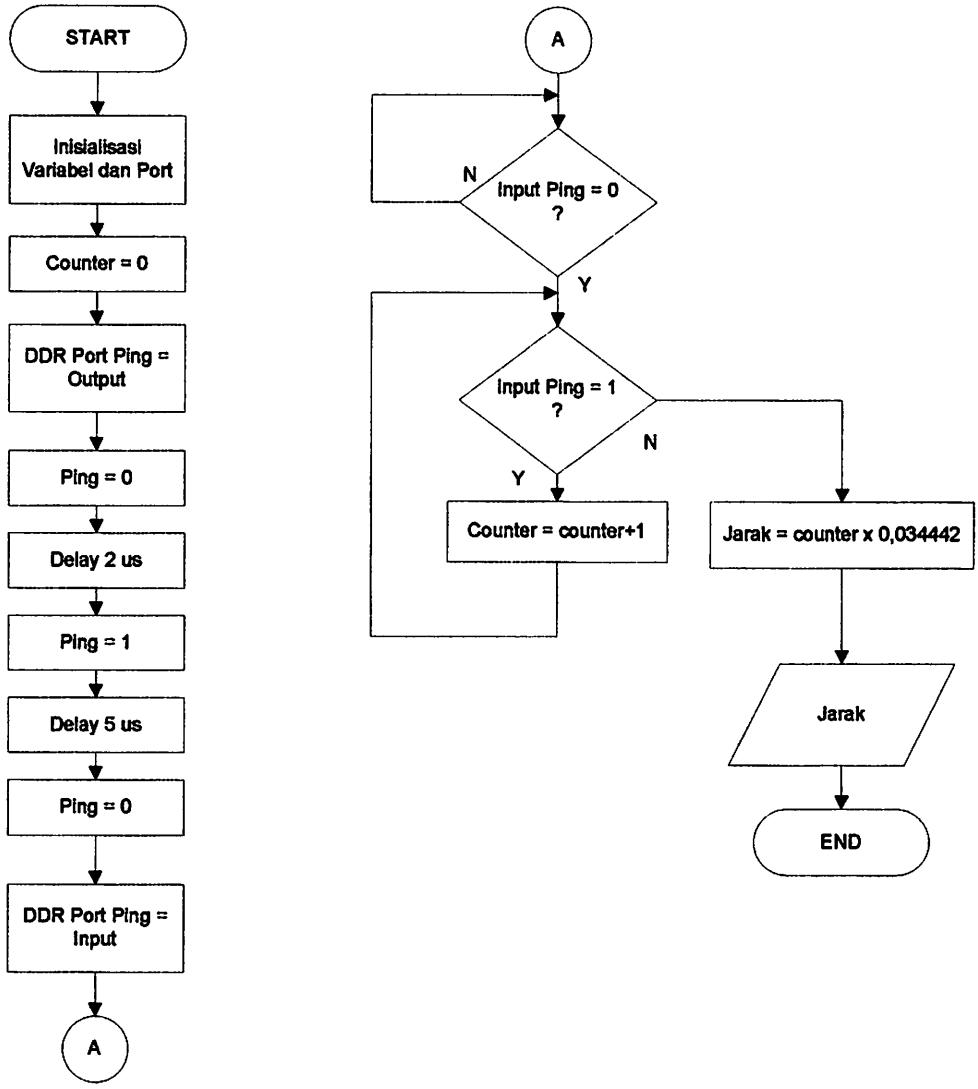
$$Jarak\ (cm) = \frac{Periode\ pulsa \times 0.034442}{2}$$



Gambar 3.11 Pulsa Ping Ultrasonik Range Finder

Tabel 3.8 Keterangan pin Sensor Ultrasonik

Sensor	**Keterangan Pin Mikrokontroler**
Sensor Ping Kiri	PINA.0
Sensor Ping Depan	PINA.1
Sensor Ping Kanan	PINA.2



Gambar 3.12 Flowchart Pembacaan data Sensor Ping

3.4.2.2 Pembacaan Push Button

Push button pada sistem ini difungsikan sebagai input dari user untuk menentukan mode dan sebagai cara mengaktifkan/menonaktifkan robot. Mode yang digunakan pada sistem ini adalah menentukan mode untuk proses mengikuti dinding kanan atau kiri. Data mode yang diinputkan selanjutnya akan disimpan pada EEPROM agar ketika robot dimatikan, maka mode yang terakhir dipilih dapat langsung digunakan sehingga tidak perlu memasukkan pilihan lagi.

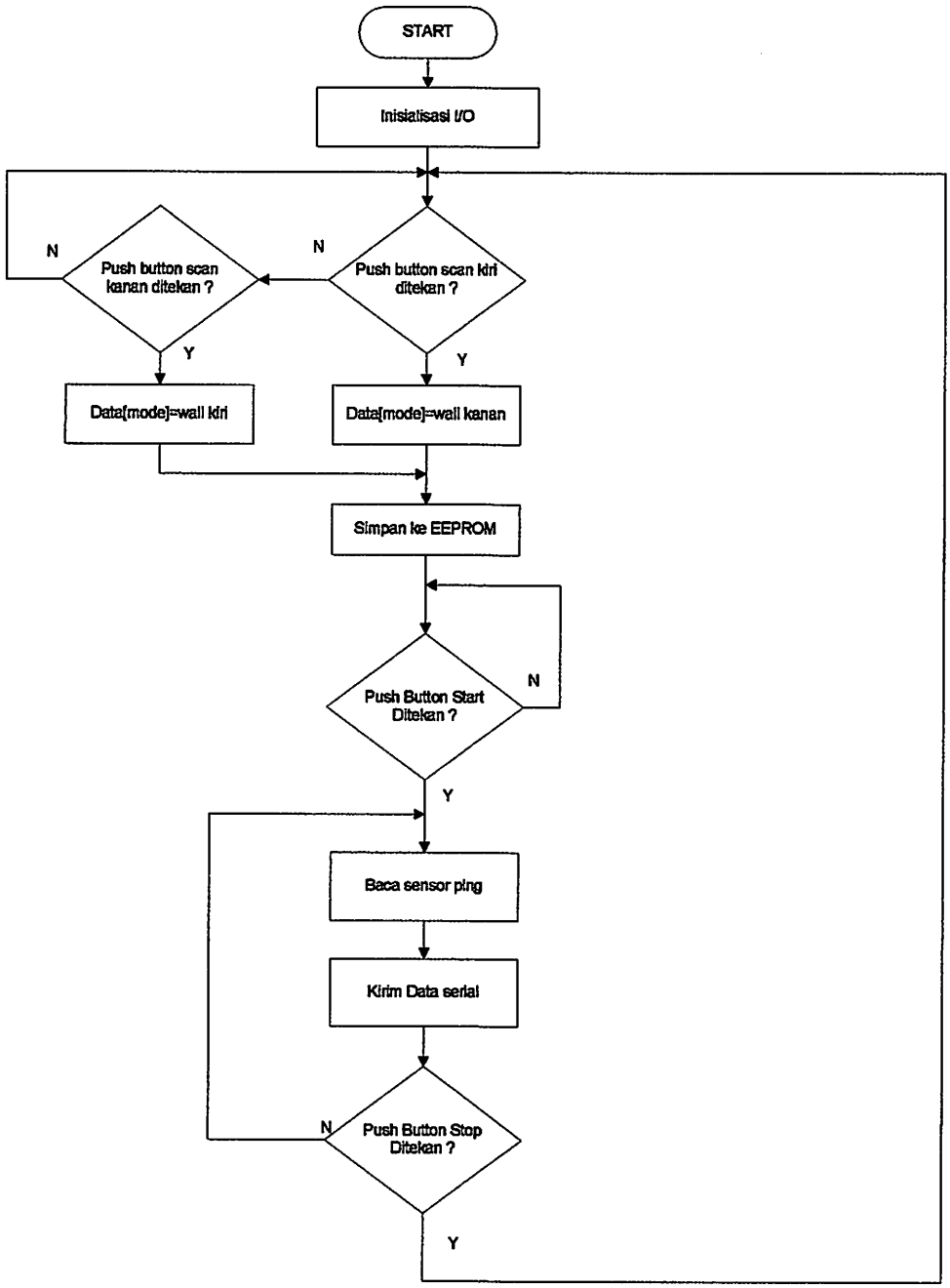
Push button disusun dengan mode aktif low yang artinya apabila terjadinya penekanan maka pin akan terhubung ke ground (GND). Penyusunan program mikrokontroler akan mengkonfigurasikan port sebagai pin input dengan mengatur register DDR (*Data Direction Register*) pada nilai 0 (*low*) dan mengaktifkan pullup internal yaitu mengatur register PORT pada nilai 1 (*high*).

Gambar 3.13 Bentuk Fisik Rangkaian Push Button

Tabel 3.9 Keterangan Pin Push Button

Push Button	**I/O Status**	**Pullup Status**	**Register**		Fungsi
			DDR	PORT	
S1*					Power ON/OFF
S2	Input	Diaktifkan	DDRA.3=1	PORTA.3=1	Start
S3	Input	Diaktifkan	DDRA.4=1	PORTA.4=1	Stop/Standby
S4	Input	Diaktifkan	DDRA.5=1	PORTA.5=1	Mode Tracking dinding kiri
S5	Input	Diaktifkan	DDRA.6=1	PORTA.6=1	Mode Tracking dinding kanan

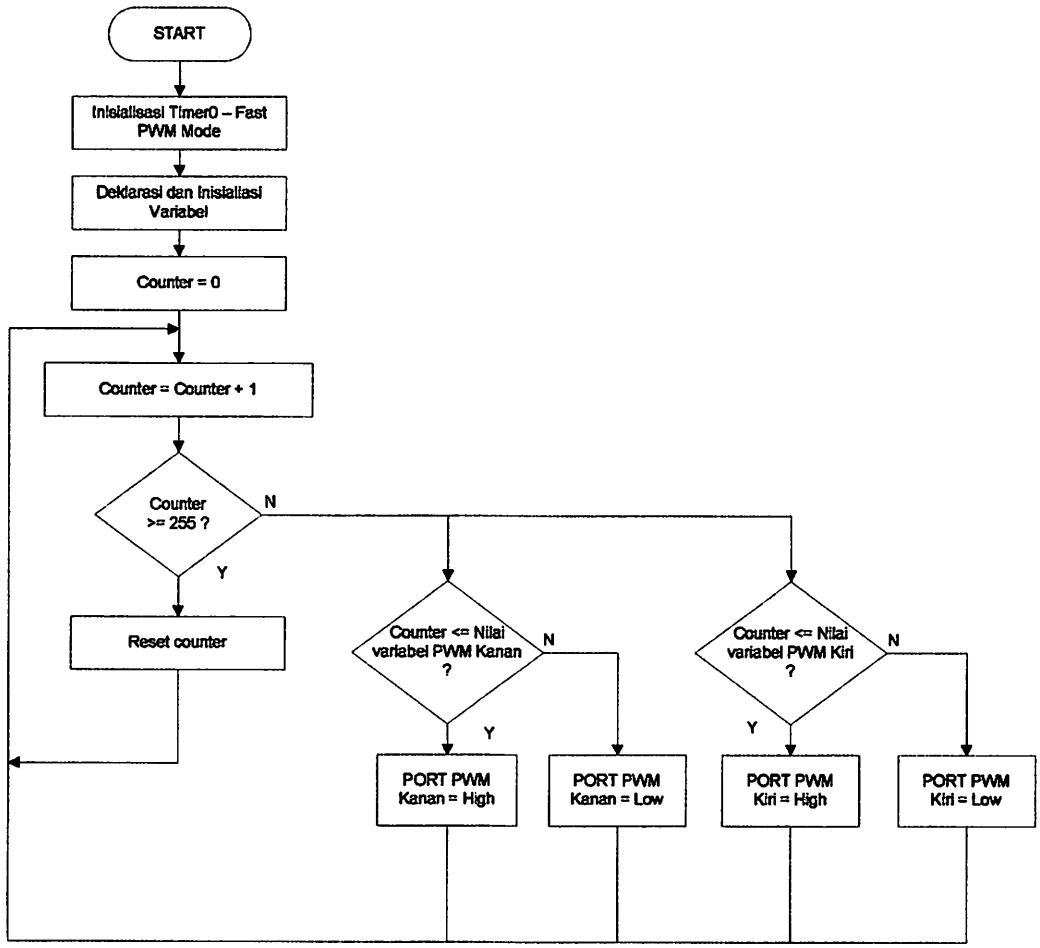
* Dihubungkan ke rangkaian Main Power Switch



Gambar 3.14 Flowchart Pembacaan Input Push Button

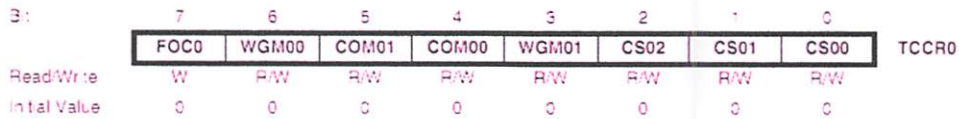
3.4.2.3 Pembangkitan PWM (*Pulse Width Modulation*)

PWM (*Pulse Width Modulation*) pada mikrokontroler dapat dibangkitkan dengan menggunakan fitur interrupt timer. Pada mikrokontroler ATMega162 terdapat 2 timer dengan resolusi 8 bit dan 2 timer dengan resolusi 16 bit. Pada perancangan sistem dipakai timer dengan resolusi 8 bit yaitu menggunakan timer 0. Flowchart program pembangkitan PWM dapat dilihat pada gambar berikut :



Gambar 3.15 Flowchart Pembangkitan PWM dengan Timer 0

Pada Flowchart di atas, proses pembangkitan PWM dimulai dengan menginisialisasi Timer 0 untuk bekerja pada mode Normal dan menggunakan fitur Overflow Interrupt . Pengaturan Mode dan fitur yang digunakan pada timer dapat ditentukan pada register TCCR0 (*Timer/Counter Control Register*) .



Gambar 3.16 Register *TCCR0* (Timer/Counter Control Register)

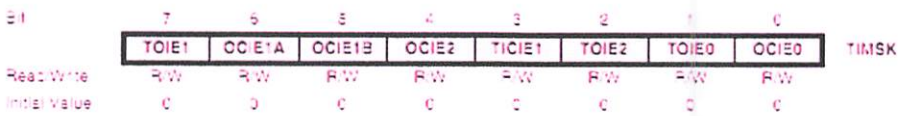
Menurut datasheet ATMega162, Mode *Normal* dapat diaktifkan dengan memberikan logic 0 pada bit *WGM00* dan *WGM01* (*WGM00* = 0, *WGM01*=0). Sedangkan untuk mengatur frekuensi timer yang digunakan dapat diatur dengan memberikan nilai pada bit *CS00,CS01* dan *CS02*. Pada perancangan sistem ini menggunakan frekuensi Timer sebesar 11059200 (*prescaler* = *1*) sehingga nilai CS02=0, CS01=0 dan CS00=1. Penentuan nilai ini didasarkan pada deskripsi bit Clock Select pada Tabel 3.10 berikut :

Tabel 3.10 Nilai Clock Select Timer 0

Frekuensi Clock	**Prescaler**	**F Timer**	**Clock Select**		
			CS02	**CS01**	**CS00**
11059200 Hz	1	11059200 Hz	0	0	1
	8	1382400 Hz	0	1	0
	64	172800 Hz	0	1	1
	256	43200 Hz	1	0	0
	1024	10800 Hz	1	0	1

*). Sumber : Datasheet ATMega162

Karena pembangkitan mode PWM ini menggunakan fitur rutin Overflow Interrupt maka bit *TOIE0* pada register *TIMSK* diset 1.



Gambar 3.17 Register TIMSK (Timer Interrupt Mask Register)

Timer/Counter 8 bit dapat menghitung maksimal hingga 255 (0 – 255), dimana periode setiap perhitungan (*Clock bergantung dari nilai prescaler*)
Sehingga untuk menentukan frekuensi pwm (f_{pwm}) dapat dihitung dengan persamaan :

$$f_{pwm} = \frac{1}{\left(\frac{Top\ Value \times PWM\ value}{f_{osc} / N}\right)}$$

Dimana :

F_{pwm} = Frekuensi Output PWM

Top Value = Nilai maksimal Hitungan *Timer*

PWM value = Nilai PWM 8 bit

fosc = Frekuensi Kristal yang dipakai

N = Prescaler (*1,8,64,256 dan 1024*)

Pada perancangan sistem diketahui Frekuensi Kristal yang dipakai = 11059200 Hz, prescaler = 1 (*no prescaler*), top value = 255 (*Timer 8 bit*) dan PWM Value = 255 (maks) maka frekuensi pwm output dapat dicari dengan :

$$f_{pwm} = \frac{1}{\left(\frac{Top\ Value \times PWM\ value}{f_{osc} / N}\right)}$$

$$f_{pwm} = \frac{1}{\left(\frac{255 \times 255}{11059200/1}\right)}$$

$$f_{pwm} = \frac{1}{\left(\frac{65025}{11059200}\right)}$$

$$f_{pwm} = \frac{1}{(0.0058797)}$$

$$f_{pwm} = 170.07 \text{ Hz}$$

Untuk nilai Periode PWM dapat dihitung dengan persamaan :

$$T_{PWM} = \frac{1}{f_{PWM}}$$

Sehingga Periode PWM Adalah :

$$T_{PWM} = \frac{1}{170.09\ Hz}$$

$T_{PWM} = 0.005879$ S atau 5.87 mS

Duty Cyle PWM adalah perbandingan antara periode on (T_{ON}) dengan periode PWM ($T_{ON} + T_{OFF}$). Untuk mengetahui nilai periode pulsa ON (T_{ON}) dapat dicari dengan persamaan :

$$T_{ON} = \frac{Top\ Value \times PWM\ value}{f_{OSC} / N}$$

Dimana :

Top Value = Nilai maksimal Hitungan Timer (8 bit = 255, 16 bit = 65535)

PWM Value = Nilai PWM (0 – 255)

fosc = Frekuensi Kristal yang dipakai

N = *Prescaller (1,8,64,256 dan 1024).*

Nilai *duty Cycle* dinyatakan dalam persamaan :

$$Duty\ Cycle = \frac{\left(\frac{Top\ Value \times PWM\ value}{f_{OSC} / N}\right)}{T_{PWM}} x\,100\%$$

Atau

$$Duty\ Cycle = \frac{T_{ON}}{T_{PWM}} x\,100\%$$

Sehingga pada saat nilai PWM = 127 maka *Duty Cycle* ditentukan dengan

$$Duty\ Cycle = \frac{\left(\frac{Top\ Value \times PWM\ value}{f_{OSC} / N}\right)}{T_{PWM}} x\,100\%$$

$$Duty\ Cycle = \frac{\left(\frac{255 \times 127}{11059200/1}\right)}{0.005879} x\,100\%$$

$$Duty\ Cycle = \frac{(0.002928)}{0.005879} x\,100\%$$

$$Duty\ Cycle = \frac{(0.002928)}{0.005879} x\,100\%$$

$$Duty\ Cycle = 49.81\%$$

Dari nilai perbandingan periode pulsa ON (T_{ON}) dengan periode pulsa PWM (T_{ON} + T_{OFF}) maka nilai tegangan keluaran (V_{OUT}) dapat dihitung dengan persamaan :

$$V_{OUT} = \frac{T_{ON}}{T_{PWM}} x V_S$$

Dimana Vs adalah 10.42 volt sehingga nilai tegangan keluaran pada saat Duty Cycle = 50 % adalah :

$$V_{OUT} = \frac{(0.002928)}{0.005879} x 10.42$$

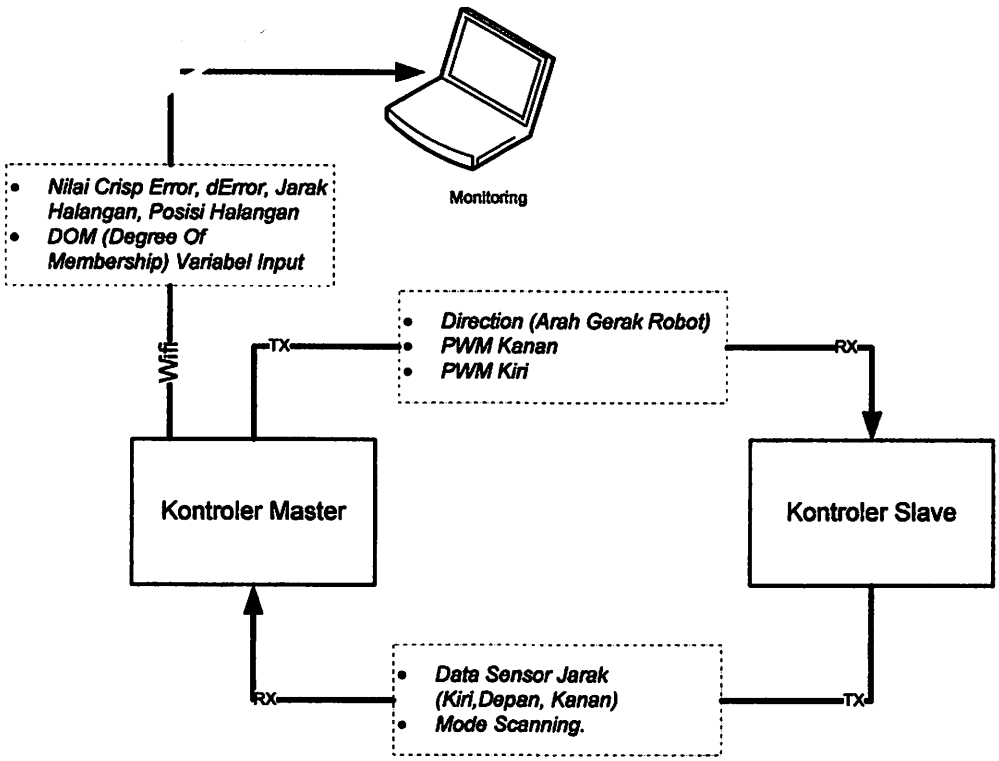
$$V_{OUT} = 0.498 x 10.42$$

$$V_{OUT} = 5.189 \text{ volt}$$

3.4.3 Perancangan Sistem Komunikasi

Komunikasi dalam sistem ini dibagi menjadi dua ,yaitu :

1. Komunikasi antara Kontroler Slave dengan Kontroler Master.
2. Komunikasi Kontroler Master dengan PC



Gambar 3.18 Diagram Blok Sistem Komunikasi

3.4.3.1 Komunikasi Master-Slave

Komunikasi antara kontroler Master dengan kontroler *slave* menggunakan metode poling dengan sistem paket data. sistem polling yaitu proses pengiriman data yang menunggu adanya data yang diterima terlebih dahulu. Sedangkan sistem paket data adalah suatu metode pengiriman data dimana data yang dikirimkan adalah lebih dari satu dan tidak saling mempengaruhi antara data yang satu dengan yang lain.

Tabel 3.11 Pengiriman Data Dari Kontroler *Slave* ke Kontroler *Master*

Urutan Pengiriman	1	2	3	4	5	6	7	8	9
Data	Jarak Kiri (cm)	;	Jarak Depan (cm)	;	Jarak Kanan (cm)	;	Data Mode (cm)	;	Enter (\n)

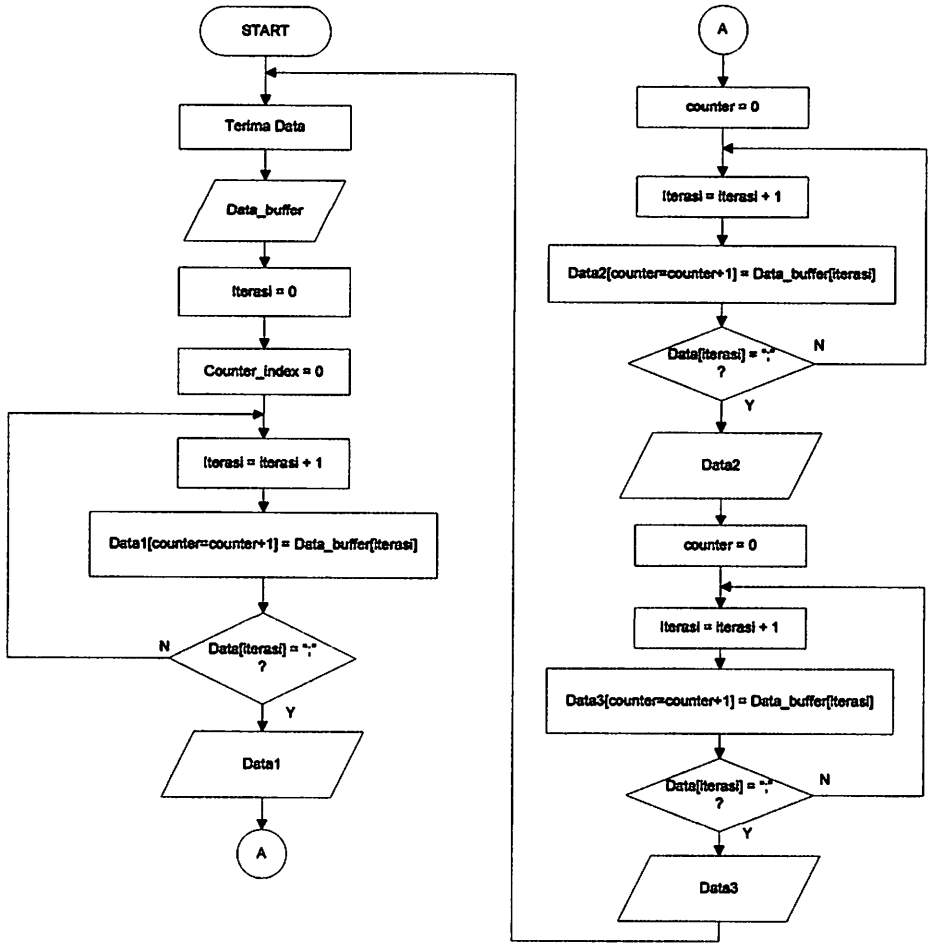
Data yang dikirimkan oleh kontroler slave ke kontroler master ditunjukkan pada Tabel 3.14 diatas. Pada pengiriman sistem paket,antara data yang satu dengan data yang lain dipisahkan oleh karakter pemisah yaitu dalam perancangan dipakai karakter titik koma (";"). Tujuan pemisahan ini adalah agar pada saat diterima pada kontroler slave data yang dikirim dapat dipisahkan untuk selanjutnya diolah secara independen. Pengiriman paket diakhiri oleh karakter Enter untuk menandakan akhir pengiriman paket.

Tabel 3.12 Pengiriman Data dari Kontroler *Master* ke Kontroler *Slave*

Urutan Pengiriman	1	2	3	4	5	6
Data	Arah Gerak	;	PWM Motor Kanan	;	PWM Motor Kiri	Enter ("\n")

Pada pengiriman data dari kontroler slave, format data yang dikirimkan adalah sama dengan pengiriman dari kontroler *slave* ke *master*. Perbedaannya terletak pada variabel data yang dikirimkan. Pada Tabel 3.15, terdapat 3 data yang merepresentasikan gerakan robot. Data yang pertama adalah data arah gerakan robot yaitu gerak maju, mundur, belok kanan dan belok kiri. Sedangkan data ke-3 dan ke-5 adalah data kecepatan yang berupa PWM motor kanan dan motor kiri hasil dari pengolahan kontroler fuzzy.

