PENGEMBANGAN *DRIVE HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

SKRIPSI

Disusun Oleh :

TITO DINTI KOSHARS
NIM : 11.12.211

PROGRAM STUDI TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2015

LEMBAR PERSETUJUAN

PENGEMBANGAN *DRIVE HEXAPOD ROBOT* UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

SKRIPSI

Disusun dan diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai gelar Sarjana Teknik

Disusun oleh :
TITO DINTI KOSHARS
NIM : 11.12.211

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
M. Ibrahim Ashari, ST, MT	Irmalia Suryani Faradisa, ST, MT
NIP. P. 1030100358	NIP. P. 1030000365

Mengetahui,
Ketua Program Studi Teknik Elektro S-1

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

PROGRAM STUDI TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2015

PENGEMBANGAN *DRIVE HEXAPOD ROBOT* UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

Tito Dinti Koshars, Nim 1112211
Dosen Pembimbing : M. Ibrahim Ashari, ST,MT dan
Irmalia Suryani Faradisa, ST,MT

Konsentrasi Teknik Elektronika, Jurusan Teknik Elektro S-1
Fakultas Teknologi Industri, Institut Teknologi Nasional Malang
Jl. Raya Karanglo Km.2 Malang
E-mail : titodintikoshars@gmail.com

ABSTRAK

Pada perancangan dan pembuatan robot *hexapod* sebelumya yang sudah memakai sistem *embedded* pada sistem kontrolnya masih terdapat kekurangan, hal tersebut disebabkan belum adanya metode yang digunakan untuk mengontrol pergerakan kaki robot dalam berjalan maka pergerakan robot masih kurang halus. Oleh karena itu dalam perancangan sistem ini diimplementasikan konsep kendali cerdas dengan menggunakan logika *fuzzy* yang ditanamkan pada sistem *embedded* Arduino Mega 2560 pada robot *hexapod.*

Dalam perancangan robot *hexapod* ini, digunakan kontroler Arduino Mega 2560 dan *Module* SSC 32 untuk mengendalikan delapan belas motor servo sebagai aktuator kaki robot *hexapod* ini. Dalam komunikasi data antara kontroler Arduino Mega 2560 dan Module SSC 32 berkomunikasi dengan komunikasi serial dengan *baudrate* 115200 bps. Pada robot ini terdapat juga tiga sensor jarak SRF HC04 yang digunakan sebagai parameter *input* logika *fuzzy* yang hasil *output* dari perhitungan *fuzzy* tersebut akan digunakan untuk menentukan aksi kendali dari robot *hexapod* tersebut.

Dari hasil beberapa pengujian metode logika *fuzzy* berhasil ditanam pada sistem Arduino Mega 2560, maka dari hasil tersebut telah terealisasi robot *hexapod* yang dapat bergerak otomatis pada suatu trajektori dengan presentase keberhasilan sebesar 84% dan dengan *Error* 16% dalam melakukan pergerakan yang *smooth* untuk menyusuri dinding dan menghindar halangan yang terdapat pada lintasan labirin.

Kata Kunci : *Arduino Mega2560, Logika Fuzzy, Robot Hexapod, Sensor SRF HC04*

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 "**PENGEMBANGAN *DRIVE HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK 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. Dr.Ir. Lalu Mulyadi, 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. Bapak Dr. Eng I Komang Somawirata, ST, MT selaku Sekretaris Jurusan Teknik Elektro S-1 Institut Teknologi Nasional Malang.
5. Irmalia Suryani Faradisa, ST, MT selaku Dosen Pembimbing Dua Skripsi.
6. 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, 3 Agustus 2015

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. Salah satu hal yang menjadi perhatian akhir-akhir ini adalah bagaimana membuat 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 beroda maupun aktuator yang disusun menyerupai kaki sehingga dapat bergerak dari suatu tempat ke tempat yang lain untuk melakukan tugas tertentu. *Hexapod* robot adalah salah satu robot berkaki yang banyak dikembangkan oleh para peneliti di dalam maupun di luar negeri karena salah satu keunggulanya terhadap medan yang di laluinya. Salah satu fokus dalam pembuat suatu robot hexapod ini adalah sistem navigasi. Perancangan navigasi suatu robot *mobile* memiliki beberapa acuan yang digunakan yaitu : Kontur (bentuk) dari lingkungan dan jaraknya terhadap suatu benda tersebut. Untuk itu dibutuhkan suatu sistem kendali cerdas yang dapat digunakan untuk mengetahi lingkungan yang akan delewati robot tersebut sesuai dengan *planing*.

Pada Perancangan dan Pembuatan Robot *Hexapod* (Daris Alfi, 2013) yang telah dibuat di laboratorium robotika ITN Malang sistem navigasi yang digunakan pergerakan robotnya masih kurang *smooth* saat berbelok atau pada saat menghindar dari halangan. Oleh karena itu pada pembuatan *hexapod* robot ini akan ditanamkan sistem kontrol pengendali robot dengan logika *fuzzy* yang nantinya nilai outputan gerak dari sistem ini lebih *smooth* pada saat berbelok atau pun pada saat menghindari halangan. Sistem ini mengadopsi dari sistem navigasi robot mobil dengan menggunakan metode kendali logika *fuzzy* (Cosmas Eric Septian, 2014) pada robot tersebut menggunakan metode logika *fuzzy* untuk robot beroda jadi output dari *fuzzy* yang dihasilkan hanya untuk mengendalikan dua

buah motor DC saja dan terbukti berhasil diterapkan, oleh karena itu pada pembuatan robot *hexapod* ini kita akan mengimplementasikan metode kendali logika *fuzzy* yang memiliki tiga buah sensor ultrasonik sebagai nilai input untuk parameter yang akan dipetakan dalam proses fuzzyfikasi dengan derajat keanggotaan (*grade of memberhip*) masing-masing. Maka selanjutnya data akan di evaluasi dengan *rule* untuk menentukan nilai output pada kontroler yang diubah ke dalam nilai pwm dan di outputkan ke 18 buah motor servo pada kaki *hexapod* robot ini yang berkedudukan sebagai aktuator.

Dari beberapa kondisi tersebut di atas, penulis ingin membuat suatu mobile robot yang dapat melakukan navigasi (pergerakan) pada lingkungan yang tak terprediksi dengan metode *logika fuzzy*. Untuk itu pada perancangan sistem, akan dibuat kombinasi dari sistem sensor ultrasonic. Harapannya sistem ini dapat dikembangkan dan diterapkan pada sistem robot *hexapod* pada robot KRPAI ITN Malang untuk navigasi terhadap ruang labirin yang keadaanya sudah ditetapkan oleh panitia lomba jadi robot ini dapat bergerak dengan *flexibel* terhadap dinding labirin tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas, maka dapat disimpulkan permasalahan yang dituangkan dalam karya ilmiah ini, yaitu :

1. Bagaimana membuat suatu robot berkaki *hexapod* yang dapat melakukan navigasi pada suatu lingkungan yang tidak diketahui secara pasti pada bentuk labirin lintasanya.
2. Bagaimana menerapkan suatu kendali *fuzzy* kedalam mikrokontroler Atmega 2560(Arduino Mega 2560).

1.3 Tujuan

Pembuatan robot berkaki *hexapod* yang cerdas ini bertujuan untuk merancang suatu sistem kendali cerdas yang dapat menyusuri dinding arena dengan pergerakan *smooth* dan dapat juga menghindar dari halangan yang berupa halangan yang di tentukan pada rule KRPAI 2015 dengan mengenali parameter-parameter lingkungan untuk digunakan sebagai acuan dalam melakukan navigasi

secara otomatis. Sehingga nantinya sistem ini dapat dikembangkan untuk konsep robot cerdas masa depan dan dapat membantu untuk mempersiapkan KRPAI(Kontes Robot Pemadam Api Idonesia) berkaki ITN MALANG.

1.4 Batasan Masalah

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

1. *Hexapod* robot ini hanya digunakan di dalam arena yang sudah ditentukan dengan arena yang memiliki tinggi dinding 30 cm terbuat dari papan rata dan berlantai karpet.
2. Berat robot tidak lebih dari 7,7 kg/cm sesuai dengan nilai maksimum torsi motor servo.
3. Jarak sensor ultrasonic maksimal adalah 3 m.
4. Bentuk dimensi dan pergerakan robot saat bergerak ukuranya sesuai dengan rule KRPAI (Kontes Robot Pemadam Api Indonesia) berkaki p x l x t = 31 cm x 31 cm x 27 cm.
5. Memiliki halangan sesuai ketentuan rule KRPAI Berkaki 2015 yaitu berbentuk tabung dengan diameter 9 cm dan tinggi 30 cm berwarna merah.

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 ada hubunganya dengan permasalahan pada perancangan alat.
2. Perancangan Sistem

   Pembuatan design dan pencarian bahan untuk pembuatan mekanik serta pembuatan design rangakaian elektronika seperti : rangkaian *Push Button*, lampu indikator dan *wearing* sensor.
3. Pembuatan Hardware

   Pembuatan rangkaian *Push Button*, lampu indikator dan *wearing* sensor.

4. Pengujian Sistem

   Proses ujicoba rangkaian dan keseluran sistem untuk mengetahui adanya kesalahan agar sistem sesuai dengan konsep yang telah dirancang.
5. Pelaporan hasil pengujian dan kesimpulan.

1.6 Sistematika Penulisan

Untuk mempermudah dan memahami pembahasan penulisanskripsi 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 perancangan dan analisa: sensor SRF, arduino mega, *lyxnmotion visual squencer*, motor servo dan regulator 5volt.

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.

DAFTAR PUSTAKA

BAB II
DASAR TEORI

2.1 Pengertian Robot Hexapod

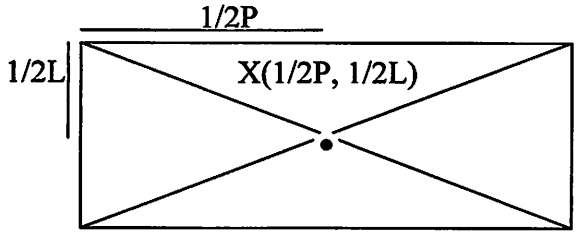
Hexapod robot adalah termasuk dalam robot berkaki yang mempunyai kaki berjumlah 6 tungkai, *hexapod* termasuk robot yang masuk kedalam kategori robot berkaki. Maka dari itu perencanaan pada robot *hexapod* yang menjadi tema penulis untuk perancangan robot *hexapod* dan pemilihan gait yang baik sangat berpengaruh untuk kondisi gerak robot, karena robot secara statis dapat stabil dengan menggunakam 3 kaki atau lebih. Maka dari itu robot *hexapod* memiliki fleksibelitas yang tinggi jika ada salah satu kaki yang tidak berfungsi maka masih ada kemungkinan robot dapat berjalan karena tidak terlalu dibutuhan untuk mencapai titik kestabilan karena kaki lainya masih memiliki kesempatan untuk mencari pijakan lainnya.

Robot *hexapod* memiliki dua bagian yaitu bagian tubuh serta bagian kaki dan memiliki tiga derajat kebebasan 3 (DOF) yang dibentuk pada tiga sendi kaki yaitu *coxa, femur*, dan *tibia*. Kemudian pada sendi kaki robot direprensentasikan dalam satu buah motor servo 180 derajat [1].

Dalam analisa robot terbagi menjadi 2 (dua), yaitu :

1. Analisa Pusat Masa

   Pusat masa adalah jika posisi dimana bentuk tubuh persegi panjang maka pusat masa berada di tengah.



Gambar 2.1. Titik Pusat Masa Pada Bidang Persegi Panjang [1]

Sangat pentingnya pusat masa yaitu untuk menentukan keseimbangan robot sehingga pada saat posisi berdiri diam ataupun pada saat

berjalan robot tidak terjatuh karena pusat masa dapat berubah sesuai gaya yang didapat dari pergerakan kaki robot sesuai dengan sudut pergerakan yang dibuat.

2. Analisa Pada Gaya Dan Sendi Pada Keseimbangan Robot

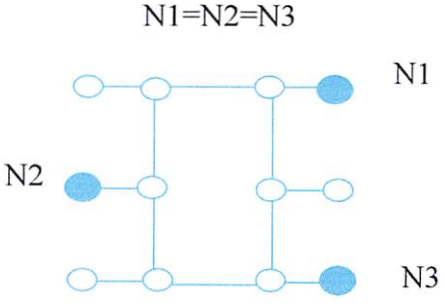
Analisa pada gaya dan sendi dibedakan menjadi 2 yaitu *dinamix stability* dan *static stability.*

- *Dynamix Stability*

  Robot yang berjalan stabil pada saat bergerak harus terus bergerak agar tidak jatuh. Jika tidak berjalan atau mendadak berhenti maka pusat masa akan membuat robot terjatuh.

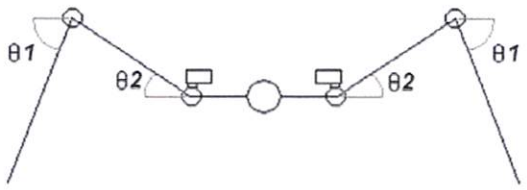
- *Static Stability*

  Robot yang dapat dinamakan *static stability* adalah robot yang dapat diberhentikan pergerakanya pada titik manapun pada gaitnya dan tidak akan jatuh. Pada robot *Hexapod* selama ada kaki manapun yang bersentuhan dengan lantai dan pusat massa ditempatkan pada segitiga yang terbentuk pada kaki-kaki robot. Maka robot akan berdiri stabil saat posisi statis.



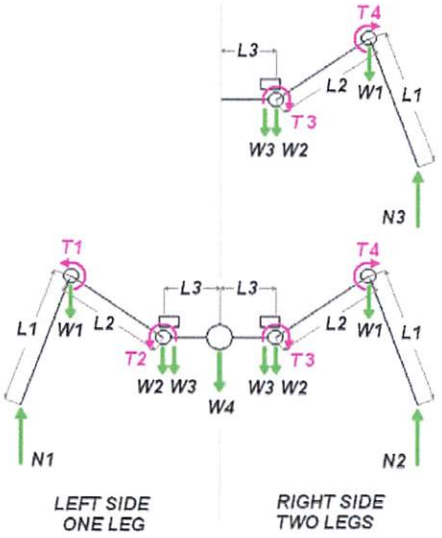
3 *Legs Raised*, 3 *Legs Supporting*

Gambar 2.2. Tripot Gait [1]



Gambar 2.3. Sudut Pada Kaki Hexapod [1]

Gaya yang terjadi pada setiap joint kaki Hexapod :



Gambar 2.4. Gaya Pada Sendi Robot[1]

Keterangan:

N1: Normal (Reaction) force

L1, 2, 3: Length of link

W1, 2, 3 : Weight of each actuator (W2, 3 are assumed tobe very close)

W4 : Weight acting at the center of mass

T1, 2, 3 : Torque acting at each joint (each side is defferent)

2.2 Logika Fuzzy

Istilah *Fuzzy Set* pertama kali diperkenalkan oleh Prof. Lutfi 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 [2].

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 [2].

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*.

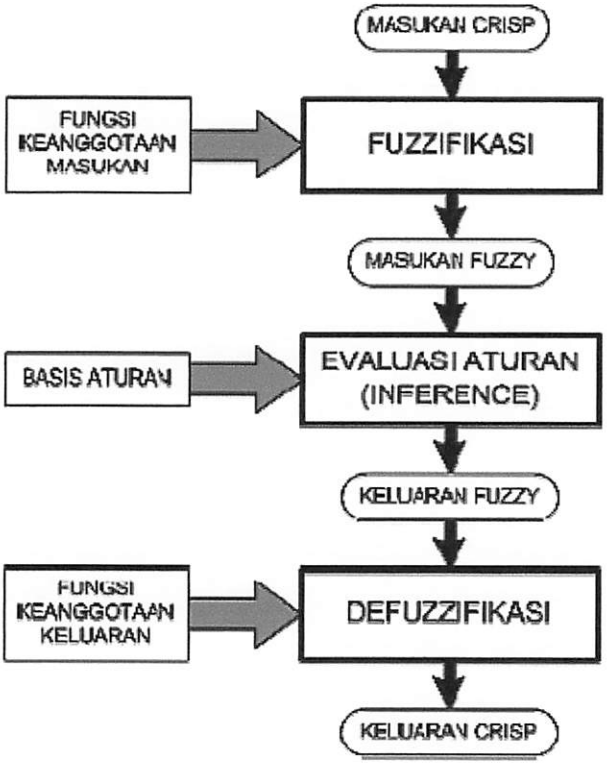
2.2.1 Struktur Dasar Pengendalian Logika *Fuzzy*

Metode-metode perancangan pengendalian klasik 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*



Gambar 2.5 Struktur Sistem Kendali *Fuzzy* [3]

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

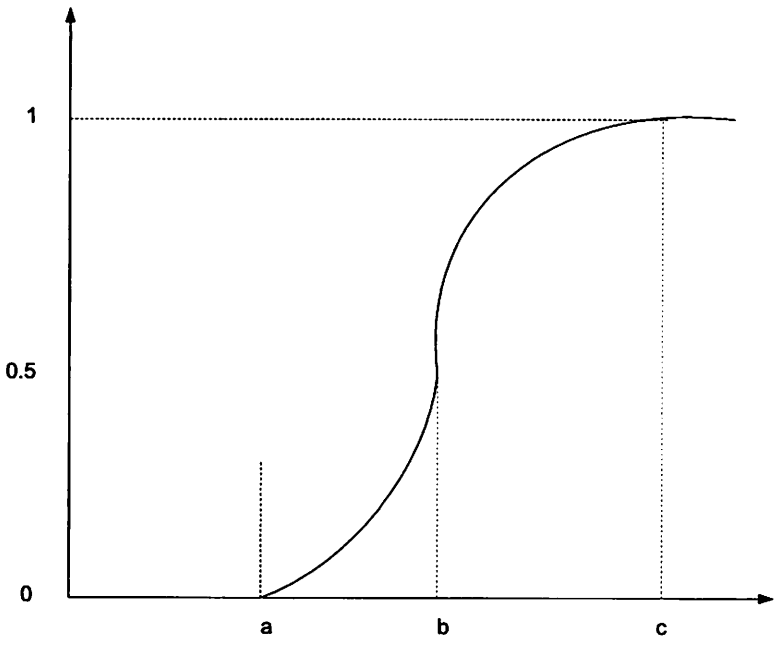
macam fungsi keanggotaan yang dinyatakan dalam fungsi keanggotaan S, π, dan T (Triangular).

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 :



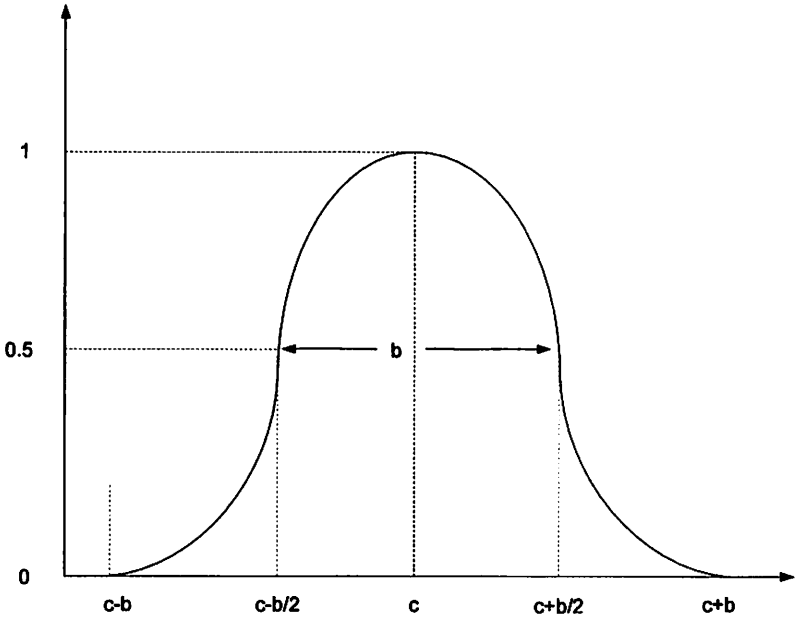
Grafik 2.1 S-function [2]

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 :



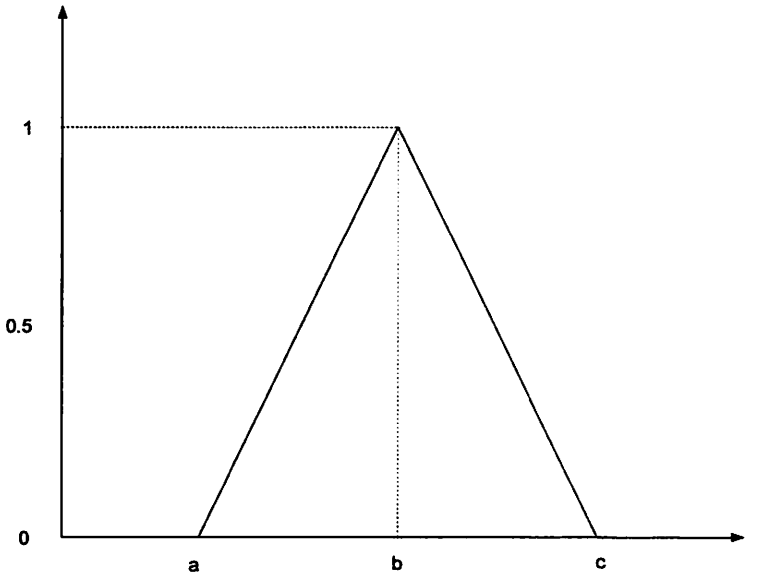
Grafik 2.2 *π-function* [2]

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 T-function [2]

2. Penyusunan Aturan Atau *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* [2].
Aturan Dasar *fuzzy* adalah dalam bentuk umum :

$R^{(l)}$: IF x_1 is F_1^1 AND .... AND.... X_n is F_n^1 , THEN y is G^1

Dimana F_1^1 dan G^1 adalah himpunan *fuzzy* masing-masing di $U_i C$ 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)

$A_{1,\ldots,}$ A_n adalah *antecedent* yaitu masukan yang defuzzifikasi, sedangkan $B_{1,\ldots}B_n$ 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 [2]. 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 :

$$V_o = \sum_{i=1}^{j} \frac{v_i}{J} \quad \text{..... (2.4)}$$

$$v_j = \max_{v \in V} \mu_v(v) \quad \text{..... (2.5)}$$

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

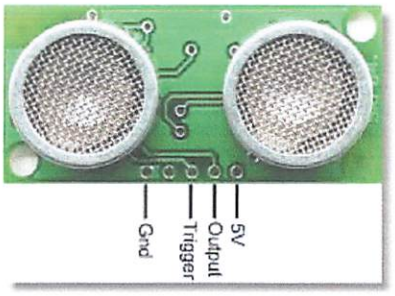
m = tingkat kuantisasi

v_k = elemen ke-k

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

V = semesta pembicaraan (*Universe of Discourse*) [2]

2.3 Sensor Ultrasonik



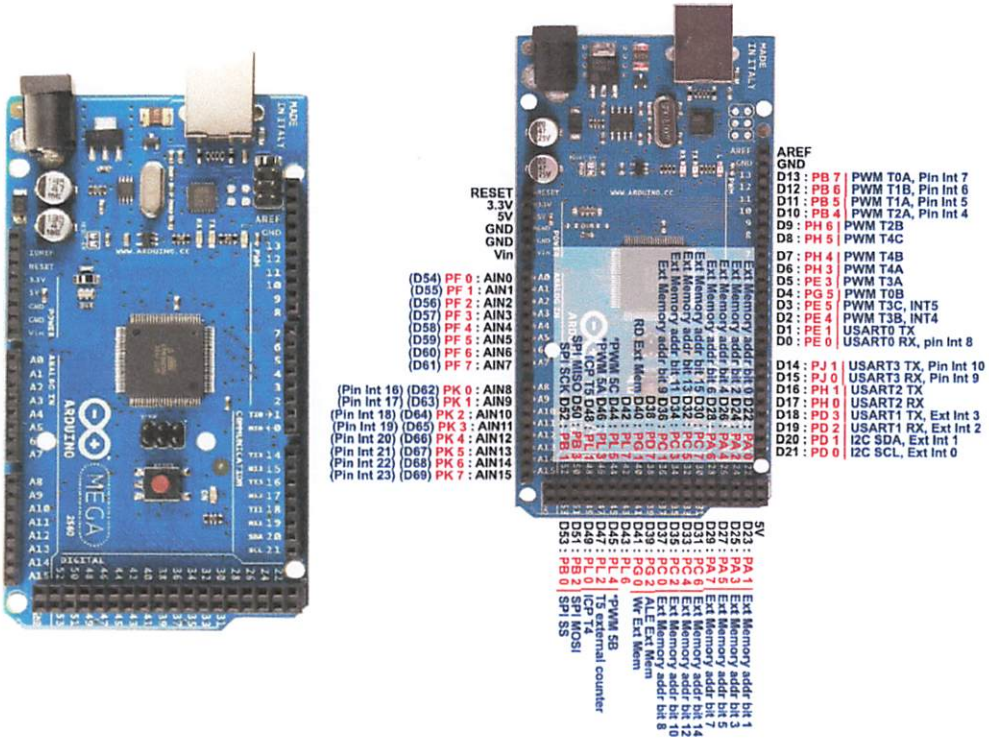
Gambar 2.6 Modul Sensor Ultrasonik [4]

Sensor SRF 04 ini sensor ultrasonic yang berfungsi untuk mengukur jarak suatu benda dengan prinsip memancarkan gelombang ultrasonic kemudian menangkap pantulannya. Sensor ini dapat mengukur jarak antara 3 cm sampai 300 cm. Lebar pulsanya sekitar 100 uS sampai 18 mS. Sensor ini membangkitkan sinyal ultrasonok sebesar 40 KHz. Pada saat Trigger diaktifkan gelombang ultrasonic ini akan merambat diudara dengan kecepatan 344.424 m/detik atau 1 cm setiap 29.034 μs [4].

Gelombang tersebut akan mengenai objek kemudian terpantul kembali ke sensor. Selama menunggu pantulan, sensor akan menghasilkan sebuah pulsa (*high*) pulsa ini akan berhenti (*low*) ketika gelombang suara pantulan terdeteksi oleh sensor. Lebar pulsa tersebutlah yang yang dipresentasikan sebagai jarak antara sensor ping dengan objek. Sensor Ultrasonik dalam perancangan sistem ini digunakan untuk mendeteksi jarak. Untuk dapat melakukan navigasi dengan baik maka diperlukan 3 buah sensor ultrasonic yang dipasang pada bagian samping pojok kiri, depan dan samping pojok kanan. Dari kelima sensor inilah yang nantinya dijadikan acuan sebagai input parameter kendali *fuzzy* pada Arduino Mega 2560. Dengan rumus untuk memperoleh parameter jarak dalam cm :

$$Jarak = \frac{\left(\frac{Lebar\ pulsa}{29.034\mu S}\right)}{2} \quad \text{.....(2.7)}$$

2.4 Arduino Mega 2560



Gambar 2.7. Arduino Mega 2560 [5]

Arduino Mega 2560 adalah papan mikrokontroler sesuai dengan Atmega 2560 (Atmega 2560 datasheet) . Ic ini memiliki 54 pin digital input / output (dan 14 pin input / output dapat digunakan hasil PWM), 16 input analog, 4 UART (*Universal Asynchronous Receiver / Transmitter*) untuk antarmuka dengan RS232 port serial perangkat diaktifkan termasuk komputer, 16 MHz sangat osilator, sebuah koneksi USB, jack listrik, header ICSP, bersama dengan tombol sebagai reset [5].

Fitur ATMEGA 2560

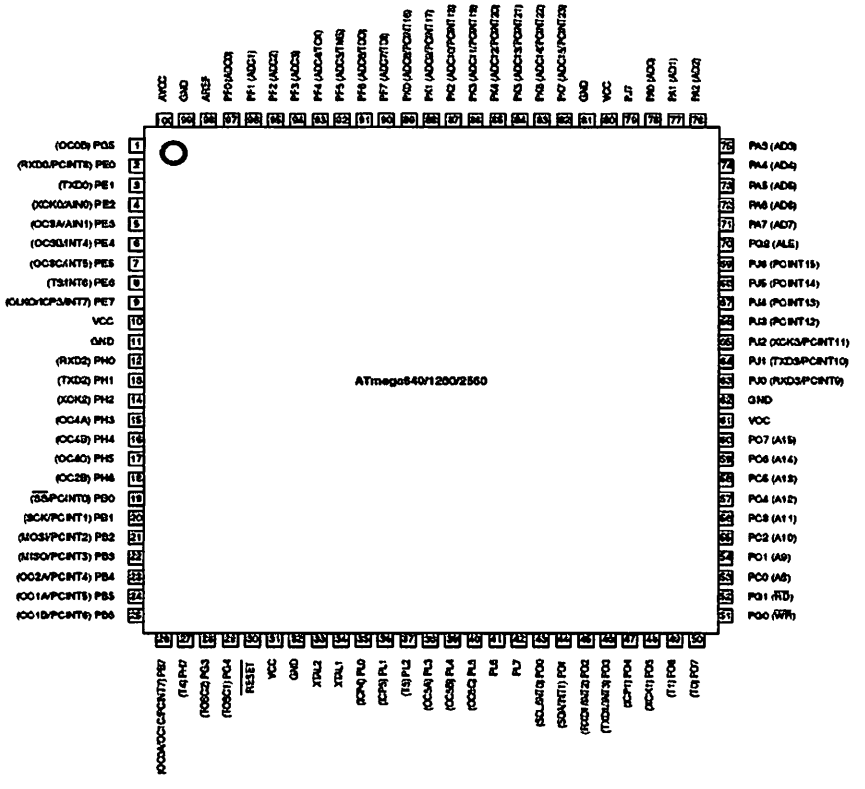
- *High Performance, Low Power AVR®*
  - *8-Bit Microcontroller*
- *Advanced RISC Architecture*
- *High Endurance Non-volatile Memory segments*
  - *8K Bytes of In-System Self-programmable Flash program memory*

- *4K Bytes EEPROM*
- *8K Bytes Internal SRAM*
- *Write/Erase cyles: 10,000 Flash/100,000 EEPROM*
- *Data retention: 20 years at 85°C/100 years at 25°C*
- *Programming Lock for Software Security*

• *Peripheral Features*

- *16-channel ADC*
- *Programmable Serial USART*
- *Master/Slave SPI Serial Interface*
- *Byte-oriented 2-wire Serial Interface (Philips IC compatible)*
- *Interrupt and Wake-up on Pin Change* [5]

2.4.1 Konfigurasi Pin Atmega 2560



Gambar 2.8. *Konfigurasi Pin Out* ATmega 2560 [6]

Deskripsi *Konfigurasi Pin Out* ATmega 2560

VCC : *Digital supply voltage.*

GND : *Ground.*

Port A (PA7..PA0) : Pada Port A terdapat 8-bit bi-directional I / O port dengan resistor pull up internal (bagian ini digunakan untuk setiap bit). Pada Port A juga di fasilitasi *Output buffer* yang memiliki karakteristik *drive* simetris dengan dua sink yang tinggi dan kemampuan sumber tegangan. Sebagai input, pin Port A secara eksternal menggunakan tegangan rendah. Jika resistor *pull-up* diaktifkan. Pin Port A dapat dinyatakan saat posisi reset.

Port B (PB7..PB0) : Pada Port B terdapat 8-bit bi-directional I / O port dengan resistor pull-up internal (digunakan untuk setiap bit). Port B *buffer* memiliki karakteristik drive yang simetris dengan kedua sink tinggi dan kemampuan sumbernya. Sebagai input, pin port B yang secara eksternal sumber yang rendah saat resistor *pull-up* diaktifkan. Pin di Port B adalah tri-dinyatakan saat kondisi reset diaktifkan.

Port G (PG5..PG0) : Pada port G terdapat 6 - bit port I / O dengan resistor *pull- up* internal (dipilih untuk setiap bit). Port G memiliki *output buffer* dengan karakteristik drive simetris dengan kedua sink yang tinggi dan tegangan sumber. Sebagai input pin Port G yang secara eksternal rendah pada sumber arus jika *pull- up* resistor diaktifkan. PortG adalah *tri -stated* ketika kondisi reset.

Port H (PH7. PH0) : Port H adalah 8 - bit bi - directional I / O port dengan resistor *pull- up* internal (digunakan untuk setiap bit). Port H *Output buffer* memiliki karakteristik *drive* yang simetris dengan kedua sink tinggi dan kemampuan pada sumber tegangan. Sebagai input, Port H yang secara eksternal ditarik rendah saat posisi *pull-up* diaktifkan. Port H tri - menyatakan saat reset kondisi menjadi aktif.

Port K (PK7. . PK0) Port K berfungsi sebagai input analog ke A / D Converter. Port K adalah 8 - bit bi - directional I / O port dengan resistor pull- up internal (dipilih untuk setiap bit) . Port K *Output buffer* memiliki karakteristik *drive* yang simetris dengan kedua sink tinggi dan kemampuan tegangannya. Sebagai input Port K yang secara eksternal rendah dan jika

resistor *pull-up* diaktifkan . Port K menyatakan saat reset.

Port L (PL7. . PL0) Port L adalah 8 - bit *bi - directional* I / O port dengan resistor *pull- up* internal (dipilih untuk setiap bit) . Port Output L *buffer* memiliki karakteristik drive yang simetris dengan kedua sink tinggi dan kemampuan tegangan. Sebagai input, port L yang secara eksternal ditarik rendah akan sumber jika resistor pull-up diaktifkan. Pin Pelabuhan L adalah tri - dinyatakan saat reset kondisi menjadi aktif, bahkan jika jam tidak berjalan. Port L juga menfasilitasi berbagai fitur.

RESET pengulangan masukan . Tingkat rendah pada pin ini selama lebih dari panjang pulsa minimum maka akan mengulang , bahkan jika jam tidak berjalan. Panjang pulsa minimum akan di suplai.

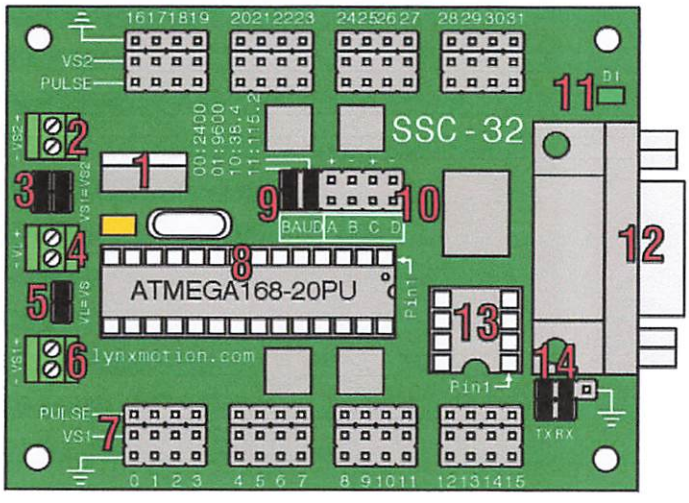
XTAL1 Input untuk penguat *inverting Oscillator* dan masukan ke sirkuit waktu operasi intern.

XTAL2 Output dari penguat pembalik Oscillator .

AVCC adalah pin tegangan suplai Port F dan A / D Converter. diharuska arus eksternal terhubung ke VCC, bahkan jika ADC tidak digunakan. Jika ADC digunakan, diharuskan con - dihubungkan ke VCC melalui *low-pass* filter.

AREF ini adalah pin referensi analog untuk A / D Converter [6].

2.5 *Module SSC -32*



Gambar 2.9. *Module* SSC-32 [7]

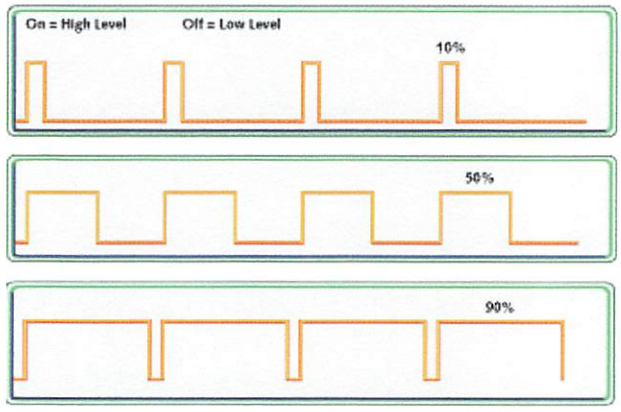
Module SSC - 32 (servo *controller serial*) adalah servo controller kecil *preassembled* dengan beberapa fitur besar. Controller ini memiliki resolusi tinggi (1uS) untuk posisi yang akurat, dan bergerak sangat halus. Rentang ini 0.50mS untuk 2.50mS untuk jarak sekitar 180 °. Kontrol gerak dapat respon langsung, kecepatan dikontrol, gerakan waktu, atau kombinasi keduanya. Yang unik memungkinkan setiap kombinasi servo untuk memulai dan gerak akhir pada saat yang sama, bahkan jika servos harus pindah jarak yang berbeda. Ini adalah fitur yang sangat kuat untuk membuat sebuah prototipe berjalan kompleks untuk robot multi servo berjalan. Posisi servo atau gerakan dapat menjawab pertanyaan untuk memberikan umpan balik ke komputer host. Bahkan sequencer 18 servo *hexapod* ini memungkinkan kontrol penuh terhadap semua aspek bolak balik *hexapod* hanya dengan mentransfer beberapa nilai dari *host controller*. Setiap output dapat digunakan sebagai output tingkat TTL. Ada 4 input digital yang statis atau terkunci, sehingga Anda tidak perlu khawatir kehilangan data. Controller ini juga dapat digunakan sebagai input analog. Ada tiga blok terminal untuk pilihan powering. Pada DB9 input memiliki tingkat keakuratan RS - 232 untuk digunakan dengan PC . Dan juga sebuah socket untuk 24LC32P EEPROM , yang akan digunakan dalam pengembangan dari firmware.

Fitur Module SSC-32

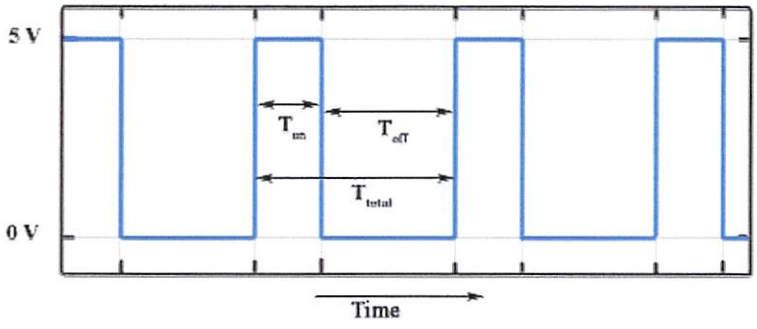
- *EEPROM = 24LC32P (Required for 2.01GP)*
- *Speed = 14.75 MHz*
- *Internal Sequencer = 12 Servo Hexapod (Alternating Tripod)*
- *Serial input = True RS-232 or TTL, 2400, 9600, 38.4k, 115.2k,*
- *Outputs = 32 (Servo or TTL)*
- *Current requirements = 31mA*
- *PC interface = DB9F*
- *Microcontroller interface = Header posts*
- *Servo control = Up to 32 servos plug in directly*
- *Servo type supported = Futaba or Hitec*
- *Servo travel range = 180°*
- *Servo resolution = 1uS, .09°*
- *Servo speed resolution = 1uS / Second*
- *PC board size = 3.0" x 2.3"*
- *VS current capacity = 15 amps per side, 30 amps max* [7]

2.6 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 %.



Gambar 2.10 Variasi Presentase *Duty Cycle* [3]



Gambar 2.11 Penjelasan *Duty Cycle* [3]

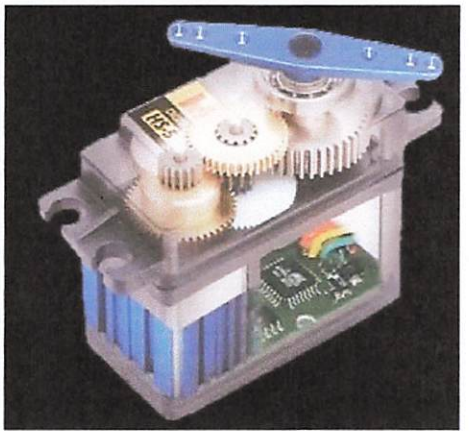
Dengan persamaan untuk mencari persentase *Duty Cycle* :

$$Duty\ Cycle = \frac{Ton}{Ton+Toff}\ x\ 100\% \quad \text{.....(2.8)}$$

$\mathbf{T_{on}}$ adalah waktu dimana tegangan keluaran berada pada posisi tinggi (baca: *high* atau *1*)

$\mathbf{T_{off}}$ adalah waktu dimana tegangan keluaran berada pada posisi LOW atau 0

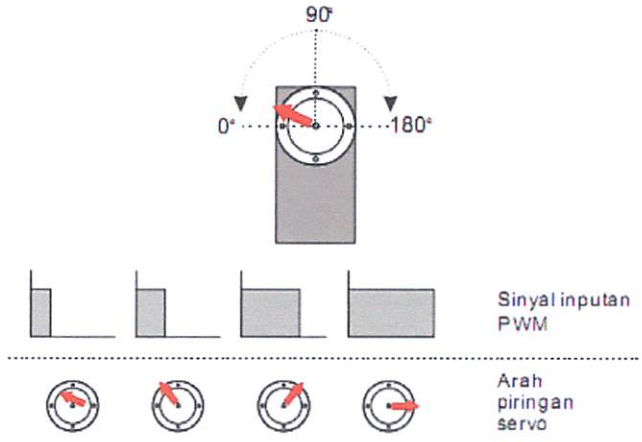
2.7 Motor Servo HS 645 MG



Gambar 2.12 Bentuk Fisik Hitec Servo HS-645MG [8]

Motor servo adalah motor yang mampu bekerja dua arah yaitu searah jarum jam (*clockwise*) dan berlawanan jarum jam (*counterclockwise*) dimana arah dan sudut pergerakan rotornya dapat dikendalikan hanya dengan memberikan pengaturan *duty cycle* sinyal PWM pada bagian pin kontrolnya.

Motor servo pada umumnya terdiri dari dua jenis, yaitu servo *continuous* dan servo standar. Servo *continuous* dapat berputar sebesar 360 derajat, sedangkan motor servo tipe standar hanya mampu berputar 180 derajat.



Gambar 2.13 Prinsip Kerja Motor Servo [8]

Spesifikasi Motor Servo HS-645MG [8]

Control System: +Pulse Width Control 1500usec Neutral

Required Pulse: 3-5 Volt Peak to Peak Square Wave

Operating Voltage: 4.8-6.0 Volts

Operating Temperature Range: -20 to +60 Degree C

Operating Speed (4.8V): 0.24sec/60° at no load

Operating Speed (6.0V): 0.20sec/60° at no load

Stall Torque (4.8V): 106.93 oz/in. (7.7kg.cm)

Stall Torque (6.0V): 133.31 oz/in. (9.6kg.cm)

Operating Angle: 45 Deg. one side pulse traveling 400 usec

Continuous Rotation Modifiable: Yes

Direction: Clockwise/Pulse Traveling 1500 to 1900 usec

Current Drain (4.8V): 8.8mA/idle and 350mA no load operating

Current Drain (6.0V): 9.1mA/idle and 450mA no load operating

Dead Band Width: 8usec

Motor Type: 3 Pole Ferrite

Potentiometer Drive: Indirect Drive

Bearing Type: Dual Ball Bearing

Gear Type: 3 Metal Gears and 1 Resin Metal Gear

Connector Wire Length: 11.81" (300mm)

Dimensions: 1.59" x 0.77"x 1.48" (40.6 x 19.8 x 37.8mm)

Weight: 1.94oz. (55.2g)

BAB III
PERANCANGAN DAN ANALISA SISTEM

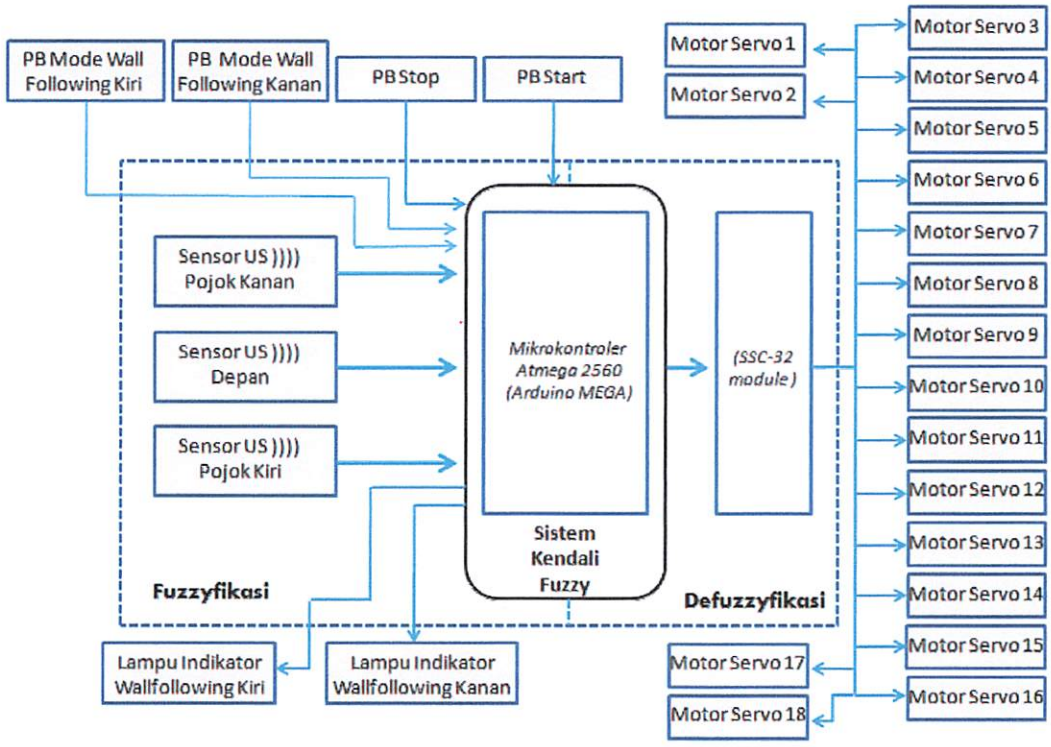
Pada bab ini akan membahas mengenai perencanaan sistem, prinsip kerja dan perancangan perangkat keras (*hardware*) serta perangkat lunak (*software*) yang berkaitan dengan kontrol robot yaitu mengenai design sistem kendali *fuzzy* yang diimplementasikan pada sistem *embedded Arduino Mega 2560.* Pada 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.