Selanjutnya untuk menguraikan/ memecah data yang dikirim dalam bentuk paket dapat dilakukan dengan teknik *parsing* data. Teknik *parsing* atau pemisahan data dapat dilakukan dengan menggunakan bantuan *buffer* data dengan *counter indeks* . Alur atau flowchart dapat dijelaskan pada gambar berikut :



Gambar 3.19 Flowchart *Parsing* Data

Data yang diterima disimpan pada variabel aray dengan setiap elemen menyimpan 1 karakter ASCII dengan kapasitas 8 bit (1 Byte). Proses pemisahan (*parsing*) mengunakan perintah perulangan dimana indeks perulangan digunakan untuk melakukan *scanning* pada setiap elemen *array* sampai ditemukan karakter pemisah berupa karakter titik koma (";"). Selama belum menemukan karakter pemisah, data pertama akan disimpan pada buffer data pertama untuk kemudian

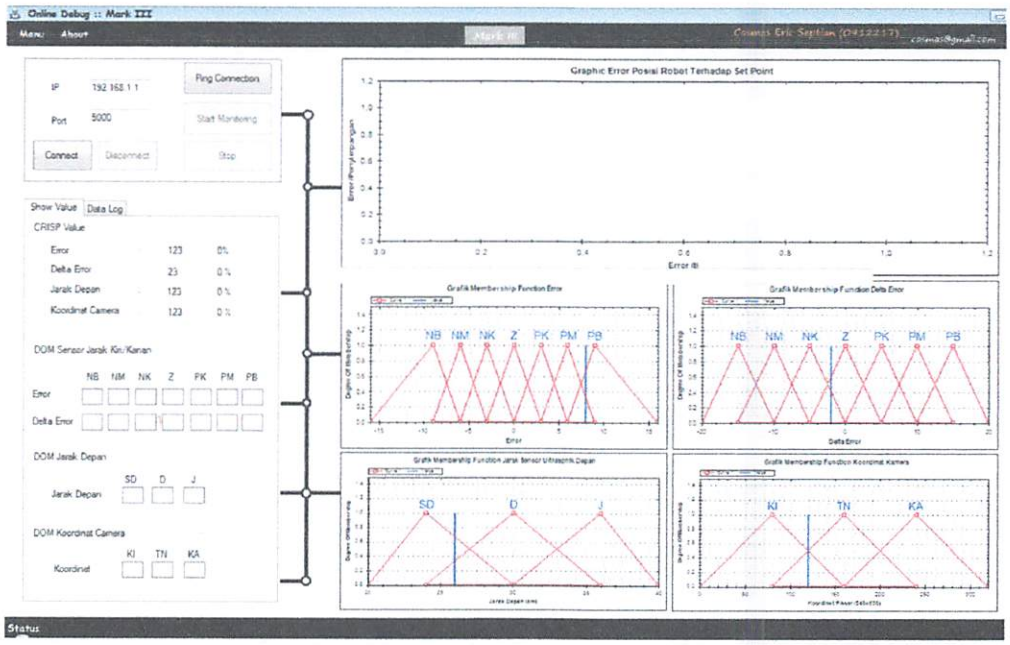
dikonversi dari format *array* ke dalam bilangan *float* sehingga dapat diolah lebih lanjut pada algoritma program selanjutnya.

3.4.3.2 Komunikasi Kontroler Master dengan PC (*Online Debugging*)

Perancangan komunikasi ini dimaksudkan untuk mengetahui variabel error dan delta error aktual serta variabel jarak halangan dan posisi halangan ketika robot melakukan proses telusur dinding dan menghindar halangan. Komunikasi antara kontroler master dengan PC/laptop menggunakan wifi dengan kontroler master difungsikan sebagai *server* dan PC sebagai *client*. Format data yang digunakan sama dengan komunikasi kontroler *master* dengan kontroler *slave*.

Tabel 3.13 Urutan Pengiriman Pada Komunikasi dengan PC

No	Data
1	Error
2	Delta Error
3	Jarak Halangan
4	Posisi Halangan
5	μError 1
6	μError 2
7	μError 3
8	μError 4
9	μError 5
10	μError 6
11	μError 7
12	μDerr 1
13	μDerr 2
14	μDerr 3
15	μDerr 4
16	μDerr 5
17	μDerr 6
18	μDerr 7
19	μJarak Sangat Dekat 1
20	μJarak Dekat 2
21	μJarak Jauh 3
22	μPosisi Kiri 1
23	μPosisi Tengah 2
24	μPosisi Kanan 3



Gambar 3.20 Tampilan Aplikasi *Online Debugging*

Aplikasi *Online Debugging* ini dibuat dengan menggunakan bahasa C# (C Sharp) dilengkapi dengan grafik error secara *real time*. Selain itu ditampilkan juga nilai Derajat Keanggotaan dari masing-masing input fuzzy.

Langkah-langkah untuk menghubungan aplikasi *Online Debugging* dengan kontroler master via wifi adalah :

1. Aktifkan kontroler master (*raspberry pi*) dan pastikan PC atau laptop telah terkoneksi dengan SSID "***mark3-AP***" yaitu nama untuk mengientifikasi wifi yang digunakan oleh kontroler Master.
2. Cek koneksi dengan menjalankan perintah ping ke IP 192.168.1.1
3. Jika aplikasi dengan kontroler master telah terkoneksi dengan baik maka selanjutnya dapat langsung untuk dilakukan monitoring .
4. Start robot dan grafik akan terlihat secara *real-time*.

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

4.1 Pendahuluan

Pada bab ini ditujukan untuk melakukan pengujian dan pembahasan dari sistem yang telah dirancang sebelumnya agar dapat diketahui bagaimana kinerja dari keseluruhan sistem maupun kinerja masing-masing bagian. Dari hasil pengujian tersebut akan dijadikan dasar untuk menentukan kesimpulan serta point-point kekurangan yang harus segera diperbaiki agar kinerja keseluruhan sistem dapat sesuai dengan perencanaan dan perancangan yang telah dibuat.
Pengujian dan pembahasan sistem ini akan dibagi dalam 2 bagian yaitu :

1. Pengujian dan pembahasan Perangkat keras (*Hardware*) yang meliputi pembahasan Rangkaian sensor Ultrasonik, Rangkaian Minimum sistem ATMega162 (*Kontroler Slave*), pengujian Rangkaian Level Konverter, pengujian Driver Motor L298 dan pengujian Rangkaian *Push Button.*
2. Pengujian dan pembahasan Perangkat Lunak (*Software*) yang meliputi pengujian Kontroler Fuzzy, pengujian Sensor Kamera dan Pengujian Komunikasi Antara kontroler *master* dengan kontroler *slave.*

4.2 Pengujian Minimum Sistem ATMega162 (*Kontroler slave*)

Tujuan pengujian pada Minimum Sistem ATMega162 ini adalah agar perangkat lunak yang akan ditanamkan pada mikrokontroler dapat berjalan sesuai dengan yang diinginkan. Pengujian terutama dilakukan untuk menguji port keluaran dan inputan serta pin USART yang akan digunakan untuk berkomunikasi dengan kontroler *Master.*

4.2.1 Peralatan yang digunakan

1. Multimeter Digital
2. Kabel Penghubung
3. Catu daya 5 Volt

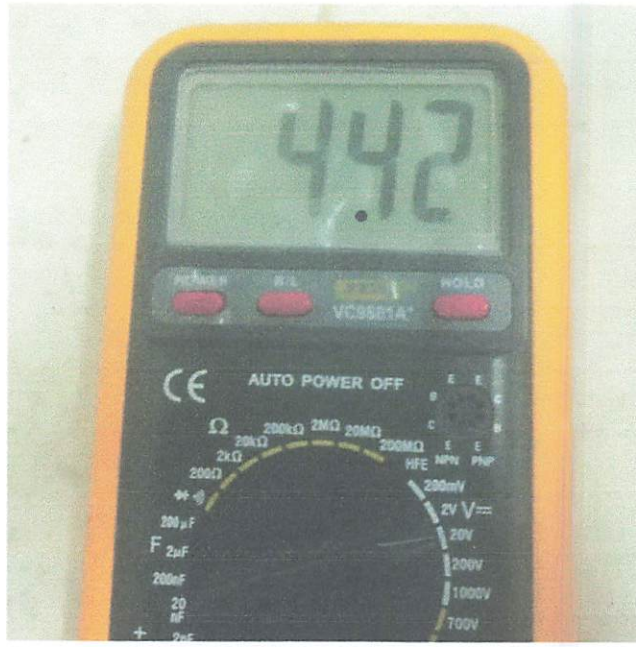
4.2.2 Langkah-langkah yang dilakukan

1. Hubungkan Minimum Sistem dengan catu daya 5 volt
2. Memprogram mikrokontroler untuk mengeluarkan logika 0 dan logika 1 pada masing-masing pin.
3. Hubungkan *probe* positif dari multimeter digital ke masing-masing pin mikrokontroler dan *probe* negatif ke pin ground
4. Mengukur tegangan dari masing-masing pin mikrokontroler
5. Mencatat hasil pengamatan yang telah dilakukan.

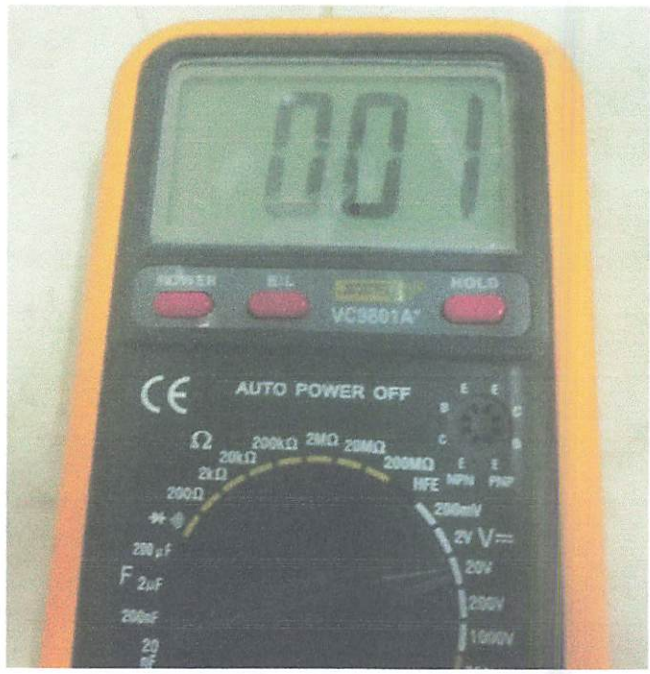
4.2.3 Hasil Pengujian

Tabel 4.1 Hasil Pengujian Tegangan Output Mikrokontroler

Pin	*Logic* Output (bit)	Tegangan Output (volt)
A.0	1	4,36
A.1	1	4,36
A.2	1	4,37
A.3	1	4,38
A.4	0	0.01
A.5	0	0.03
A.6	0	0.01
A.7	0	0.00



Gambar 4.1 Hasil Pengujian Tegangan PORTC Mikrokontroler Pada Keadaan Logika High



Gambar 4.2 Hasil Pengujian Tegangan PORTC Mikrokontroler Pada keadaan Logika *Low*

4.2.4 Analisa Pengujian

Output mikrokotroler pada saat digunakan sebagai output (Nilai register DDRxn = *High*) maka nilai register PORTxn harus diset 1 agar dapat mengeluarkan logic 1 dengan nilai tegangan 5 V sedangkan ketika register PORTxn diset 0 maka nilai tegangan rata-rata adalah sebesar 0.3 V.

4.3 Pengujian Rangkaian Level Konverter

Rangkaian level konverter ini digunakan sebagai konverter level tegangan dari level CMOS ke level TTL dan sebaliknya. Rangkaian level konverter ini terdiri dari dua rangkaian yaitu rangkaian *switching* transistor dan rangkaian pembagi tegangan (*attenuator*) .

4.3.1 Peralatan yang digunakan

1. Power Supply 5 V untuk Minimum Sistem
2. *Adjustable power supply*
3. Multimeter Digital
4. Kabel penghubung

4.3.2 Langkah-langkah yang dilakukan

1. Hubungkan power supply untuk catu daya ke rangkaian level konverter
2. Atur power supply adjustable hingga mendapatkan nilai tegangan 3,3 V untuk merepresentasikan logika *High* pada level CMOS dan hubungkan ke input TX CMOS.
3. Ukur dan catat pada bagian output TX TTL.
4. Atur tegangan hingga mendapatkan tegangan 0 volt untuk merepresentasikan logika *low* dan ukur pada bagian output TTL
5. Atur power supply hingga mendapatkan tegangan 5 V untuk merepresentasikan Logika High dan hubungkan ke pin input TX TTL.
6. Catat dan ukur tegangan pada pin output TX CMOS
7. Rubah tegangan input menjadi 0 Volt untuk merepresentasikan logika Low
8. Catat dan ukur kembali tegangan output.

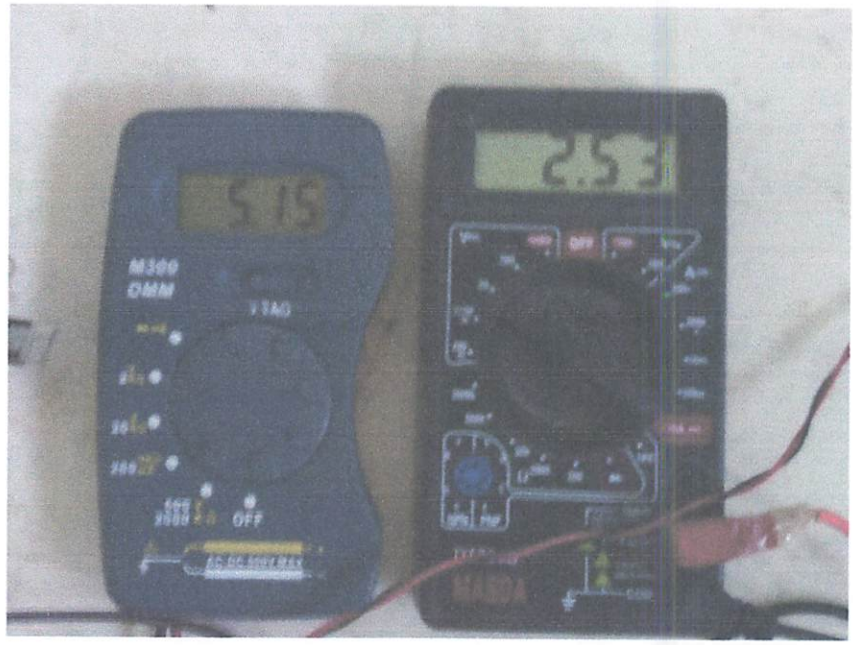
4.3.3 Hasil Pengujian

Tabel 4.2 Hasil Pengujian Switching transistor pada Rangkaian *Logic Level Converter*

No	Tegangan Input (volt)	Tegangan Output (volt)	Keterangan		Logic Status TTL
			Q1	Q2	
1	0	0	Cutoff	Saturasi	Low
2	0.4	0.01	Cutoff	Saturasi	Low
3	0.5	0.02	Cutoff	Saturasi	Low
4	0.6	0.04	Cutoff	Saturasi	Low
5	2.2	4.41	Saturasi	Cutoff	High
6	2.5	4.41	Saturasi	Cutoff	High
7	3.0	4.40	Saturasi	Cutoff	High
8	3.3	4.40	Saturasi	Cutoff	High

Tabel 4.3 Hasil Pengujian Rangkaian Level Konverter pada Bagian Pembagi tegangan

No	Tegangan Input (volt)	Tegangan Output (volt)	Logic Status CMOS
1	0	0	Low
2	1,5	0.75	Low
3	2	1.02	Floating
4	3,5	1.72	High
5	5	2.52	High



Gambar 4.3 Hasil Pengujian Rangkaian Level Logic Converter Pada bagian Atenuator (Pembagi Tegangan)

Gambar 4.4 Hasil Pengujian Rangkaian Level Converter Pada Bagian *Switching* Transistor

4.3.4 Analisa Pengujian

Pada tabel 4.2 diatas pengujian pada rangkaian level konverter pada bagian switching memiliki dua kemungkinan keluaran yaitu *Cutoff* (*logic low*) dan Saturasi (*logic high*). Dari hasil pengujian di atas logic low di dapat pada saat nilai tegangan input berkisar di antar nilai 0 volt sampai dengan 0.6 volt. Keadaan *saturasi* menurut perhitungan didapat ketika nilai $I_b \geq \frac{I_c}{Hfe}$ sedangkan keadaan *cutoff* didapat ketika nilai $I_b < \frac{I_c}{Hfe}$. Nilai i_b adalah arus basis yang mengalir ke transistor 2N2222 dengan nilai hfe=100 sehingga nilai ib pada saat tegangan input=0.6 volt dapat dicari dengan :

$$i_b = \frac{V - V_{BE}}{R_b}$$

$$i_b = \frac{0.6 - 0.6}{1000}$$

$$i_b = \frac{0}{1000}$$

$$i_b = 0A$$

Sedangkan nilai i_c didapat dengan persamaan :

$$i_C = \frac{vcc}{R_C}$$

$$i_C = \frac{5}{1000}$$

$$i_C = 0.005A$$

Nilai $\frac{I_C}{Hfe} = 0.00005A$ sehingga Transistor Q1 bekerja pada daerah *cutoff*. Ketika Q1 cutoff maka nilai VCE = VCC = 5 volt sehingga nilai i_b yang mengalir ke Q2 dapat dihitung :

$$i_b = \frac{V - V_{BE}}{R_b}$$

$$i_b = \frac{5 - 0.6}{1000}$$

$$i_b = \frac{4.4}{1000}$$

$i_b = 0.0044A \; atau \; 4.4mA$

Pada kondisi ini nilai $Ib \geq \frac{Ic}{Hfe}$ yang akan menyebabkan Q2 berada pada kondisi saturasi sehingga nilai VCE_{sat} dapat dicari dengan :

$$VCE_{Sat} = VCC - (Ic \cdot Rc)$$

$$VCE_{Sat} = 5 - (0.005 \cdot 1000)$$

$$VCE_{Sat} = 5 - (5)$$

$$VCE_{Sat} = 0 \; volt$$

Pada tabel 4.3 adalah hasil pengujian rangkaian level logic converter pada bagian pembagi tegangan (*atenuator*). Nilai tegangan output berdasarkan tegangan input yang masuk dapat dihitung dengan :

$$Vout = \left(\frac{R2}{R1 \times R2} \right) \times Vin$$

$$Vout = \left(\frac{10000}{8200 \times 10000} \right) \times 5$$

$$Vout = (0.549) \times 5$$

$$Vout = 2.74 \; Volt$$

Pada saat nilai tegangan input = 5 volt maka hasil keluaran tegangan = 2.7 volt . pada nilai ini level tegangan masuk dalam kondisi logika *High* CMOS.

4.4 Pengujian Rangkaian Driver Motor L298

4.4.1 Peralatan yang digunakan

1. Multimeter digital
2. Osiloskop digital
3. Driver Motor L298
4. Dua buah motor DC Gearbox
5. Kabel penghubung
6. Mikrokontroler
7. Catu daya 5 volt dan 12 Volt

4.4.2 Langkah-langkah Pengujian

1. Hubungkan Catu Daya 5 volt ke Minimum Sistem dan Catu daya 12 volt ke rangkaian driver Motor L298.
2. Hubungkan PORTD.2 mikrokontroler ke pin IN4 Driver Motor
3. Hubungkan PORTD.4 mikrokontroler ke pin IN3 Driver Motor
4. Hubungkan PORTD.3 mikrokontroler ke pin ENB Driver Motor
5. Hubungkan PORTD.5 mikrokontroler ke pin IN2 Driver Motor
6. Hubungkan PORTD.7 mikrokontroler ke pin IN1 Driver Motor
7. Hubungkan PORTD.6 mikrokontroler ke pin ENA Driver Motor
8. Memprogram mikrokontroler untuk memberikan logika ke rangkaian driver motor.
9. Catat pergerakan Motor
10. Hubungkan probe Osiloskop Chanel A ke pin PWM Mikrokontroler dan Channel B ke Output Driver Motor
11. Ukur dan catat *Duty Cycle* dan Frekuensi PWM

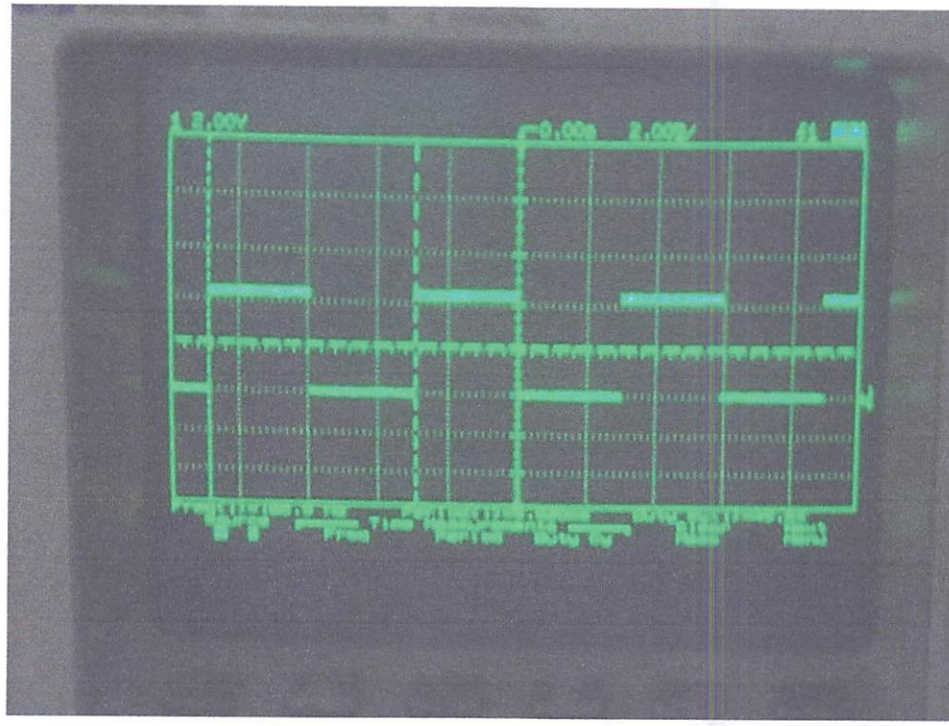
4.4.3 Hasil Pengujian

Tabel 4.4 Hasil Pengujian Arah Putaran Motor

PWM	Input				Arah Putaran	
	IN1	IN2	IN3	IN4	Motor 1	Motor 2
0	0	0	0	0	Fast Brake	Fast Brake
255	1	0	1	0	CW	CW
255	0	1	0	1	CCW	CCW
255	1	1	1	1	Fast Brake	Fast Brake

Tabel 4.5 Hasil Pengujian PWM Driver Motor

No	PWM	Frekuensi (Hz)	T (ms)	Duty Cycle Input (%)	Vout
1	50	168.9	5.93	19.6	2.25
2	127	168.9	5.92	49.6	5.28
3	150	168.9	5.93	58.6	6.20
4	250	168.9	5.93	97.6	9.96
5	255	168.9	5.93	100	10.42



Gambar 4.5 Bentuk Gelombang PWM Pada Keadaan *Duty Cycle* 50 %

4.4.4 Analisa Pengujian

A. Analisa Putaran Motor

Pada tabel 4.4 diatas ditunjukkan pengujian arah putaran motor dengan berbagai kemungkinan logika Input dengan nilai PWM 0 dan 255. Pada saat keadaan IN1=0 dan IN2=0 atau IN3=1 dan IN4=1 keadaan motor berada pada kondisi stop dengan status *Fast Brake* yang artinya motor sedikit mengalami pengereman. Bila keadaan IN1 dan IN2 memiliki polaritas berlawanan maka motor akan bergerak CW (*Counter Wise*)/ CCW(*Counter Clock Wise*).

B. Analisa Frekuensi

Pada pengujian PWM, nilai frekuensi adalah sebesar 168,9 Hz sedangkan menurut perhitungan didapat :

$$f_{pwm} = \frac{1}{\left(\frac{Top\ Value \times PWM\ value}{f_{osc} / N}\right)}$$

$$f_{pwm} = \frac{1}{\left(\frac{255 \times 255}{11059200/1}\right)}$$

$$f_{pwm} = \frac{1}{\left(\frac{65025}{11059200}\right)}$$

$$f_{pwm} = \frac{1}{(0.0058797)}$$

$$f_{pwm} = 170.07 \text{ Hz}$$

Sehingga error pengujian terhadap perhitungan dapat dihitung dengan persamaan :

$$\% \, Error = \left| \frac{Hasil\ Pengujian - Hasil\ Perhitungan}{Hasil\ Perhitungan} \right| \times 100\,\%$$

$$\% \, Error = \left| \frac{168.90 - 170.09}{170.09} \right| \times 100\,\%$$

$$\% \, Error = 0.00699 \times 100\,\%$$

$$\% \, Error = 0.69\,\%$$

C. Analisa Periode

Nilai Periode PWM (T_{PWM}) ditentukan dengan persamaan :

$$T_{PWM} = \frac{1}{f_{PWM}}$$

$$T_{PWM} = \frac{1}{170.09\ Hz}$$

$$T_{PWM} = 0.005879 \text{ S atau } 5.87 \text{ mS}$$

Periode pulsa ON (T_{ON}) ditentukan dengan persamaan :

$$T_{ON} = \frac{Top\ Value \times PWM\ value}{f_{OSC} / N}$$

$$T_{ON} = \frac{255 \times 255}{11059200/1}$$

$$T_{ON} = 0.002928\text{S}$$

D. Analisa Duty Cycle dan Tegangan Keluaran (V_{OUT})

Nilai Duty Cycle dapat diketahui dengan perbandingan antara Periode Pulsa ON (T_{ON}) dengan periode pulsa PWM ($T_{ON} + T_{OFF}$).

$$Duty\ Cycle = \frac{T_{ON}}{T_{PWM}} x\,100\%$$

Dan tegangan keluaran V_{out} dihitung dengan persamaan :

$$V_{OUT} = \frac{T_{ON}}{T_{PWM}} x\, V_S$$

Dimana V_S = Tegangan Sumber ke Driver

Pada percobaan 1 pengujian PWM, Duty Cycle dapat dihitung :

$$Duty\ Cycle = \frac{\left(\frac{Top\ Value \times PWM\ value}{f_{OSC} / N}\right)}{T_{PWM}} x\,100\%$$

$$Duty\ Cycle = \frac{\left(\frac{255 \times 50}{11059200/1}\right)}{0.005879} x\,100\%$$

$$Duty\ Cycle = \frac{\left(\frac{255 \times 50}{11059200/1}\right)}{0.005879} x\,100\%$$

$$Duty\ Cycle = \frac{(0.001152)}{0.005879} x\,100\%$$

$$Duty\ Cycle = 19.61\%$$

Nilai V_{OUT} Motor dengan nilai V_S = 10.42 volt dapat dihitung dengan :

$$V_{OUT} = \frac{T_{ON}}{T_{PWM}} x\, V_S$$

$$V_{OUT} = \frac{(0.001152)}{0.005879} x\, 10.42\ volt$$

$$V_{OUT} = 0.196102 x\, 10.42\ volt$$

$$V_{OUT} = 2.04\ volt$$

Dengan hasil perhitungan dan pengujian didapat nilai *error duty cycle* dengan persamaan :

$$\%\,error = \left| \frac{Duty\ Cycle\ Perhitungan - Duty\ Cycle\ Pengujian}{Duty\ Cycle\ Pengujian} \right| \times 100\%$$

$$\%\,error = \left| \frac{19.61 - 19.60}{19.61} \right| \times 100\%$$

$$\%\,error = 0.05\,\%$$

Untuk analisa data selanjutnya dapat dihitung dengan persamaan yang sama seperti pada percobaan 1, sehingga error rata-rata *Duty Cycle* setiap percobaan dapat dihitung dengan :

$$\overline{\%Error} = \frac{\sum \%Error}{Jumlah\ Percobaan}$$

$$\overline{\%Error} = \frac{0.05 + 0.42 + 0.39 + 0.45 + 0}{5}$$

$$\overline{\%Error} = 0.26\ \%$$

4.5 Pengujian Sensor Jarak (*Ultrasonik Ping*)

4.5.1 Peralatan yang diperlukan

1. Sensor jarak Ultrasonik Ping
2. Alat Ukur Panjang
3. Mikrokontroler
4. *Downloader* untuk memasukkan program ke mikrokontroler
5. Catu daya 5 volt
6. LCD 16 x 2