3.1 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 dan *output*. Pada bagian input terdiri dari rangkaian sensor yang dipakai yaitu 3 buah modul sensor jarak berbasis ultrasonik dan rangkaian *push button* untuk menentukan mode *following* dan yang diinputkan oleh *user*.

Bagian kontroler terdiri dari modul Arduino Mega 2560 yang dikombinasikan dengan modul SSC-32 yang berfungsi sebagai output untuk menggerakan 18 motor servo yang berfungsi sebagai aktuator robot. Dengan pergerakan sudut servo tersebut dikendalikan dengan mengubah nilai PWM yang di inputkan oleh *controler*.

Diagram blok keseluruhan sistem dapat dilihat pada gambar berikut :



Gambar 3.1 Diagram Blok Sistem

Penjelasan Diagram Blok :

1) Input , masukan nilai aktual dari Sensor Ultrasonik dan *Push Button* dari *user*. Bagian input terdiri dari :
    a. PB *Start* berfungsi untuk menjalankan robot *hexapod* ini sesuai dengan mode *Wall Following*nya.
    b. PB *Mode Wall Following* Kiri digunakan untuk menentukan mode penyusuran robot *hexapod* ini terhadap dinding sebelah kiri robot pada saat dijalankan.
    c. PB *Mode Wall Following* Kanan digunakan untuk menentukan mode penyusuran robot *hexapod* ini terhadap dinding sebelah kanan robot pada saat dijalankan.
    d. PB *Stop* berfungsi untuk mematikan robot *hexapod* sesuai kondisi awal yaitu diam.

e. Sensor US Pojok Kiri adalah sensor ultrasonik SRF pojok bagian kiri berfungsi untuk membaca nilai jarak aktual robot terhadap dinding kiri sebagai inputan *fuzzy*.
f. Sensor US Depan adalah sensor ultrasonik SRF depan berfungsi untuk membaca nilai jarak aktual robot terhadap dinding depan atau halangan sebagai inputan *fuzzy*.
g. Sensor US Pojok Kanan adalah sensor ultrasonik SRF pojok bagian kanan berfungsi untuk membaca nilai jarak aktual robot terhadap dinding kanan sebagai inputan *fuzzy*.

2) Mikrokontroler Atmega 2560, yaitu bagian pengolahan hasil nilai yang dibaca oleh sensor. Kontroler pada perancangan sistem ini menggunakan *board Arduino Mega 2560*. Kontroler disini digunakan untuk pembacaan 3 sensor ultrasonic SRF, pembacaan mode yang diinputkan oleh user dan mengolah data dari hasil *defuzzyfikasi* oleh *Arduino Mega 2560* menjadi nilai gerakan yang akan dikirim ke *module* SSC-32 melalui komunikasi Rx/ Tx dan nilai PWM tersebut yang akan menghasilkan nilai derajat pada Motor Servo untuk menggerakan kaki robot *hexapod* tersebut. Kontroler ini digunakan sebagai kontroler utama yang digunakan untuk mengolah semua *input* yang akan menentukan gerakan dari robot *hexapo*d ini melalui sistem kendali logika *fuzzy*.
3) *Output*, yaitu bagian yang akan memproses hasil *defuzzyfikasi* yang dilakukan oleh kontroler menjadi suatu gerakan robot (maju dan belok). Bagian ouput terdiri dari beberapa bagian, yaitu :
    a. *SSC-32 Module* yang akan mengubah hasil *defuzzifikasi* menjadi nilai PWM yang akan menggerakkan motor servo pada kaki *robot hexapod*.
    b. Lampu Indikator *Wallfollowing Kiri* digunakan untuk membantu user dalam mengetahui mode yang sedang aktif atau yang sedang digunakan yaitu *mode wallfollowing* sebelah kiri.

c. Lampu Indikator *Wallfollowing* Kanan digunakan untuk membantu *user* dalam mengetahui mode yang sedang aktif atau yang sedang digunakan yaitu *mode wallfollowing* sebelah kanan.
d. Motor Servo 1-18 digunakan sebagai aktuator kaki pada robot *hexapod* ini dengan menerima nilai PWM dan menjadikan suatu nilai sudut pada kaki untuk melakukan gerakan maju dan berbelok sesuai perintah yang diberikan dari *module* SSC-32.

3.1.1 Prinsip Kerja

Cara mengaktifkan robot *hexapod* ini yaitu pertama kita memilih jenis mode *wallfollowing* yang akan kita gunakan dengan cara menekan salah satu PB *Mode Wallfollowing* kanan atau kiri untuk mode navigasi robot *hexapod* ini untuk menyusuri dinding lintasan. Bila sudah lampu indikator *following* akan menyala maka selanjutnya kita menekan PB *Start* untuk menjalankan robot *hexapod* ini. Pada robot ini untuk bergerak secara otomatis dibutuhkan sensor yang dapat merepresentasikan sebuah parameter keadaan suatu lingkungan ke dalam sebuah nilai. Pada diagram blok di atas terdapat 3 buah sensor ultrasonic yang akan memberikan suatu nilai jarak robot terhadap dinding sebelah kanan, depan dan samping kiri robot. Dari 3 buah sensor ultrasonic yang di pasang pada pojok kanan, pojok kiri dan depan bagian robot tersebut melalui proses berikut dapat menghasilkan nilai jarak dengan prinsip kerja, unit *trigger* pada sensor SRF-04 akan mengeluarkan sonar Ultrasonik yang akan dipantulkan oleh dinding dalam waktu tertentu dan diterima pada bagian *echo* sensor SRF-04 tersebut dan menghasilkan pulse yang terhubung pada pin digital Arduino Mega 2560, pulse tersebut yang akan diolah dan dikonversi menjadi nilai jarak pada Arduino Mega 2560 dan dari data jarak tersebutlah yang akan digunakan sebagai input kontroler *fuzzy* pada Arduino Mega 2560 untuk proses *fuzzyfikasi*, evaluasi *rule* dan proses *defuzzyfikasi*. Output yang dihasilkan dari proses tersebut adalah nilai pwm yang akan dikirimkan dari Arduino Mega 2560 ke *module* SSC-32 dengan melalui komunikasi *Serial* RX1/TX1 dan data yang dikirim tersebut menjadi acuan untuk menggerakan motor servo yang berkedudukan sebagai aktuator yang akan merubah derajat sudut servo pada masing-masing persendian dari kaki robot

hexapod, maka dari sinilah pergerakan navigasi robot untuk penjejak dinding dan menghindari halangan menjadi lebih *smooth* atau halus karena pergerakan robot *hexapod* ini tergantung dari nilai jarak yang dihasilkan sensor SRF-04 (sensor ultrasonik) yang berfungsi sebagai inputan sistem *fuzzy* robot *hexapod* ini.

3.2 Perancangan Perangkat Keras

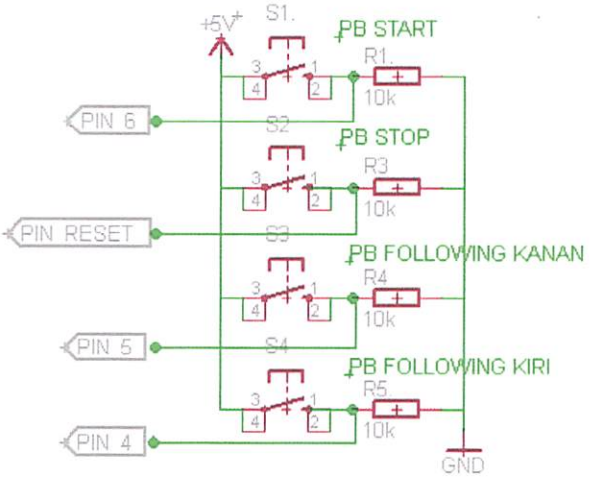
3.2.1 Perancangan Rangkaian Push Button (*User Input*)

Rangkaian *push button* di pasang pada pin Arduino Mega 2560 pada perancangan sistem ini digunakan untuk menentukan pilihan *mode wall following* dengan konfigurasinya yaitu *mode wall following* kanan (PIN5) atau *wall following* kiri (PIN4) terhadap dinding dan juga untuk *Start* (PIN6) dan *Stop* robot (PIN *Reset*). Rangkaian yang digunakan adalah menggunakan *push button* menggunakan *resistor pulldown* dengan tegangan sumber 5V karena pada pin input arduino ini membutuhkan nilai arus untuk inputan atau trigger antara 0,5mA -40mA maka dapat dihitung untuk resistor sebagai berikut.

$V = I \cdot R$ .....(3.1)

$5\ V = 0{,}5mA \cdot R$

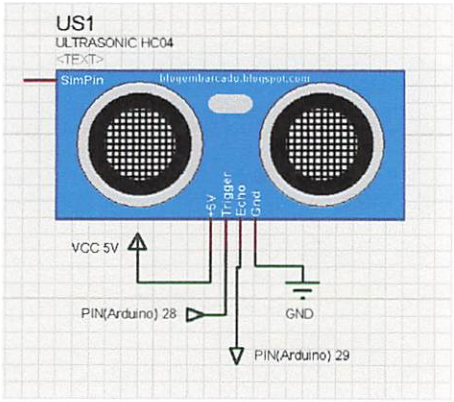
$$\frac{5v}{0{,}5x10^{-3}} = 10k\Omega$$



Gambar 3.2 *Push Button User Input*

3.2 .2 Perancangan Sensor Ultrasonik SRF HC-04

Perancangan sensor Ultrasonik pada sistem ini digunakan untuk mengetahui jarak robot dengan dinding atau halangan yang berfungsi sebagai inputan pada unit kontroler untuk memberi input nilai fuzyfikasi yang akan digunakan untuk mendriver robot *hexapod* dalam melakukan pergerakan yang inputanya terdiri dari sensor ultrasonik depan, pojok kanan dan pojok kiri. Dengan prinsip kerja alat ini memancarkan suara ultrasonik pada *trigger* dan menerima suara pantulanya pada *echo* maka dengan susunan pin yang akan difungsikan sebagai pemicu unit *trigger* dan *echo* untuk penerima data sebagai berikut konfigurasinya antara SRF HC04 dengan Arduino Mega 2560 dengan salah satu contoh gambar konfigurasi rangkaian.



Gambar 3.3 Sensor Ultrasonic SRF HC-04

Tabel 3.1 Keterangan Pin Sensor Ultrasonic SRF HC-04

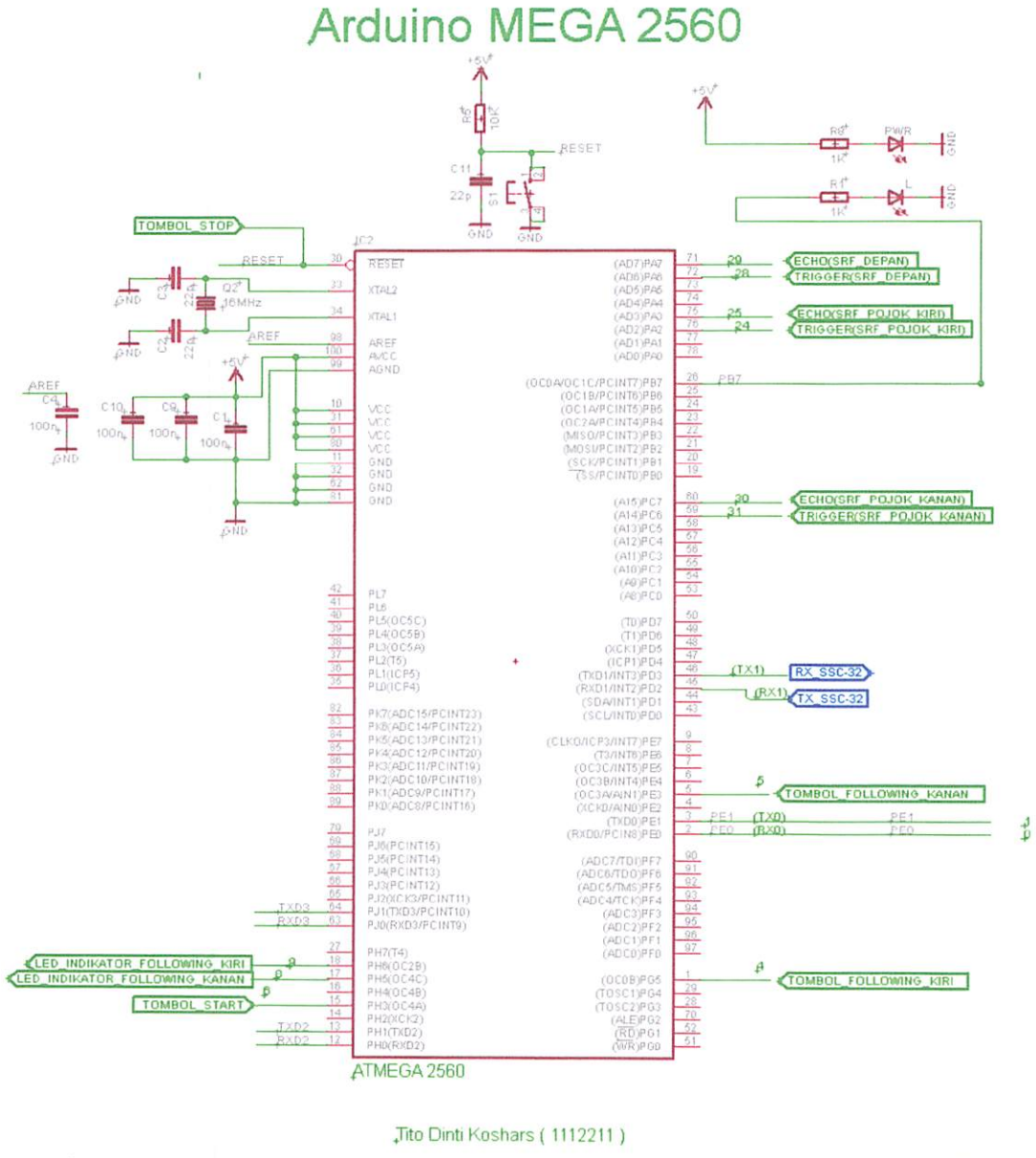
Posisi SRF	Trigger	Echo
Depan	Pin 28	Pin 29
Pojok Kanan	Pin 30	Pin 31
Pojok Kiri	Pin 24	Pin 25

*Dihubungkan ke Board Arduino

3.2.3 Perancangan Rangkaian Arduino Mega 2560

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 ATMega 2560. Mikrokontroler ini sengaja dipilih karena pertimbangan *memory flash* (memori program) yang cukup besar untuk mengolah data nilai *fuzzy* yang akan dipergunakan untuk mengontrol robot *hexapod* ini, serta memiliki fitur 4 buah USART (Serial Rx/Tx). Dengan perancangan sistem untuk mengontrol robot *hexapod* ini yaitu seperti gambar di bawah ini.



Gambar 3.4 Rangkaian Minimum Sistem ATMega 2560 (Arduino Mega 2560)

3.2.3.1 Perancangan Rangkaian *Clock Generator*

Pada perancangan *clock generator* ini menggunakan kristal berfrekuensi 16.000 Mhz, sedangkan nilai kapasitor menggunakan 22pF sampai 22pF. Nilai kapasitor ini diperoleh dari tebel *data sheet* tentang penggunaan kapasitor untuk rangkaian oscilator/sistem *clock* pada Atmega 2560. Penggunaan kristal 16.000 Mhz ini bertujuan agar perhitungan *baudrate* tidak mengalami *error* yang disebabkan kerena selisih perhitungan. Pada Arduino Mega 2560 ini menggunakan kristal 16.000 Mhz, dimana *baudrate* yang akan diinginkan adalah 115200 bps, maka nilai pada UBRR (*USART Baud Rate Register*) dapat di tentukan dengan perhitungan :

UBRR = (fosc / 16 x *Baud*) – 1 .....(3.2)

UBRR = (16000000 / 16 x 115200) – 1

UBRR = (16000000 / 1843200) – 1

UBRR = 8,68 – 1

= 7,68d atau 8H

UBRR = *USART Baud Rate Register*

fosc = Frekuensi kristal *osilator*

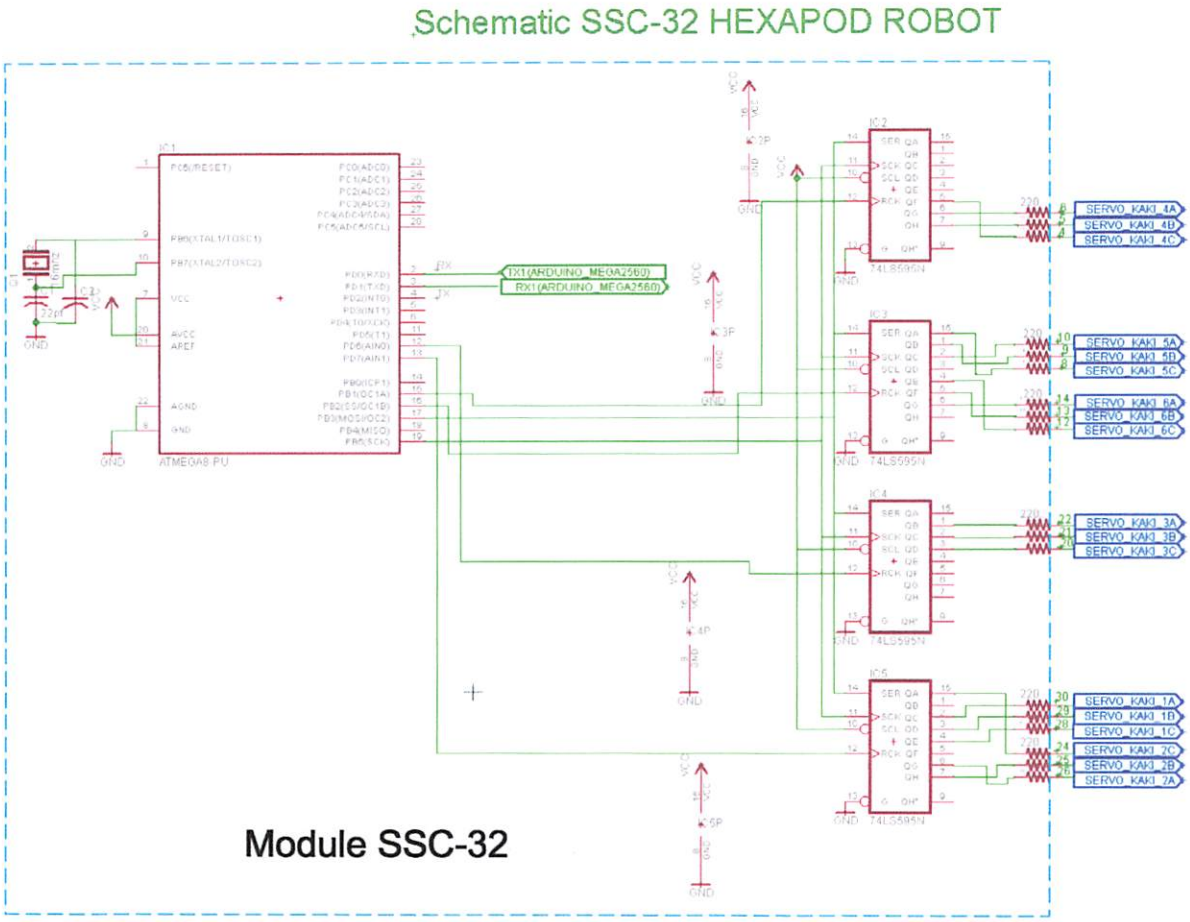
Baud = *baud rate* (bit per detik)

3.2.3.2 Perancangan Rangkaian Reset

Reset digunakan untuk menginisialisasi 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.*

3.2.4 Perancangan Rangkaian *Module* SSC-32

SSC - 32 (*servo controller serial*) adalah servo *controller* kecil *preassembled* dengan beberapa fitur besar. Control ini memiliki resolusi (1uS) untuk posisi yang akurat , dan bergerak sangat halus. Rentang ini 0.50mS untuk 2.50mS untuk jarak sekitar 180 °. Dalam sistem robot *hexapod* ini untuk komunikasi data antara Arduino Mega 2560 dengan *module* SSC-32 dalam mengatur sudut kaki robot *hexapod* ini menggunakan komunikasi *USART1* dengan *Baud Rate* 115200 bps karena untuk mempercepat perolehan data yang masuk ke *module* SSC-32. Berikut adalah skematik *module* SSC-32 untuk robot *hexapod* ini.

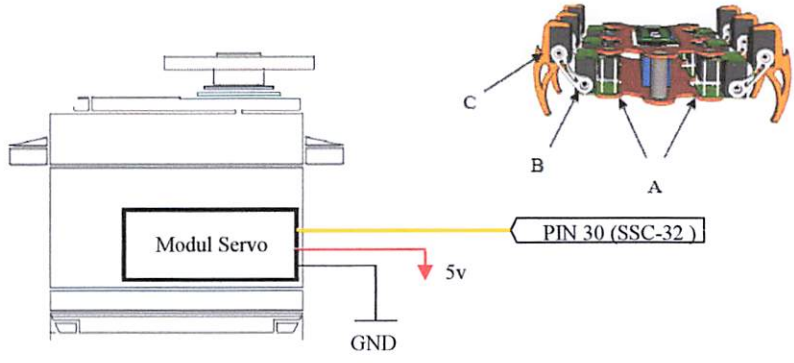


Gambar 3.5 Rangkaian *Module* SSC-32

3.2.5 Perancangan Pin Data Motor Servo Kaki Robot Hexapod

Motor servo disini berjumlah 18 motor servo berfungsi sebagai penggerak atau aktuator kaki robot *hexapod*, motor ini bekerja dua arah yaitu searah jarum jam (*clockwise*) dan berlawanan jarum jam (*counterclockwise*) dimana arah dan sudut pergerakan rotornya dapat dikendalikan hanya dengan memberikan pengaturan *duty cycle* sinyal PWM pada bagian pin kontrolnya.