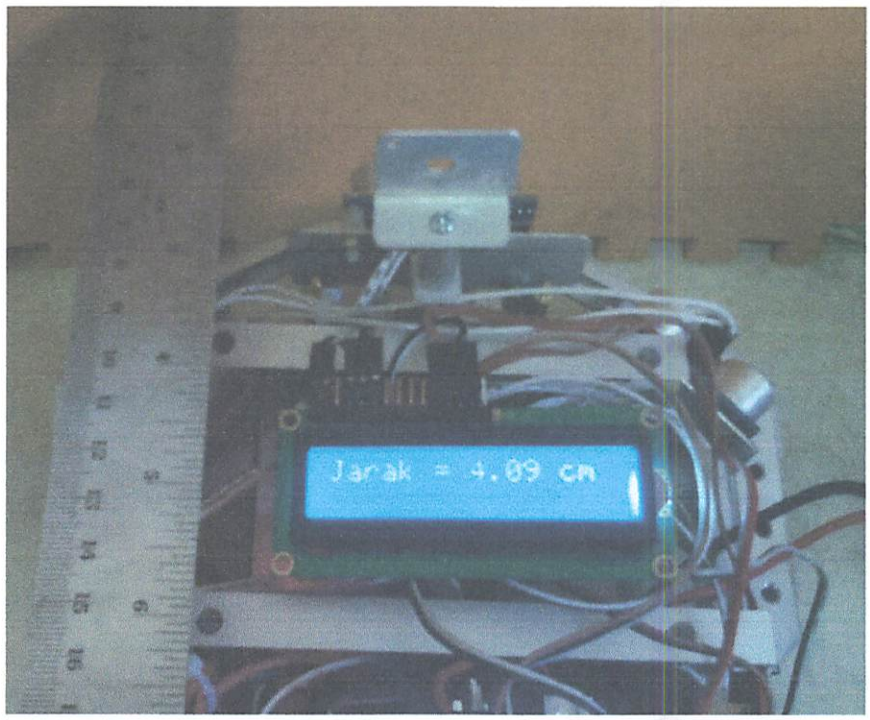
4.5.2 Langkah-langkah Pengujian

1. Hubungkan catu daya 5 volt ke sensor Ultrasonik
2. Hubungkan pin data Ultrasonik ke Pin A.0 Mikrokontroler
3. Hubungkan LCD ke PORTC Mikrokontroler
4. Hubungkan catu daya 5 volt ke Minimum sistem ATMega162
5. Program mikrokontroler untuk membaca jarak dengan sensor ultrasonik dan menampilkan data jarak ke LCD
6. Catat dan bandingkan jarak terukur di LCD dengan Alat ukur

4.5.3 Hasil Pengujian

Tabel 4.6 Hasil Perbandingan Pengujian dan Pengukuran pada Objek Rata

No	Alat Ukur (cm)	Pengujian Pada Objek Rata (cm)	Pengujian Pada Objek Tidak Rata (cm)
1	4	4.09	5.06
2	8	7.89	9.12
3	12	12.10	8.81
4	16	16.01	12.06
5	20	20.23	17.04



Gambar 4.6 Pengujian Sensor Jarak

4.5.4 Analisa Pengujian

Prinsip pengukuran sensor jarak adalah mengukur pulsa yang dikirimkan oleh sensor ping dengan menggunakan counter. Pertama kali mikrokontroler akan mengirimkan pulsa dengan periode pulsa high (T_{high}) = 5 µS dan periode pulsa low (T_{low}) = 5 µS. selanjutnya mikrokontroler akan menunggu pantulan gelombang suara ultrasonik diterima oleh bagian RX ping . selama menunggu inilah *counter* akan bertambah sampai pantulan suara ultrasonik diterima. Periode pulsa counter tersebut kemudian digunakan untuk dikonversi ke dalam jarak. Persamaan yang digunakan adalah :

$$Jarak\ (cm) = \frac{Periode\ pulsa \times 0.098620}{2}$$

Dengan menggunakan persamaa diatas, jika pulsa terukur di dapat nilai sebesar 82 µs, maka jarak dapat dicari dengan :

$$Jarak\ (cm) = \frac{82 \times 0.098620}{2}$$

$$Jarak\ (cm) = \frac{8.08684}{2}$$

$$Jarak\ (cm) = 4{,}04342\ \text{cm}$$

Sedangkan nilai error didapat dengan membandingkan nilai sensor jarak hasil pengujian dengan hasil pengukuran. Persamaan yang digunakan adalah :

$$\%\ error = \left| \frac{Hasil\ Pengujian - Hasil\ Pengukuran}{Hasil\ Pengukuran} \right| \times 100\%$$

Dari hasil pengujian :

1. $\%\ error = \left| \frac{4.09 - 4.00}{4.00} \right| \times 100\%$

   $\%\ error = 2.25\ \%$

2. $\%\ error = \left| \frac{7.89 - 8.00}{8.00} \right| \times 100\%$

   $\%\ error = 1.3\ \%$

3. $\%\ error = \left| \frac{12.10 - 12.00}{12.00} \right| \times 100\%$

   $\%\ error = 0.8\ \%$

4. $\%\ error = \left| \frac{16.01 - 16.00}{16.00} \right| \times 100\%$

   $\%\ error = 0.06\ \%$

5. $\%\ error = \left| \frac{20.23 - 20.00}{20.00} \right| \times 100\%$

   $\%\ error = 1.15\ \%$

Dari nilai error tersebut dapat diambil nilai rata-rata error dengan persamaan :

$$\overline{\%Error} = \frac{\sum \%Error}{Jumlah\ Percobaan}$$

$$\overline{\%Error} = \frac{2.25 + 1.3 + 0.8 + 0.06 + 1.15}{5}$$

$$\overline{\%Error} = 1.11\ \%$$

Tabel 4.7 Nilai Error Pengujian pada Objek rata

No	Jarak (cm)		Error (%)
	Pengujian	Alat Ukur	
1	4.09	4	2.25
2	7.89	7	1.30
3	12.1	12	0.80
4	16.01	16	0.06
5	20.23	20	1.15
Error Rata-rata			1.11

Untuk mengetahui nilai error rata-rata pengujian sensor jarak pada objek tidak rata dapat dicari dengan persamaan yang sama seperti di atas.

Dari hasil pengujian :

1. $\%\,error = \left|\frac{5.06 - 4.00}{4.00}\right| \times 100\%$

   $\%\,error = 26.5\,\%$

2. $\%\,error = \left|\frac{9.12 - 8.00}{8.00}\right| \times 100\%$

   $\%\,error = 14\,\%$

3. $\%\,error = \left|\frac{8.81 - 12.00}{12.00}\right| \times 100\%$

   $\%\,error = 26.58\,\%$

4. $\%\,error = \left|\frac{12.06 - 16.00}{16.00}\right| \times 100\%$

   $\%\,error = 24.62\%$

5. $\%\,error = \left|\frac{17.04 - 20.00}{20.00}\right| \times 100\%$

   $\%\,error = 14.80\,\%$

Error rata-rata pengujian sensor pada objek tidak rata dapat dicari dengan persamaan :

$$\overline{\%Error} = \frac{\sum \%Error}{Jumlah\ Percobaan}$$

$$\overline{\%Error} = \frac{26.5 + 14 + 26.62 + 24.62 + 14.80}{5}$$

$$\overline{\%Error} = 21{,}30 \quad \%$$

Tabel 4.8 Nilai error pengujian sensor Ultrasonik Pada objek tidak rata

No	Jarak (cm)		Error (%)
	Pengujian	Alat Ukur	
1	5.06	4	26.50
2	9.12	7	14.00
3	8.81	12	26.62
4	12.06	16	24.62
5	17.04	20	14.80
Error Rata-rata			21.30

Error yang terjadi diakibatkan oleh pantulan gelombang suara ultrasonik yang tidak dipantulkan secara sempurna karena permukaan yang tidak rata/ bergelombang. selain itu dari pengujian didapatkan kesimpulan bahwa sensor jarak yang berbasis ultrasonik juga tidak dapat digunakan pada lingkungan yang dapat menyerap suara seperti busa atau kain.

4.6 Pengujian Sensor Kamera

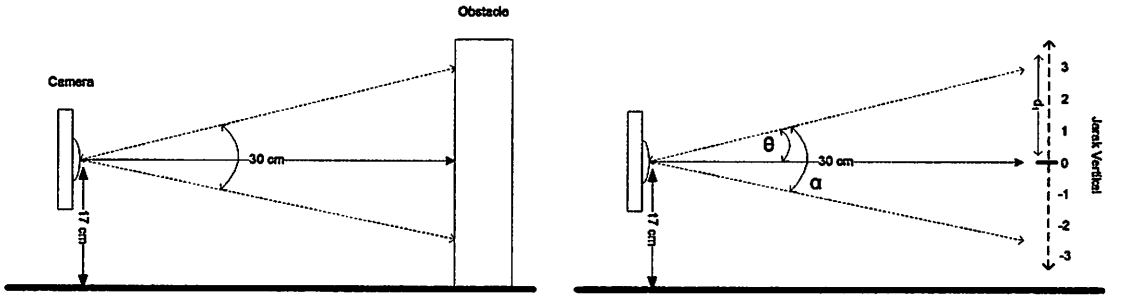
4.6.1 Peralatan yang diperlukan

1. Modul Kamera Raspberry Pi (*RaspiCam*)
2. Raspberry Pi
3. Catu daya 5 volt/2A
4. Konverter HDMI to VGA
5. LCD Monitor
6. Object Berwarna Merah (*R=255, G=0, B=0*) H=100,S=0,V=0

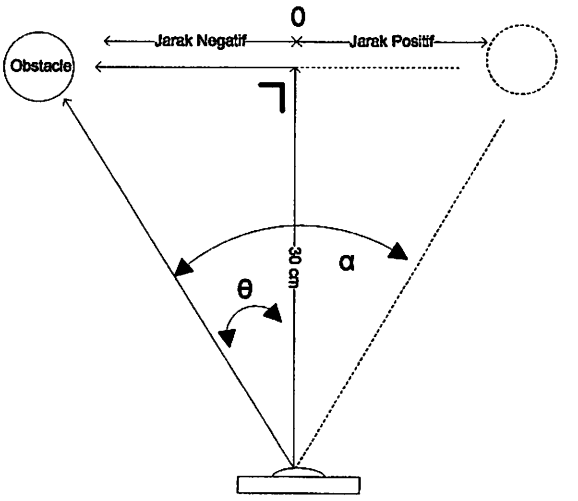
4.6.2 Langkah-langkah Pengujian

1. Masukkan kabel RaspiCam ke soket CSI.
2. Hubungkan HDMI konverter ke socket HDMI dan sambungkan soket VGA *female* ke soket VGA *Male* Monitor
3. Hubungkan Catu daya 5 volt/2A ke Raspberry Pi
4. Jalankan program pendeteksi warna pada Raspberry Pi dan Tampilkan data posisi objek dalam format piksel pada LXTerminal
5. Catat hasilnya pada berbagai kondisi pencahayaan dan jarak

4.6.3 Hasil Pengujian



Gambar 4.7 Pengujian Sensor Kamera pada Posisi sudut Vertikal



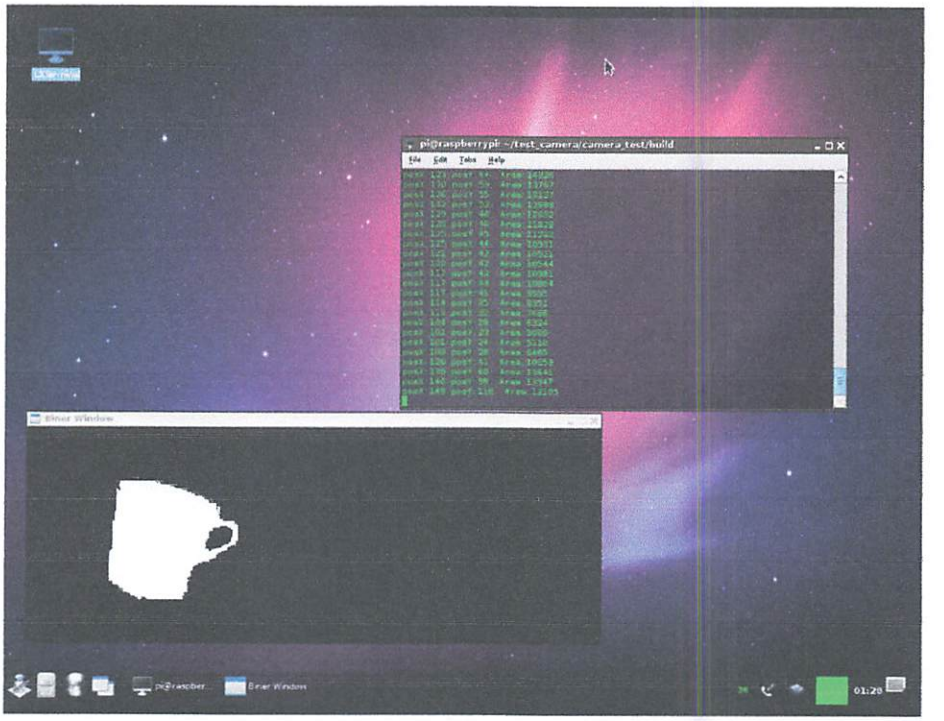
Gambar 4.8 Pengujian Sensor Kamera pada Posisi Sudut Horizontal

Tabel 4.9 Hasil pengujian Koordinat Piksel Horizontal RaspiCam

No	Jarak (d_1)	Koordinat Pixel Horizontal
1	-16	0
2	-14	0
3	-12	4
4	-10	26
5	-8	54
6	-6	75
7	-4	102
8	-2	125
9	0	150
10	2	180
11	4	214
12	6	244
13	8	281
14	10	295
15	13	313
16	16	0

Tabel 4.10 Hasil Pengujian Sudut Piksel Vertikal RaspiCam

No	Jarak (d_1)	Koordinat Pixel Vertikal
1	-12	0
2	-10	224
3	-8	201
4	-6	187
5	-4	162
6	-2	145
7	0	118
8	2	84
9	4	63
10	6	42
11	8	18
12	10	4
13	12	0



Gambar 4.9 Hasil Deteksi Objek dengan tampilan Nilai *Threshold*

4.6.4 Analisa Pengujian

Pada pengujian sensor kamera di atas dibagi menjadi dua bagian yaitu pengujian deteksi pada sudut vertikal dan pengujian pada sudut horizontal. Prinsip pengukuran sudut menggunakan persamaan segitiga siku-siku dimana tinggi segitiga adalah jarak depan dan alas segitiga adalah jarak halangan dengan titik tengah kamera.

1. Sudut vertikal

   Pengujian dilakukan seperti pada Gambar 4.7 di atas dengan hasil diketahui bahwa maksimal panjang alas segitiga adalah +10 cm dan -10 cm dan tinggi segitga = 30 cm sehingga sudut θ dapat dicari dengan persamaan :

$$\tan\theta = \frac{alas}{tinggi}$$

$$\tan\theta = \frac{10}{30}$$

$$\tan\theta = 0.3$$

$$\theta = arc\tan 0.3$$

$\theta = 16.69^0$

$\alpha = 2 \times \theta = 33.39^0$

Nilai Sudut jangkauan vertikal kamera adalah sebesar 33.39^0

2. Sudut Horizontal

Pengujian sudut jangkauan horizontal dilakukan seperti pada gambar 4.8 di atas dengan diketahui bahwa maksimal panjang alas segitiga adalah -12 dan +13 sedangkan tinggi segitiga = 30 cm sehingga sudut θ dapat dicari dengan persamaan :

$$\tan\theta = \frac{alas}{tinggi}$$

$$\tan\theta = \frac{13}{30}$$

$$\tan\theta = 0.43$$

$$\theta = arc\tan 0.43$$

$$\theta = 23.42^0$$

$$\alpha = 2 \times \theta = 46.85^0$$

Nilai jangkauan maksimal sudut horizontal kamera adalah $46{,}85^0$.

4.7 Pengujian Komunikasi Master-Slave

4.7.1 Peralatan yang dibutuhkan

1. Raspberry Pi
2. Catu daya 5 volt/2A
3. Rangkaian Level Logic Konverter
4. Minimum Sistem ATMega162

4.7.2 Langkah-langkah Pengujian

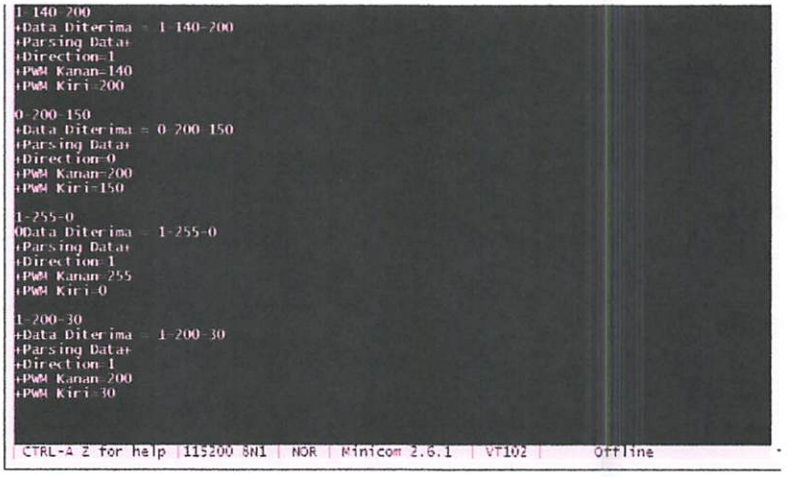
1. Hubungkan Pin TX Raspberry Pi ke Input TX rangkaian Level Konverter
2. Hubungkan Pin RX Raspberry Pi ke Input RX rangkaian Level Konverter
3. Hubungkan Output Rangkaian Level Konverter ke pin RX dan TX (USART0) mikrokontroler dengan hubungan *Cross* (disilang)

4. Memprogram mikrokontroler untuk menerima data dari USART0, menyimpan ke buffer dan mengirimkannya kembali dengan baudrate 115200 Bps.
5. Hidupkan Raspberry Pi dengan menghubungkan Catu daya 5 volt/2A ke konektor mikro USB
6. Jalankan miniCOM pada raspberry Pi dan kirimkan karakter.
7. Amati dan Catat karakter yang Dikirim dan diterima

4.7.3 Hasil Pengujian

Tabel 4.11 Hasil Pengujian Komunikasi Master-Slave

Data yang Dikirim (ASCII)	Data Yang Diterima (ASCII)
1-140-200 \n	1-140-200
0-200-150 \n	0-200-150
1-255-0 \n	1-255-0
1-200-30 \n	1-200-30
0-255-255 \n	0-255-255



Gambar 4.10 Pengujian Komunikasi Master-Slave

4.7.4 Analisa Pengujian

Pada tabel 4.6, data yang diterima adalah sama dengan data yang dikirim tanpa ada cacat (*Corrupt*) dengan kecepatan pengiriman serial adalah 115200 bps artinya dalam satu detik komunikasi serial dapat mengirimkan sampai dengan

115200 bit data. Setiap bit dikirimkan dalam waktu 1/115200 sehingga 1 bit dapat dikirimkan dalam waktu 8.68 μs. Pada tabel di atas pengiriman dilakukan dengan menggunakan karakter ASCII (8 bit) sehingga pengiriman karakter "1-140-200 \n" ditempuh dalam waktu 8.68 μs x (8 bit x 10) = 694.4 μs.

4.8 Pengujian Kontroler Fuzzy (*Offline*)

Sistem pengujian kontroler fuzzy secara *offline* menggunakan metode perbandingan antara kontroler dengan aplikasi yang sudah ada yaitu *Fuzzy Logic Toolbox* dari Matlab. Perbandingan dilakukan dengan memberikan nilai input yang sama dan membandingkan keluaran hasil defuzzyfikasi.

4.8.1 Peralatan yang dibutuhkan

1. Aplikasi Fuzzy Logic Toolbox Matlab
2. Raspberry Pi
3. Adaptor 5 volt/2A
4. Wifi dongle untuk koneksi antara Raspberry Pi dengan komputer melalui SSH
5. Aplikasi bitwise SSH Client

4.8.2 Langkah-langkah Pengujian

1. Pasang wifi dongle pada port USB Raspberry Pi
2. Hidupkan Raspbery Pi dengan menghubungkan Adaptor 5 volt/2A pada konektor mikro USB.
3. Hubungkan komputer/laptop dengan Rapberry Pi melalui wifi dengan SSID "Mark-III".
4. Koneksikan Komputer ke Raspberry Pi dengan memasukkan IP 192.168.1.1 pada aplikasi Bitvise SSH Client.
5. Buka terminal Raspberry Pi
6. Jalankan Program *Fuzzy Logic Controler* dan masukkan nilai parameter Input
7. Buka fuzzy Logic Toolbox™ pada Matlab.
8. Bandingkan dan Catat Hasil keluaran Defuzzyfikasi.

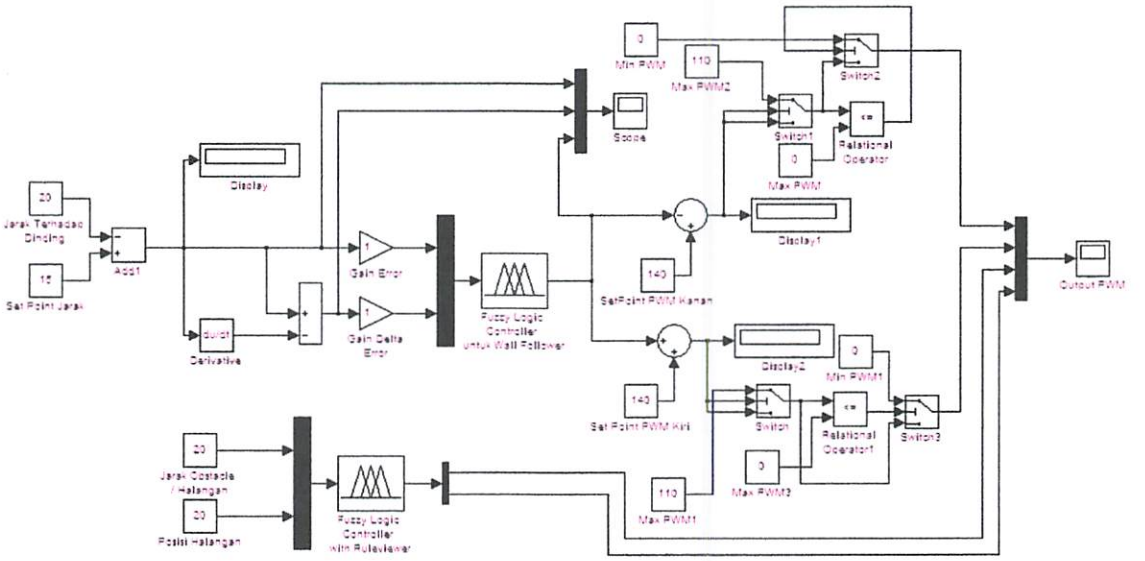
4.8.3 Hasil Pengujian

Tabel 4.12 Hasil pengujian output fuzzy untuk Penelusur Dinding

No	Input		Output Fuzzy	
			PWM	
	Error	Delta Error	Kontroler Master Raspberry Pi	Fuzzy Logic Toolbox Matlab
1	0	0	0.00	0
2	4.62	1	133.71	134
3	5.28	2	147.03	147
4	6.39	2	168.25	168
5	4.62	-1	110.86	111
6	5.28	-1	123.43	123
7	6.39	2	168.25	168
8	8.59	4	210.26	210

Tabel 4.13 Hasil Pengujian Output Fuzzy Untuk Penghindar Halangan

NO	Input		Output			
	Jarak Halangan	Posisi Kordinat Pixel (x)	Kontroler Master		Fuzzy Logic Toolbox Matlab	
			PWM Kiri	PWM Kanan	PWM Kiri	PWM Kanan
1	10.40	100	20	220	20	220
2	12.23	130	20	220	20	220
3	15.10	160	20	154.90	20	155
4	17.50	200	36.05	127	36.1	127
5	20.72	220	127	68.10	127	68.1
6	23.46	250	127	127	127	127
7	25.88	270	127	127	127	127
8	29.43	300	127	127	127	127

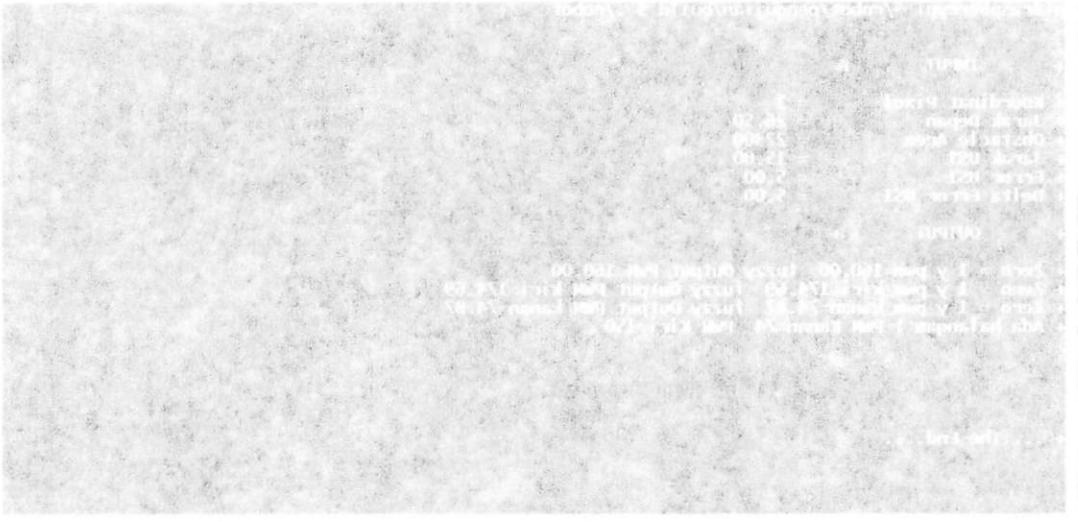


Gambar 4.11 Pengujian Kontroler Dengan Simulink

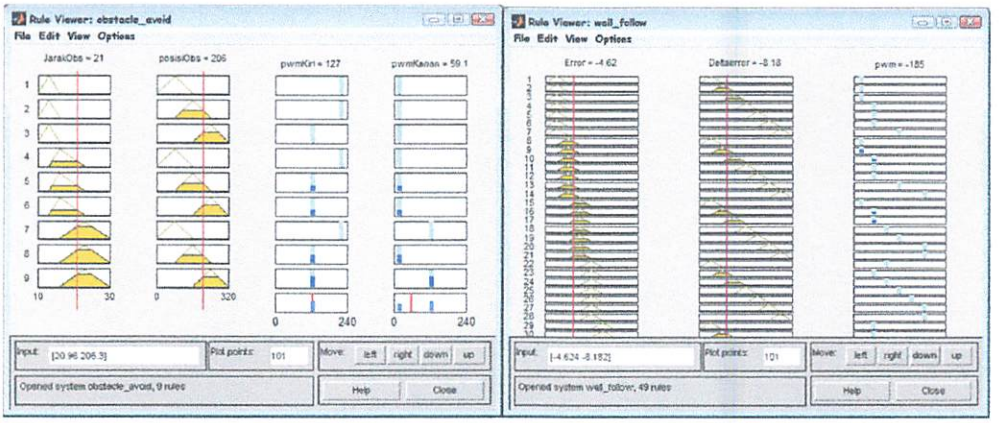
```
pi@raspberrypi ~/robot/pengujian/build $ ./robot

+       INPUT       +

+ Koordinat Pixel        = 1
+ Jarak Depan            = 26.50
+ Obstacle Area          = 22300
+ Jarak US1              = 15.00
+ Error US1              = 5.00
+ Delta Error US1        = 5.00

+       OUTPUT      +

+ Zero = 1 y_pwm=160.00  fuzzy Output PWM=160.00
+ Zero = 1 y_pwm_kiri=174.69  fuzzy Output PWM kiri=174.69
+ Zero = 1 y_pwm_kanan=74.87  fuzzy Output PWM kanan=74.87
+ Ada halangan ! PWM Kanan=74  PWM Kiri=150



+ ....The End....
```

Gambar 4.12 Hasil Pengujian Kontroler Master

Gambar 4.11 Pengujian Komputer Dengan Simulink



Gambar 4.12 Hasil Pengujian Komputer Virtual



(a) (b)

Gambar 4.13 Hasil Rule Viewer Pada Fuzzy Logic Toolbox (a). Penghindar Halangan (b). Penelusur Dinding

4.8.4 Analisa Hasil Pengujian

Pada hasil pengujian di atas selisih nilai output defuzzyfikasi antara kontroler master dengan *Fuzzy Logic Toolbox* sangat kecil sehingga dapat dikatakan bahwa program yang ditanamkan pada kontroler master berfungsi sesuai dengan persamaan dan aturan dari logika fuzzy. Kontroler fuzzy untuk penelusur dinding menggunakan konsep MISO (*Multiple Input Single Output*) dengan nilai input adalah *error* dan *delta error* dari pembacaan sensor jarak kiri/kanan dan output tunggal berupa kecepatan PWM yang dibagi lagi menjadi PWM kanan dan PWM kiri dengan melakukan pengurangan dan penambahan dengan set point PWM. Sedangkan kontroler fuzzy untuk penghindar halangan menggunakan konsep MIMO (*Multiple Input Multiple Output)* dengan input berupa jarak dengan halangan dan posisi koordinat piksel terhadap halangan.