Pada motor servo ini memiliki 3 kabel,satu kabel untuk input pulsa pengontrol ,dua kabel untuk positif dan negative *supply*. Berikut adalah contoh konfigurasi pin servo dengan *module* SSC32 dan tabel susunan pin yang akan digunakan pada robot hexapod ini yang tersambung dengan SSC32 *module* :



Gambar 3.6 Motor Servo HS-645 MG

Tabel 3.2 Tabel Konfigurasi *Motor Servo*

Kaki	Pin SSC-32 yang digunakan 18 servo		
	A	B	C
Kaki 1	Pin 30	Pin 29	Pin 28
Kaki 2	Pin 26	Pin 25	Pin 24
Kaki 3	Pin 22	Pin 21	Pin 20
Kaki 4	Pin 6	Pin 5	Pin 4
Kaki 5	Pin 10	Pin 9	Pin 8
Kaki 6	Pin 14	Pin 13	Pin 12

**di pasang ke pin SSC-32*

3.2.6 Perancangan Rangkaian Lampu Indikator

Indikator pada perancangan sistem ini digunakan sebagai penanda untuk memberitahukan kondisi dan mode *following* yang sedang digunakan pada robot. Rangkaian ini menggunakan LED berwarna merah berukuran 3 mm yang terhubung ke Arduino Mega 2560. 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 :

$$R_S = \frac{Vs-Vf}{I_F} \quad \text{.....(3.3)}$$

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 minimal 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 :

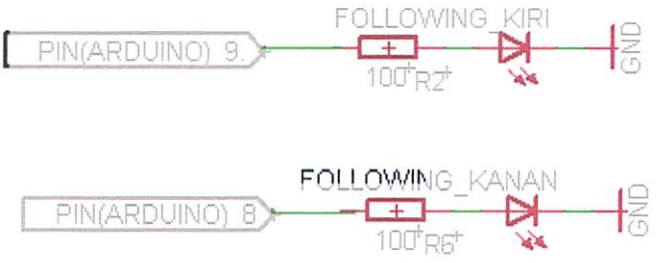
$$R_s = \frac{Vs - Vf}{I_F}$$

$$R_s = \frac{5 - 3.5}{25\ x\ 10^{-3}}$$

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

$R_s = 60\ \Omega$ dibulatkan menjadi $100\ \Omega$

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.



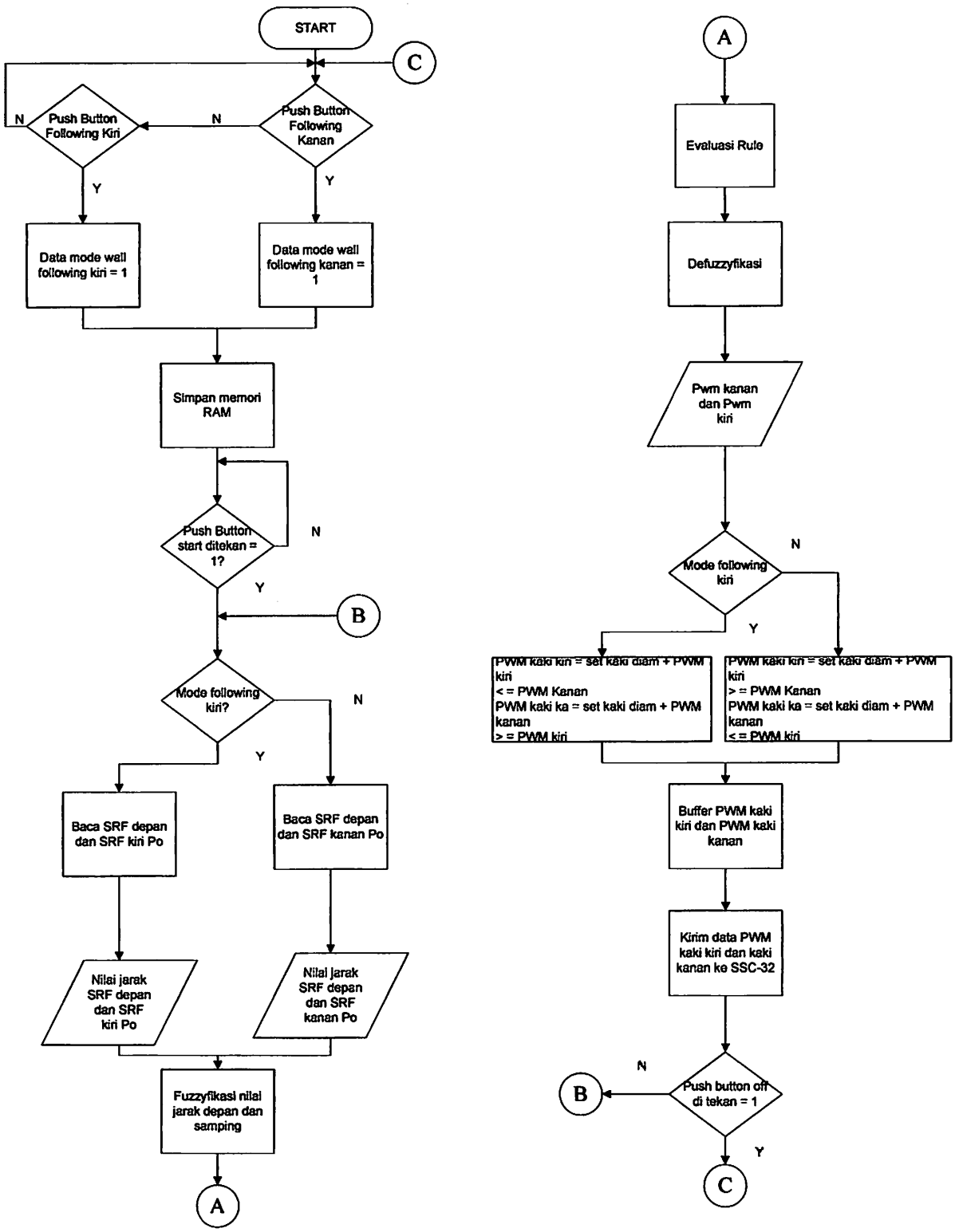
Gambar 3.7 Rangkaian Lampu Indikator

3.3 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. Perancangan kontroler Master (Arduino Mega 2560), terdiri dari program pembacaan *input push button*, program pembacaan sensor Ultrasonik SRF, perancangan kontroler *Fuzzy* dan program untuk komunikasi USART (Serial Rx/Tx) dengan *module* SSC-32.

3.3.1 Perancangan Kontroler Master

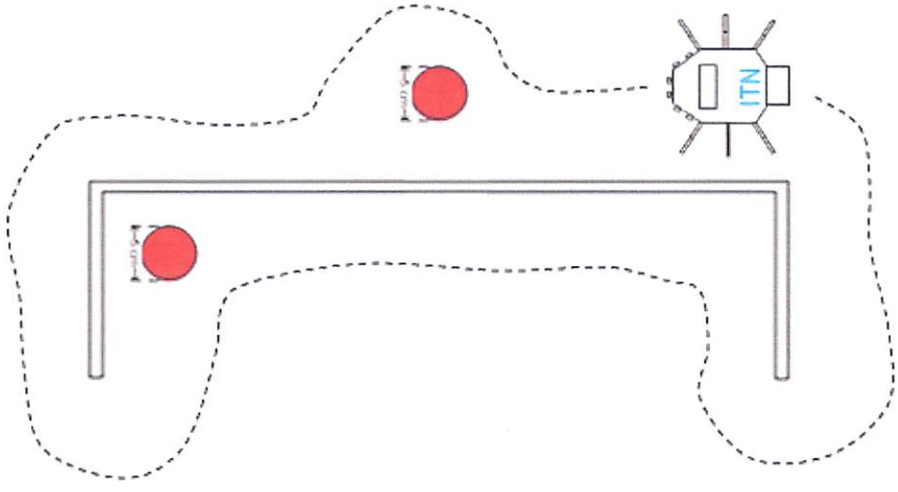
Pembuatan program pada kontroler Master (Arduino Mega 2560) menggunakan bahasa pemrograman Arduino dan dengan menggunakan juga library EFLL (*Embedded Fuzzy Logic Library*) yang sudah tersedia di dalamnya. Dengan gambaran alur kerja seperti pada gambar *flowchart* berikut.



Gambar 3.8 *Flowchart* Kontroler Master

3.3.1.1 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 3.9 dengan urutan robot berangkat dari posisi start kemudian menelusuri dinding dan melewati halangan sampai mencapai titik akhir (*finish*).

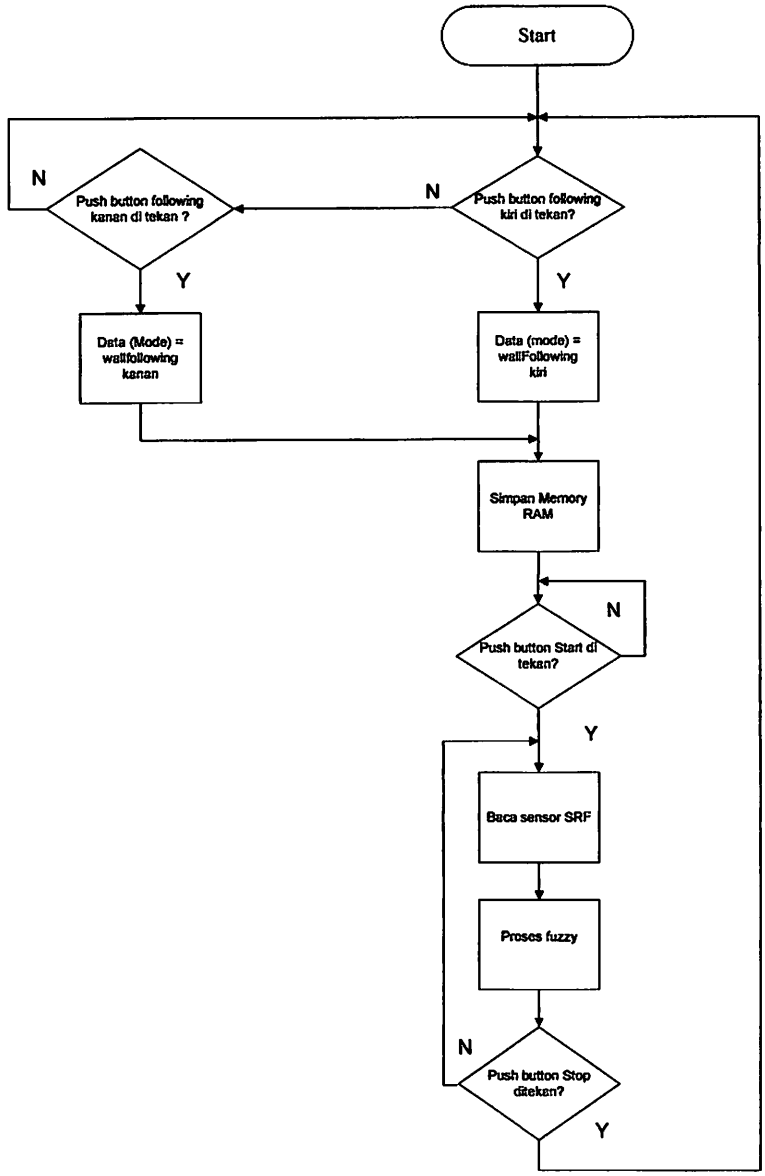


Gambar 3.9 *Path Planning*

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.

3.3.1.2 Perancangan Program Pembacaan *Push Button*

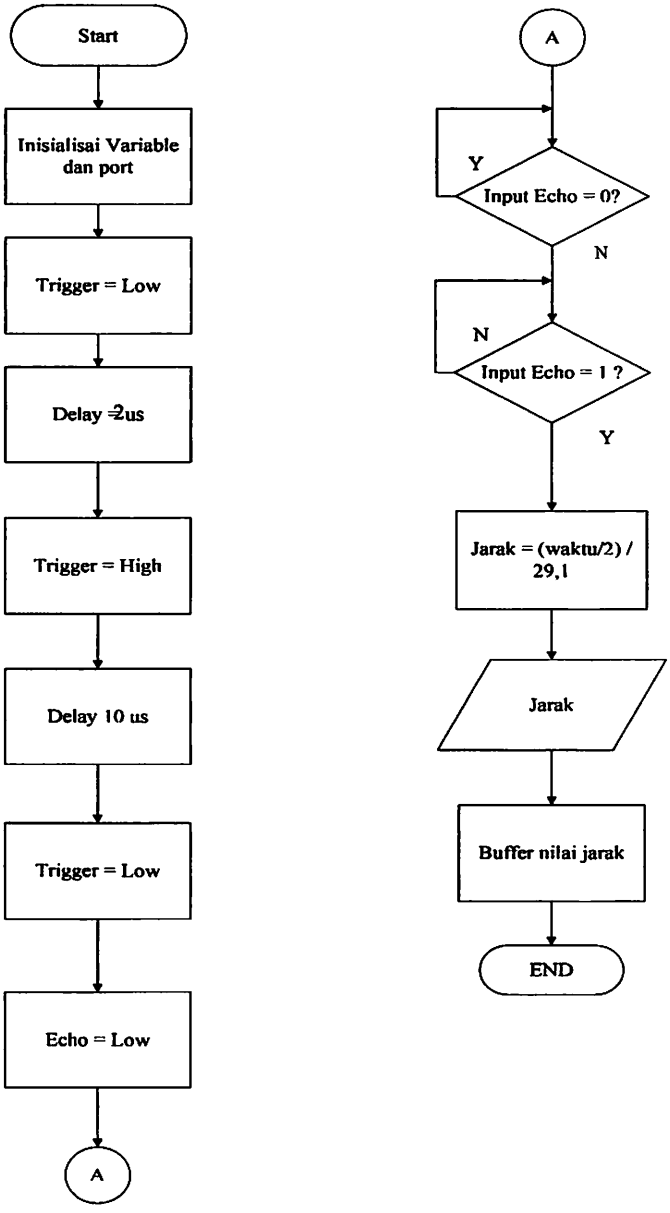
Push button disini di fungsikan sebagai *Human Intervace* yaitu untuk pengoperasian robot pada saat *Start, Stop* dan juga jenis pemakaian mode *Wallfollowing* yang akan digunakan. Disini kita menggunakan logik 1 untuk menginputkan ke arduino mega 2560 sebagai parameter *True*. Dengan flowchart sebagai berikut.



Gambar 3.10 *Flowchart* Pembacaan *Input Push Button*

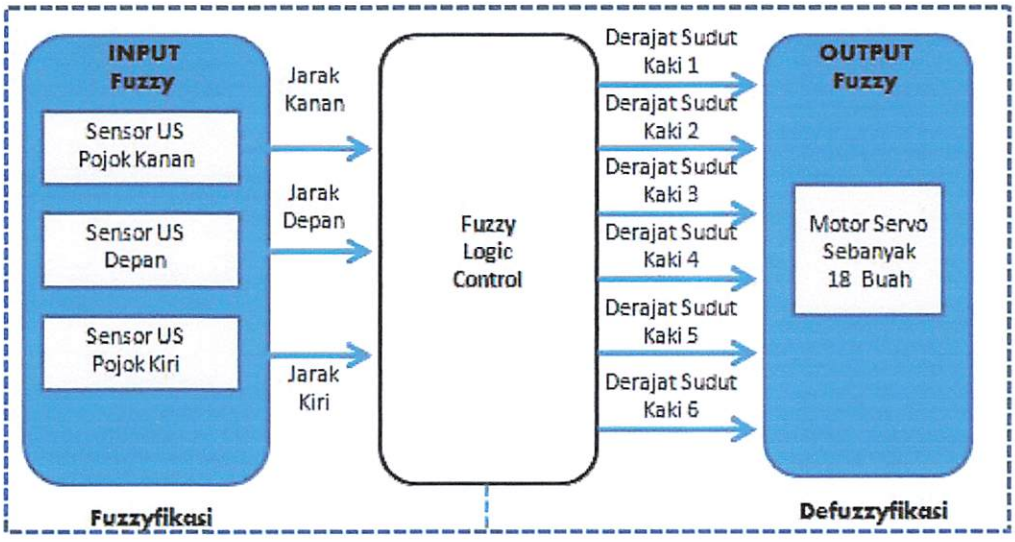
3.3.1.3 Perancangan Program Pembacaan Sensor Ultrasonik SRF

Di perancangan program pembacaan sensor jarak ini kita memakai sensor SRF HC04 yang membaca jarak dari sinyal ultrasonik yang di pancarkan oleh *Trigger* selama 10 us yang akan memantul pada benda di depanya dan diterima kembali oleh *Echo* dan di olah pada Arduino Mega 2560 sehingga menghasilkan parameter jarak yang akan di jadikan inputan pada fuzzyfikaksi untuk proses *fuzzy* yang akan digunakan untuk mengontrol pergerakan robot *hexapod* ini. Dengan gambar *flowchart* sebagai berikut.



Gambar 3.11 *Flowchart* Pembacaan Data Sensor SRF HC04

3.3.1.4 Perancangan Sistem Kendali *Fuzzy*



Gambar 3.12 Diagram Blok Sistem Kendali

Pada gambar diatas menjelaskan bagaimana proses mulai dari *fuzzy input*, proses sampai ke *aktuator outputnya*, jadi sensor ultrasonik disini sebagai nilai *input* untuk parameter jarak yang akan dipetakan dalam proses *fuzzyfikasi* dengan derajat keanggotaan (*grade of memberhip*) masing-masing. Maka selanjutnya data akan di evaluasi dengan *rule* untuk menentukan nilai *output* pada kontroler yang diubah ke dalam nilai pwm dan di outputkan ke 18 buah motor servo pada kaki *hexapod* robot ini yang berkedudukan sebagai *aktuator*. Prinsip jalan yang dihasilkan dari metode logika *fuzzy* robot ini yaitu menghasilkan nilai derajat sudut yang *smooth*, pada saat *wall following* kanan dan robot berjalan pada trek lurus maka robot akan menstabilkan posisinya terhadap dinding dengan *rule* yang telah ada pada metode kendali logika *fuzzy* yang telah di tanam pada Arduino Mega 2560 yang telah menggunakan sensor ultrasonik sebagai inputan. Salah satu contoh jika pada saat sensor US pojok kanan dan US depan mendeteksi bahwa jarak robot dengan dinding jauh maka pergerakan kaki robot sebelah kiri melangkah lebih jauh dari derajat sudut kaki sebelumnya dan derajat sudut kaki sebelah kanannya mengambil langkah sangat sedikit. Semua derajat perubahan sudut kakinya berubah secara otomatis sesuai dengan jarak robot dengan dinding. Maka hal tersebut akan menghasilkan pergerakan robot bergerak ke arah kanan dan akan terjadi sampai sensor US pojok kanan menemukan titik jarak stabil pada

rule yang telah di tanam. Pergerakan kaki robot disini diatur dengan derajat sudut kaki yang dihasilkan dari *rule fuzzy* jadi derajat sudut yang dihasilkan pada aktuator bisa berubah ubah dengan beberapa fariasi sudut yang di hasilkan oleh defuzzyfikasi dan dari hal tersebut akan menjadikan pergerakan kaki yang *smooth* atau halus, jadi disinilah inti dari jalan pada robot hexapod yang akan di buat. Perubahan derajat sudut kakinya berubah ubah sesuai jarak robot dengan dinding yang berfungsi sebagai penghalus gerakan dari robot tersebut.

A. Fuzzyfikasi

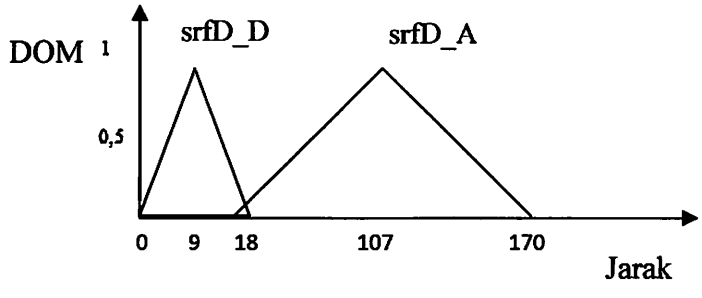
Model yang digunakan untuk proses fuzzyfikasi adalah dengan menggunakan model *triangular function* (fungsi keanggotaan segitiga). Pada perancangan sistem terdapat 3 variabel *input fuzzy* yaitu nilai jarak SRF depan, jarak SRF pojok kiri dan jarak SRF pojok kanan.

Maka dari beberapa percobaaan untuk batas aman dinding atau halangan dengan robot *hexapod* untuk bagian SRF depan menghasilkan parameter sebagai berikut :

Tabel 3.3 Range Fungsi Keanggotaan SRF Depan

Fungsi Keanggotaan SRF Depan	**Range**		
	a	**b**	**c**
srfD_D (Dekat)	0	9	18
srfD_A (Aman)	15	107	170

Grafik *membership function* untuk SRF depan digambarkan sebagai berikut :



Grafik 3.1 Fungsi Keanggotaan SRF Depan

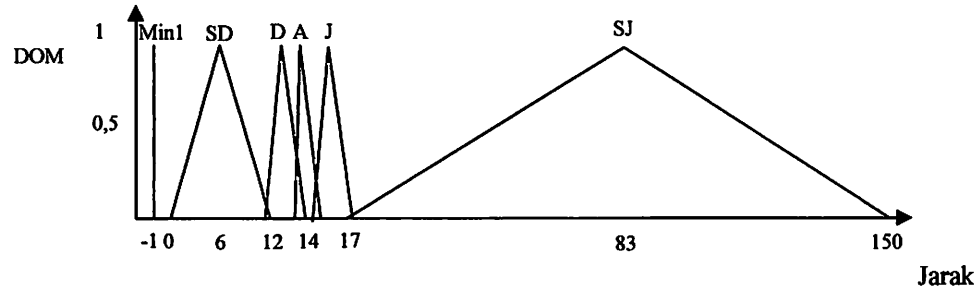
Keterangan : DOM (*Degree Of Membersip*)

Untuk melihat parameter jarak sebelah kiri kita memakai SRF pojok kiri .

Tabel 3.4 Range Fungsi Keanggotaan SRF Pojok Kiri

Fungsi Keanggotaan SRF pojok kiri	Range		
	a	b	c
srfKiPo_SD	0	6	12
srfKiPo_D	11	12.5	14
srfKiPo_A	13	14	15
srfKiPo_J	14	15.5	17
srfKiPo_SJ	16	83	150

Grafik fungsi keanggotaan segitiga dari SRF kiri pojok dapat dilihat pada gambar berikut :



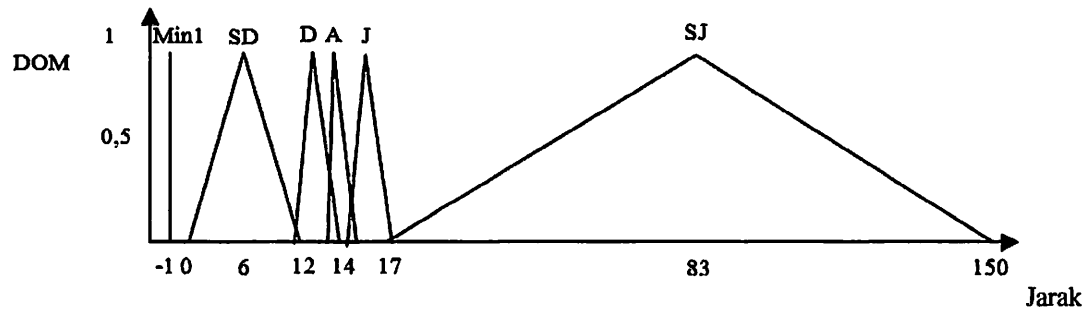
Grafik 3.2 Fungsi Keanggotaan SRF Kiri Pojok

Untuk melihat parameter jarak sebelah kanan kita memakai SRF pojok kanan maka kita membuat :

Tabel 3.5 Range Fungsi Keanggotaan SRF Pojok Kanan

Fungsi Keanggotaan SRF pojok kiri	Range		
	a	b	c
srfKaPo_SD	0	6	12
srfKaPo_D	11	12.5	14
srfKaPo_A	13	14	15
srfKaPo_J	14	15.5	17
srfKaPo_SJ	16	83	150

Grafik fungsi keanggotaan segitiga dari SRF kanan pojok adalah sebagai berikut :



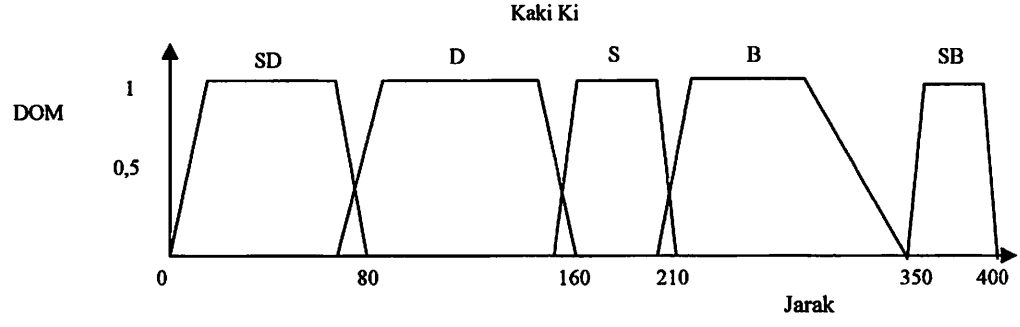
Grafik 3.3 Fungsi Keanggotaan dari SRF Kanan Pojok

Dan dengan *set output* untuk kaki sebelah kiri pada perubahan *duty* nya

Tabel 3.6 Range Fungsi Keanggotaan *Output* Kaki Sebelah Kiri

Fungsi Keanggotaan Output Kaki kiri	**Range**			
	a	**b**	**c**	**d**
kakiKi_SD	0	10	70	80
kakiKi_D	70	80	150	160
kakiKi_S	150	160	200	210
kakiKi_B	200	210	270	350
kakiKi_SB	350	360	390	400

Maka dapat digambarkan fungsi keanggotaan outputnya dengan grafik sebagai berikut untuk *output* kaki kiri.



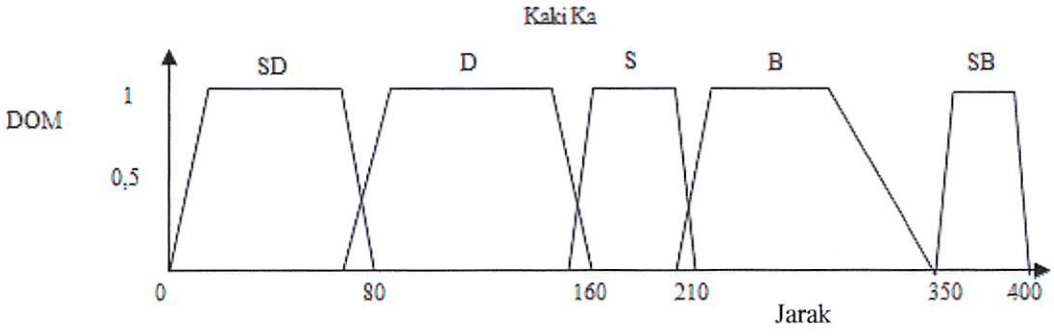
Grafik 3.4 Fungsi Keanggotaan Dari *Output* Kaki Kiri

Dan dengan *set output* untuk kaki sebelah kanan pada perubahan *duty* nya

Tabel 3.7 Range Fungsi Keanggotaan *Output* Kaki Sebelah Kanan

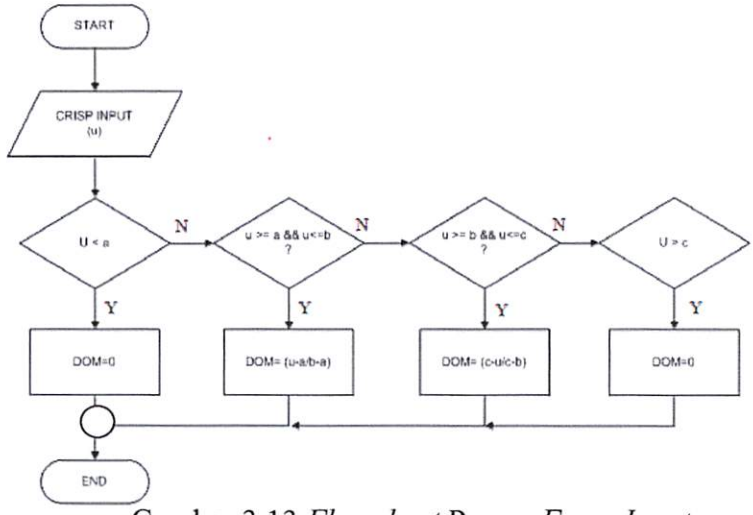
Fungsi Keanggotaan Output Kaki kiri	**Range**			
	a	**b**	**c**	**d**
kakiKa_SD	0	10	70	80
kakiKa_D	70	80	150	160
kakiKa_S	150	160	200	210
kakiKa_B	200	210	270	350
kakiKa_SB	350	360	390	400

Maka dapat digambarkan fungsi keanggotaan outputnya dengan grafik sebagai berikut untuk *output* kaki kanan.



Grafik 3.5 Fungsi Keanggotaan Dari *Output* Kaki Kanan

Untuk *flowchart fuzzy input* dengan salah satu contoh :



Gambar 3.13 *Flowchart* Proses *Fuzzy Input*

B. Evaluasi Rule

Dalam perancangan sistem ini, tujuan utama robot adalah menjaga jarak aktual agar robot dapat berjalan secara seflexibel mungkin dan presentase tabrakan (*collision*) sekecil mungkin. Maka dari itu dalam perancangan sistem ini, *metode* pengambilan data evalusai *rule* dengan mengubah ubah nilai derajat pada servo robot untuk membuat gerakan sehalus mungkin. Kita membuat tabel *rule* untuk setiap mode *Wallfollowing* yaitu sebanyak 10 *rule* dalam satu mode sebagai berikut.

Untuk penyusuran dinding sebelah kiri mempergunakan *rule*

Tabel 3.8 Aturan Pada Kontroler fuzzy Untuk Penghindar Halangan Dan Menyusuri Dinding Sebelah Kiri

Jarak Depan dan Pojok Kiri	**SD**	**D**	**A**	**J**	**SJ**
D	kakiKi_SB	kakiKi_B	kakiKi_S	KakiKi_S	kakiKi_SD
	kakiKa_SD	kakiKa_S	kakiKa_S	kakiKa_B	kakiKa_SB
A	kakiKi_SB	kakiKi_B	kakiKi_S	KakiKi_S	kakiKi_SD
	kakiKa_SD	kakiKa_S	kakiKa_S	kakiKa_B	kakiKa_SB

Untuk penyusuran dinding sebelah kanan mempergunakan *rule*

Tabel 3.9 Aturan Pada Kontroler *Fuzzy* Untuk Penghindar Halangan Dan Menyusuri Dinding Sebelah Kanan

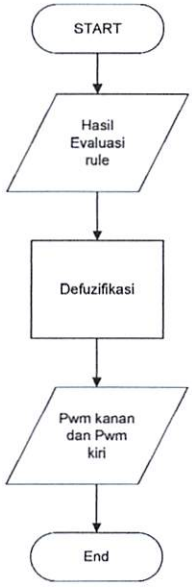
Jarak Depan dan Pojok Kanan	**SD**	**D**	**A**	**J**	**SJ**
D	kakiKi_SD	kakiKi_S	kakiKi_S	KakiKi_B	kakiKi_SB
	kakiKa_SB	kakiKa_B	kakiKa_S	kakiKa_S	kakiKa_SD
A	kakiKi_SD	kakiKi_S	kakiKi_S	KakiKi_B	kakiKi_SB
	kakiKa_SB	kakiKa_B	kakiKa_S	kakiKa_S	kakiKa_SD

Keterangan:

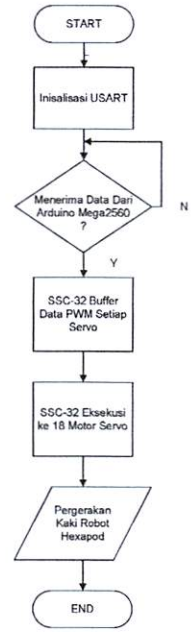
Jarak Depan :		Pojok kanan atau kiri :		Pojok kanan atau kiri :	
D	: Dekat	SD	: Sangat Dekat	SD	: Sangat Dikit
A	: Aman	D	: Dekat	D	: Dikit
		A	: Aman	S	: Sedang
		J	: Jauh	B	: Banyak
		SJ	: Sangat Jauh	SB	: Sangat Banyak

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 yang akan dikirimkan ke *module* SSC-32 untuk di proses menjadi gerak servo yang akan menggerakan kaki robot *Hexapod* ini. 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*. Metode COA dipilih dikarenakan metode ini tidak membutuhkan perhitungan yang terlalu rumit sehingga memudahkan dalam pengimplementasian pada sistem *embedded*.



Gambar 3.14 *Flowchart Process Fuzzy Output*



Gambar 3.15 *Flowchart* Proses Eksekusi Data PWM Pada *Module* SSC32

BAB IV
PENGUJIAN DAN PEMBAHASAN SISTEM

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 pengujian Rangkaian *Push Button*, Rangkaian sensor Ultrasonik, Rangkaian Minimum sistem Arduino Mega 2560 , Rangkaian Lampu Indikator *Following* dan pengujian Modul SSC-32 dengan motor servo.
2. Pengujian dan pembahasan Perangkat Lunak (*Software*) yang meliputi pengujian Kontroler *Fuzzy* pada Arduino Mega 2560.

4.1 Pengujian Rangkaian Push Button

Tujuan pengujian pada Rangkaian *Push Button* ini adalah untuk mengetahui tegangan yang akan masuk untuk mentriger modul Arduino Mega 2560 sebagai nilai logik 1 dan 0 dapat bekerja dengan baik maka dilakukan pengujian sebagai berikut.

4.1.1 Peralatan yang digunakan

1. Multimeter Digital
2. Kabel Penghubung
3. Catu daya 5 Volt
4. Arduino Mega 2560
5. Rangkaian *Push Button*

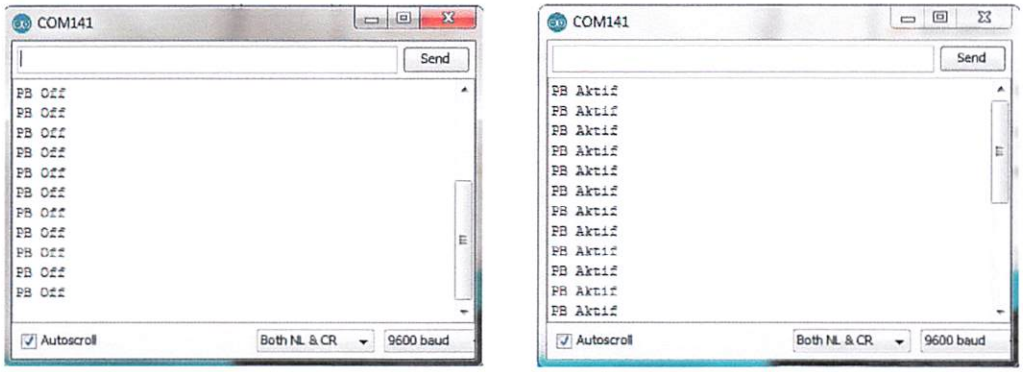
4.1.2 Langkah-langkah yang dilakukan

1. Hubungkan Rangkaian *Push Button* dengan catu daya 5 volt dan pin 2 Arduino Mega 2560 ke bagian *trigger Push Button output*.
2. Arahkan skala multimeter digital pada pengukuran tegangan DC.
3. Hubungkan *probe* positif dari multimeter digital ke masing-masing *output* pin rangkaian *push button* dan *probe* negatif ke pin *ground*.
4. Upload program penguji *push button* pada Arduino Mega 2560.
5. Tampilkan serial monitor untuk memantau keterangan bahwa *push button* Aktif atau Off.
6. Mengukur tegangan dari masing – masing *output push button*.
7. Mencatat hasil pengamatan yang telah dilakukan.

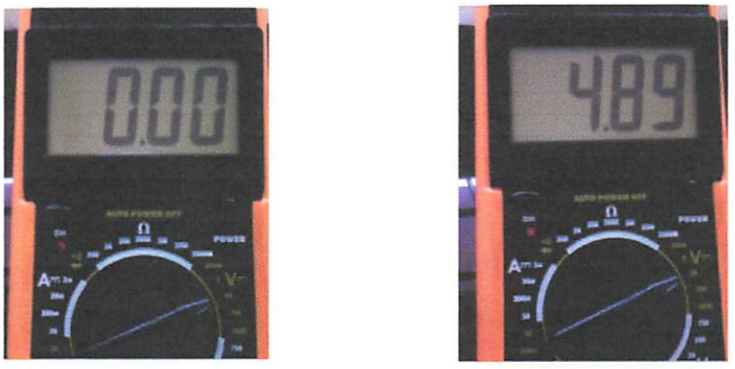
4.1.3 Hasil Pengujian

Tabel 4.1 Hasil Pengujian Tegangan *Output* Rangkaian *Push Button* Dengan Arduino Mega 2560

Phush Button	***Logic* Out put (bit)**	**Tegang an Out put (volt)**	**Keterangan Pengujian Arduino Mega 2560**	***Logic* Out put (bit)**	**Tegang an Out put (volt)**	**Keterangan pengujian Arduino Mega 2560**
Push Button Start	0	0,00	PB Off	1	4,88	PB Aktif
Push Button Wallfollowing Kanan	0	0,00	PB Off	1	4,89	PB Aktif
Push Button Wallfollowing Kiri	0	0,00	PB Off	1	4.89	PB Aktif
Push Button Stop	0	0,00	PB Off	1	4.89	PB Aktif



Gambar 4.1 Pengujian *Push Button* Dengan Arduino Mega 2560 (a) *Low* (b) *High*



(a) (b)

Gambar 4.2 Hasil Pengujian Tegangan *Output Push Button (a)*Keadaan Logika *Low dan (b) Keadaan Logika High*

4.1.4 Analisa Pengujian

Output tegangan dari masing -masing rangkaian *push button* yaitu 4,89 V untuk logika *High* dan tegangan 0 V untuk logika *Low* yang akan di inputkan ke Arduino Mega 2560 sebagai *trigger* nilai inputan pin digital untuk pemicu logika. Ketentuan Arduino dapat menerima tegangan input sebesar 3.3 V - 5 V sebagai *trigger* atau inputan oleh sebab itu hasil pengujian rangkaian *Push Button* ini telah **berhasil 100%**. Karena telah di coba juga pada Arduino Mega 2560 pada saat di inputkan pada pin input digital dan pada saat tombol di tekan pada serial monitor menampilkan PB Aktif ini membuktikan pengujian ini berhasil.

4.2 Pengujian Sensor Ultrasonik SRF HC 04

Pengujian ini bertujuan untuk melihat tingkat presisi dari sensor Ultrasonik ini untuk mengukur jarak terhadap objek yang rata dan tidak rata. dilakukan dengan membandingkan hasil perhitungan dari Arduino Mega 2560 dengan hasil perhitungan alat ukur panjang.

4.2.1 Peralatan yang digunakan

1. Sensor jarak Ultrasonik SRF HC04
2. Alat Ukur Panjang
3. Arduino Mega2560
4. Kabel Data USB
5. Catu daya 5 volt
6. *Software* Arduino Serial Monitor

4.2.2 Langkah-langkah yang dilakukan

1. Hubungkan catu daya 5 volt ke Sensor Ultrasonik.
2. Hubungkan Pin *Trigger* Sensor Ultrasonik ke Pin 28 Arduino Mega 2560.
3. Hubungkan Pin *Echo* Sensor Ultrasonik ke Pin 29 Arduino Mega 2560.
4. Hubungkan kabel data USB ke Arduino Mega 2560.
5. Program Arduino Mega 2560 untuk membaca jarak dengan sensor ultrasonik dan menampilkan data jarak ke Serial monitor.
6. Catat dan bandingkan jarak terukur di Serial Monitor dengan alat ukur.

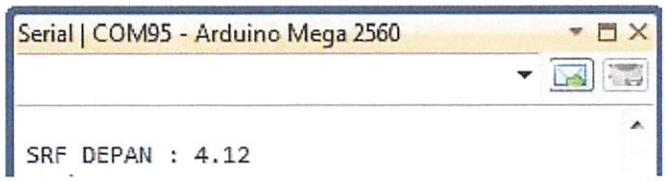
4.2.3 Hasil Pengujian

Tabel 4.2 Hasil Perbandingan Pengujian dan Pengukuran pada Objek Rata dan Objek Tidak Rata

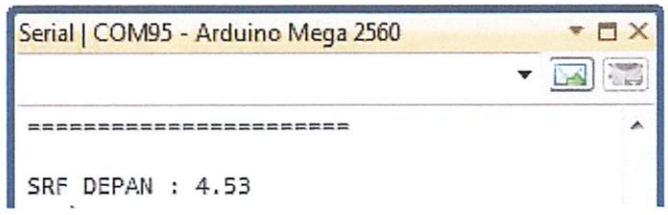
No	Alat Ukur (cm)	Pengujian Pada Objek Rata (cm)	Error %	Pengujian Pada Objek Tidak Rata (cm)	Error %
1	4	4.12	3	4.53	13.25
2	8	8.09	1.13	8.74	9.25
3	12	12.22	1.83	13.38	11.5
4	16	16.28	1.75	17.21	7.56
5	20	20.31	1.55	21.84	9.2
Error Rata-rata			1.5		10.15

(a) (b)

Gambar 4.3 Pengujian Sensor Jarak ke Objek Rata (a) dan ke Objek tidak Rata (b)



Gambar 4.4 Tampilan Serial Monitor Pengujian Sensor Jarak ke Objek Rata



Gambar 4.5 Tampilan Serial Monitor Pengujian Sensor Jarak ke Objek Tidak Rata

4.2.4 Analisa Pengujian

Prinsip pengukuran sensor jarak adalah mengukur pulsa yang diterima pada unit *Echo* sensor ultrasonik SRF HC04. Yang alur kerjanya sebagai berikut Pertama unit *Trigger* akan aktif *low* selama 2 us lalu mengirimkan suara ultrasonik atau *high* selama 10 us dan suara tersebut akan merambat di udara dan ahirnya akan terpantul kembali dan akan diterima oleh unit *Echo* pada sensor SRF HC04 ini dan akan di jadikan nilai pulsa yang akan dihitung oleh Arduino Mega 2560 sesuai dengan persamaan berikut untuk memperoleh nilai jarak dalam cm :

$$Jarak\ (cm) = \frac{(Periode\ pulsa / 29.1)}{2}$$

Dengan menggunakan persamaa diatas, jika pulsa terukur di dapat nilai sebesar 120 µs, maka jarak dapat dicari dengan :

$$Jarak\ (cm) = \frac{235/29.1}{2}$$

$$Jarak\ (cm) = 4{,}0378\ \text{cm}$$

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 sensor ultrasonik terhadap objek rata :

$$\%\ error = \left|\frac{4.12 - 4.00}{4.00}\right| \times 100\%$$

$$\%\ error = 3\ \%$$

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

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

$$\overline{\%Error} = \frac{3 + 1.13 + 1.83 + 1{,}75 + 1.55}{5}$$

$$\overline{\%Error} = 1.5\ \ \%$$

- Jadi kesimpulan dari pengujian sensor ultrasonik SRF HC04 ini terhadap objek rata memiliki Error sebesar 1.5% dan presentase keberhasilan sebesar **98.5%**

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 :

$$\%\ error = \left|\frac{4.53 - 4.00}{4.00}\right| \times 100\%$$

$$\%\ error = 13.25\ \%$$

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

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

$$\overline{\%Error} = \frac{13.25 + 9.25 + 11.5 + 7.56 + 9.2}{5}$$

$$\overline{\%Error} = 10.15\ \ \%$$

- Kesimpulan Error yang terjadi diakibatkan oleh pantulan gelombang suara ultrasonik yang tidak dipantulkan secara sempurna karena permukaan yang tidak rata/ *oval*. Dalam pengujian Sensor Ultrasonik SRF HC04 ini terhadap objek tidak rata memiliki *Error* sebesar 10.15% dan persentase keberhasilan sebesar **89.85%.**

4.3 Pengujian Output Arduino Mega 2560

Tujuan pengujian pada Pin *output* Arduino Mega 2560 adalah agar perangkat lunak yang akan ditanamkan pada mikrokontroler dapat berjalan sesuai dengan yang diharapkan. Pengujian terutama dilakukan untuk menguji berapa besar tegangan yang di*Output*kan dari pin *Output* digital arduino yang dikelurkan .