Sebagai contoh pada pengujian ke-8 Tabel 4.7 Diketahui nilai Error = 8,59 dan nilai delta error = 4 maka proses penentuan derajat keanggotaan error dan delta error dapat dihitung dengan persamaan :

$$\mu(k) = \begin{cases} 0 & k < a \\ \dfrac{k-a}{b-a} & a \le k \le b \\ \dfrac{c-k}{c-b} & b \le k \le c \\ 0 & k > c \end{cases}$$

Nilai error 8,59 termasuk dalam fungsi keanggotaan PM (Positif Medium) dengan range a=3, b=6, c=9 dan PB (Positif Besar) dengan range a=6, b=9, c=12.

Untuk derajat keanggotan PM :

$$\mu PM = \frac{c - error}{c - b}$$

$$\mu PM = \frac{9 - 8.59}{9 - 6}$$

$$\mu PM = 0.14$$

Untuk derajat keanggotaan PB :

$$\mu PB = \frac{error - a}{b - a}$$

$$\mu PB = \frac{8.59 - 6}{9 - 6}$$

$$\mu PB = 0.86$$

Sedangkan untuk Nilai delta error = 4 termasuk dalam fungsi keanggotaan Z (zero) dengan range a=-5, b=0, c=5 dan PK (Positif Kecil) dengan range nilai a=0, b=5, c=10.

Untuk derajat keanggotaan Z :

$$\mu Z = \frac{c - derror}{c - b}$$

$$\mu Z = \frac{5 - 4}{5 - 0}$$

$$\mu Z = 0.2$$

Untuk derajat keanggotaan PK :

$$\mu PK = \frac{derror - a}{b - a}$$

$$\mu PK = \frac{4 - 0}{5 - 0}$$

$$\mu PK = 0.8$$

Dengan demikian dari hasil fuzzyfikasi nilai error dan delta error selanjutnya akan diolah berdasarkan aturan-aturan yang telah ditetapkan dengan fungsi keanggotaan output minimum sehingga :

[38] If Error = PM AND Delta_error = Z THEN y_{38} = yPM, $\mu(y_{38})$=0.14

[39] If Error = PM AND Delta_error = PK THEN y_{39} = yPM, $\mu(y_{39})=0.14$

[45] If Error = PB AND Delta_error = Z THEN y_{45} = yPM , $\mu(y_{45})=0.2$

[46] If Error = PB AND Delta_error = PK THEN y_{46} = yPB, $\mu(y_{46})=0.8$

Output Crisp dapat hasil defuzzyfikasi dapat dihitung dengan persamaan :

$$Crisp\ Output = \frac{\sum_{k=0}^{m} y_k\ \mu(y_k)}{\sum_{k=0}^{m} \mu(y_k)}$$

Sehingga nilai output Crisp adalah :

$$Crisp\ Output = \frac{y_{38}\ \mu(y_{38}) + y_{39}\ \mu(y_{39}) + y_{45}\ \mu(y_{45}) + y_{46}\ \mu(y_{46})}{\mu(y_{38}) + \mu(y_{39}) + \mu(y_{45}) + \mu(y_{46})}$$

$$Crisp\ Output = \frac{0.14 \bullet 160 + 0.14 \bullet 160 + 0.2 \bullet 160 + 0.8 \bullet 240}{0.14 + 0.14 + 0.2 + 0.8}$$

$$Crisp\ Output = \frac{22.4 + 22.4 + 32 + 192}{1.28}$$

$$Crisp\ Output = 210$$

Untuk proses penghitungan nilai output yang lain dilakukan dengan proses dan persamaan yang sama.

4.9 Pengujian Kontroler Fuzzy (*Online*)

Pengujian kontroler fuzzy secara online dilakukan untuk mengetahui parameter-parameter fuzzy dalam bentuk tampilan grafik secara real-time ketika robot sedang running (berjalan).

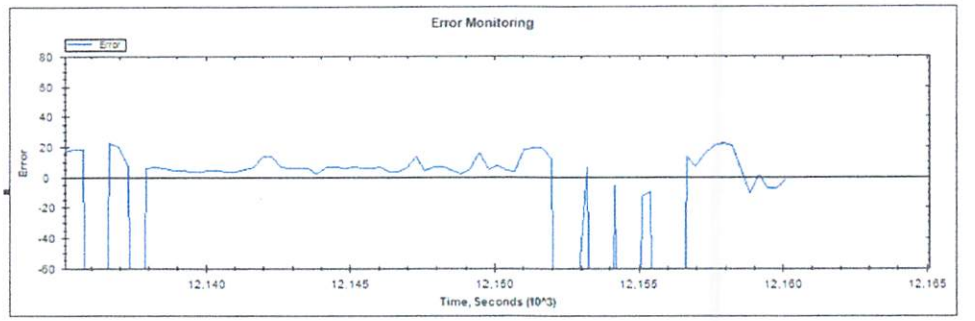
4.9.1 Peralatan Yang digunakan

1. PC/ Laptop dengan Koneksi Wifi
2. Software Online Debuging (C#)

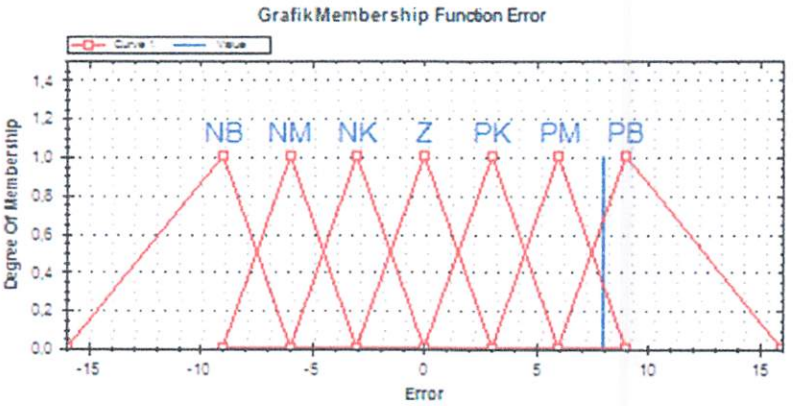
4.9.2 Langkah-langkah Pengujian

1. Nyalakan Robot
2. Koneksikan Laptop/PC dengan Robot (SSID "mark3-AP")
3. Ping Koneksi dengan Alamat IP 192.168.1.1
4. Start Robot dan Koneksikan dengan Software Online Debugging

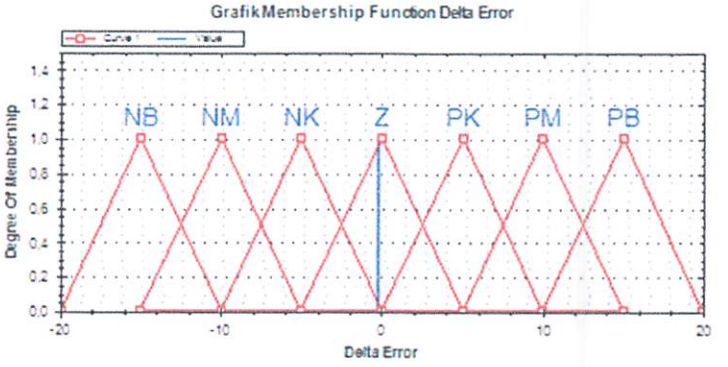
4.9.3 Hasil Pengujian



Grafik 4.1 Nilai Error Robot secara Real Time

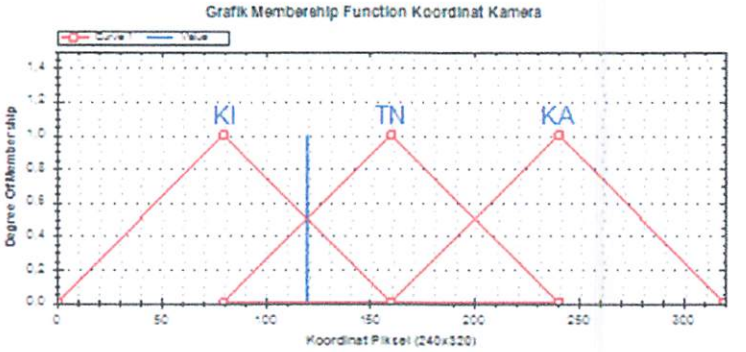


Grafik 4.2 Nilai Fungsi Keanggotaan Error Secara Real Time



Grafik 4.3 Nilai Fungsi Keanggotaan Delta Error Secara Real Time

Grafik 4.4 Nilai Fungsi Keanggotaan Jarak halangan Secara Real Time



Grafik 4.5 Nilai Fungsi Keanggotaan Posisi Halangan Secara Real Time

4.9.4 Analisa Hasil Pengujian

Pada Tampilan Grafik Di atas diunjukkan parameter dari kendali fuzzy secara real time yaitu nilai error , fungsi keanggotaan error, fungsi keanggotaan delta error, fungsi keanggotaan posisi jarak halangan dengan robot dan fungsi keanggotaan posisi koordinat piksel halangan . Pada grafik error yang ditunjukkan pada Grafik 4.1, nilai error mengalami fluktuasi dengan range nilai antara positif dan negatif dimana nilai negatif adalah keadaan yang dialami robot ketika menjauhi dinding sedangkan nilai error positif adalah ketika robot mendekat pada dinding sedangkan nilai ideal didapatkan ketika robot berada pada posisi 0 (nol) atau jarak aktual sama dengan set point jarak. Pengambilan nilai untuk ditampilkan pada grafik menggunakan waktu sampling sebesar 100 mS.

4.10 Pengujian Sistem

Pengujian dilakukan dengan menguji kemampuan robot untuk mengikuti dinding kiri atau dinding kanan serta kemampuan untuk menghindar halangan dengan warna tertentu. Pengujian telusur dinding dibagi menjadi dua mode yaitu telusur dinding kiri atau telusur dinding kanan pada lintasan lurus, belokan 90^0 dan belokan 180^0. Sedangkan pengujian penghindar halangan dilakukan pada posisi sudut arah datang robot. Tujuan utama dari pengujian sistem ini adalah untuk mengetahui performa metode kendali fuzzy logic dalam pergerakan robot untuk menjaga nilai jarak terhadap dinding agar mendekati nilai set point atau nilai error sekecil mungkin. Pengujian telusur dinding menggunakan mode telusur kiri saja karena telusur kiri atau kanan pada dasarnya adalah sama.

4.10.1 Langkah Pengujian

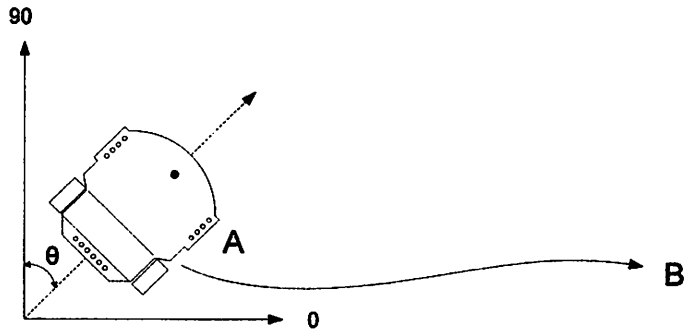
4.10.1.1 Pengujian pada Lintasan Lurus

Lintasan pengujian menggunakan track lurus dengan panjang 80 cm dan tinggi 30 cm. Robot bergerak dari titik A sampai titik B dengan indikator keberhasilan adalah kemampuan robot untuk bergerak maju tanpa menyentuh dinding.

Gmbar 4.14 Pengujian Pergerakan robot Pada Lintasan Lurus

4.10.1.2 Pengujian Sudut Start

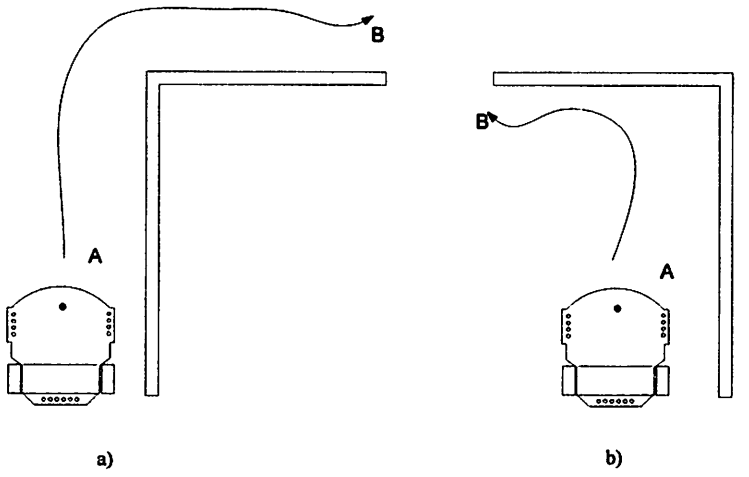
Pengujian sudut menggunakan lintasan lurus dengan mengubah sudut awal robot pada saat keadaan start. Indikator keberhasilan robot yaitu kemampuan untuk dapat menyetabilkan posisi dengan menyejajarkan dengan acuan dinding. Pada pengujian dipakai mode telusur dinding kiri dengan nilai *set point* 26 cm terhadap dinding.



Gambar 4.15 Pengujian Sudut Start

4.10.1.3 Pengujian Belokan 90^0

Pengujian dilakukan dengan pada lintasan dengan belokan sebesar 90^0 atau siku. Indikator keberhasilan robot yaitu kemampuan untuk menentukan posisi haluan dengan nilai yang proporsional tanpa tabrakan dengan dinding.

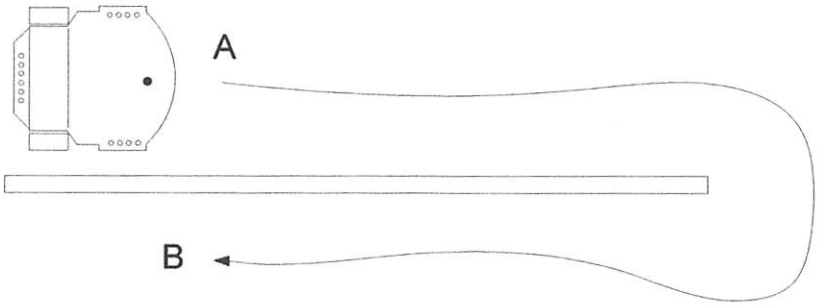


Gambar 4.16 Pengujian Belokan 90^0

Gambar 4.16 (a) adalah lintasan dengan sudut haluan 90^0 dengan posisi robot berada pada posisi luar sudut sedangkan pada gambar 4.16 (b) lintasan dengan sudut 90^0 dengan posisi robot berada di dalam sudut.

4.10.1.4 Pengujian Belokan 180^0

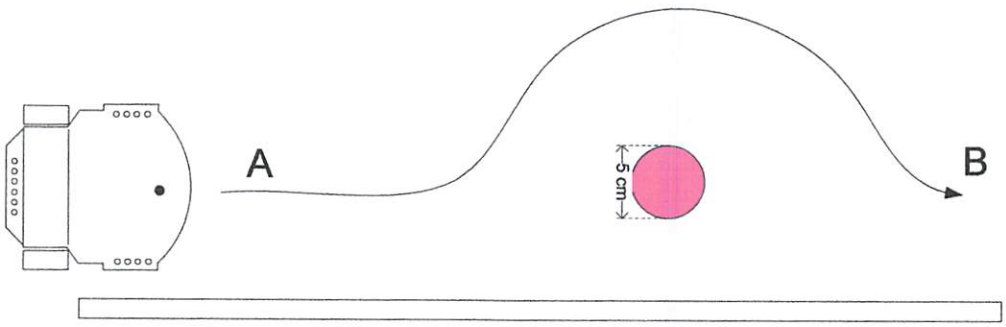
Pengujian dilakukan pada lintasan lurus dengan sudut belokan sebesar 180^0 dengan posisi start robot pada titik A menuju titik B. indikator keberhasilan robot dalah kemampuan untuk bergerak maju dan melakukan belokan 180^0 tanpa menyentuh dinding.



Gambar 4.17 Pengujian Belokan 180^0

4.10.1.5 Pengujian Penghindar Halangan

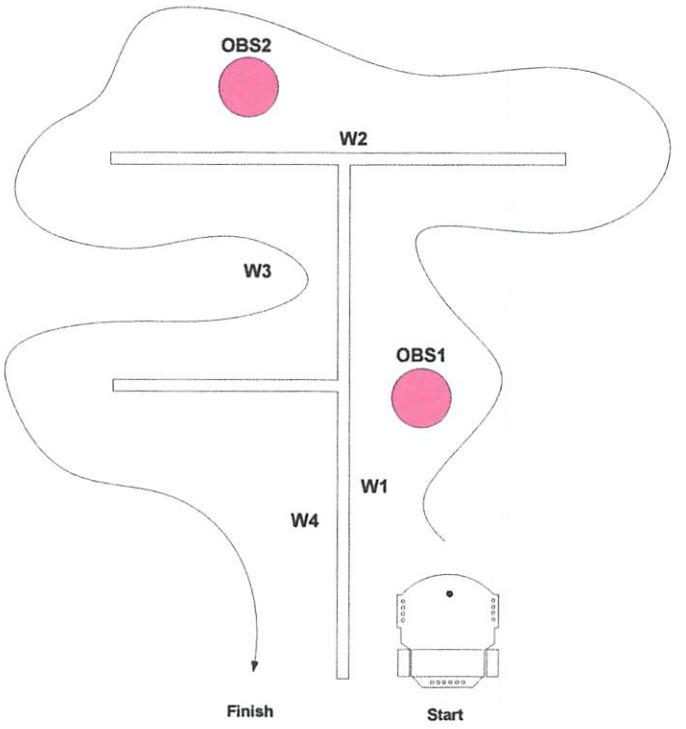
Pengujian menggunakan penghalang berbentuk tabung dengan diameter 5 cm dengan warna merah (*H=100, S=0, V=0*). Lintasan yang dipakai adalah Lintasan lurus ditambah dengan halangan pada posisi tengah lintasan. Indikator kerberhasilan pergerakan robot adalah kemampuan untuk menghindari halangan tanpa menabrak halangan.



Gambar 4.18 Pengujian *Obstacle Avoidance* (Penghindar Halangan)

4.10.1.6 Pengujian Telusur Dinding dan Penghindar Halangan

Pengujian dilakukan untuk melihat kemampuan robot dalam menelusur dinding dan menghindar halangan dengan berbagai sudut belokan dan posisi halangan. Prosedur pengujian adalah robot bergerak dari posisi start menuju finish dengan indikator keberhasilan adalah kemampuan untuk bergerak maju dengan presentase menyentuh dinding dan menabrak halangan sekecil mungkin.



Gambar 4.19 Pengujian Telusur Dinding dan Penghindar Halangan

4.10.2 Hasil Pengujian

Tabel 4.14 Hasil Pengujian Pergerakan Robot Pada Lintasan Lurus

Percobaan	1	2	3	4	5	6	7	8	9	10
Hasil	√	√	√	√	√	√	√	√	√	√

Keterangan :

√ = Berhasil

X = Menyentuh Dinding

Tabel 4.15 Hasil Pengujian Pergerakan Robot Dengan Variasi Sudut Start

Sudut	Percobaan				
	1	2	3	4	5
0	√	√	√	√	√
20	√	√	√	√	√
30	√	√	√	√	√
40	√	√	√	√	√
50	√	√	√	√	√
60	√	√	X	√	√
70	√	X	X	√	√
80	√	√	√	√	X
90	√	√	√	√	√

Keterangan :
√ = Berhasil
X = Menyentuh Dinding

Tabel 4.16 Hasil Pengujian Pergerakan Robot Pada belokan 90°

Percobaan	1	2	3	4	5	6	7	8	9	10
Hasil	X	√	√	√	√	√	X	√	√	√

Keterangan :

√ = Berhasil

X = Menyentuh Dinding

Tabel 4.17 Hasil Pengujian Pergerakan Robot Pada Belokan 180°

Percobaan	1	2	3	4	5	6	7	8	9	10
Hasil	√	√	√	√	√	√	√	√	√	√

Keterangan :

√ = Berhasil

X = Menyentuh Dinding

Tabel 4.18 Hasil Pengujian Pergerakan Robot Untuk Penghindar Halangan

Percobaan	1	2	3	4	5	6	7	8	9	10
Hasil	√	√	√	X	√	√	√	√	√	√

Keterangan :

√ = Berhasil

X = Menyentuh Dinding / Halangan

Tabel 4.19 Hasil Sistem Pergerakan Robot Keseluruhan

Percobaan	W1	W2	W3	W4	OBS1	OBS2
1	√	√	√	√	X	√
2	√	√	X	√	√	√
3	√	X	√	√	√	√
4	√	X	X	√	√	√
5	√	X	√	√	√	X
6	√	√	√	√	√	X
7	√	√	√	√	√	√
8	√	√	√	√	X	√
9	√	√	√	√	√	√
10	√	X	√	√	√	√

Keterangan :

√ = Berhasil

X = Menyentuh Dinding / Halangan

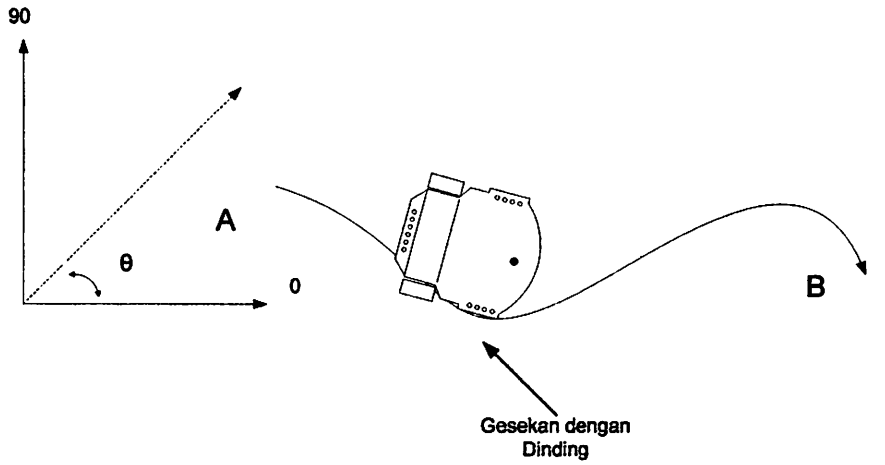
4.10.3 Analisa Hasil Pengujian

4.10.3.1 Analisa Pada lintasan Lurus

Pada pengujian pertama yang ditunjukkan pada tabel 4.14 di atas, robot berhasil bergerak maju pada lintasan lurus dengan persentase keberhasilan 100 %. Sehingga dapat disimpulkan bahwa tingkat kepresisian dari metode kontrol fuzzy telah dapat diterapkan dengan baik.

4.10.3.2 Analisa dengan Variasi Sudut Start

Pengujian sudut start dimaksudkan untuk menguji tingkat kestabilan robot dengan nilai sudut yang berubah. Dari tabel 4.15 di atas, dilakukan 5 kali percobaan untuk setiap nilai sudut dengan nilai 0^0 berarti robot sejajar dengan dinding dan posisi 90^0 berarti start robot berada pada posisi tegak lurus dengan dinding. Dari Hasil pengujian didapat bahwa robot mengalami 1 kali error pada posisi sudut 60^0, dua kali error pada sudut 70^0 dan 1 kali error pada sudut 90^0.

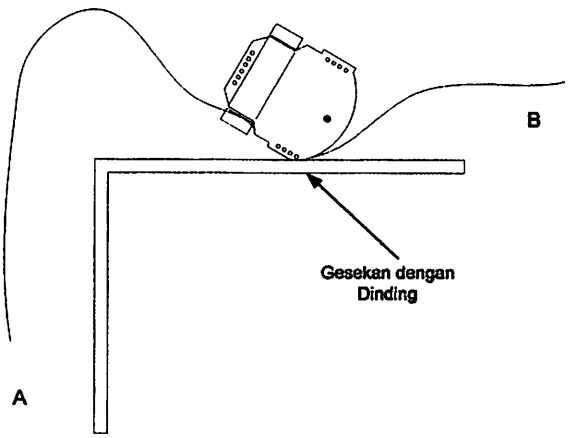


Gambar 4.20 Error Yang Terjadi pada Pengujian Dengan Variasi Sudut Start

Error yang terjadi bisa disebabkan oleh terjadinya pemberian sinyal kontrol yang kurang sesuai sehingga terjadi osilasi yang terlalu besar. Solusi yang dapat diterapkan untuk mengatasi error ini adalah dengan menurunkan kecepatan atau mengatur aturan pada kontrol fuzzy sesuai dengan karakteristik delta error yaitu meredam terjadinya overshoot.

4.10.3.3 Analisa Pengujian Pada Belokan 90^0

Pada Tabel 4.16 ditunjukkan hasil pengujian pergerakan robot pada saat berbelok dengan sudut 90^0. Dari hasil pengujian didapat gesekan dengan dinding pada pengujian 1 dan 7 namun robot dapat kembali bergerak maju dan menstabilkan posisi kembali.

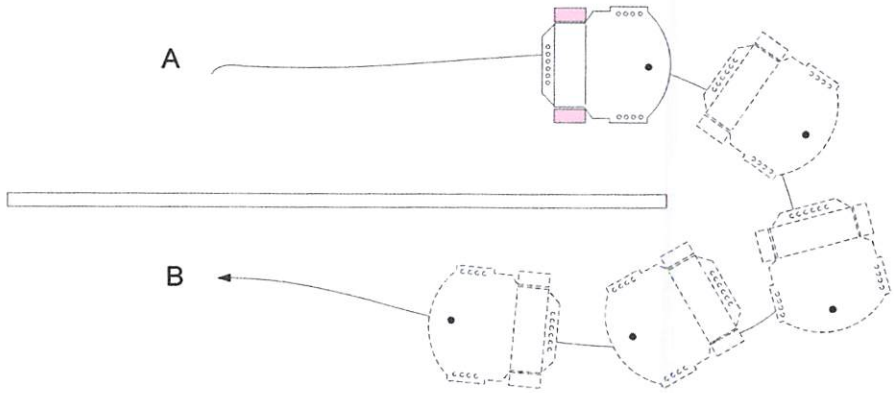


Gambar 4.21 Error pengujian Pada Pengujian belokan 90^0

Error yang terjadi disebabkan karena terjadinya efek dinamik dari robot dimana nilai percepatan akan menyebabkan robot begerak terlalu jauh sehingga haluan robot semakin melebar. selain itu efek dari motor / roda yang slip juga dapat menyebabkan pergerakan robot menjadi tidak stabil. Efek dinamik pada robot dapat dikurangi dengan merancang desain mekanik dengan bahan yang lebih ringan,

4.10.3.4 Analisa Pengujian Pada Belokan 180^0

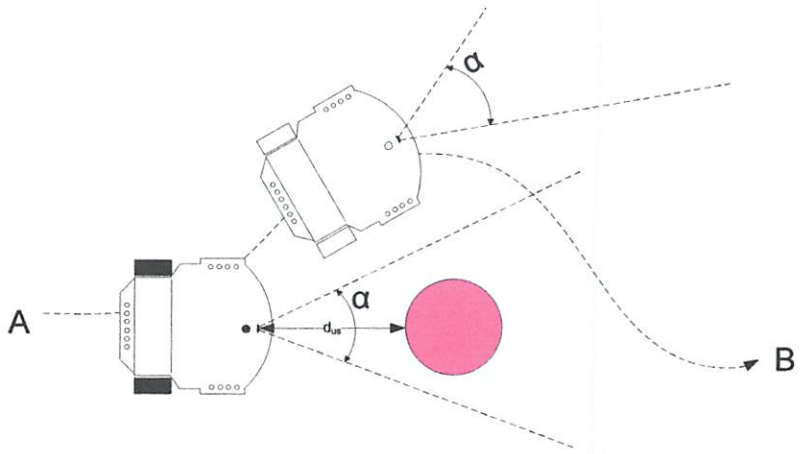
Pada Tabel 4.18 ditunjukkan hasil pergerakan robot pada belokan 180^0 dengan percobaan sebanyak 10 kali. Hasil yang didapat adalah robot mampu berbelok dengan sempurna tanpa menyentuh dinding sehingga dapat dikatakan bahwa tingkat kepresisian robot lebih baik pada belokan 180^0. Gerakan belokan 180^0 ini merepresentasikan gerakan *nonholonomic* pada konstruksi DDMR (*Differential Drive Mobile Robot*).