4.3.1 Peralatan Yang Digunakan

1. Multimeter Digital
2. Kabel Penghubung
3. Catu daya 5 Volt
4. Lampu LED
5. Komputer
6. *Software* Arduino

4.3.2 Langkah-Langkah Yang Dilakukan

1. Hubungkan Arduino Mega 2560 dengan catu daya 5 volt
2. Memprogram Arduino Mega 2560 untuk mengeluarkan logika 0 dan logika 1 pada masing-masing pin.
3. Hubungkan *probe* positif dan rangkaian indikator LED positif ke pin output Arduino Mega 2560 serta negatif dari LED indikator dan *probe* negatif ke pin ground Arduino Mega 2560 .
4. Mengukur tegangan dari masing-masing pin Arduino Mega 2560 dan melihat apakah lampu indikator Led menyala.
5. Mencatat hasil pengamatan yang telah dilakukan.

4.3.3 Hasil Pengujian

Tabel 4.3 Hasil Pengujian Pin *Output* Arduino Mega2560

Pin	*Logic* Output (bit)	Tegangan Output (volt)	LED Indikator
3	1	4,25	Nyala
4	1	4,25	Nyala
5	1	4,24	Nyala
6	1	4,26	Nyala
7	0	0.00	Mati
8	0	0.00	Mati
9	0	0.00	Mati
10	0	0.00	Mati

Gambar 4.6 Hasil Pengujian *Output* Tegangan Pin Digital Arduino Mega 2560 Pada Keadaan Logika *High*

Gambar 4.7 Hasil Pengujian *Output* Tegangan Pin Digital Arduino Mega 2560 Pada Keadaan Logika *Low*

4.3.4 Analisa Pengujian

Pin *Output* Arduino Mega 2560 pada saat diset *High* dapat mengeluarkan *logic* 1 dengan tegangan *output* 4,25 V dan menyalakan lampu Led sedangkan ketika Pin *output* diset *Low* maka nilai tegangan *output 0 V* dan lampu indikator pada kondisi mati. Maka dari kondisi ini dapat disimpulkan *output* Arduino baik.

4.4 Pengujian *Output* Modul SSC-32

Pengujian ini bertujuan untuk mengetahui *output* lebar pulsa PWM pada pin *output* SSC-32 dibandingkan dengan nilai pada *software Visual Squenser*.

4.4.1 Peralatan yang dibutuhkan

1. Module SSC-32
2. Catu daya 5 volt
3. Kabel USB serta DB9 konverter
4. Kabel penghubung
5. Osiloskop
6. Laptop
7. Software Visual Squencer

4.4.2 Langkah-langkah Pengujian

1. Hubungkan hubungkan *module* SSC- 32 dengan tegangan sumber 5 V.
2. Hubungkan *module* SSC-32 ke DB9 konverter dan USB ke komputer.
3. Hubungkan pin 0 *Output* pulsa dari *Module* SSC-32 ke probe positif osiloskop dan pin *ground* (-) ke *probe* Osiloskop *ground.*
4. Buka *software Visual Squencer* yaitu software kusus untuk menjalankan *module* SSC 32 dari komputer atau laptop.
5. Seting baudrate yang digunakan yaitu 115200 bps clik *connect* pada tampilan lalu pilih *squencer* dan centang pada tampilan kontrol servo pin 0 maka akan muncul gui untuk mengatur gerak derajat servo.
6. Geser scrol pada gui ke kanan atau ke kiri untuk mengubah nilai timenya untuk mengubah nilai lebar pulsa yang akan di outputkan pada servo kita geser ke 1500 us sebagai senter servo.

7. Seting osiloskop ke *time/div* menjadi 0,5 ms untuk mendapatkan sekala waktunya.
8. Catat hasil setiap pengujian sesuai pergantian nilai inputan.

4.4.3 Hasil Pengujian

Tabel 4.4 Hasil Pengujian *Output* Lebar Pulsa *Module* SSC 32

No	Nilai Time High yang dikirim dari Gui SSC-32	Nilai Time dari Alat Ukur Osiloscop	Error %
1	500us	500us	0
2	1000us	1000us	0
3	1500us	1500us	0
4	2000us	2000us	0
5	2500us	2500us	0
Rata – Rata Error			0

Gambar 4.8 Pengujian *Output* Lebar Pulsa Module SSC32 Dengan Osiloscop dan Visual Squenser Software

Gambar 4.9 Pengujian *Output* Lebar Pulsa *Module* SSC32 Dengan Osiloscop

4.4.4 Analisa Pengujian

Pada Output Module SSC32 ini didapatkan nilai time pada saat *High* sama dengan nilai input maka dapat dihitung sebagai contoh sebagai berikut:

$$\%\, error = \left| \frac{1500 - 1500}{1500} \right| \times 100\%$$

$$\%\, error = 0\,\%$$

Error rata-rata pengujian output module SSC32 dapat dicari dengan persamaan :

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

$$\overline{\%Error} = \frac{0+0+0+0+0}{5}$$

$$\overline{\%Error} = 0\,\%$$

- Kesimpulan dari pengujian *output* dari *module* SSC 32 ini yaitu dapat memberi nilai *output* pulse *high* (lebar pulsa *High*) sesuai kontrol yang diberikan dan memiliki nilai *Error* sebesar 0% dan persentase keberhasilan **100%**

4.5 Pengujian Kontroler *Fuzzy*

Sistem pengujian kontroler *fuzzy* disini dengan membandingkan nilai defuzifikasi yang dikeluarkan dari software matlab dan *output* nilai defuzifikasi dari Arduino Mega 2560. Dengan memberikan nilai input yang sama dan kita lihat hasilnya.

4.5.1 Peralatan yang dibutuhkan

1. Aplikasi *Fuzzy Logic* Matlab
2. Arduino Mega 2560
3. Sumber tegangan 5 volt
4. Kabel USB
5. Komputer
6. Sensor SRF HC04

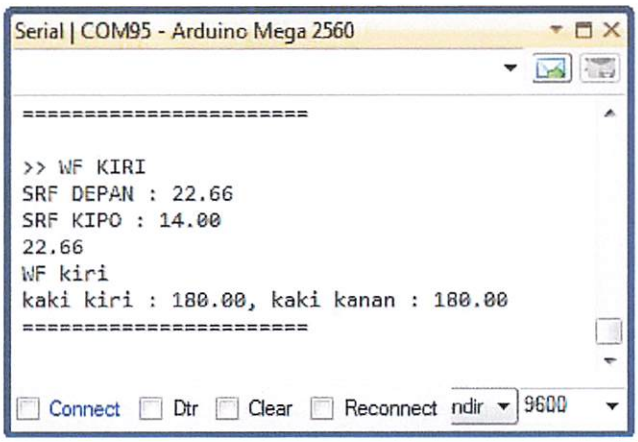
4.5.2 Langkah-langkah Pengujian

1. Buka *Software Matlab fuzzy,* buat bagan *input output* ,pilih jenis metode membership yang segitiga untuk *input*, trapesium untuk *output* ,setelah itu isi nilai membersip inputan dan *output fuzzy* kemudian buat *rule*, masukan nilai jarak untuk pengujian yang nilainya sama dengan hasil sensor pada arduino secara manual pada *Fuzzy Matlab*, *running* dan catat hasilnya.
2. Hubungkan tegangan sumber 5 volt ke Ultrasonik SRF HC04 dan Arduino Mega 2560.
3. Hubungkan pin *trigger* dan *echo* sensor Ultrasonik SRF HC04 ke Pin Arduino Mega isikan Program *Fuzzy* yang telah dibuat dan monitoring lewat Serial monitor.
4. Catat hasil pengujian menurut inputan *fuzzy* dan bandingkan hasil defuzyfikasinya.

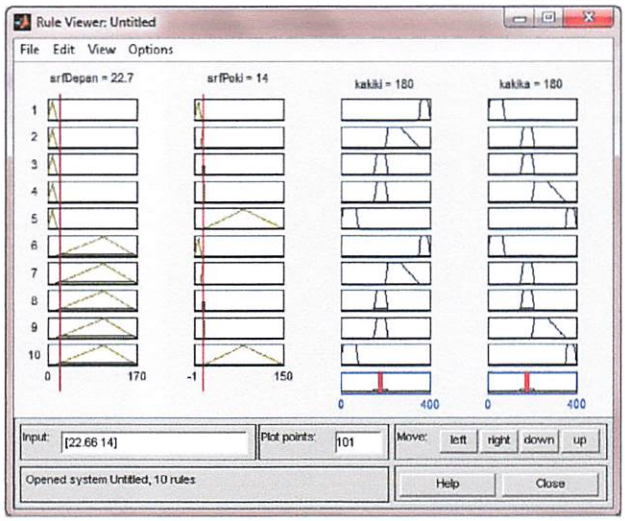
4.5.3 Hasil Pengujian

Tabel 4.5 Hasil Pengujian *Output Fuzzy* Untuk Penelusur Dinding Kiri

No	Input		*Output Fuzzy*				Error %	
	Jarak		Pengujian *Fuzzy* Aduino Mega 2560		*Fuzzy Logic* Matlab			
	Depan	Pojok Kiri	Kaki Kiri	Kaki Kanan	Kaki Kiri	Kaki Kanan	Kaki Kiri	Kaki Kanan
1	22.66	10.53	375	40	374	40	0.27	0
2	22.78	13.88	265	180	274	180	3.28	0
3	22.66	14.00	180	180	180	180	0	0
4	22.66	15.28	180	265	180	274	0	3.28
5	22.66	17.41	40	375	40	374	0	0.27
Error Rata-rata							0.71	0.71



Gambar 4.10 Hasil Pengujian *Output Fuzzy* Pada Arduino Mega2560



Gambar 4.11 Hasil *Rule Viewer* Pada *Fuzzy Logic* Matlab

4.5.4 Analisa Hasil Pengujian

Untuk mengetahui nilai *error* rata-rata pengujian *output* nilai *fuzzy* dari perhitungan matlab dan *output fuzzy* dari Arduino Mega 2560 dapat dicari dengan persamaan yang sama yaitu:

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

Dari hasil pengujian :

$$\%\ error = \left|\frac{375-374}{374}\right| \times 100\%$$

$$\%\ error = 0.27\ \%$$

Error rata-rata pengujian fuzzy output antara perhitungan Matlab dan dari hasil Arduino Mega 2560 dapat dicari dengan persamaan :

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

$$\overline{\%Error} = \frac{0.27+3.28+0+0+0}{5}$$

$$\overline{\%Error} = 0.71\ \ \%$$

- Dari hasil pengujian ini dapat disimpulkan penerapan *fuzzy* pada Arduino Mega 2560 ini berhasil karena beda *output* dari perhitungan Matlab dengan *output fuzzy* dari Arduino Mega 2560 ini tidak terlalu jauh hasil outputnya dengan nilai **Error 0.71%** dan indikasi **Keberhasilan** sebesar **99.29%**

4.6 Pengujian Motor Servo

Pengujian ini bertujuan untuk membandingkan kepresisian perubahan derajat putar motor servo dengan derajat putar alat ukur.

4.6.1 Peralatan yang digunakan

1. Motor Servo HS645 mg
2. Alat Ukur derajat (busur derajat)
3. Arduino Mega2560
4. Module SSC32
5. Kabel Data USB
6. Catu daya 5 volt
7. Software Visual Squenser

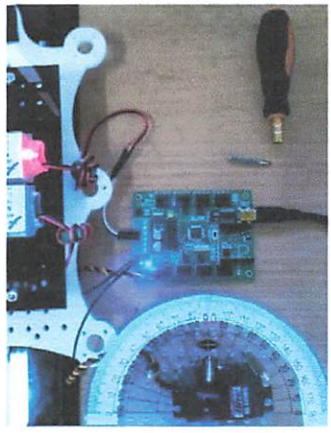
4.6.2 Langkah-langkah yang dilakukan

1. Hubungkan catu daya 5 Volt ke *module* SSC 32.
2. Hubungkan kabel USB ke laptop dan DB 9 konverter ke module SSC32.
3. Hubungkan pin Motor servo ke pin *output module* SSC 32 (pin 0) pulse pada bagian pulse dan bagian catu daya servo ke catu daya 5 volt.
4. Buka software visual squenser *set baudrate* ke 115200 bps lalu click *connect* dan pilih squenser lalu centang 0 maka akan keluar gui untuk mengubah nilai derajat servo (Pin 0).
5. Geser skrol pada gui tersebut sesuai derajat yang di tentukan .
6. Amati dan catat hasil dari pengujian.

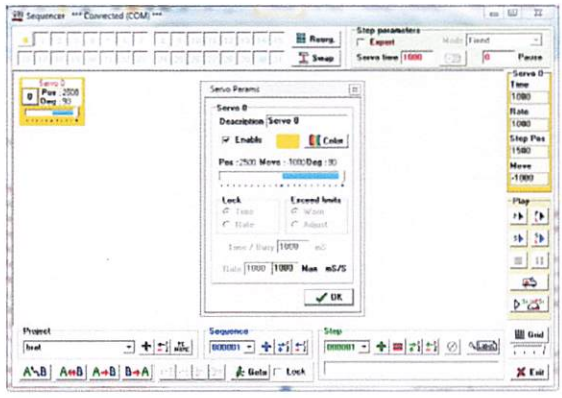
4.6.3 Hasil Pengujian

Tabel 4.6 Hasil Perbandingan Pengujian dan Pengukuran pada Motor Servo

No	Besar Sudut Putaran Motor Servo		Error %
	Alat Ukur (derajat)	Pengujian (derajat)	
1	-90	-90,2	0.2
2	-45	-45,1	0.2
3	0	0	0
4	45	45,2	0.4
5	90	90,2	0.2
Rata – Rata Error			0.36



Gambar 4.12 Pengujian Motor Servo Terhadap Derajat Posisi



Gambar 4.13 *Shoftware* Visual Squenser

4.6.4 Analisa Pengujian

Prinsip pengukuran derajat motor servo ini bila di gui visual squenser kita tentukan derajatnya maka *output* dari servo menunjuk pada derajat berapa pada alat ukur tersebut dan akan dibandingkan.

Sedangkan nilai *error* dapat dihitung dengan persamaan :

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

Dari hasil pengujian :

$$\%\ error = \left| \frac{-90.2 - (-90)}{(-90)} \right| \times 100\%$$

$$\%\ error = 0.2\ \%$$

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

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

$$\overline{\%Error} = \frac{0.2 + 0.2 + 0 + 0.4 + 0.2}{5}$$

$$\overline{\%Error} = 0.36 \quad \%$$

- Kesimpulan dari pengujian perbedaan derajat perputaran motor servo pada alat ukur dengan nilai kontrol yang diberikan dari Software Visual Squenser tidak terlalu besar yaitu persentase Errornya 0.36% dan persentase keberhasilanya yaitu **99.64%** maka pengujian ini dianggap berhasil.

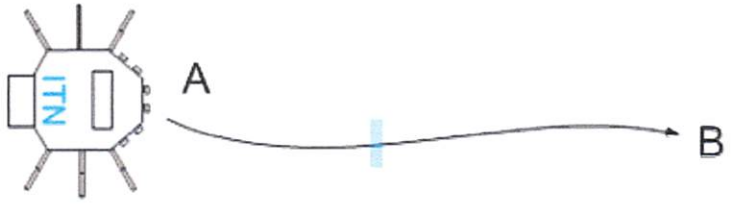
4.7 Pengujian Sistem Keseluruhan

Pengujian dilakukan dengan menguji kemampuan robot untuk mengikuti dinding kiri atau dinding kanan serta kemampuan untuk menghindar halangan. 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 berjalan sebaik mungkin. Pengujian telusur dinding menggunakan mode telusur kanan saja karena telusur kiri atau kanan pada dasarnya adalah sama.

4.7.1 Langkah Pengujian

4.7.1.1 Pengujian pada Lintasan Lurus

Lintasan pengujian menggunakan *track* lurus dengan panjang 100 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

Tabel 4.7 Hasil Pengujian Pergerakan Robot Pada Lintasan Lurus

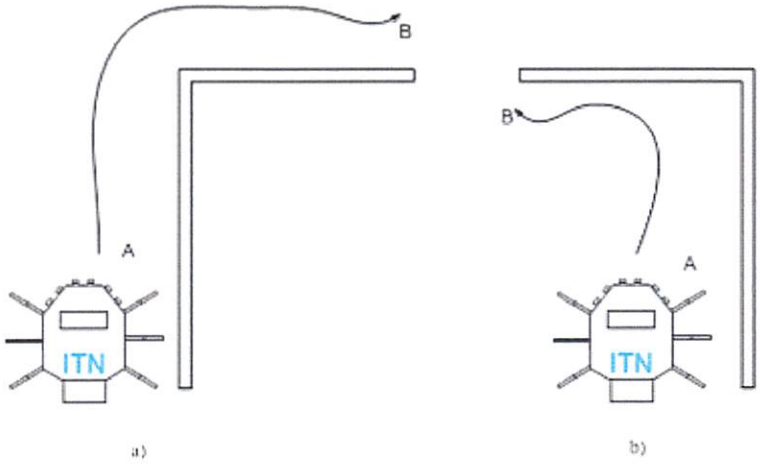
Percobaan	1	2	3	4	5	6	7	8	9	10
Hasil	√	√	√	√	√	√	√	√	√	√

Keterangan : √ = Berhasil X = Menyentuh Dinding

- Pada pengujian gerak robot pada lintasan lurus ini robot tidak menyentuh dinding lintasan satu kalipun dalam 10 kali percobaan oleh karena itu dapat disimpulkan bahwa pada lintasan lurus ini persentase keberhasilan sebesar **100 %** karena robot dapat berjalan dan bergerak dengan pergerakan *smooth* .

4.7.1.2 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 4.15 Pengujian Belokan 90^0

Gambar 4.15 (a) adalah lintasan dengan sudut haluan 90^0 dengan posisi robot berada pada posisi luar sudut sedangkan pada gambar 4.15 (b) lintasan dengan sudut 90^0 dengan posisi robot berada di dalam sudut.

Tabel 4.8 Hasil Pengujian Pergerakan Robot Pada belokan 90°

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

Keterangan :

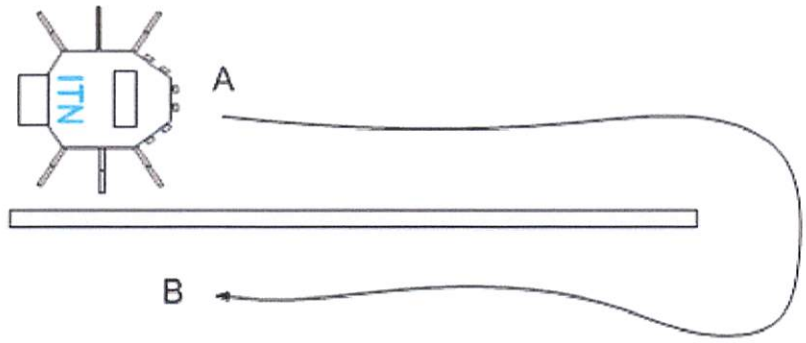
√ = Berhasil

X = Menyentuh Dinding

- Pada pengujian belok 90° ini robot dapat bergerak secara *smooth* (halus) tanpa suatu kendala dan berhasil bergerak dan berbelok dengan persentase keberhasilan sebesar **100%** pada 10 kali percobaan.

4.7.1.3 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 4.16 Pengujian Belokan 180^0

Tabel 4.9 Hasil Pengujian Pergerakan Robot Pada Belokan 180°

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

Keterangan :

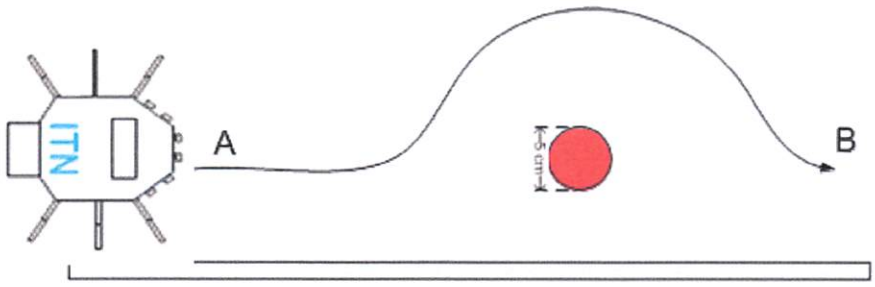
√ = Berhasil

X = Menyentuh Dinding

- Pada pengujian pergerakan robot pada belok 180° ini robot melakukan kesalahan sebanyak 2 kali karena robot bergesekan dengan dinding, dengan ini dapat disimpulkan bahwa pada percobaan ini persentase keberhasilan robot dalam berbelok 180° yaitu **80%**.

4.7.1.4 Pengujian Penghindar Halangan

Pengujian menggunakan penghalang berbentuk tabung dengan diameter 5 cm. 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 4.17 Pengujian *Obstacle Avoidance* (Penghindar Halangan)

Tabel 4.10 Hasil Pengujian Pergerakan Robot Untuk Penghindar Halangan

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

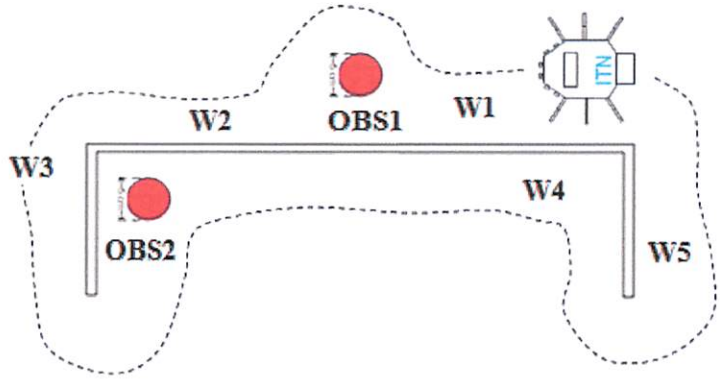
Keterangan :

√ = Berhasil X = Menyentuh Dinding / Halangan

- Pada pengujian halangan terdapat 2 kali menabrak halangan dari 10 kali percobaan ini disebabkan karena sensor ultrasonik terganggu oleh bidang pantul yang tidak rata. Jadi pada percobaan jalan robot dengan adanya halangan tingkat keberhasilanya sebesar **80%**.

4.7.1.5 Pengujian Telusur Dinding dan Penghindar Halangan Dengan Menggunakan Metode Logika *Fuzzy*

Pengujian ini dilakukan untuk melihat kemampuan robot dalam menelusur dinding dengan prosedur pengujian bila menggunakan logika fuzzy pada robot adalah robot bergerak dari posisi *start* menuju *finish* dengan indikator keberhasilan adalah presentase menyentuh dinding, menabrak halangan sekecil mungkin dan berapa waktu yang dibutuhkan untuk mencapai finish.



Gambar 4.18 Pengujian Telusur Dinding dan Penghindar Halangan Dengan *Fuzzy*

Tabel 4.11 Hasil Sistem Pergerakan Robot Keseluruhan Dengan *Fuzzy Logic*

Percobaan	W1	W2	W3	W4	W5	OBS1	OBS2	Waktu
1	√	√	√	√	√	√	√	0:2:10
2	√	√	√	√	√	X	√	0:2:11
3	√	X	X	√	√	X	√	0:2:14
4	√	√	X	√	X	√	X	0:2:15
5	√	√	√	√	√	√	√	0:2:09

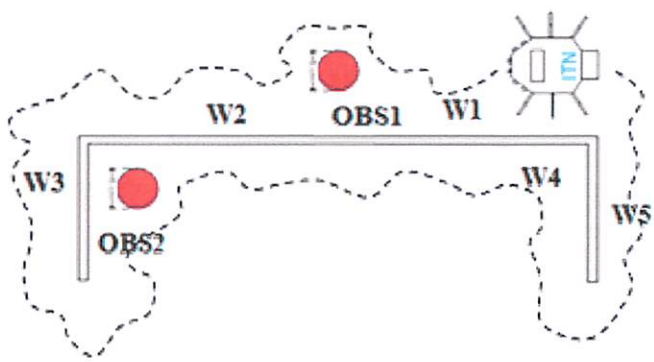
Keterangan :

√ = Berhasil

X = Menyentuh Dinding / Halangan

4.7.1.6 Pengujian Pembanding Telusur Dinding dan Penghindar Halangan Tanpa Menggunakan Metode Logika *Fuzzy*

Pengujian ini dilakukan untuk melihat kemampuan robot dalam menelusur dinding dengan prosedur pengujian bila tidak menggunakan logika *fuzzy* pada robot adalah robot bergerak dari posisi *start* menuju *finish* dengan indikator keberhasilan adalah presentase menyentuh dinding, menabrak halangan sekecil mungkin dan berapa waktu yang dibutuhkan untuk mencapai finish.



Gambar 4.19 Pengujian Telusur Dinding dan Penghindar Halangan Tanpa *Fuzzy*

Tabel 4.12 Hasil Sistem Pergerakan Robot Keseluruhan Tanpa *Fuzzy Logic*

Percobaan	W1	W2	W3	W4	W5	OBS1	OBS2	Waktu
1	X	√	X	√	X	X	X	0:4:35
2	√	X	X	X	X	X	√	0:4:38
3	X	X	X	√	X	X	X	0:5:14
4	X	√	X	√	X	X	X	0:4:40
5	X	√	X	X	X	X	X	0:5:24

Keterangan :

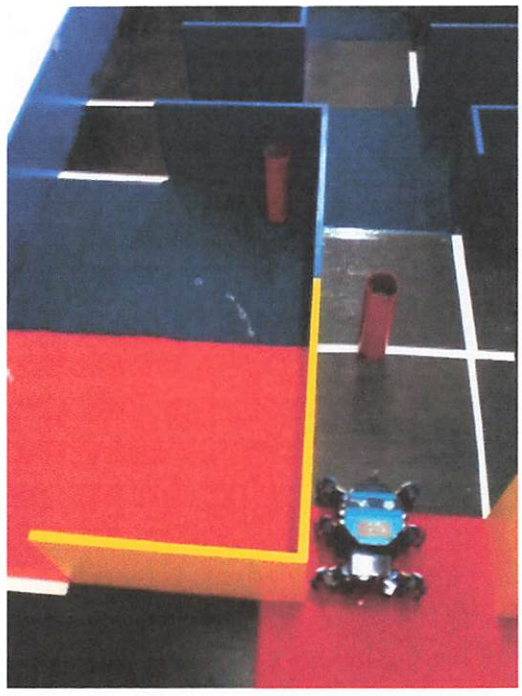
√ = Berhasil

X = Menyentuh Dinding / Halangan

4.7.2 Analisa Hasil Pengujian

Pada pengujian berjalan pada lintasan lurus dan belok 90° robot dapat berhasil 100% dan dapat berjalan dengan smooth atau halus pada lintasan, sedangkan pada belok 180° masih ada *error* dan menyentuh dinding lintasan labirin karena terlalu melebar saat berbelok ini disebabkan pijakan kaki yang kurang kesat terhadap lintasan.

Pada saat pengujian terhadap halangan masih juga terdapat *error* atau menabrak halangan ini disebabkan bentuk dari halangan yang berbentuk lingkaran sehingga sonar ultrasonik yang dipancarkan tidak terpantul dengan baik sehingga robot menjadi terlambat untuk menghindarinya. Dari hasil pengujian pada lintasan keseluruhan dengan menggunakan metode *Fuzzy Logic* yang paling sering terjadi *error* yaitu pada belok 180° dan pada saat menghindar halangan dengan hasil waktu tercepat untuk menempuh dari titik *start* ke *finish* 0:2:09 dan jika dibandingkan dengan pergerakan robot tanpa metode *Fuzzy Logic* waktu tempuhnya 0:4:35 ini disebabkan robot sering menabrak dinding dan terjebak pada belokan, maka pergerakan robot dengan menggunakan metode *Fuzzy Logic* yang di tanam pada robot dua kali lebih cepat dalam melakukan pergerakan untuk menyusuri dinding lintasan dengan perhitungan persentase keberhasilan dari pengujian gerak robot dengan lintasan keseluruhan dapat dilihat pada perhitungan berikut.



Gambar 4.20 Pengujian Pada Lintasan

Berdasarkan presentase keberhasilan pergerakan mengikuti dinding dan menghindar halangan sebanyak 45 kali percobaan dapat dihitung dengan persamaan :

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

$$\%\ Keberhasilan = \frac{38}{45} x\ 100\%$$

$$\%\ Keberhasilan = 84\%$$

Dengan demikian dapat disimpulkan bahwa pengujian sistem secara keseluruhan telah berhasil **84%** dalam berjalan menyusuri dinding labirin dan juga menghindari halangan dengan bergerak secara *smooth* (halus). Ada juga beberapa perbaikan yang diperlukan untuk memperbaiki performa *system* agar robot ini bergerak lebih baik lagi.

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. Dari hasil uji coba Push Button yang dibuat baik karena dapat mentrigger Arduino Mega 2560 dengan mengeluarkan logik 1 aktif dengan tegangan 4.98volt dan logik 0 dengan 0 volt.
2. Untuk pengujian Sensor Ultrasonik SRF HC04 pada bidang datar baik dengan error 1,5% dan pada bidang tidak rata memiliki *error* sebesar 10,15% ini disebabkan suara ultrasonik yang di pancarkan tidak terpantul dengan baik oleh bahan tidak rata.
3. Pada pengujian output hasil nilai *fuzzy* dari perhitungan *Software* Matlab dan *output* dari Arduino Mega 2560 tidak terlalu berbeda dengan *Error* nilai 0,71% ini yang mengindikasikan bahwa penerapan kontrol *fuzzy* pada Arduino Mega 2560 berhasil dengan persentase 99,29% .
4. Pada pengujian motor servo HS645 MG ini untuk mengetahui pergerakan sudutnya ternyata sesuai dengan yang diharapkan dan dapat digunakan sebagai aktuator kaki robot *Hexapod* ini dengan Error 0,36%.
5. Posisi Pada pengujian pergerakan robot pada lintasan lurus dan belok 90^0 robot dapat berjalan secara *Smooth* 100% pada lintasan belok 180^0 Robot masih mengalami kendala pada pijakan kaki yang kurang kesat sehingga pada saat berbelok sering kali melebar. Sedangkan pada pengujiah penghindar halangan robot masih menenyentuh hal ini disebabkan karna bentuk dari halangan yang tidak rata sehingga pantulan suara ultrasonik yang dipancarkan sensor tidak terpantul dengan sempurna ini yang menyebabkan robot terlambat menghindar.

6. Pada hasil pengujian keseluruhan maka dapat disimpulkan bahwa robot *hexapod* ini dapat menyusuri dinding lintasan labirin dan menghindar halangan dengan *smooth* (halus) telah berhasil **84%.**
7. Pada hasil pengujian perbandingan antara pergerakan robot menggunakan metode *Fuzzy Logic* yang ditanam pada robot bergerak lebih cepat dan halus dengan waktu 0:2:09 sedangkan pergerakan robot tanpa menggunakan metode *Fuzzy Logic* mendapat catatan waktu tempuh 0:4:35 dan pergerakan robotnya sangat kasar maka dengan hasil pengujian perbandingan tersebut robot hexapod yang ditanami metode Logika Fuzzy dua kali lebih cepat dan halus.
8. Setelah melakukan pengujian robot *hexapod* ini motor Servo HS 645 MG kurang handal dalam melakukan pergerakan yang halus tapi cepat ini karena dalam konstruksi mekanik servo ini tidak semua gear terbuat dari metal gear ada satu gear yang terbuat dari bahan kuningan ini yang menyebabkan servo cepat aus atau rusak bila digunakan dengan pergerakan yang cepat.

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 untuk menghindar halangan maka diperlukan sensor dengan penjejak object lebih cepat seperti camera.
2. Untuk meningkatkan pergerakan kaki robot yang lebih kuat dan lebih cepat lagi maka dibutuhkan Motor servo yang lebih handal seperti motor servo AX 12
3. Untuk mendapatkan nilai *rule fuzzy* yang paling optimal maka pada pengembangan selanjutnya dapat digunakan metode *neuro fuzzy*.
4. Untuk mengurangi selip pada saat robot berjalan maka gunakan kaki robot yang lunak supaya kaki robot dapat mencengkram lintasan dengan baik.

DAFTAR PUSTAKA

[1] Daris, *Perancangan dan Pembuatan Sistem Gerak Robot Laba-Laba (Hexapod) Menggunakan Lynxmotion Visual Squencer Dan Arduino Mega 2560*. Malang: ITN Malang, 2013.

[2] Son Kuswadi, *Kendali Cerdas Teori dan Aplikasinya*. Yogyakarta: Andi, 2007.

[3] Eric Kosmas, *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*. Malang: ITN Malang, 2014.

[4] Datasheet SRF 04. [Online]. inside.mines.edu/~whoff/./SRF04%20Technical%20Documentation.pdf, diakses tanggal 5/3/2015

[5] Eko Juzi Istiyanto, *Pengantar Elektronika dan Instrumentasi (Pendekatan Project Arduino dan Android)*, Andi, Ed. Yogyakarta: Arieta, 2013.

[6] Atmel 2560. [Online]. http://www.atmel.com/devices/atmega2560.aspx

[7] Lynxmotion Visual Sequencer V1.16 Manual. [Online]. http://www.lynxmotion.com/p-395-ssc-32-servo-controller.aspx, diakses tanggal 5/3/2015

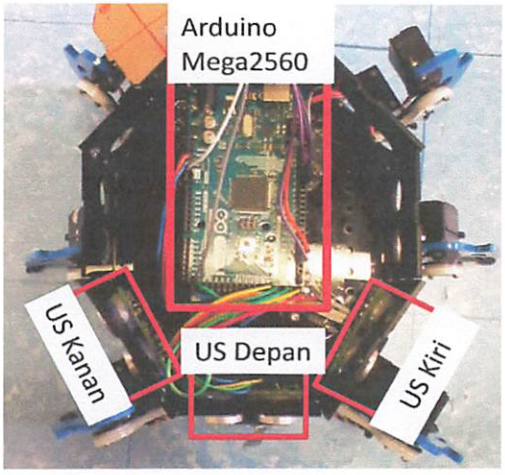
[8] Datasheet Hitec Servo HS-645MG. [Online]. http://www.servocity.com/html/hs 645mg_ultra_torque.html, diakses tanggal 5/3/2015

[9] Library EFLL. [Online]. https://github.com/zerokol/eFLL, diakses tanggal 8/5/2015

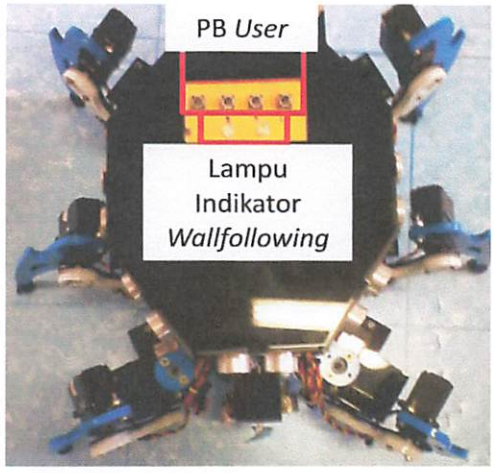
[10] Pungky Eka Sasmita, Dr.Tri Arief Sardjono, ST. MT., Ir. Harris Pirngadi, MT. Kontrol Penjejak Pada Robot Pemadam Api Menggunakan Sistem Pengindera Api Dan Posisi Jarak Dengan Metode Fuzzy Logic, *digilib.its.ac.id/.../ITS-Undergraduate-18346-Paperpdf,* diakses tanggal 5/3/2015

LAMPIRAN

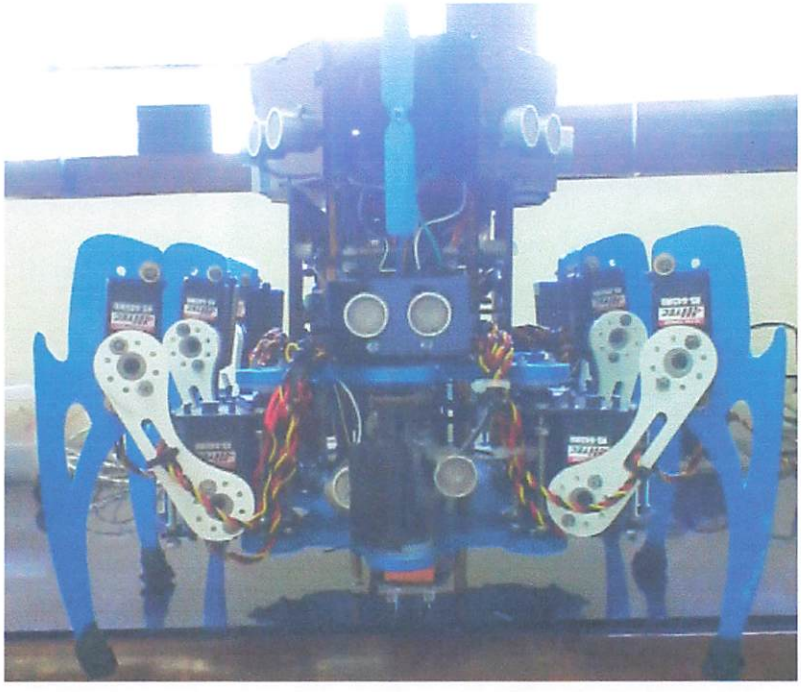
GAMBAR ROBOT HEXAPOD



Robot Hexapod Tampak Atas
Tanpa Tutup



Robot Hexapod Tampak Atas
Dengan Tutup



Robot Hexapod Tampak Depan

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang Bertanda Tangan Dibawah Ini:

Nama : Tito Dinti Koshars
NIM : 111221
Semester : 8 (delapan)
Fakultas : Teknologi Industri
Jurusan : Teknik Elektro S-I
Konsentrasi : **TEKNIK ENERGI LISTRIK**
TEKNIK ELEKTRONIKA
TEKNIK KOMPUTER DAN INFORMATIKA
TEKNIK KOMPUTER
TEKNIK TELEKOMUNIKASI
Alamat : Jl. Jend. A. Yani 215 RT 30 RW 4 Desa. Karanglates Sumberpucung Malang

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat SKRIPSI Tingkat Sarjana. Untuk melengkapi permohonan tersenut, bersama ini kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan- persyaratan pengambilan SKRIPSI adalah sebagai berikut:

1. Telah melaksanakan semua praktikum sesui dengan konsentrasinya (...)
2. Telah lulus dan menyerahkan laporan Praktek Kerja (...)
3. Telah lulus seluruh mata kuliah keahlian (MKB)sesuai konsentrasinya (...)
4. Telah menempuh matakuliah> 134 sks dengan IPK > 2 dan tidak ada nilai E (...)
5. Telah mengikuti secara aktif kegiatan seminar Skripsi yang diadakan Jurusan (...)
6. Memenuhi persyaratan administrasi (...)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenarannya data tersebut diatas
Recording Teknik Elektro S-I

(Joni Handayani)

Malang,.....201
Pemohon

(Tito Dinti Koshars)

Disetujui
Ketua Prodi Teknik Elektro S-I

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

Mengetahui
Dosen Wali

(Irmalia Suryani F, ST, MT)

BERITA ACARA RAPAT PERSETUJUAN JUDUL/PROPOSAL SKRIPSI

PROGRAM STUDI TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Elektronika S1

Tanggal :

1.	NIM	1112211
2.	Nama	Tito Dinti Koshars
3.	Judul yang diajukan	Pengembangan Drive Hexapod Robot utk penjejak dinding & penghindar halangan dg navigasi sensor ultrasonik. menggunakan met kendali logika FUZZY.
4.	**Disetujui/~~Ditolak~~***	
5.	Catatan:	
6.	Pembimbing yang diusulkan: 1. Ibrahim Ashari ✓ 2. Irmalia SF ✓	

Menyetujui

1. Koordinator Dosen Kelompok Keahlian

2. Dosen Kelompok Keahlian (Terlampir)

* : Coret yang tidak perlu

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Telp. (0341) 417636 Malang

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S-1
Konsentrasi : Teknik Elektronika

1.	Nim	: **1112211**	
2.	Nama	: **TITO DINTI KOSHARS**	
3.	*Konsentrasi Jurusan*	: ***Teknik Elektronika***	
4.	Jadwal Pelaksanaan:	Waktu	Tempat
	09 April 2015	09.00	III.1.5
5.	Judul proposal yang diseminarkan Mahasiswa	PENGEMBANGAN DRIVE HEXAPOD ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA FUZZY	
6.	Perubahan judul yang diusulkan oleh Kelompok Dosen Keahlian		
7.	Catatan : - Pengujian sistem Fuzzy -		
8.	Catatan :		
	Persetujuan judul Skripsi		
	Disetujui, Dosen Keahlian I (.....)	Disetujui, Dosen Keahlian II (.....)	Disetujui, Dosen Keahlian III (.....)
	Mengetahui, Ketua Program Studi Teknik Elektro S-1 **M. Ibrahim Ashari ,ST, MT** NIP. P 1030100358	Disetujui, Calon Dosen Pembimbing ybs Pembimbing I (.....)	Pembimbing II (.....)

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

BERITA ACARA SEMINAR PROGRESS SKRIPSI
PROGRAM STUDI TEKNIK ELEKTRO S1

KONSENTRASI	Elektronika

1.	Nama Mahasiswa	TITO DINTI KOSHARS		NIM	1112211
2.	Keterangan	Tanggal	Waktu		Tempat / Ruang
	Pelaksanaan				
3.	Judul Skripsi	PENGEMBANGAN DRIVE HEXAPOD ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA FUZZY			
4.	Perubahan Judul	.....			
5.	Catatan : Rangkaian Interpolasi Video				
6.	Mengetahui, Ketua Jurusan. M. Ibrahim Ashari, ST, MT	Disetujui, Dosen Pembimbing			
		Pembimbing I M. Ibrahim Ashari, ST, MT	Pembimbing II Irmalia Suryani Faradisa, ST, MT		

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK 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

Nomor Surat : ITN-183/EL-FTI/2015 Tanggal, 28 Mei 2015
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Bapak/Ibu **M. Ibrahim Ashari, ST, MT**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **TITO DINTI KOSHARS**
Nim	: **1112211**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Elektronika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

"Semester Genap Tahun Akademik Genap 2014 - 2015"

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

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

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG

INSTITUT TEKNOLOGI NASIONAL MALANG

FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK

PT. BNI (PERSERO) MALANG
BANK 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

Nomor Surat : ITN-183/EL-FTI/2015 Tanggal, 28 Mei 2015
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Bapak/Ibu **Irmalia Suryani Faradisa, ST, MT**
Dosen Teknik Elektro S-1
ITN MALANG

Dengan Hormat

Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi untuk mahasiswa :

Nama	: **TITO DINTI KOSHARS**
Nim	: **1112211**
Fakultas	: **Teknologi Industri**
Program Studi	: **Teknik Elektro S-1**
Konsentrasi	: **Teknik Elektronika**

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/i selama masa waktu :

"Semester Genap Tahun Akademik Genap 2014 - 2015"

Demikian agar maklum dan atas perhatian serta bantuannya kami sampaikan terima kasih.

Mengetahui
Ketua Program Studi Teknik Elektro S-1

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

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI

SEMESTER GENAP TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : Tito Dinti Koshars
NIM : 1112211
Nama Pembimbing : M.Ibrahim Ashari,ST, MT
Judul Skripsi : PENGEMBANGAN DRIVE *HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
1	22 mei 2015	10.00 - 10.15	acc Bab I dan II	
2	3 juni 2015	10.00 10.15	Revisi bab II	
3	3 juni 2015	13.30 13.45	acc Bab III	
4	6 juli 2015	10.00 - 10.45	Seminar Progres	
5	7 juli 2015	11.00 11.15	acc Bab IV	
6	9 juli 2015	13.10 13.20	acc Bab V	
7	10 juli 2015	13.00 13.15	acc makalah seminar	

Malang, 28 April 2015
Pembimbing

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

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI

SEMESTER GENAP TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : Tito Dinti Koshars
NIM : 1112211
Nama Pembimbing : M.Ibrahim Ashari,ST, MT
Judul Skripsi : PENGEMBANGAN DRIVE *HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
1	1 agust 2015	13.00 13.15	acc laporan skripsi	
2				
3				
4				
5				
6				
7				

Malang, Juli 2015
Pembimbing

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

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Malang

MONITORING BIMBINGAN SKRIPSI

SEMESTER GANJIL TAHUN AKADEMIK 2014-2015

Nama Mahasiswa : Tito Dinti Koshars
NIM : 1112211
Nama Pembimbing : Irmalia Suryani Faradisa,ST, MT
Judul Skripsi : PENGEMBANGAN DRIVE *HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY*

Minggu Ke-	Hari, Tanggal	Waktu Bimbingan	Materi Bimbingan	Paraf
1	13-5-15		- Perancangan fuzzy rule - Penulisan Bab 3	
2	26-5-15		- flowchart fuzzy - Keterangan dan Block diagram	
3	4-6-15		Acc bab 3	
4	3-7-15		Revisi bab 4	
5	7-7-15		Acc Bab 4	
6	9 Juli 2015		revisi bab 5 Acc sidang	
7	1 agustus 2015		Acc. ujian kompre	

Malang, 28 April 2015

Pembimbing

Irmalia Faradisa Suryani, ST,MT
NIP.P. 10300003675

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

Formulir Perbaikan Ujian Skripsi

Dalam Pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T.Energi Listrik,/ T. Elektronika, /T. Komputer, / T.Telekomunikasi, Maka Perlu Adanya Perbaikan Skripsi Untuk Mahasiswa:

Nama : Tito Rinti
NIM : 11122111
Perbaikan Meliputi :

- Ditambahkan pengujian dengan yg lama (tanpa fuzzy)
- Pengujian dg MATLAB dijelaskan lebih detail.
- Penulisan kutipan & daftar pustaka

Malang, 19/8/2015

(.....)
Anggraini

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

Formulir Perbaikan Ujian Skripsi

Dalam Pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T.Energi Listrik,/ T. Elektronika, /T. Komputer, / T.Telekomunikasi, Maka Perlu Adanya Perbaikan Skripsi Untuk Mahasiswa:

Nama : TITO DWIPUTRA K.
NIM : 1112211
Perbaikan Meliputi :

(1) Penulisan daftar Pustaka harus mengacu Buku (HARDWARE) atau (IEEE) → urut abjadi atau penomoran
Lihat kutipan yg diambil dari Buku Referensi wajib di tulis Pustaka.