Gambar 4.22 Pergerakan Robot pada Belokan 180^0

4.10.3.5 Analisa Percobaan *Obstacle Avoider* (Penghindar Halangan)

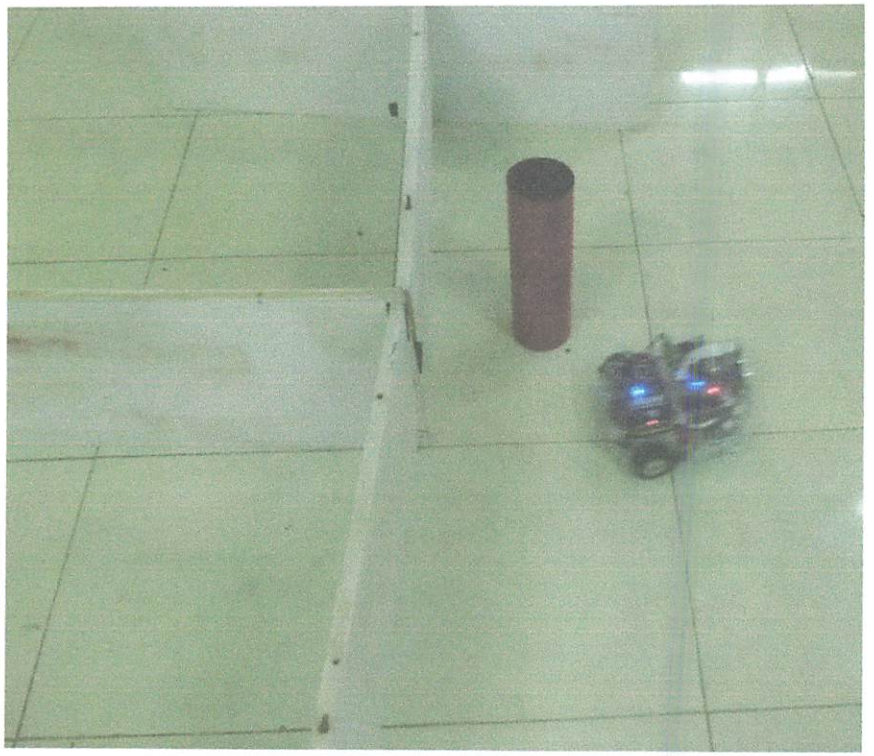
Pada percobaan penghindar halangan, presentase keberhasilan adalah 90 % dengan 1 kali kegagalan. Kegagalan disebabkan karena faktor area jangkauan kamera yang terbatas yaitu 46.85^0 pada area horizontal dan faktor pencahayaan. Selain itu faktor kecepatan robot juga berpengaruh terhadap pembacaan sensor karena kecepatan pembacaan *frame* kamera yang terbatas yaitu sekitar 10 fps (*frame per second*).



Gambar 4.23 Pergerakan Robot Untuk Menghindar Halangan

4.10.3.6 Analisa Pengujian Keseluruhan Sistem Pergerakan Robot

Pada pengujian keseluruhan sistem robot didapatkan 4 kali error ketika menghindar halangan sedangkan pada saat melakukan telusur dinding terdapat error sebanyak 6 kali yaitu keadaan menyentuh dinding namun masih bisa melakukan pergerakan. Error yang sering terjadi ketika melakukan pergerakan adalah keadaan robot dengan haluan yang terlalu besar. Error terjadi karena terjadinya momentum pergerakan robot pada lintasan lurus sehingga robot tidak mampu menahan gaya ke depan karena faktor berat. Untuk itu agar presentase keberhasilan dapat maksimal, perlu dilakukan pengaturan kecepatan ideal untuk robot bergerak maju , belok dan menghindar halangan.



Gambar 4.24 Pengujian pada Lintasan

Berdasarkan tabel 4.19 presentase keberhasilan pergerakan mengikuti dinding dan menghindar halangan dapat dihitung dengan persamaan :

$$\%\ Keberhasilan = \frac{pengujian berhasil}{total\ Pengujian} x\ 100\%$$

$$\%\ Keberhasilan = \frac{50}{60} x\ 100\%$$

$$\%\ Keberhasilan = 83\%$$

Dengan demikian dapat disimpulkan bahwa pengujian sistem secara keseluruhan telah berhasil dengan beberapa perbaikan yang diperlukan untuk memperbaiki performa sistem.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah dilakukan perancangan, pengujian dan analisa sistem, maka dapat disimpulkan beberapa hal yang dapat digunakan untuk perbaikan dan pengembangan selanjutnya, yaitu :

1. Kontroler fuzzy memiliki efektifitas yang sama seperti pengendali manusia namun membutuhkan memori yang cukup besar untuk dapat menyimpan variabel dan aturan-aturan fuzzy.
2. Efek Dinamik sangat berpengaruh terhadap sistem kendali robot namun dapat dikurangi dengan merancang konstruksi mekanik yang ringan dan sistem roda yang anti slip.
3. Penggunaan konstruksi DDMR (*Differential Drive Mobile Robot*) lebih memudahkan pergerakan robot untuk maju, belok dan mundur jika dibandingkan dengan konstruksi 4 roda dengan *belt* namun kurang kokoh pada saat digunakan pada beban yang berat.
4. Penggunaan banyak sensor ultrasonik dapat mengakibatkan terjadinya interferensi antar sensor dan dapat diatasi dengan memberikan jeda pembacaan namun berdampak pada respon sistem kontrol.
5. Posisi sensor ultrasonik kiri dan kanan sangat berpengaruh terhadap pergerakan robot terutama saat berbelok 90^0 dan 180^0. dari hasil pengujian diperoleh posisi sensor optimal pada posisi 40^0 - 45^0.
6. Penggunaan sensor kamera dapat memudahkan robot untuk mengenali objek atau penghalang namun sangat terpengaruh oleh faktor pencahayaan.
7. Pengimplementasian metode fuzzy dengan mengadaptasi karakteristik dari PD (*Proportional Derrivative*) mampu membuat robot bergerak secara otomatis mengikut dinding dan menghindar halangan dengan presentase sebesar 83 %.

5.2 Saran

Pada perancangan dan pembuatan tugas akhir ini tak lepas dari kekurangan-kekurangan baik dari perancangan sistem maupun peralatan yang digunakan, untuk itu agar sistem dapat menjadi lebih baik maka diperlukan beberapa perbaikan dan penyempurnaan, yaitu :

1. Untuk meningkatkan respon kontrol maka diperlukan sensor dengan kecepatan yang lebih cepat seperti sensor SRF (*Sensor Range Finder*)
2. Untuk mendapatkan nilai haluan yang proporsional terhadap sudut lintasan maka disarankan untuk menggunakan motor DC *gearbox* yang dilengkapi dengan *brake* (rem).
3. Untuk meningkatkan presentase keberhasilan robot dalam menelusuri jalur (*path*) maka direkomendasikan menggunakan dua buah sensor yang diletakkan di sebelah pojok kanan/kiri dan di sebelah kanan/kiri.
4. Untuk mendapatkan nilai rule fuzzy yang paling optimal maka pada pengembangan selanjutnya dapat digunakan metode *neuro fuzzy*.

DAFTAR PUSTAKA

Andrianto, Heri.(2008). "*Pemrograman Mikrokontroler Atmega16 Menggunakan Bahasa C*". Informatika:Bandung

Atmel, 2007. "*Atmega 162 Datasheet*". URL : http://www.atmel.com/Images/Atmel-2513-8-bit-AVR-Microntroller ATmega162_Datasheet.pdf

Iqbal, Rully Muhammad, Dikairono, Rufy. 2012. "*Implementasi Sistem Navigasi Behaviour-based Robotic dan Kontroller Fuzzy pada Manuver Robot Cerdas Pemadam Api*". JURNAL TEKNIK POMITS Vol. 1 No. 1

Kuswadi, Son (2007), "*Kendali Cerdas Teori dan Aplikasinya*".Andi: Yogyakarta
Pitowarno, Endra.(2006). "*Robotika Disain, Kontrol, dan kecerdasan Buatan*". Andi : Yogyakarta.

Raspberry Wiki Page (Rpi CameraModule). URL: http://elinux.org/Rpi_Camera_Module

Raspberry Wiki Page (Rpi Hardware) . URL : http://elinux.org/RPi_Hardware

Soebhakti, Hendawan. (2008). "*AVR Application Note : Ping))) Parallax Ultrasonic Range Finder*". URL : http://hendawan.files.wordpress.com/2009/02/ping-paralax-application.pdf

Suksmadana, I Made Budi, Mubarak,Fithra.(2011). "*Rancang Bangun Robot Mobil Menggunakan Logika Fuzzy Untuk Bernavigasi Berbasiskan Mikrokontroler AVR Atmega 8535*". Jurnal Dielektrika ISSN 2086-9487, Vol.2, No.1:9-17, Februari 2011. URL : *ejournal.ftunram.ac.id/FullPaper/budi.pdf*

Wiratran, Helmi. "*Perancangan dan Implementasi Embedded Fuzzy Logic Controller untuk Pengaturan Gerak Robot Segway Mini*". Sumber http://digilib.its.ac.id/public/ITS-Undergraduate-16888-Paper-747606.pdf . Diakses Tanggal 7 Juli 2014

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
NI (PERSERO) MALANG
NK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : COSMAS ERIC SEPTIAN
NIM : 09.12.217
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Elektronika
MASA BIMBINGAN : SEMESTER GENAP 2013/2014
JUDUL : ***RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY***

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Rabu
Tanggal : 13 Agustus 2014
Dengan Nilai : 90,25 (A)

PANITIA UJIAN SKRIPSI

Ketua Majelis Penguji

M. IbrahimAshari, ST, MT
NIP.P. 1030100358

Sekretaris Majelis Penguji

Dr. Eng. Aryuanto S, ST, MT
NIP.Y.1030800417

ANGGOTA PENGUJI

Dosen Penguji I

Irmalia Suryani Faradisa, ST, MT
NIP. P. 1030100365

Dosen Penguji II

Ir. Eko Nurcahyo, MT
NIP. Y. 1028700172

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA	: COSMAS ERIC SEPTIAN
NIM	: 09.12.217
JURUSAN	: Teknik Elektro S-1
KONSENTRASI	: Teknik Elektronika
MASA BIMBINGAN	: SEMESTER GENAP 2013/2014
JUDUL	: RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PEMETAAN RUANG DAN PENGHINDAR HALANGAN DENGAN MEMAKAI SENSOR ULTRASONIC DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari	: Rabu
Tanggal	: 13 Agustus 2014
Dengan Nilai	: 90,25 (A)

PANITIA UJIAN SKRIPSI

Ketua Majelis Penguji	Sekretaris Majelis Penguji
M. Ibrahim Ashari, ST, MT	Dr. Eng. Aryuanto S, ST, MT
NIP.P. 1030100358	NIP.Y.1030800417

ANGGOTA PENGUJI

Dosen Penguji I	Dosen Penguji II
Irmalia Suryani Faradisa, ST, MT	Ir. Eko Nurcahyo, MT
NIP. P. 1030100365	NIP. Y. 1028700172

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
PROGRAM STUDI TEKNIK ELEKTROS-1
KONSENTRASI TEKNIK ENERGI LISTRIK

PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi Program studi Teknik Elektro jenjang strata satu (S-1) yang diselenggarakan pada :

Hari : Rabu
Tanggal : 13 Agustus 2014

Telah dilakukan perbaikan skripsi oleh :

Nama : Cosmas Eric Septian
NIM : 0912217
Perogram Studi : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika
Judul Skripsi : **RANCANG BANGUN DIFFERENTIAL DRIVE MOBILE ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK DAN MODUL KAMERA RASPBERRY PI MENGGUNAKAN METODE KENDALI LOGIKA FUZZY**

No	Materi Perbaikan	Keterangan
1.	Kesimpulan	
2.	Bab III Tata Tulis, Blok Diagram, Perancangan Transistor	
3.	Yang Sudah ada di Bab 2 tidak usah dimasukkan di bab 3.	

Dosen Penguji I

Irmalia Suryani Faradisa, ST, MT
NIP. P. 1030100365

Dosen Pembimbing I

M. Ibrahim Ashari, ST, MT
NIP. P. 1030100358

Dosen Pembimbing II

Dr. Eng. Aryuanto Soetedjo, ST, MT
NIP. Y. 1030800417

SURAT PERNYATAAN ORISINALITAS

Yang bertandatangan di bawah ini :

Nama : Cosmas Eric Septian

NIM : 09.12.217

Program Studi : T.Elektro S-1

Konsentrasi : Teknik Elektronika

Dengan ini menyatakan bahwa Skripsi yang saya buat adalah hasil karya sendiri , tidak merupakan plagiasi dari karya orang lain. Dalam Skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, 28 Agustus 2014

Yang membuat Pernyataan,

Cosmas Eric Septian

NIM : 0912217

```
**********************************************************************
rogram Raspberry Master yang digunakan untuk
 mengolah data yang dikirim dari uC Slave ATMEGA162
 Kontroller fuzzy
 mengirim data pwm dan direction ke uC Slave MEGA162
 mengirim data ke PC Client via wifi untuk monitoring debug

reated By        : Cosmas Eric Septian
roject           : Tugas Akhir Skripsi 2013/2014
ibrary Serial    : WiringPi (@Gordon Project)
mage Processing  : OpenCV ( Open Computer Vision) -- Color Detecting

**********************************************************************/

lude <iostream>
lude <stdlib.h>
lude <stdio.h>
lude <string.h>
lude <errno.h>
lude <pthread.h>
lude <sys/types.h>
lude <sys/socket.h>
lude <netinet/in.h>
lude <netdb.h>
lude <unistd.h>
lude <arpa/inet.h>
lude <math.h>

lude <cv.h>
lude <highgui.h>
lude "RaspiCamCV.h"

lude <wiringPi.h>
lude <wiringSerial.h>

 namespace std;
 namespace cv;

ine     N_OUT               49
ine     N_RULE              49

 state_send_ready = 0;


.ne     set_point_pwm              50
ine     max_pwm                    200
ine     min_pwm                    0


ned char set_point_pwm = 50;
ned char max_pwm = 200;
ned char min_pwm = 0;
ned char max_pwm_temp = 0;

.ne     maju               0
.ne     mundur             1
.ne     belok_kanan        2
.ne     belok_kiri         3
ne      stop               4

ne      RX_DATA_COUNT      36
ne      TX_DATA_COUNT      36

ne        scan_kiri            0
.ne       scan_kanan           1

**********************************************
 Output Defuzzifikasi
uzzyfikasi degan menggunakan metode Sugeno
gan mengambil titik Tengah (Center Area Methode)
**********************************************/
```

```
l = PWM
.ne      yNB          -220
.ne      yNM          -160
.ne      yNK          -80
.ne      yZ           0
.ne      yPK          80
.ne      yPM          160
.ne      yPB          220

)ut PWM Kanan
)ut PWM Kiri
.ne      SLOW         6
.ne      MEDIUM       80
.ne      FAST         256


 eNB,eNM,eNK,eZ,ePK,ePM,ePB;
 deNB,deNM,deNK,deZ,dePK,dePM,dePB;

lembership function untuk jarak Ultrasonik depan
 obs_jauh,obs_dekat,obs_sangat_dekat;
lembership function untuk posisi koordinat
orizontal halangan depan terhadap posisi robot
lengan menggunakan sensor kamera
 obs_kiri,obs_tengah,obs_kanan;

*******************************
ll Follower
*******************************/
 out_value_wall_follow[N_OUT];
)UT_RULE_WALL_FOLLOW[N_RULE];

*******************************
stacle Avoider
*******************************/
 out_value_obs[9];
)UT_RULE_PWM_KI[9];
)UT_RULE_PWM_KA[9];

os;
e obs_area;

***************************
 Sensor Jarak Pojok Kiri
 Sensor Jarak Depan
 Sensor Jarak Pojok Kanan
**************************/

 set_US1=0,set_US3=0;
 err_US1,err_US3,prev_err_US1,prev_err_US3,derr_US1,derr_US3;
 jarak_depan;
 fuzzy_output_pwm=0;
 fuzzy_output_pwm_kiri=0,fuzzy_output_pwm_kanan=0;

ned int count_rx = 0;
rx_data[RX_DATA_COUNT];
data_rx;
d;
r_ready = 0;
jarak_us1[6],jarak_us2[6],jarak_us3[6],mode_following[2],signal_start[2];
 jarak_us1_f,jarak_us2_f,jarak_us3_f;
ned int mode_following_ui,signal_start_ui;

tx_data[TX_DATA_COUNT];
wm_kanan,pwm_kiri,dir;

************************************
zyfikasi dengan menggunakan fungsi
nggotaan segitiga (Triangle Function)
************************************/
ne      unlimited_upper      0
```

```
ine     unlimited_lower     1
ine     limited             2
t trimf(float crisp_input,int a_val,int b_val,int c_val,unsigned char mode)

float numerator=0, denumerator=0;

if (crisp_input < a_val )
{
    switch(mode)
    {
    case limited :
        return (0);
        break;
    case unlimited_lower :
        return (1);
        break;
    case unlimited_upper :
        return (0);
        break;
    }

}
else
{
    if((crisp_input>=a_val) && (crisp_input<=(b_val)))
    {
        numerator = (crisp_input-(a_val));
        denumerator = ((b_val)-(a_val));
        return (float)(numerator/denumerator);
    }
    else
    {
        if((crisp_input>=(b_val))&&(crisp_input<=c_val))
        {
            numerator = (c_val-(crisp_input));
            denumerator = ((c_val)-(b_val));
            return (float)(numerator/denumerator);
        }
        else
        {
            switch(mode)
            {
            case limited :
                return (0);
                break;
            case unlimited_lower :
                return (0);
                break;
            case unlimited_upper :
                return (1);
                break;
            }
        }
    }


gsi untuk evaluation rule dengan menggunakan tiga variable

ne     AND     0
ne     OR      1
 rule_wall_follow(int  rule_index,  float  var_linguistik1,  unsigned  char  OPERATOR  ,  float
inguistik2,int y1)

UT_RULE_WALL_FOLLOW[rule_index] = y1;

witch(OPERATOR)

ase AND :
   out_value_wall_follow[rule_index] = fmin(var_linguistik1,var_linguistik2);
```

```
    break;
case OR :
    out_value_wall_follow[rule_index] = fmax(var_linguistik1,var_linguistik2);
    break;
}


 rule_obs_avoid(int rule_index, float var_linguistik1, unsigned char OPERATOR ,
                float var_linguistik2,int pwm_ki,int pwm_ka)

OUT_RULE_PWM_KA[rule_index] = pwm_ka;
OUT_RULE_PWM_KI[rule_index] = pwm_ki;

switch(OPERATOR)
{
case AND :
    out_value_obs[rule_index] = fmin(var_linguistik1,var_linguistik2);
    break;
case OR :
    out_value_obs[rule_index] = fmax(var_linguistik1,var_linguistik2);
    break;
}


 out_value_reset()

unsigned char i=0;
for(i=0; i<N_OUT; i++)
{
    out_value_wall_follow[i] = 0;
}

for(i=0; i<9; i++)
{
    out_value_obs[i] = 0;
}


ine     WALL_FOLLOW         0
ine     OBSTACLE_AVOID      1
ine     PWM_KANAN           2
ine     PWM_KIRI            3
: defuzzifikasi_cog_methode(int *out_rule, unsigned char output, int *check)

float num = 0,
           denum = 0;
unsigned char i = 0;

switch(output)
{
    /**
    ** Output = PWM (Kecepatan) robot
    **/
case WALL_FOLLOW :
    for(i=0; i<N_OUT; i++)
    {
        num += (out_rule[i]*out_value_wall_follow[i]);
        denum += out_value_wall_follow[i];
    }
    break;
case OBSTACLE_AVOID :
    /**
    ** DEfuzzyfikasi Obstacle Avoidance
    ** Output 1 = PWM Kiri
    ** Output 2 = PWM Kanan
    **/
    for(i=0; i<9; i++)
    {
        num += (out_rule[i]*out_value_obs[i]);
        denum += out_value_obs[i];
    }
```

```
    break;
default :
    break;
}

/** Cek Pembagian dengan NOL*/
if (denum == 0)
{
    *check = 0;
    return (0);
}
else
{
    *check = 1;
}

return (float)(num/denum);



////////////// FUZZYFIKASI//////////////////////////////////////
 fuzzyfikasi_wall_follow(unsigned char mode)

switch(mode)
{
case scan_kiri :
    eNB = trimf(err_US1,-16,-9,-6,unlimited_lower);
    eNM = trimf(err_US1,-9,-6,-3,limited);
    eNK = trimf(err_US1,-6,-3,0,limited);
    eZ = trimf(err_US1,-3,0,3,limited);
    ePK = trimf(err_US1,0,3,6,limited);
    ePM = trimf(err_US1,3,6,9,limited);
    ePB = trimf(err_US1,6,9,16,unlimited_upper);

    deNB = trimf(derr_US1,-20,-15,-10,unlimited_lower);
    deNM = trimf(derr_US1,-15,-10,-5,limited);
    deNK = trimf(derr_US1,-10,-5,0,limited);
    deZ = trimf(derr_US1,-5,0,5,limited);
    dePK = trimf(derr_US1,0,5,10,limited);
    dePM = trimf(derr_US1,5,10,15,limited);
    dePB = trimf(derr_US1,10,15,20,unlimited_upper);
    break;
case scan_kanan :
    eNB = trimf(err_US3,-12,-9,-6,unlimited_lower);
    eNM = trimf(err_US3,-9,-6,-3,limited);
    eNK = trimf(err_US3,-6,-3,0,limited);
    eZ = trimf(err_US3,-3,0,3,limited);
    ePK = trimf(err_US3,0,3,6,limited);
    ePM = trimf(err_US3,3,6,9,limited);
    ePB = trimf(err_US3,6,9,12,unlimited_upper);

    deNB = trimf(derr_US3,-24,-18,-12,unlimited_lower);
    deNM = trimf(derr_US3,-18,-12,-6,limited);
    deNK = trimf(derr_US3,-12,-6,0,limited);
    deZ = trimf(derr_US3,-6,0,6,limited);
    dePK = trimf(derr_US3,0,6,12,limited);
    dePM = trimf(derr_US3,6,12,18,limited);
    dePB = trimf(derr_US3,12,18,24,unlimited_upper);
    break;
default :
    break;
}



 fuzzyfikasi_obstacle_avoidance()

obs_sangat_dekat = trimf(jarak_us2_f,20,16,20,limited);
obs_dekat = trimf(jarak_us2_f,16,20,30,limited);
obs_jauh = trimf(jarak_us2_f,20,30,40,limited);
```

```
)bs_kiri = trimf(pos,0,80,160,limited);
)bs_tengah = trimf(pos,80,160,250,limited);
)bs_kanan = trimf(pos,160,240,320,limited);


'////////////////////// EVALUASI RULE ////////////////////////
'*************************************************************
ıluasi Rule Untuk Wall Following
ıgan Parameter Error dan Delta Error
'*************************************************************/
evaluasi_rule_wall_follow()

:ule_wall_follow(0,eNB,AND,deNB,yNB);
:ule_wall_follow(1,eNB,AND,deNM,yNB);
'ule_wall_follow(2,eNB,AND,deNK,yNB);
:ule_wall_follow(3,eNB,AND,deZ,yNM);
'ule_wall_follow(4,eNB,AND,dePK,yNM);
:ule_wall_follow(5,eNB,AND,dePM,yNM);
:ule_wall_follow(6,eNB,AND,dePB,yZ);

:ule_wall_follow(7,eNM,AND,deNB,yNB);
:ule_wall_follow(8,eNM,AND,deNM,yNB);
'ule_wall_follow(9,eNM,AND,deNK,yNM);
'ule_wall_follow(10,eNM,AND,deZ,yNM);
'ule_wall_follow(11,eNM,AND,dePK,yNM);
'ule_wall_follow(12,eNM,AND,dePM,yZ);
'ule_wall_follow(13,eNM,AND,dePB,yPM);

:ule_wall_follow(14,eNK,AND,deNB,yNB);
'ule_wall_follow(15,eNK,AND,deNM,yNM);
'ule_wall_follow(16,eNK,AND,deNK,yNM);
'ule_wall_follow(17,eNK,AND,deZ,yNK);
'ule_wall_follow(18,eNK,AND,dePK,yZ);
'ule_wall_follow(19,eNK,AND,dePM,yPM);
'ule_wall_follow(20,eNK,AND,dePB,yPM);

'ule_wall_follow(21,eZ,AND,deNB,yNK);
'ule_wall_follow(22,eZ,AND,deNM,yNM);
'ule_wall_follow(23,eZ,AND,deNK,yNK);
'ule_wall_follow(24,eZ,AND,deZ,yZ);
'ule_wall_follow(25,eZ,AND,dePK,yPK);
'ule_wall_follow(26,eZ,AND,dePM,yPM);
'ule_wall_follow(27,eZ,AND,dePB,yPM);

'ule_wall_follow(28,ePK,AND,deNB,yNM);
'ule_wall_follow(29,ePK,AND,deNM,yNM);
'ule_wall_follow(30,ePK,AND,deNK,yZ);
'ule_wall_follow(31,ePK,AND,deZ,yPK);
'ule_wall_follow(32,ePK,AND,dePK,yPM);
'ule_wall_follow(33,ePK,AND,dePM,yPM);
'ule_wall_follow(34,ePK,AND,dePB,yPB);

'ule_wall_follow(35,ePM,AND,deNB,yNM);
'ule_wall_follow(36,ePM,AND,deNM,yZ);
'ule_wall_follow(37,ePM,AND,deNK,yPM);
'ule_wall_follow(38,ePM,AND,deZ,yPM);
'ule_wall_follow(39,ePM,AND,dePK,yPM);
'ule_wall_follow(40,ePM,AND,dePM,yPB);
'ule_wall_follow(41,ePM,AND,dePB,yPB);

'ule_wall_follow(42,ePB,AND,deNB,yZ);
'ule_wall_follow(43,ePB,AND,deNM,yPM);
'ule_wall_follow(44,ePB,AND,deNK,yPM);
'ule_wall_follow(45,ePB,AND,deZ,yPM);
'ule_wall_follow(46,ePB,AND,dePK,yPB);
'ule_wall_follow(47,ePB,AND,dePM,yPB);
'ule_wall_follow(48,ePB,AND,dePB,yPB);
```

```
**********************************************************
aluasi Rule untuk penghindar halangan (Obstacle Avoidance)
mlah Output = 2
tput 1 PWM Kiri
tput 2 PWM Kanan
**********************************************************/
  evaluasi_rule_obstacle_avoidance(unsigned char mode)


switch(mode)
{
case scan_kiri :
    rule_obs_avoid(0,obs_sangat_dekat,AND,obs_kiri,FAST,SLOW);
    rule_obs_avoid(1,obs_sangat_dekat,AND,obs_tengah,FAST,SLOW);
    rule_obs_avoid(2,obs_sangat_dekat,AND,obs_kanan,MEDIUM,SLOW);

    rule_obs_avoid(3,obs_dekat,AND,obs_kiri,FAST,SLOW);
    rule_obs_avoid(4,obs_dekat,AND,obs_tengah,MEDIUM,SLOW);
    rule_obs_avoid(5,obs_dekat,AND,obs_kanan,MEDIUM,SLOW);

    rule_obs_avoid(3,obs_jauh,AND,obs_kiri,FAST,MEDIUM);
    rule_obs_avoid(4,obs_jauh,AND,obs_tengah,MEDIUM,SLOW);
    rule_obs_avoid(5,obs_jauh,AND,obs_kanan,MEDIUM,SLOW);
    break;

case scan_kanan :
    rule_obs_avoid(0,obs_sangat_dekat,AND,obs_kiri,SLOW,FAST);
    rule_obs_avoid(1,obs_sangat_dekat,AND,obs_tengah,SLOW,FAST);
    rule_obs_avoid(2,obs_sangat_dekat,AND,obs_kanan,SLOW,MEDIUM);

    rule_obs_avoid(3,obs_dekat,AND,obs_kiri,SLOW,MEDIUM);
    rule_obs_avoid(4,obs_dekat,AND,obs_tengah,SLOW,MEDIUM);
    rule_obs_avoid(5,obs_dekat,AND,obs_kanan,SLOW,MEDIUM);

    rule_obs_avoid(3,obs_jauh,AND,obs_kiri,SLOW,SLOW);
    rule_obs_avoid(4,obs_jauh,AND,obs_tengah,SLOW,MEDIUM);
    rule_obs_avoid(5,obs_jauh,AND,obs_kanan,SLOW,MEDIUM);
    break;
default :
    break;
}


////////////////////// End Of Evaluasi Rule ////////////////////////////

 clear()

system("cls");



 1

p variable
 pwmKanan,pwmKiri;

**********************************************************
in untuk terima data dengan menggunakan threding
 parsing data menjadi data us dan data mode + start_signal
a#1 = data sensor ultrasonik pojok kiri
a#2 = data sensor ultrasonik depan
a#3 = data sensor ultrasonik pojok kanan
a#4 = data mode following
a#5 = data signal start dari uc Slave
**********************************************************/
 *terima_data(void *threadid)

unsigned char i=0,j=0,k=0;
nsigned char dataLen = 0;
```

```
while(1)
{
    while(serialDataAvail(fd))
    {
        /*
        * Terima data serial
        */
        data_rx = serialGetchar(fd);
        rx_data[count_rx++] =  data_rx;

        if((data_rx==13)||(count_rx>=RX_DATA_COUNT))
        {
            //reset counter index
            count_rx = 0;
            dataLen = strlen(rx_data);

            //parsing data sensor jarak us1 / sensor pojok kiri
            for(i=0; i<dataLen; i++)
            {
                if(rx_data[i]==';') break;
                jarak_us1[j++] = rx_data[i];
            }
            jarak_us1_f = atof(jarak_us1);
            for(k=0; k<6; k++) jarak_us1[k] = 0;

            //parsing data sensor jarak us2 / sensor depan
            j=0;
            for(i=i+1; i<dataLen; i++)
            {
                if(rx_data[i]==';') break;
                jarak_us2[j++] = rx_data[i];
            }
            jarak_us2_f = atof(jarak_us2);
            for(k=0; k<6; k++) jarak_us2[k] = 0;

            //parsing data sensor us3 / jarak pojok kanan
            j=0;
            for(i=i+1; i<dataLen; i++)
            {
                if(rx_data[i]==';') break;
                jarak_us3[j++] = rx_data[i];
            }
            jarak_us3_f = atof(jarak_us3);
            for(k=0; k<6; k++) jarak_us3[k] = 0;

            //parsing data mode
            j=0;
            for(i=i+1; i<dataLen; i++)
            {
                if(rx_data[i]==';') break;
                mode_following[j++] = rx_data[i];
            }
            mode_following_ui = atoi(mode_following);
            for(k=0; k<2; k++) mode_following[k] = 0;

            //parsing data signal start
            j=0;
            for(i=i+1; i<dataLen; i++)
            {
                if(rx_data[i]==';') break;
                signal_start[j++] = rx_data[i];
            }
            signal_start_ui = atoi(signal_start);
            for(k=0; k<2; k++) signal_start[k] = 0;
            j=0;

            // ready signal
            r_ready = 1;
        }
    }
```

```
pthread_exit(NULL);


*******************************
utin untuk mengirim data ke Wifi
enggunakan Threading
*******************************/
 data_buff[200];
socket_desc , client_sock , c , read_size;
ct sockaddr_in server , client;
 client_message[500];

 *kirim_data_to_wifi(void *threadid)

socket_desc = socket(AF_INET , SOCK_STREAM , 0);
if (socket_desc == -1)
{
    printf("Could not create socket");
}
puts("Socket created");

server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons( 5000 );

if( bind(socket_desc,(struct sockaddr *)&server , sizeof(server)) < 0)
{
    perror("bind failed. Error");
    //      return 1;
}
puts("bind done");

while(1)
{
    listen(socket_desc , 3);
    puts("Waiting for incoming connections...");
    c = sizeof(struct sockaddr_in);
    client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t*)&c);
    if (client_sock < 0)
    {
        perror("accept failed");
          return 1;
    }
    puts("Connection accepted");

    //Receive a message from client
    while( (read_size = recv(client_sock , client_message , 2000 , 0)) > 0 )
    {
        /*while(1)
                {

        //printf("Masuk di While Kirim\n");

tf(data_buff,"%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;
;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f",
                            err_US1,derr_US1,jarak_us2_f,pos,

   eNB,eNM,eNK,eZ,ePK,ePM,ePB,deNB,deNM,deNK,deZ,dePK,dePM,dePB,
                                obs_jauh,obs_dekat,obs_sangat_dekat,
                                obs_kiri,obs_tengah,obs_kanan);
        write( client_sock, data_buff, strlen(data_buff));
        //recv(client_sock, client_message, 2000, 0);
        usleep(100000);*/
    }

v(client_sock, client_message, 2000, 0);
    printf("Close Client Sock\n");
    close (client_sock);

    if(read_size == 0)
    {
```

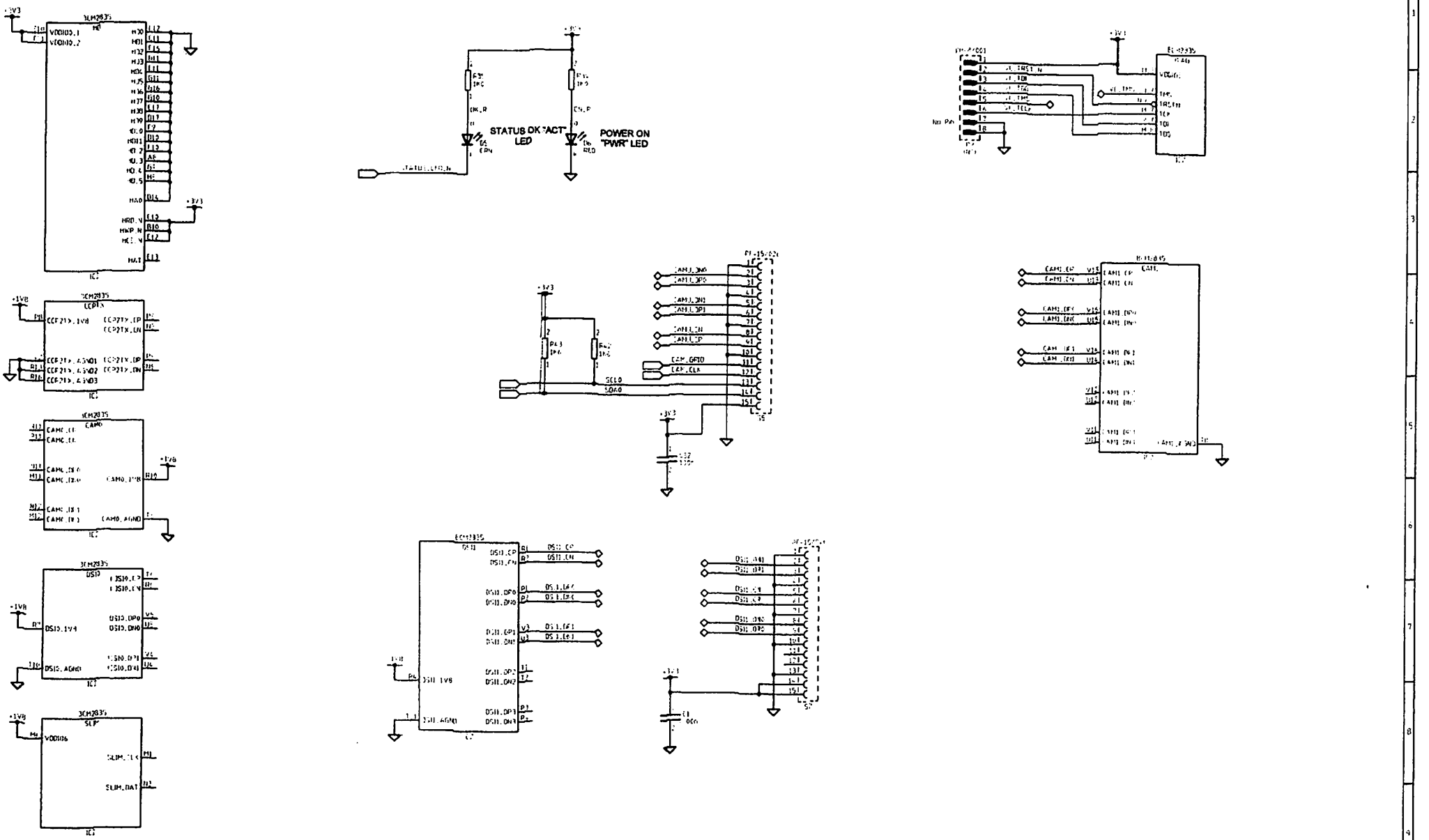
```
        puts("Client disconnected");
        fflush(stdout);
    }
    else if(read_size == -1)
    {
        perror("recv failed");
    }

)thread_exit(NULL);


********** Image Processing ******************
engohan Citra Digunakan untuk mendeteksi halangan
engan ciri berdsarkan warna
.brary yang digunakan adalah openCV (open Computer Vision)
eknik yang digunakan adalah dengan Thresholding
ida streaming picture.....................
*********************************************/
nage* GetThresholdedImage(IplImage* img)

'//Convert Image dari Format RGB ke HSV
:plImage* imgHSV = cvCreateImage(cvGetSize(img),8,3);
:vCvtColor(img,imgHSV,CV_BGR2HSV);

'/Create new image to hold Threshold image
:plImage* imgThreshed = cvCreateImage(cvGetSize(img),8,1);

'//cvInRangeS(imgHSV, cvScalar(HueMin,sMin, vMin), cvScalar(HueMax, sMax, vMax), imgThreshed);
:vInRangeS(imgHSV, cvScalar(0,100, 100), cvScalar(10, 256, 256), imgThreshed);

:vReleaseImage(&imgHSV);
'eturn imgThreshed;


e get_posx(unsigned int xpos)

'eturn xpos;


*color_detect(void * threadid)

aspiCamCvCapture * capture = raspiCamCvCreateCameraCapture(0); // Index doesn't really
r

plImage* frameCSI_Cam;
hile(1)

    frameCSI_Cam = raspiCamCvQueryFrame(capture);
    if (!frameCSI_Cam) break;

    IplImage* imgThreshold = GetThresholdedImage(frameCSI_Cam);

    // Calculate the moments to estimate the position
    CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
    cvMoments(imgThreshold, moments, 1);

    // The actual moment values
    double moment10 = cvGetSpatialMoment(moments, 1, 0);
    double moment01 = cvGetSpatialMoment(moments, 0, 1);
    double area = cvGetCentralMoment(moments, 0, 0);

    // Last Position
    static int posX = 0;
    static int posY = 0;

    posX = moment10/area;
    pos = posX;
    obs_area = area;

    // Print it out for debugging purposes
    //printf("posX:%d  Area:%.0f\n", posX,area);
```

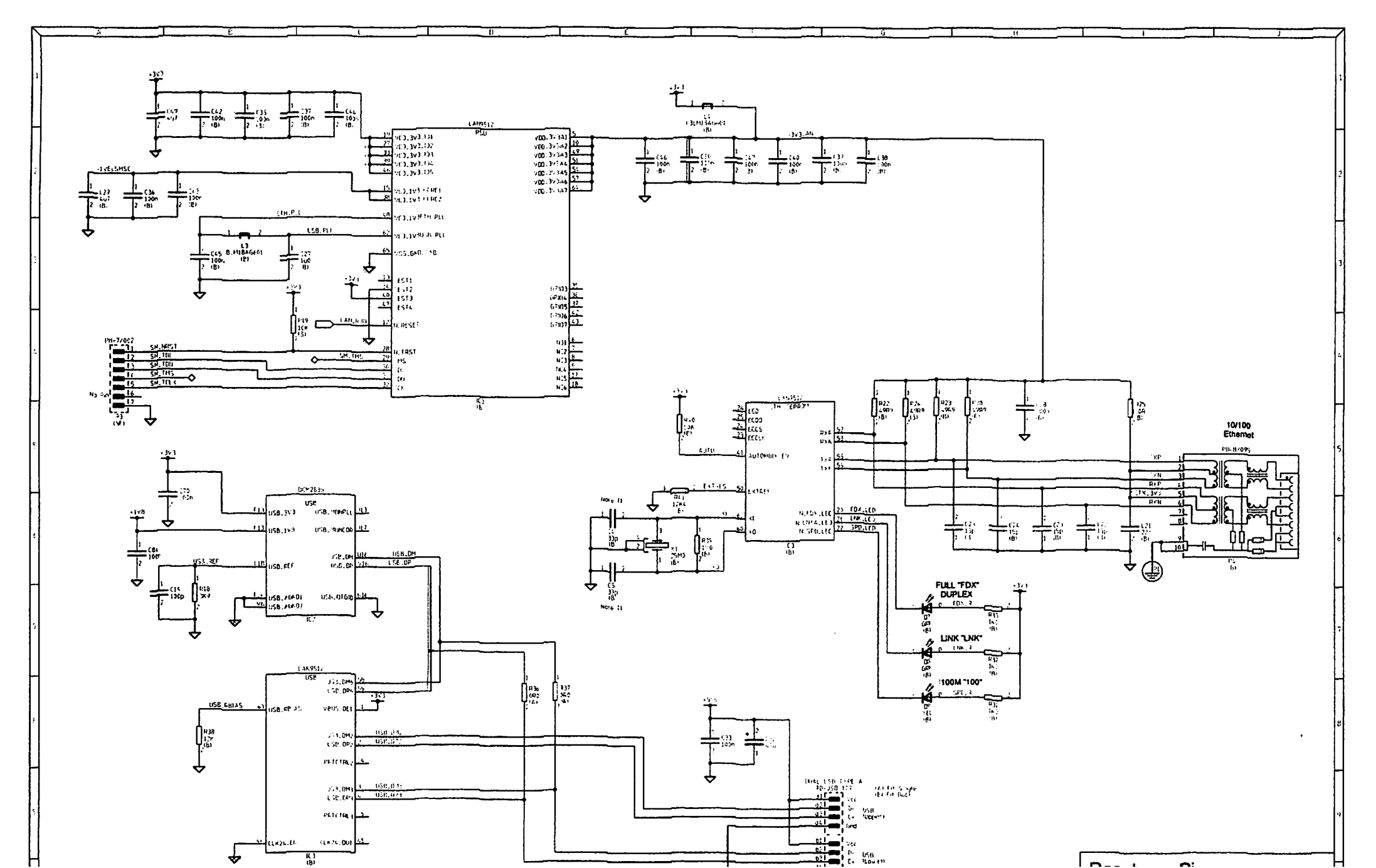
```
    // Release the thresholded image+moments
    cvReleaseImage(&imgThreshold);
    delete moments;

}
raspiCamCvReleaseCapture(&capture);
pthread_exit(NULL);


main()

int zero_status;
float y_pwm=0,y_pwm_kiri=0,y_pwm_kanan=0;
float y1_pwm=0,y1_pwm_kiri=0,y1_pwm_kanan=0;

// create Thread
pthread_t threads[3];

int rc1=0,rc2=0,rc3=0;
int n = 0;

// Thread untuk Terima data Serial dari
// Kontroller Slave MEGA162
rc1 = pthread_create(&threads[0], NULL, terima_data, (void *)0);
rc2 = pthread_create(&threads[1], NULL, color_detect, (void *)1);
rc3 = pthread_create(&threads[2], NULL, kirim_data_to_wifi, (void *)2);
// Check Thread
if(rc1||rc2||rc3)
{
    cout<<"Error : Unable to create Thread!"<<endl;
    exit(-1);


*************************************************
* open serial device ttyAMA0
* baudrate : 115200bps
*************************************************/
if ((fd = serialOpen ("/dev/ttyAMA0", 115200)) < 0)

    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;

if (wiringPiSetup () == -1)

    fprintf (stdout, "Unable to start wiringPi: %s\n", strerror (errno)) ;
    return 1 ;


************************************************
* Set Nilai Set Point Robot terhadap dinding (cm)
* US1 = Sensor Ultrasonik Pojok Kiri
* US3 = Sensor Ultrasonik Pojok Kanan
************************************************/
et_US1 = 28;
et_US3 = 27;
ax_pwm_temp = max_pwm;
**
* Main Looping
*/
hile(1)

   if (r_ready)
   {
       /// Reset Buffer Degree of membership
       out_value_reset();

ntf("\n+ Pos X=%d  Area=%2.2f jarak=%2.2f",pos,obs_area,jarak_us2_f);
       /*************** FUZZY CONTROLLER ***************/
       switch(mode_following_ui)
       {
```

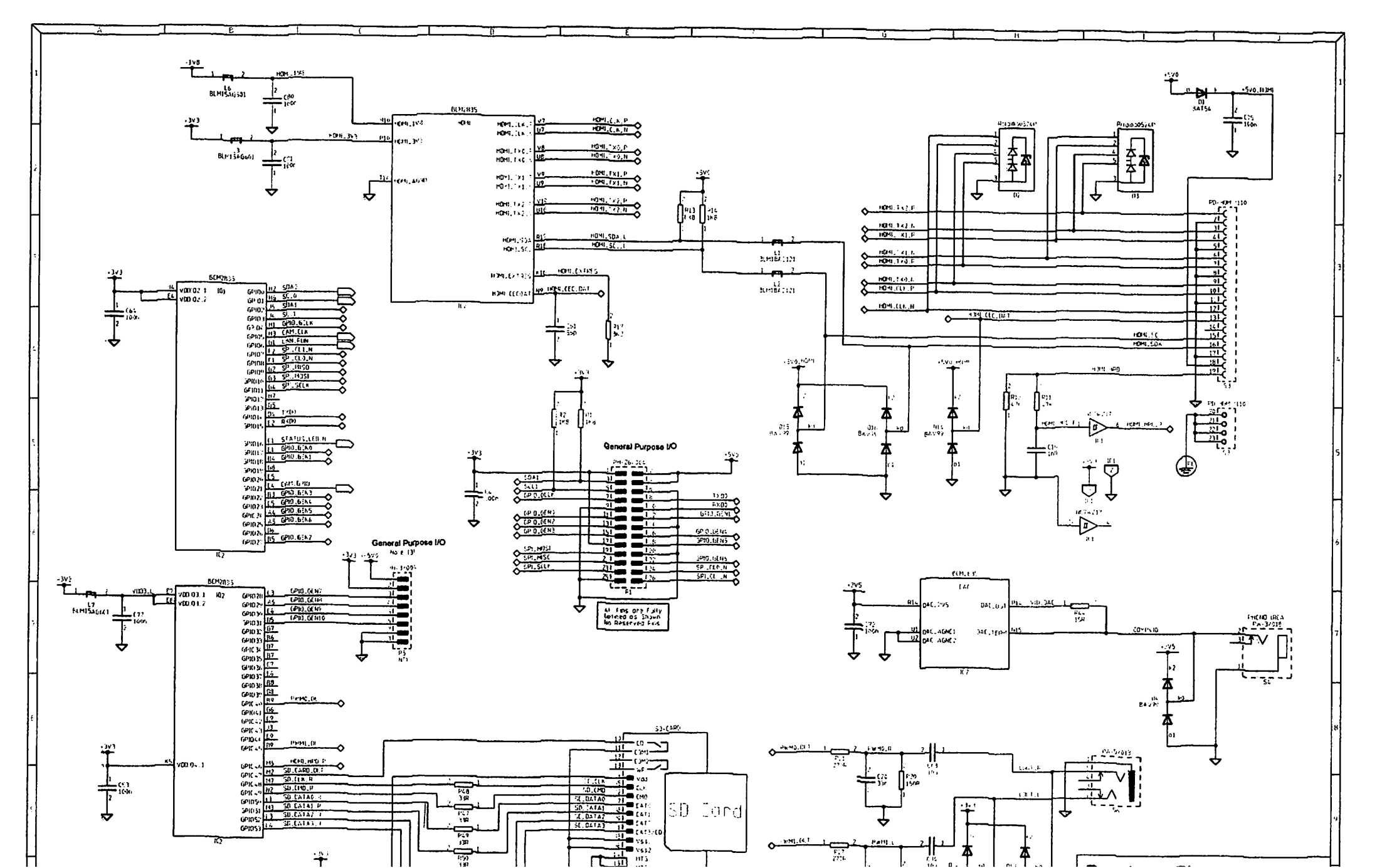
```
    case scan_kiri :
        err_US1 = (float)(set_US1 - jarak_us1_f);
        derr_US1 = err_US1 - (prev_err_US1);
        prev_err_US1 = err_US1;
        break;
    case scan_kanan :
        err_US3 = (float)(set_US3 - jarak_us3_f);
        derr_US3 = err_US3 - (prev_err_US3);
        prev_err_US3 = err_US3;
        break;
    default :
        break;
    }

    /*****************************************************************************
    * Proses Fuzzyfikasi
    **/
    fuzzyfikasi_wall_follow(mode_following_ui);
    fuzzyfikasi_obstacle_avoidance();

    /**
    ** Proses Rule Evaluation
    **/
    evaluasi_rule_wall_follow();
    evaluasi_rule_obstacle_avoidance(mode_following_ui);

    n += 1;
    if (n > 5)
    {
        n = 0;
        switch(mode_following_ui)
        {
        case scan_kiri :

tf(data_buff,"%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;
;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f",
                        err_US1,derr_US1,jarak_us2_f,pos,
                        eNB,eNM,eNK,eZ,ePK,ePM,ePB,deNB,deNM,deNK,deZ,dePK,dePM,dePB,
                        obs_jauh,obs_dekat,obs_sangat_dekat,
                        obs_kiri,obs_tengah,obs_kanan);
            break;
        case scan_kanan :

tf(data_buff,"%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;
;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f;%2.2f",
                        err_US3,derr_US3,jarak_us2_f,pos,
                        eND,eNM,eNK,eZ,ePK,ePM,ePB,deND,deNM,deNK,deZ,dePK,dePM,dePB,
                        obs_jauh,obs_dekat,obs_sangat_dekat,
                        obs_kiri,obs_tengah,obs_kanan);
            break;
        default :
            break;
        }
        write( client_sock, data_buff, strlen(data_buff));
    }

    /**
    ** Proses Defuzzifikasi dengan Metode COG (Center of Gravity) / COA (Center Of Area)
    **/
    y_pwm = defuzzifikasi_cog_methode(OUT_RULE_WALL_FOLLOW, WALL_FOLLOW,&zero_status);
    if (zero_status)
    {
        fuzzy_output_pwm = y_pwm;
        yl_pwm = y_pwm;
    }
    else
    {
        fuzzy_output_pwm = yl_pwm;
    }

    ////////////////////////////////////////////////////////////////////////
```

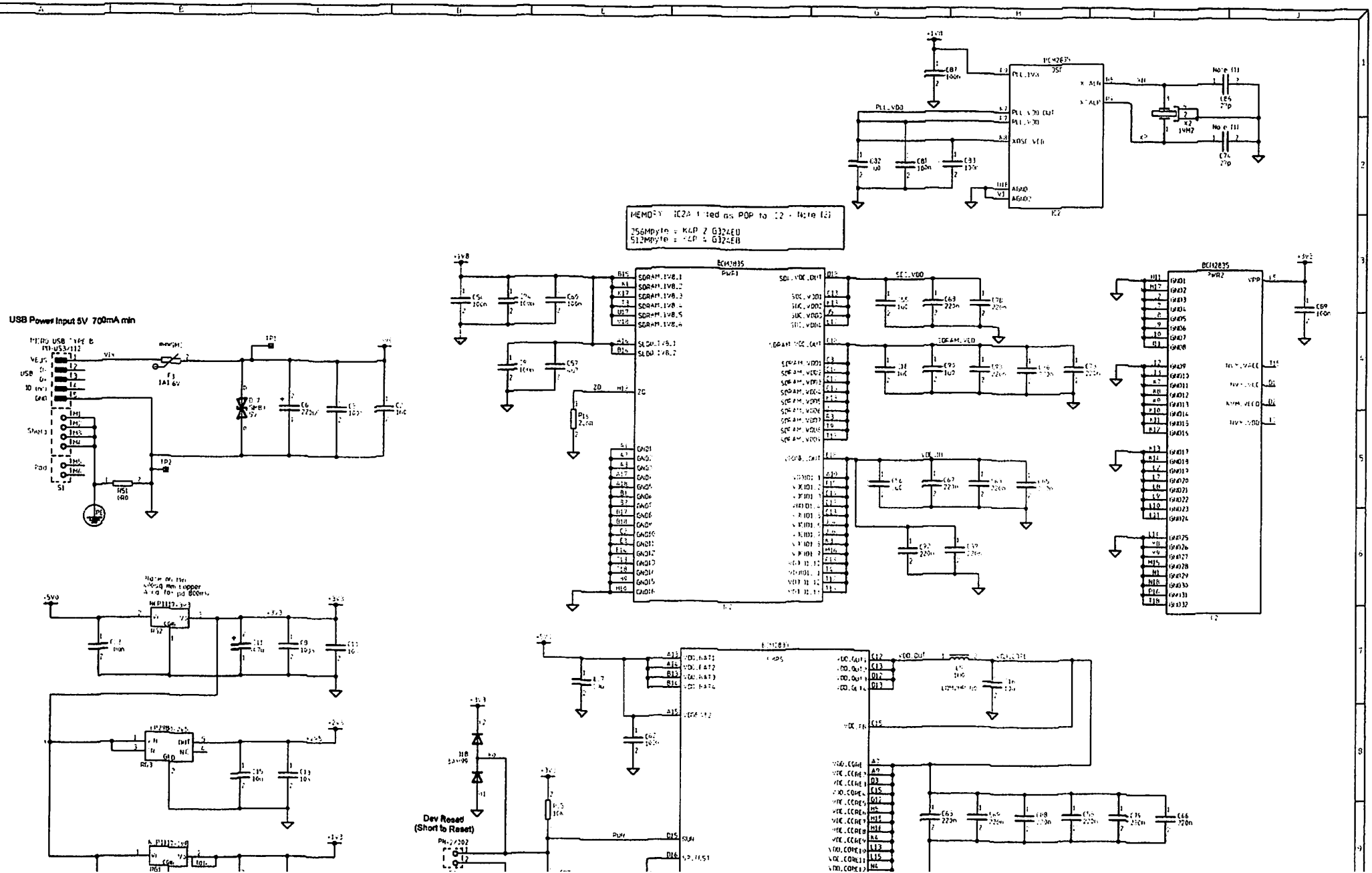
```
    y_pwm_kiri = defuzzifikasi_cog_methode(OUT_RULE_PWM_KI,OBSTACLE_AVOID,&zero_status);
    if (zero_status)
    {
        fuzzy_output_pwm_kiri = y_pwm_kiri;
        y1_pwm_kiri = y_pwm_kiri;
    }
    else
    {
        fuzzy_output_pwm_kiri = y1_pwm_kiri;
    }

    ////////////////////////////////////////////////////////////////////////
    y_pwm_kanan = defuzzifikasi_cog_methode(OUT_RULE_PWM_KA,OBSTACLE_AVOID,&zero_status);
    if (zero_status)
    {
        fuzzy_output_pwm_kanan = y_pwm_kanan;
        y1_pwm_kanan = y_pwm_kanan;
    }
    else
    {
        fuzzy_output_pwm_kanan = y1_pwm_kanan;
    }


*****************************************************************************/

    ///printf("\n+ Pos X=%d  Jarak= %2.2f  pwm ka =%
    /*****************************************************
    ** Olah data Output Fuzzy
    ** Ketentuan :
    ** Jika Terdeteksi Halangan atau terdapat penghalang
    ** maka kontroler fuzzy untuk wall follow akan dihandle
    ** oleh kontroler fuzzy untuk obstacle avoid
    *****************************************************/
    if (jarak_us2_f > 20 && jarak_us2_f < 40 && pos!=0 && pos!=320)
    {
        pwm_kanan = fuzzy_output_pwm_kanan;
        pwm_kiri = fuzzy_output_pwm_kiri;

        printf("\n+       pos       x=%d              jarak=%2.2f              pwm_kanan=%d
iri=%d",pos,jarak_us2_f,pwm_kanan,pwm_kiri);

        /// Limit PWM value
        if(pwm_kanan>max_pwm) pwm_kanan = max_pwm;
        if(pwm_kanan<min_pwm) pwm_kanan = min_pwm;
        if(pwm_kiri>max_pwm) pwm_kiri = max_pwm;
        if(pwm_kiri<min_pwm) pwm_kiri = min_pwm;
        serialPrintf(fd,"%d;%d;%d",maju,pwm_kanan,pwm_kiri);
        serialPutchar(fd,13);
    }
    else if (jarak_us2_f <= 20)
    {
        //fflush(stdout);
        switch(mode_following_ui)
        {
        case scan_kiri :
            serialPrintf(fd,"%d;%d;%d",belok_kanan,80,80);
            break;
        case scan_kanan :
            serialPrintf(fd,"%d;%d;%d",belok_kiri,80,80);
            break;
        default :
            break;
        }
        serialPutchar(fd,13);
    }
    else
    {

        switch (mode_following_ui)
        {
```

```
        case scan_kiri :
            pwm_kanan = set_point_pwm - (int)(fuzzy_output_pwm);
            pwm_kiri = set_point_pwm + (int)(fuzzy_output_pwm);
            break;
        case scan_kanan :
            pwm_kanan = set_point_pwm + (int)(fuzzy_output_pwm);
            pwm_kiri = set_point_pwm - (int)(fuzzy_output_pwm);
            break;
        default :
            break;
        }
        if (pos!=0 && obs_area > 2000)
        {
            max_pwm = 120;
        }
        else max_pwm = max_pwm_temp;
        /// Limit PWM value
        if(pwm_kanan>max_pwm) pwm_kanan = max_pwm;
        if(pwm_kanan<18) pwm_kanan = min_pwm;
        if(pwm_kiri>max_pwm) pwm_kiri = max_pwm;
        if(pwm_kiri<18) pwm_kiri = min_pwm;

        /**
        ** Kirim Data ke Kontroler Slave MEGA162
        ** Data-1 : Direction
        ** Data-2 : pwm Kanan
        ** Data-3 : pwm Kiri
        **/
        serialPrintf(fd,"%d;%d;%d",maju,pwm_kanan,pwm_kiri);
        serialPutchar(fd,13);
    }

    /// Reset Flag Ready
    r_ready = 0;
    /// delay
    //usleep(1000);
  }
}

pthread_exit(NULL);
return 0;
```


STATUS OK "ACT" LED
POWER ON "PWR" LED


10/100
Ethernet
FULL "FDX"
DUPLEX
LINK "LNK"
100M "100"


BCM2835
General Purpose I/O
SD-CARD
SD Card


USB Power Input 5V 700mA min
MEMORY IC2A fitted as POP to IC2 - Note (2)
256Mbyte = K4P 2 G324EU
512Mbyte = K4P 4 G324EB
Dev Reset
(Short to Reset)

L298

DUAL FULL-BRIDGE DRIVER

PERATING SUPPLY VOLTAGE UP TO 46 V
OTAL DC CURRENT UP TO 4 A
OW SATURATION VOLTAGE
VERTEMPERATURE PROTECTION
OGICAL "0" INPUT VOLTAGE UP TO 1.5 V
IGH NOISE IMMUNITY)

Multiwatt15

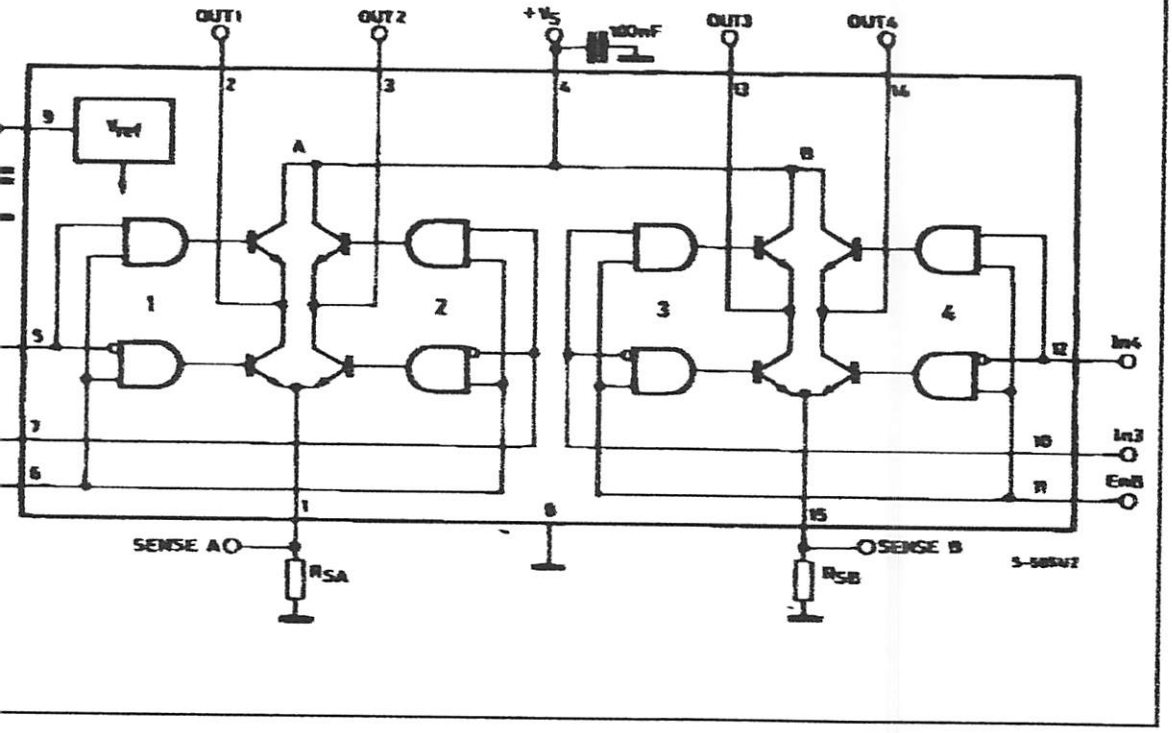
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

CRIPTION

L298 is an integrated monolithic circuit in a 15-Multiwatt and PowerSO20 packages. It is a voltage, high current dual full-bridge driver de-ed to accept standard TTL logic levels and drive ctive loads such as relays, solenoids, DC and ing motors. Two enable inputs are provided to le or disable the device independentlyof the in-gnals. The emitters of the lower transistors of bridge are connected together and the corre-ding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

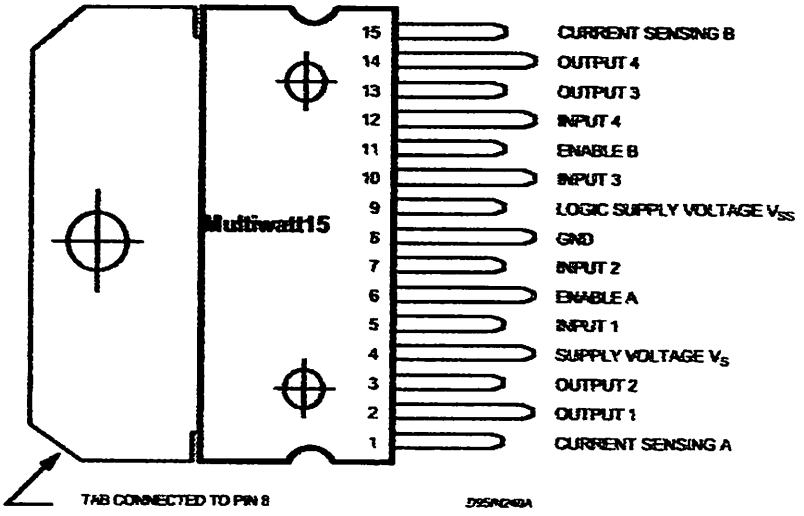
CK DIAGRAM

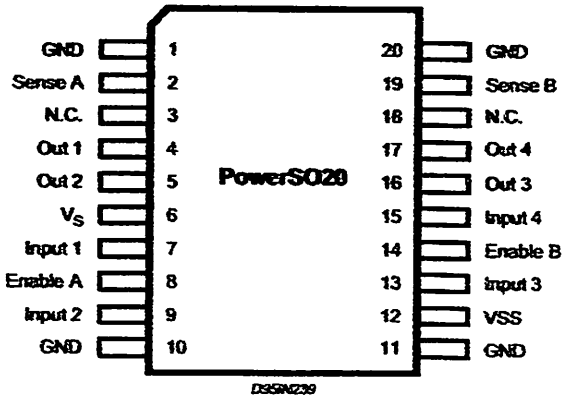


OLUTE MAXIMUM RATINGS

mbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

CONNECTIONS (top view)





MAL DATA

nbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\,j\text{-case}}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\,j\text{-amb}}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

ted on aluminum substrate

FUNCTIONS (refer to the block diagram)

N.15	PowerSO	Name	Function
;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
); 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
); 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

CTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

bol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
s	Supply Voltage (pin 4)	Operative Condition		V_{IH} +2.5		46	V
s	Logic Supply Voltage (pin 9)			4.5	5	7	V
	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0	V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L	V_i = X			4	mA
s	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0	V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L	V_i = X			6	mA
L	Input Low Voltage (pins 5, 7, 10, 12)			–0.3		1.5	V
H	Input High Voltage (pins 5, 7, 10, 12)			2.3		VSS	V
	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L				–10	µA
I	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V			30	100	µA
= L	Enable Low Voltage (pins 6, 11)			–0.3		1.5	V
= H	Enable High Voltage (pins 6, 11)			2.3		V_{SS}	V
L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L				–10	µA
H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V			30	100	µA
(H)	Source Saturation Voltage	I_L = 1A I_L = 2A		0.95	1.35 2	1.7 2.7	V V
(L)	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)		0.85	1.2 1.7	1.6 2.3	V V
sat	Total Drop	I_L = 1A (5) I_L = 2A (5)		1.80		3.2 4.9	V V
s	Sensing Voltage (pins 1, 15)			–1 (1)		2	V

CTRICAL CHARACTERISTICS (continued)

ibol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
(Vi)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
(Vi)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
(Vi)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
(Vi)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
(Vi)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
(Vi)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
(Vi)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
(Vi)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
Vi)	Commutation Frequency	I_L = 2A		25	40	KHz
Ven)	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
Ven)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
Ven)	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
Ven)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
Ven)	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
Ven)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
Ven)	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
Ven)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

nsing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
ig. 2.
ig. 4.
load must be a pure resistor.

e 1 : Typical Saturation Voltage vs. Output Current.

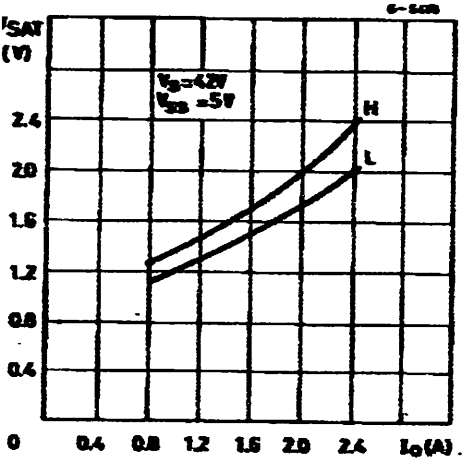
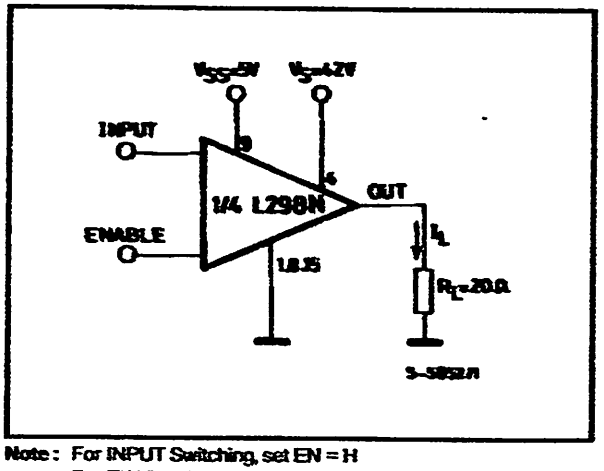
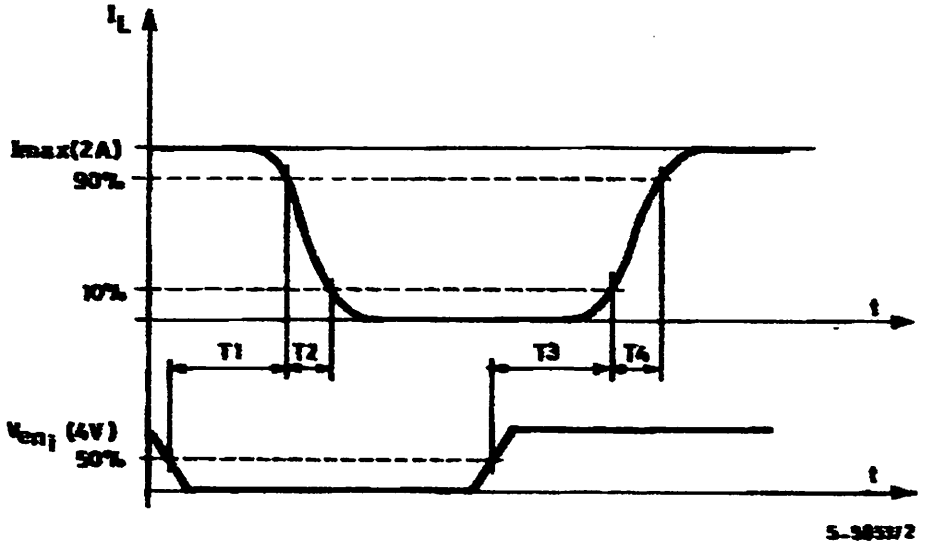


Figure 2 : Switching Times Test Circuits.

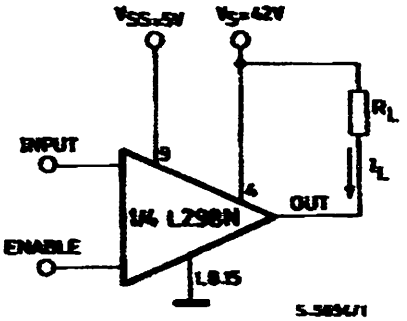


Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

re 3 : Source Current Delay Times vs. Input or Enable Switching.

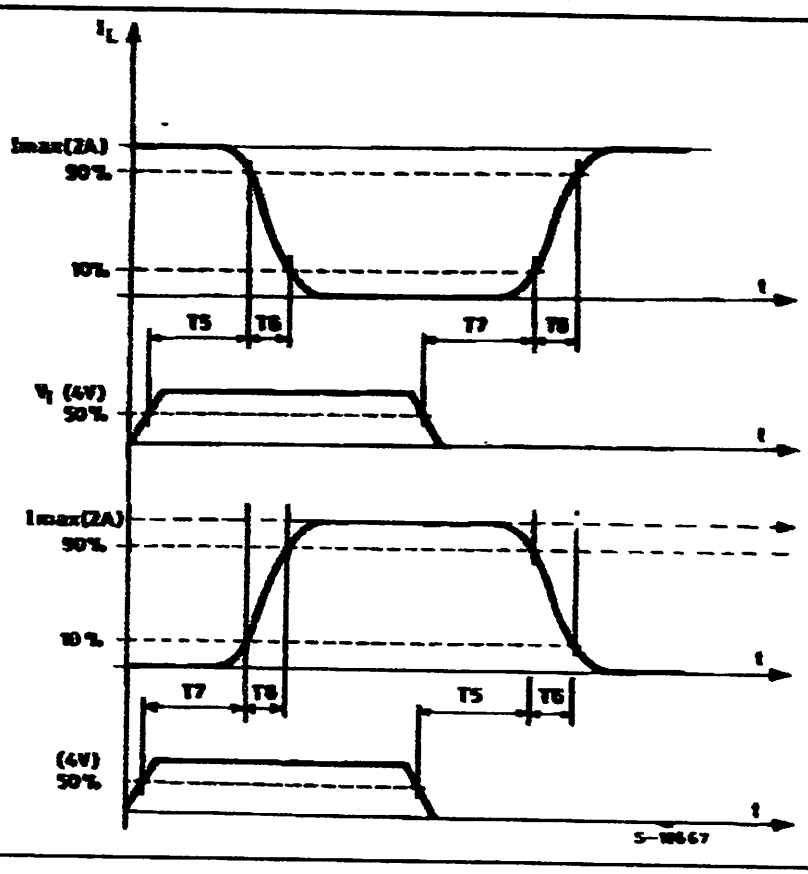


re 4 : Switching Times Test Circuits.

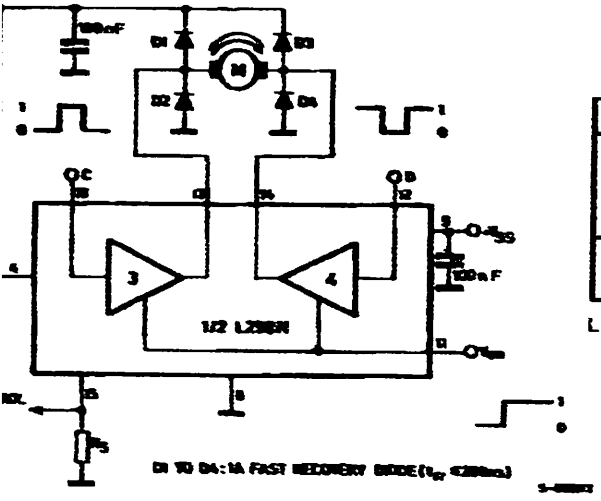


For INPUT Switching, set EN = H
For ENABLE Switching, set IN = L

re 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.



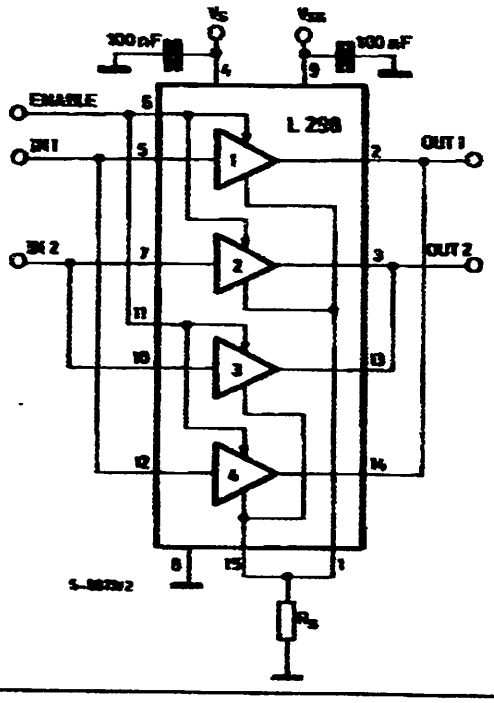
e 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

re 7 : For higher currents, outputs can be paralleled. Take care to parallel channel 1 with channel 4 and channel 2 with channel 3.



LICATION INFORMATION (Refer to the block diagram)

OWER OUTPUT STAGE

298 integrates two power output stages (A; B). ower output stage is a bridge configuration s outputs can drive an inductive load in com- r differenzial mode, depending on the state of uts. The current that flows through the load s out from the bridge at the sense output: an al resistor (R_{SA} ; R_{SB}.) allows to detect the in- / of this current.

IPUT STAGE

ridge is driven by means of four gates the in- which are In1 ; In2 ; EnA and In3 ; In4 ; EnB. inputs set the bridge state when The En input ; a low state of the En input inhibits the bridge. inputs are TTL compatible.

GGESTIONS

inductive capacitor, usually of 100 nF, must seen between both Vs and Vss, to ground, r as possible to GND pin. When the large ca- of the power supply is too far from the IC, a l smaller one must be foreseen near the

nse resistor, not of a wire wound type, must nded near the negative pole of Vs that must r the GND pin of the I.C.

Each input must be connected to the source of the driving signals by means of a very short path.

Turn-On and Turn-Off : Before to Turn-ON the Supply Voltage and before to Turn it OFF, the Enable input must be driven to the Low state.

3. APPLICATIONS

Fig 6 shows a bidirectional DC motor control Schematic Diagram for which only one bridge is needed. The external bridge of diodes D1 to D4 is made by four fast recovery elements ($t_{rr} \leq 200$ nsec) that must be chosen of a VF as low as possible at the worst case of the load current.

The sense output voltage can be used to control the current amplitude by chopping the inputs, or to provide overcurrent protection by switching low the enable input.

The brake function (Fast motor stop) requires that the Absolute Maximum Rating of 2 Amps must never be overcome.

When the repetitive peak current needed from the load is higher than 2 Amps, a paralleled configuration can be chosen (See Fig.7).

An external bridge of diodes are required when inductive loads are driven and when the inputs of the IC are chopped; Shottky diodes would be preferred.

solution can drive until 3 Amps In DC operation until 3.5 Amps of a repetitive peak current.

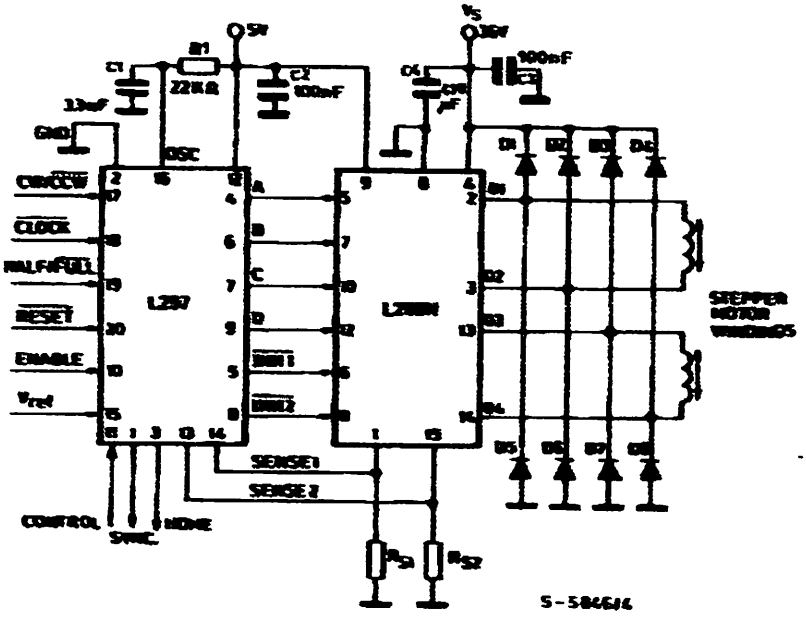
ig 8 it is shown the driving of a two phase bipolar per motor ; the needed signals to drive the in- of the L298 are generated, in this example, the IC L297.

shows an example of P.C.B. designed for the cation of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

re 8 : Two Phase Bipolar Stepper Motor Circuit.

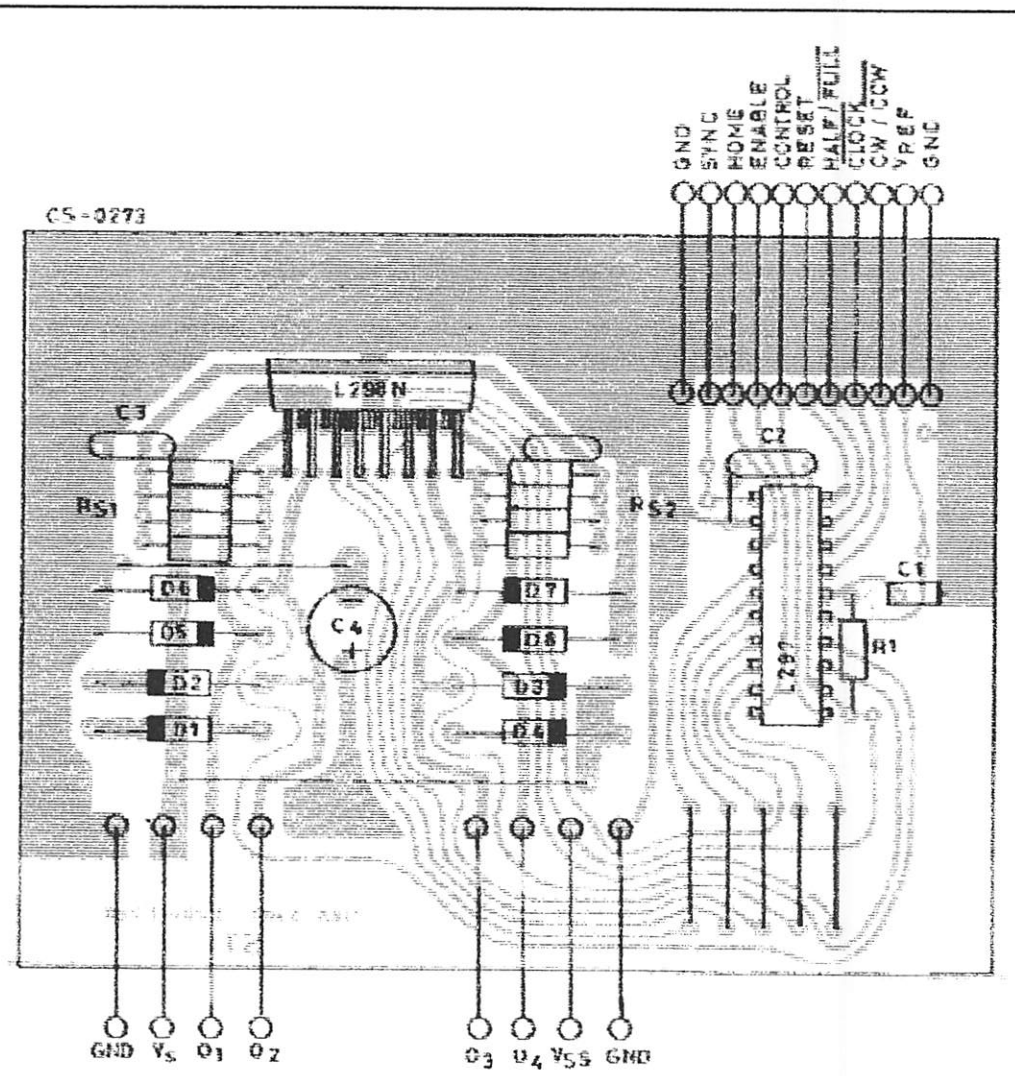
circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



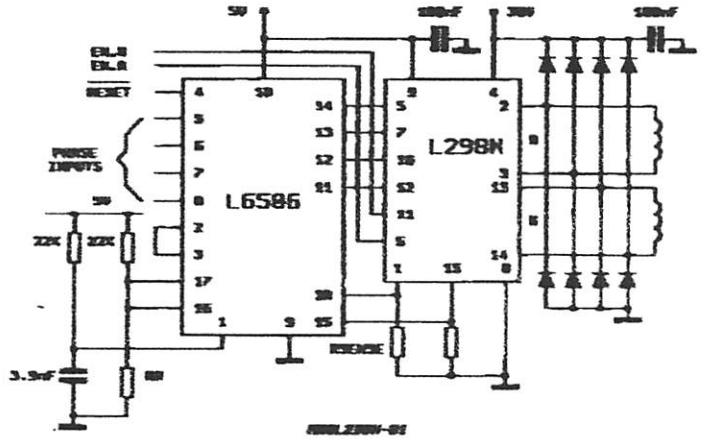
$= R_{S2} = 0.5\,\Omega$

to D8 = 2 A Fast diodes { $V_F \leq 1.2\,V$ @ $I = 2\,A$; $t_{rr} \leq 200\,ns$

re 9 : Suggested Printed Circuit Board Layout for the Circuit of fig. 8 (1:1 scale).



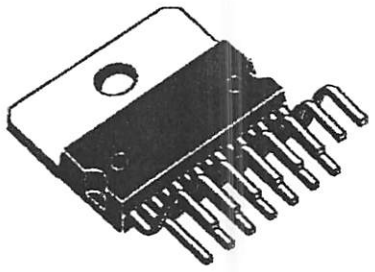
re 10 : Two Phase Bipolar Stepper Motor Control Circuit by Using the Current Controller L6506.



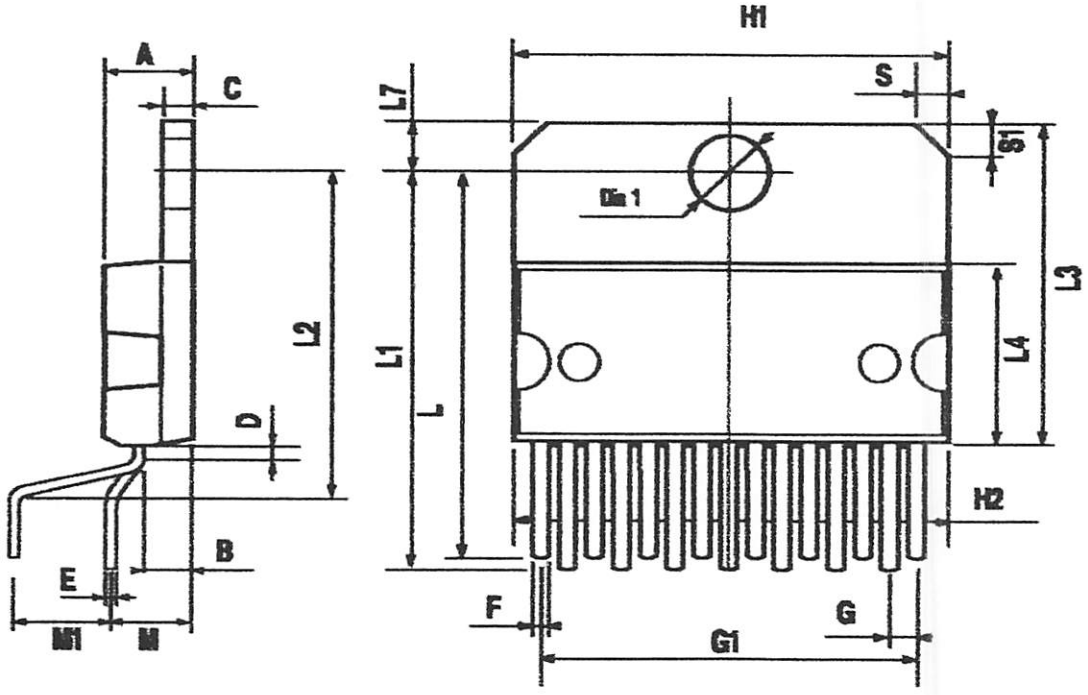
R_R and R_{sense} depend from the load current

	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
			5			0.197
			2.65			0.104
			1.6			0.063
		1			0.039	
	0.49		0.55	0.019		0.022
	0.66		0.75	0.026		0.030
	1.02	1.27	1.52	0.040	0.050	0.060
	17.53	17.78	18.03	0.690	0.700	0.710
	19.6			0.772		
			20.2			0.795
	21.9	22.2	22.5	0.862	0.874	0.886
	21.7	22.1	22.5	0.854	0.870	0.886
	17.65		18.1	0.695		0.713
	17.25	17.5	17.75	0.679	0.689	0.699
	10.3	10.7	10.9	0.406	0.421	0.429
	2.65		2.9	0.104		0.114
	4.25	4.55	4.85	0.167	0.179	0.191
	4.63	5.08	5.53	0.182	0.200	0.218
	1.9		2.6	0.075		0.102
	1.9		2.6	0.075		0.102
	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA

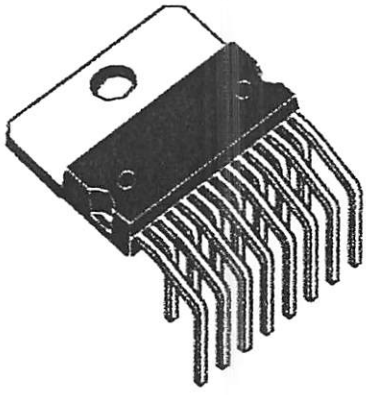


Multiwatt15 V

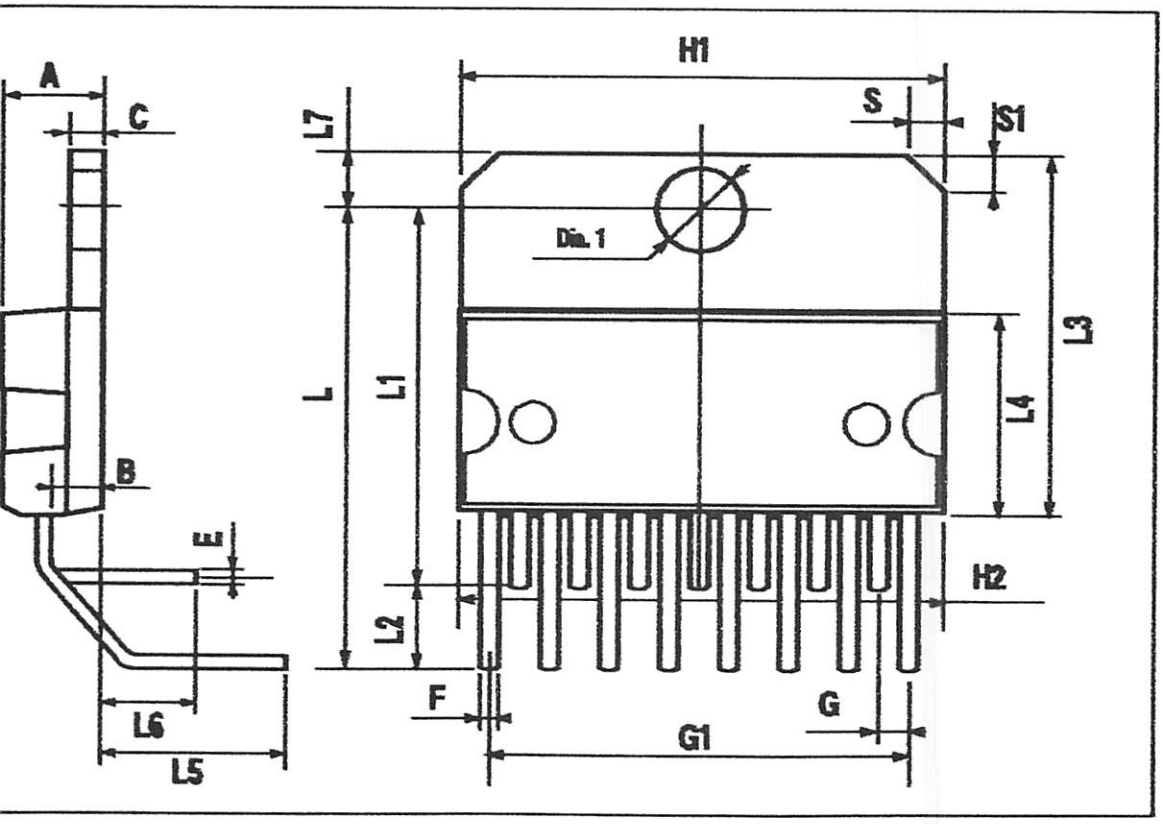


	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
			5			0.197
			2.65			0.104
			1.6			0.063
	0.49		0.55	0.019		0.022
	0.66		0.75	0.026		0.030
	1.14	1.27	1.4	0.045	0.050	0.055
	17.57	17.78	17.91	0.692	0.700	0.705
	19.6			0.772		
			20.2			0.795
		20.57			0.810	
		18.03			0.710	
		2.54			0.100	
	17.25	17.5	17.75	0.679	0.689	0.699
	10.3	10.7	10.9	0.406	0.421	0.429
		5.28			0.208	
		2.38			0.094	
	2.65		2.9	0.104		0.114
	1.9		2.6	0.075		0.102
	1.9		2.6	0.075		0.102
	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



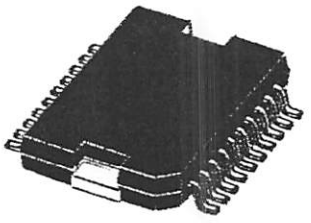
Multiwatt15 H



	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
			3.6			0.142
	0.1		0.3	0.004		0.012
			3.3			0.130
	0		0.1	0.000		0.004
	0.4		0.53	0.016		0.021
	0.23		0.32	0.009		0.013
)	15.8		16	0.622		0.630
	9.4		9.8	0.370		0.386
	13.9		14.5	0.547		0.570
		1.27			0.050	
		11.43			0.450	
)	10.9		11.1	0.429		0.437
			2.9			0.114
	5.8		6.2	0.228		0.244
	0		0.1	0.000		0.004
	15.5		15.9	0.610		0.626
			1.1			0.043
	0.8		1.1	0.031		0.043
	10° (max.)					
	8° (max.)					
		10			0.394	

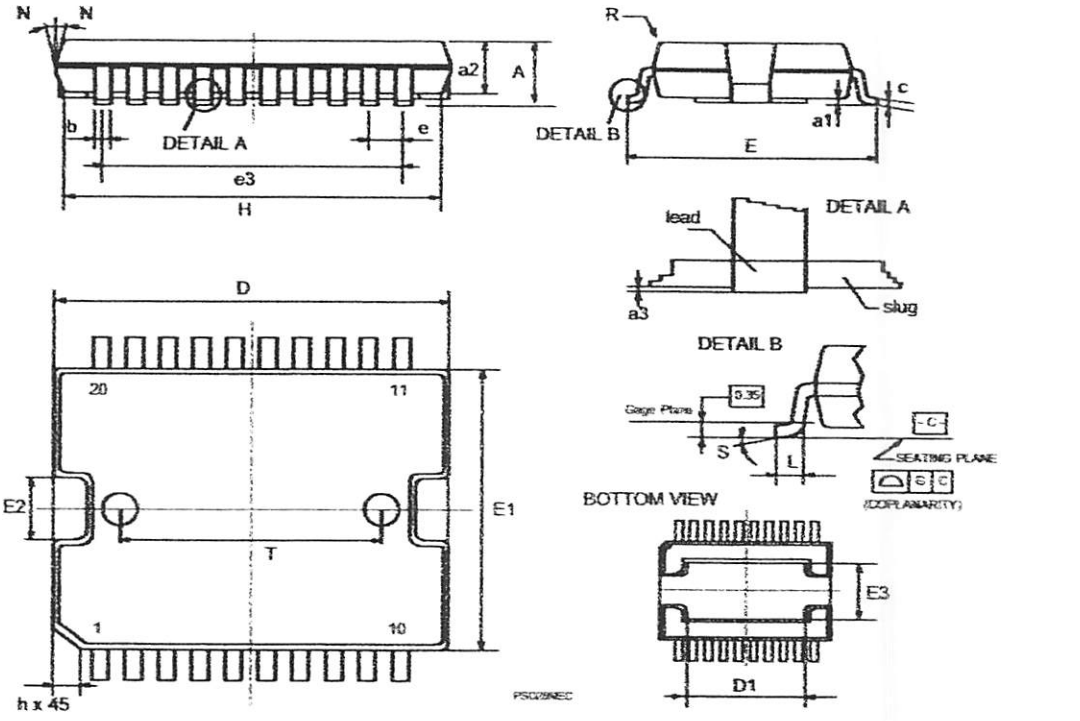
nd F" do not include mold flash or protrusions.
ash or protrusions shall not exceed 0.15 mm (0.006").
l dimensions: "E", "G" and "a3"

OUTLINE AND MECHANICAL DATA



JEDEC MO-166

PowerSO20



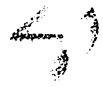
NRCHILD.

July 2014

N2222A
PN General-Purpose Amplifier

ıtures

iis device is for use as a medium power amplifier and
vitch requiring collector currents up to 500mA.



TO-92

E B C

lering Information

Part Number	Top Mark	Package	Packing Method
PN2222ABU	PN2222A	TO-92 3L	Bulk
PN2222ATA	PN2222A	TO-92 3L	Ammo
PN2222ATF	PN2222A	TO-92 3L	Tape and Reel
PN2222ATFR	PN2222A	TO-92 3L	Tape and Reel

;olute Maximum Ratings(1), (2)

ıses exceeding the absolute maximum ratings may damage the device. The device may not function or be operabove the recommended operating conditions and stressing the parts to these levels is not recommended. In addiextended exposure to stresses above the recommended operating conditions may affect device reliability. The lute maximum ratings are stress ratings only. Values are at T_A = 25°C unless otherwise noted.

ımbol	Parameter	Value	Unit
V_{CEO}	Collector-Emitter Voltage	40	V
V_{CBO}	Collector-Base Voltage	75	V
V_{EBO}	Emitter-Base Voltage	6.0	V
I_C	Collector Current	1.0	A
T_{STG}	Operating and Storage Junction Temperature Range	-55 to 150	°C

:

ese rating are based on a maximum junction temperature of 150 °C.
ese are steady-state limits. Fairchild Semiconductor should be consulted on applications involving pulsed or
v-duty-cycle operation.

ɪrmal Characteristics[3]

ɛs are at T_A = 25°C unless otherwise noted.

mbol	Parameter	Max.	Unit
P_D	Total Device Dissipation	625	mW
	Derate Above 25°C	5.0	mW/°C
$R_{\theta JC}$	Thermal Resistance, Junction to Case	83.3	°C/W
$R_{\theta JA}$	Thermal Resistance, Junction to Ambient	200	°C/W

:

CB size: FR-4, 76 mm x 114 mm x 1.57 mm (3.0 inch x 4.5 inch x 0.062 inch) with minimum land pattern size.

ctrical Characteristics

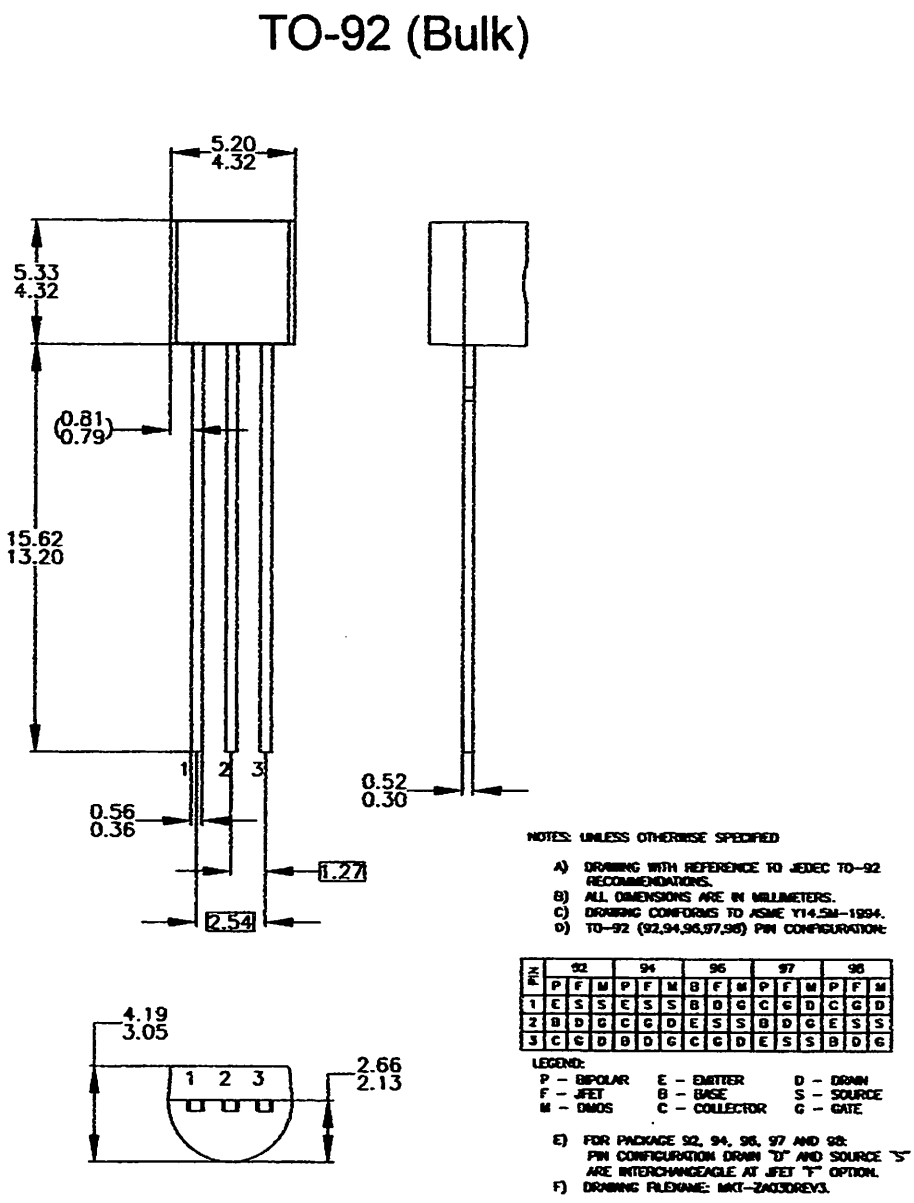
es are at T_A = 25°C unless otherwise noted.

/mbol	Parameter	Conditions	Min.	Max.	Unit
Characteristics					
$_{(BR)CEO}$	Collector-Emitter Breakdown Voltage[4]	I_C = 10 mA, I_B = 0	40		V
$_{(BR)CBO}$	Collector-Base Breakdown Voltage	I_C = 10 µA, I_E = 0	75		V
$_{(BR)EBO}$	Emitter-Base Breakdown Voltage	I_E = 10 µA, I_C = 0	6.0		V
I_{CEX}	Collector Cut-Off Current	V_{CE} = 60 V, $V_{EB(off)}$ = 3.0 V		10	nA
I_{CBO}	Collector Cut-Off Current	V_{CB} = 60 V, I_E = 0		0.01	µA
		V_{CB} = 60 V, I_E = 0, T_A = 125°C		10	
I_{EBO}	Emitter Cut-Off Current	V_{EB} = 3.0 V, I_C = 0		10	nA
I_{BL}	Base Cut-Off Current	V_{CE} = 60 V, $V_{EB(off)}$ = 3.0 V		20	nA
Characteristics					
h_{FE}	DC Current Gain	I_C = 0.1 mA, V_{CE} = 10 V	35		
		I_C = 1.0 mA, V_{CE} = 10 V	50		
		I_C = 10 mA, V_{CE} = 10 V	75		
		I_C = 10 mA, V_{CE} = 10 V, T_A = -55°C	35		
		I_C = 150 mA, V_{CE} = 10 V[4]	100	300	
		I_C = 150 mA, V_{CE} = 1 V[4]	50		
		I_C = 500 mA, V_{CE} = 10 V[4]	40		
$_{CE(sat)}$	Collector-Emitter Saturation Voltage[4]	I_C = 150 mA, I_B = 15 mA		0.3	V
		I_C = 500 mA, I_B = 50 mA		1.0	
$_{BE(sat)}$	Base-Emitter Saturation Voltage[4]	I_C = 150 mA, I_B = 15 mA	0.6	1.2	V
		I_C = 500 mA, I_B = 50 mA		2.0	
ıll Signal Characteristics					
f_T	Current Gain Bandwidth Product	I_C = 20 mA, V_{CE} = 20 V, f = 100 MHz	300		MHz
C_{obo}	Output Capacitance	V_{CB} = 10 V, I_E = 0, f = 1 MHz		8.0	pF
C_{ibo}	Input Capacitance	V_{EB} = 0.5 V, I_C = 0, f = 1 MHz		25	pF
$b'C_c$	Collector Base Time Constant	I_C = 20 mA, V_{CB} = 20 V, f = 31.8 MHz		150	pS
NF	Noise Figure	I_C = 100 µA, V_{CE} = 10 V, R_S = 1.0 kΩ, f = 1.0 kHz		4.0	dB
$e(h_{ie})$	Real Part of Common-Emitter High Frequency Input Impedance	I_C = 20 mA, V_{CE} = 20 V, f = 300 MHz		60	Ω
tching Characteristics					
t_d	Delay Time	V_{CC} = 30 V, $V_{EB(off)}$ = 0.5 V, I_C = 150 mA, I_{B1} = 15 mA		10	ns
t_r	Rise Time			25	ns
t_s	Storage Time	V_{CC} = 30 V, I_C = 150 mA, I_{B1} = I_{B2} = 15 mA		225	ns
t_f	Fall Time			60	ns

:

ılse test: pulse width ≤ 300 µs, duty cycle ≤ 2.0%.

ysical Dimensions



gure 1. 3-LEAD, TO92, JEDEC TO-92 COMPLIANT STRAIGHT LEAD CONFIGURATION (OLD TO92AM3)

ge drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner it notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the ty therein, which covers Fairchild products.

s visit Fairchild Semiconductor's online packaging area for the most recent package drawings:
ww.fairchildsemi.com/dwg/ZA/ZA03D.pdf.

ment tape and reel specifications, visit Fairchild Semiconductor's online packaging area:
ww.fairchildsemi.com/packing_dwg/PKG-ZA03D_BK.pdf.

Build Variant	Build Description

Note No	Description

Number	Details
(1)	Capacitor values tbc to crystal specification. Refer to production site specific instructions and BOM for additional information
(2)	Use flux gel or solder paste for FBP attachment process
(3)	Connector designed to be mounted underside. DO NOT FILL holes in process

Notes: A PCB assembly option ("Build Option") is formed by summing the required variants of circuit sections.
Eg: A+B+3

PCB Assembly Options

List of assembly options (valid combinations of Build variants) for reference only

Build variants	Build Description
A	Components only fitted to Model A (Basic)
B	Components only fitted to Model B (Full System)

PCB Parts List Entry

Project Code	Issue
BD-RPI-00022/001	

FID2 FID60 FID1 FID61 FID3 FID62 FID4 FID63 FID5 FID64 FID6 FID65 FID7 FID66 FID8 FID67 FID9 FID68 FID10 FID69

FID11 FID70 FID12 FID71

Raspberry Pi	
Project Code: RPI00022	Title: Raspberry Pi - Revision 2.0 Build Options/PCB layout instructions
Client: n/a	Date: 18Oct12
Scale: n/a	Sheet 05 of 05
Drawn By: PBL	File Name: RPI00022
	Issue/PMR Level: 22/0277

ʼsical Dimensions (Continued)

TO-92 (Ammo, Tape and Reel)

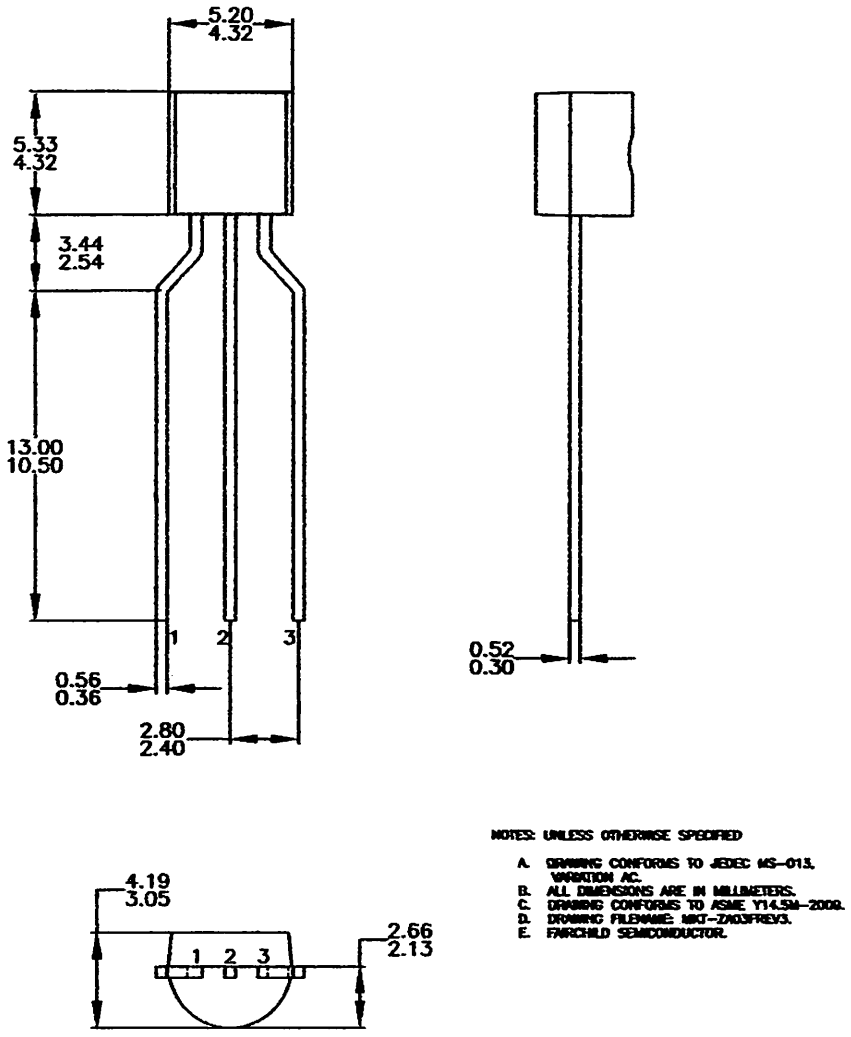


Figure 2. 3-LEAD, TO-92, MOLDED 0.200 IN LINE SPACING LEAD FORM (J61Z OPTION)

ıge drawings are provided as a service to customers considering Fairchild components. Drawings may change in any manner ıt notice. Please note the revision and/or date on the drawing and contact a Fairchild Semiconductor representative to verify or ı the most recent revision. Package specifications do not expand the terms of Fairchild's worldwide terms and conditions, specifically the ıty therein, which covers Fairchild products.

s visit Fairchild Semiconductor's online packaging area for the most recent package drawings:
www.fairchildsemi.com/dwg/ZA/ZA03F.pdf.

ırrent tape and reel specifications, visit Fairchild Semiconductor's online packaging area:
www.fairchildsemi.com/packing_dwg/PKG-ZA03F_BK.pdf.

IRCHILD.

EMARKS

lowing includes registered and unregistered trademarks and service marks, owned by Fairchild Semiconductor and/or its global subsidiaries, and is not ed to be an exhaustive list of all such trademarks.

'ower™
\P®
™
t Now™
'LUS™
'OWER™
SVOLT™

t Transfer Logic™
PEED®
Cool™
'ARK®
ntiMax™
™
)

ild®
ild Semiconductor®
Quiet Series™
)
)

Core™
ench™

F-PFS™
FRFET®
Global Power Resource℠
GreenBridge™
Green FPS™
Green FPS™ e-Series™
Gmax™
GTO™
IntelliMAX™
ISOPLANAR™
Making Small Speakers Sound Louder and Better™
MegaBuck™
MICROCOUPLER™
MicroFET™
MicroPak™
MicroPak2™
MillerDrive™
MotionMax™
mWSaver®
OptoHiT™
OPTOLOGIC®
OPTOPLANAR®

PowerTrench®
PowerXS™
Programmable Active Droop™
QFET®
QS™
Quiet Series™
RapidConfigure™
™
Saving our world, 1mW/W/kW at a time™
SignalWise™
SmartMax™
SMART START™
Solutions for Your Success™
SPM®
STEALTH™
SuperFET®
SuperSOT™-3
SuperSOT™-6
SuperSOT™-8
SupreMOS®
SyncFET™
Sync-Lock™

SYSTEM GENERAL®
TinyBoost®
TinyBuck®
TinyCalc™
TinyLogic®
TINYOPTO™
TinyPower™
TinyPWM™
TinyWire™
TranSiC™
TriFault Detect™
TRUECURRENT®*
μSerDes™
SerDes
UHC®
Ultra FRFET™
UniFET™
VCX™
VisualMax™
VoltagePlus™
XS™
仙童™

emarks of System General Corporation, used under license by Fairchild Semiconductor.

AIMER

HILD SEMICONDUCTOR RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN TO IMPROVE BILITY, FUNCTION, OR DESIGN. FAIRCHILD DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT RCUIT DESCRIBED HEREIN; NEITHER DOES IT CONVEY ANY LICENSE UNDER ITS PATENT RIGHTS, NOR THE RIGHTS OF OTHERS. THESE FICATIONS DO NOT EXPAND THE TERMS OF FAIRCHILD'S WORLDWIDE TERMS AND CONDITIONS, SPECIFICALLY THE WARRANTY THEREIN, H COVERS THESE PRODUCTS.

UPPORT POLICY

HILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE ESS WRITTEN APPROVAL OF FAIRCHILD SEMICONDUCTOR CORPORATION.

d herein:

ife support devices or systems are devices or systems which, (a) are ended for surgical implant into the body or (b) support or sustain e, and (c) whose failure to perform when properly used in coordance with instructions for use provided in the labeling, can be asonably expected to result in a significant injury of the user.

2. A critical component in any component of a life support, device, or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

COUNTERFEITING POLICY

d Semiconductor Corporation's Anti-Counterfeiting Policy. Fairchild's Anti-Counterfeiting Policy is also stated on our external website, www.fairchildsemi.com, Sales Support.

rfeiting of semiconductor parts is a growing problem in the industry. All manufacturers of semiconductor products are experiencing counterfeiting of their Customers who inadvertently purchase counterfeit parts experience many problems such as loss of brand reputation, substandard performance, failed tions, and increased cost of production and manufacturing delays. Fairchild is taking strong measures to protect ourselves and our customers from the ation of counterfeit parts. Fairchild strongly encourages customers to purchase Fairchild parts either directly from Fairchild or from Authorized Fairchild tors who are listed by country on our web page cited above. Products customers buy either from Fairchild directly or from Authorized Fairchild Distributors uine parts, have full traceability, meet Fairchild's quality standards for handling and storage and provide access to Fairchild's full range of up-to-date technical duct information. Fairchild and our Authorized Distributors will stand behind all warranties and will appropriately address any warranty issues that may arise. d will not provide any warranty coverage or other assistance for parts bought from Unauthorized Sources. Fairchild is committed to combat this global n and encourage our customers to do their part in stopping this practice by buying direct or from authorized distributors.

UCT STATUS DEFINITIONS

ion of Terms

heet Identification	Product Status	Definition
rance Information	Formative / In Design	Datasheet contains the design specifications for product development. Specifications may change in any manner without notice.
Preliminary	First Production	Datasheet contains preliminary data; supplementary data will be published at a later date. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve design.
entification Needed	Full Production	Datasheet contains final specifications. Fairchild Semiconductor reserves the right to make changes at any time without notice to improve the design.
Obsolete	*Not In Production*	Datasheet contains specifications on a product that is discontinued by Fairchild Semiconductor. The datasheet is for reference information only.

Rev. I68