Malang, 19 - 8 - 2015

(Dr. F. Edi, ST, MT)

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

PERSETUJUAN PERBAIKAN SKRIPSI

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

Hari : Rabu
Tanggal : 19 Agustus 2015

Telah dilakukan perbaikan skripsi oleh :

Nama : Tito Dinti Koshars
NIM : 1112211
Perogram Studi : Teknik Elektro S-1
Komsentrasi : Teknik Elektronika
Judul Skripsi : **Pengembangan *Drive Hexapod* Robot Untuk Penjejak Dinding Dan Penghindar Halangan Dengan Navigasi Sensor Ultrasonik Menggunakan Metode Kendali Logika *Fuzzy***

No	Materi Perbaikan	Keterangan
1.	Ditambahkan pengujian tanpa *Fuzzy*	
2.	Pengujian dengan Matlab dijelaskan nilai inputnya	
3.	Penulisan kutipan dan daftar pustaka	

Dosen Penguji I

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

Dosen Pembimbing I

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

Dosen Pembimbing II

Irmalia Suryani Faradisa,ST,MT
NIP.P.1030000365

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

PERSETUJUAN PERBAIKAN SKRIPSI

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

Hari : Rabu
Tanggal : 19 Agustus 2015

Telah dilakukan perbaikan skripsi oleh :

Nama : Tito Dinti Koshars
NIM : 1112211
Perogram Studi : Teknik Elektro S-1
Komsentrasi : Teknik Elektronika
Judul Skripsi : **Pengembangan *Drive Hexapod* Robot Untuk Penjejak Dinding Dan Penghindar Halangan Dengan Navigasi Sensor Ultrasonik Menggunakan Metode Kendali Logika *Fuzzy***

No	Materi Perbaikan	Keterangan
1.	Penulisan daftar pustaka dan kutipan harus sesuai IEEE	

Dosen Penguji II

Dr. F. Yudi Limpraptono,ST,MT
NIP.Y. 1039500274

Dosen Pembimbing I

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

Dosen Pembimbing II

Irmalia Suryani Faradisa,ST,MT
NIP.P.1030000365

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama : Tito Dinti Koshars

NIM : 1112211

Program Studi : Teknik 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, 10 September 2015

Yang membuat Pernyataan,

Tito Dinti Koshars

NIM : 1112211

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK 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

1. Nama : Tito Dinti Koshars
2. NIM : 1112211
3. Program Studi : TEKNIK ELEKTRO S-1
4. Konsentrasi : TEKNIK ELEKTRONIKA

Judul : **PENGEMBANGAN *DRIVE HEXAPOD* ROBOT UNTUK PENJEJAK DINDING DAN PENGHINDAR HALANGAN DENGAN NAVIGASI SENSOR ULTRASONIK MENGGUNAKAN METODE KENDALI LOGIKA *FUZZY***

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

Hari : Rabu
Tanggal : 19 Agustus 2015
Dengan Nilai : 92,0 (A)

Panitia Ujian Skripsi

Ketua Majelis Penguji	Sekretaris Majelis Penguji
M. Ibrahim Ashari, ST, MT	**Dr. Eng. I Komang Somawirata, ST, MT**
NIP. P. 1030100358	**NIP. P.1030100361**

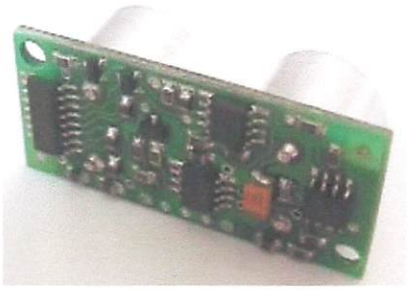
Anggota Penguji

Penguji I	Penguji II
Dr. Eng. Aryuanto Soetedjo, ST, MT	**Dr. F. Yudi Limpraptono,ST,MT**
NIP. Y. 1030800417	**NIP.Y. 1039500274**

DATASHEET

SRF04 - Ultra-Sonic Ranger

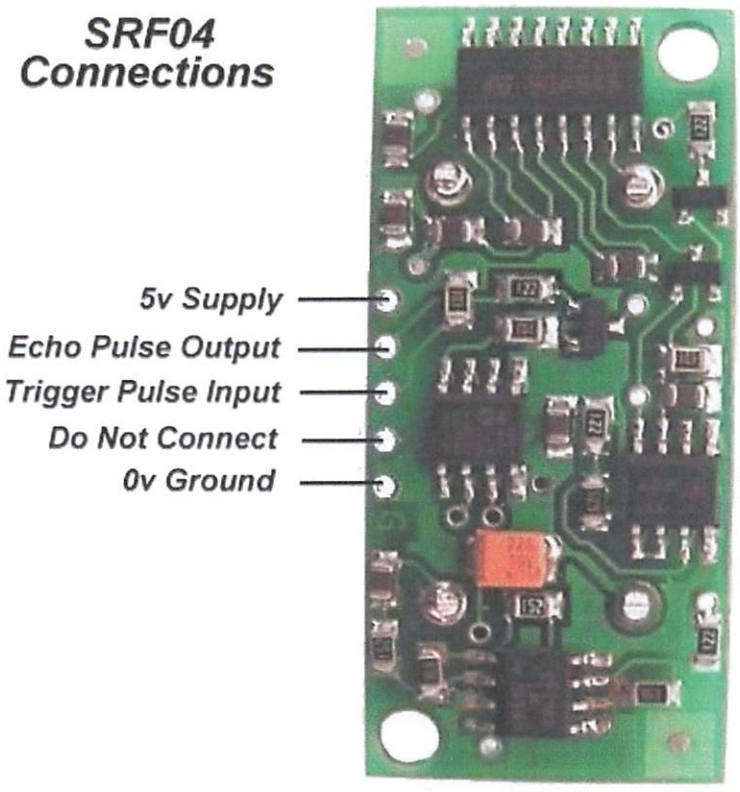
Technical Specification



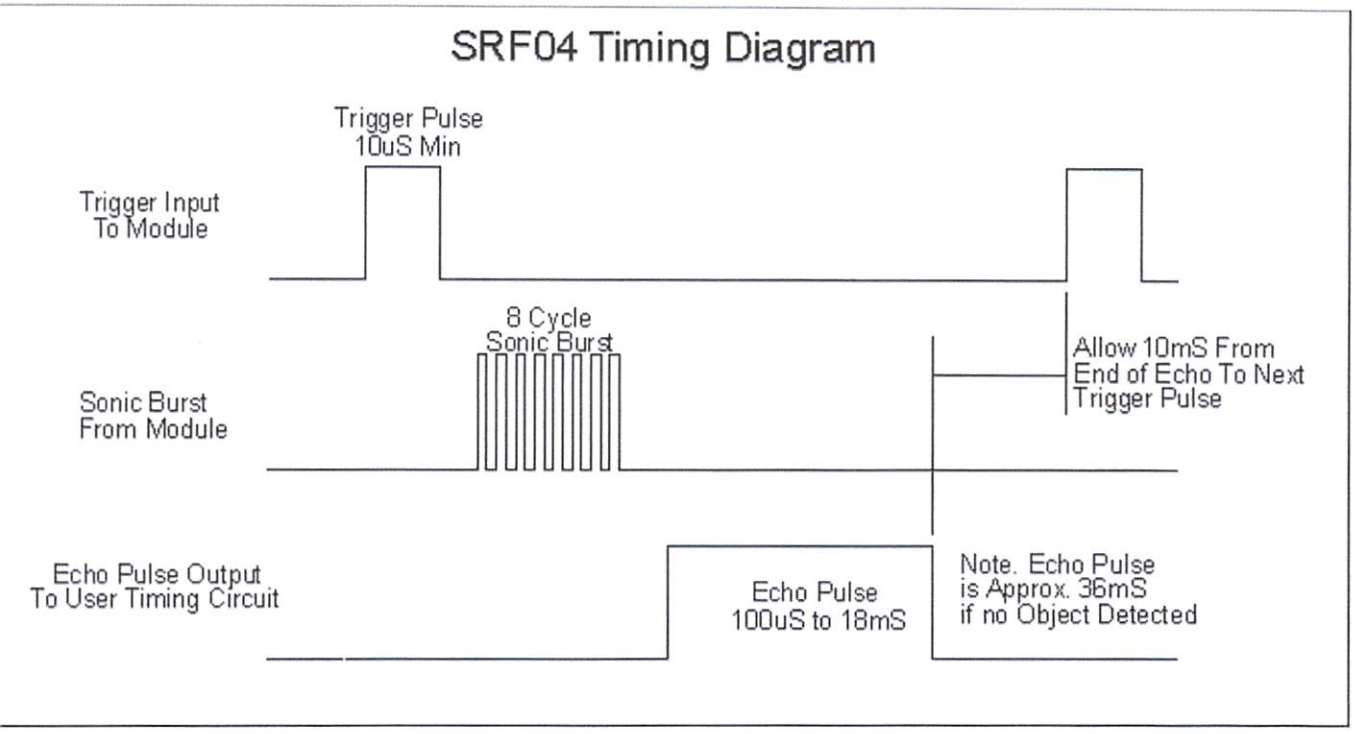
s project started after I looked at the Polaroid Ultrasonic Ranging module. It has a number of dvantages for use in small robots etc.

1. The maximum range of 10.7 metre is far more than is normally required, and as a result
2. The current consumption, at 2.5 Amps during the sonic burst is truly horrendous.
3. The 150mA quiescent current is also far too high.
4. The minimum range of 26cm is useless. 1-2cm is more like it.
5. The module is quite large to fit into small systems, and
6. It's EXPENSIVE.

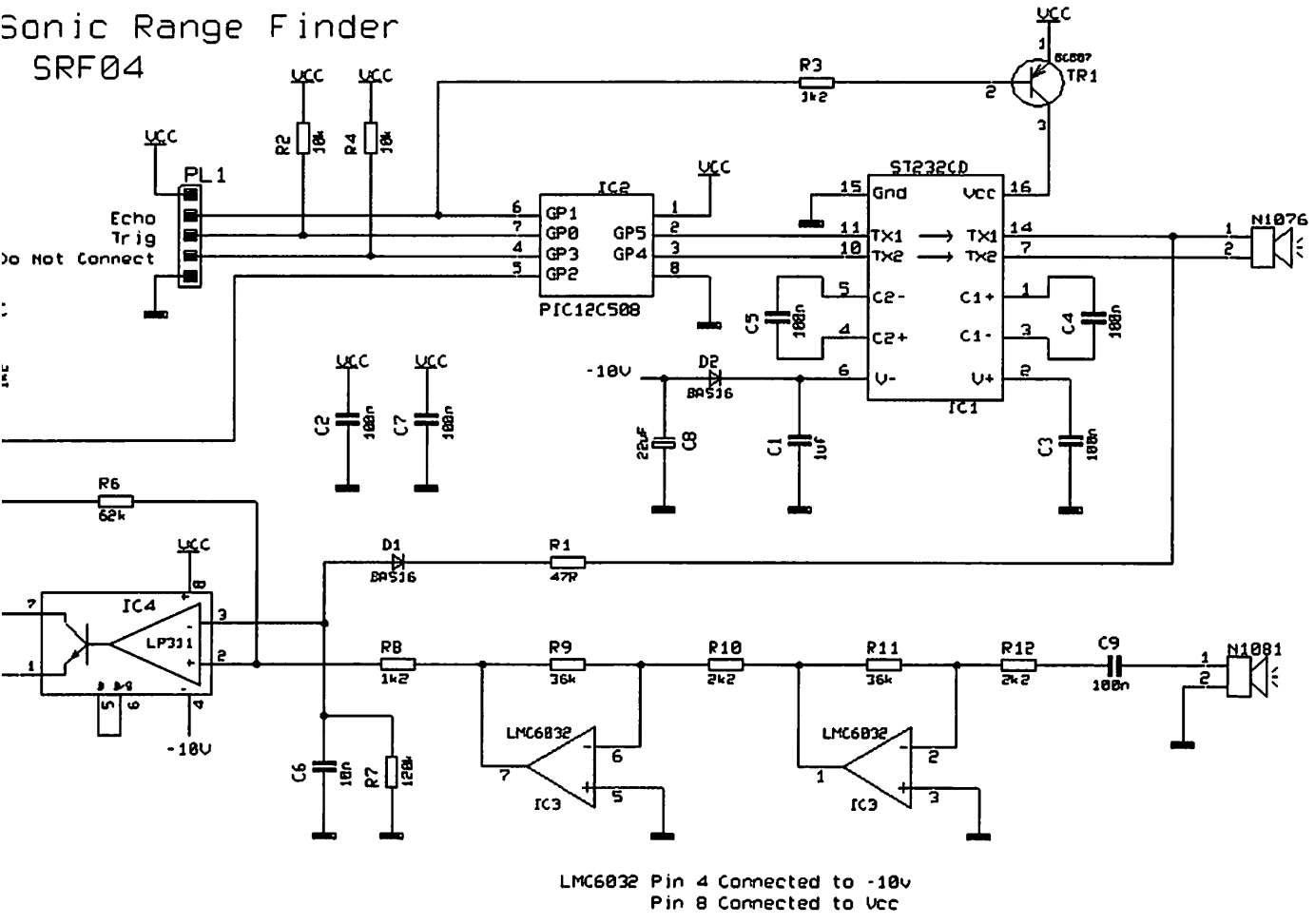
e SRF04 was designed to be just as easy to use as the Polaroid sonar, requiring a short trigger pulse and viding an echo pulse. Your controller only has to time the length of this pulse to find the range. The nections to the SRF04 are shown below:



SRF04 Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input tart the ranging. The SRF04 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line n. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is efore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to ulate the range in inches/centimeters or anything else. If nothing is detected then the SRF04 will lower its o line anyway after about 36mS.



e is the schematic, You can download a better quality pdf (161k) version srf1.pdf



LMC6032 Pin 4 Connected to -10v
Pin 8 Connected to Vcc

circuit is designed to be low cost. It uses a PIC12C508 to perform the control functions and standard hz piezo transducers. The drive to the transmitting transducer could be simplest driven directly from the . The 5v drive can give a useful range for large objects, but can be problematic detecting smaller objects. transducer can handle 20v of drive, so I decided to get up close to this level. A MAX232 IC, usually used RS232 communication makes and ideal driver, providing about 16v of drive.

receiver is a classic two stage op-amp circuit. The input capacitor C8 blocks some residual DC which ays seems to be present. Each gain stage is set to 24 for a total gain of 576-ish. This is close the 25 imum gain available using the LM1458. The gain bandwidth product for the LM1458 is 1Mhz. The imum gain at 40khz is 1000000/40000 = 25. The output of the amplifier is fed into an LM311 parator. A small amount of positive feedback provides some hysterisis to give a clean stable output.

problem of getting operation down to 1-2cm is that the receiver will pick up direct coupling from the smitter, which is right next to it. To make matters worse the piezo transducer is a mechanical object that os resonating some time after the drive has been removed. Up to 1mS depending on when you decide it stopped. It is much harder to tell the difference between this direct coupled ringing and a returning echo, ch is why many designs, including the Polaroid module, simply blank out this period. Looking at the ning echo on an oscilloscope shows that it is much larger in magnitude at close quarters than the cross- oled signal. I therefore adjust the detection threshold during this time so that only the echo is detectable. 100n capacitor C10 is charged to about –6v during the burst. This discharges quite quickly through the

resistor R6 to restore sensitivity for more distant echo's.

onvenient negative voltage for the op-amp and comparator is generated by the MAX232. Unfortunately, also generates quite a bit of high frequency noise. I therefore shut it down whilst listening for the echo. 10uF capacitor C9 holds the negative rail just long enough to do this.

peration, the processor waits for an active low trigger pulse to come in. It then generates just eight cycles 0khz. The echo line is then raised to signal the host processor to start timing. The raising of the echo line shuts of the MAX232. After a while – no more than 10-12mS normally, the returning echo will be cted and the PIC will lower the echo line. The width of this pulse represents the flight time of the sonic t. If no echo is detected then it will automatically time out after about 30mS (Its two times the WDT od of the PIC). Because the MAX232 is shut down during echo detection, you must wait at least 10mS veen measurement cycles for the +/- 10v to recharge.

formance of this design is, I think, quite good. It will reliably measure down to 3cm and will continue cting down to 1cm or less but after 2-3cm the pulse width doesn't get any smaller.

ximum range is a little over 3m. As and example of the sensitivity of this design, it will detect a 1inch k plastic broom handle at 2.4m.
rage current consumption is reasonable at less than 50mA and typically about 30mA.

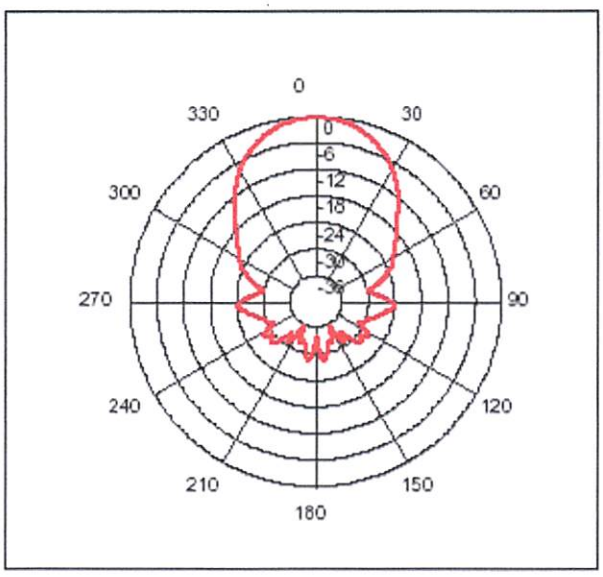
vnload the source code and a ready assembled hex file.

culating the Distance

SRF04 provides an echo pulse proportional to distance. If the width of the pulse is measured in uS, then ding by 58 will give you the distance in cm, or dividing by 148 will give the distance in inches. 58=cm or uS/148=inches.

nging beam pattern and beam width

can't! This is a question which crops up regularly, however there is no easy way to reduce or change the m width that I'm aware of. The beam pattern of the SRF04 is conical with the width of the beam being a tion of the surface area of the transducers and is fixed. The beam pattern of the transducers used on the F04, taken from the manufacturers data sheet, is shown below.



e is more information in the sonar faq.

ate - May 2003

e the original design of the SRF04 was published, there have been incremental improvements to improve ormance and manufacturing reliability. The op-amp is now an LMC6032 and the comparator is an 11. The 10uF capacitor is now 22uF and a few resistor values have been tweaked. These changes have ened over a period of time.

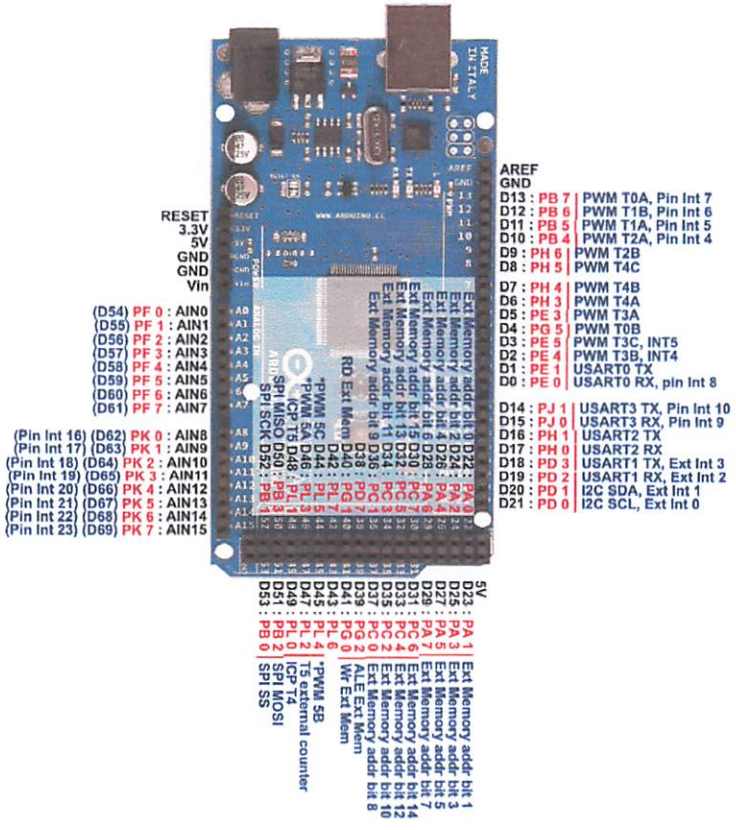
RF04's manufactured after May 2003 have new software implementing an optional timing control input g the "do not connect" pin. This connection is the PIC's Vpp line used to program the chip after mbly. After programming its just an unused input with a pull-up resistor. When left unconnected the 04 behaves exactly as it always has and is described above. When the "do not connect" pin is connected ound (0v), the timing is changed slightly to allow the SRF04 to work with the slower controllers such as icaxe. The SRF04's "do not connect" pin now acts as a timing control. **This pin is pulled high by ult and when left unconnected, the timing remains exactly as before.** With the timing pin pulled low unded) a 300uS delay is added between the end of the trigger pulse and transmitting the sonic burst. e the echo output is not raised until the burst is completed, there is no change to the range timing, but the S delay gives the Picaxe time to sort out which pin to look at and start doing so. The new code has ed in all SRF04's since the end of April 2003. The new code is also useful when connecting the SRF04 e slower Stamps such as the BS2. Although the SRF04 works with the BS2, the echo line needs to be ected to the lower numbered input pins. This is because the Stamps take progressively longer to look at igher numbered pins and can miss the rising edge of the echo signal. In this case you can connect the "do onnect" pin to ground and give it an extra 300uS to get there.

Arduino Mega 2560

High Performance, Low Power AVR®

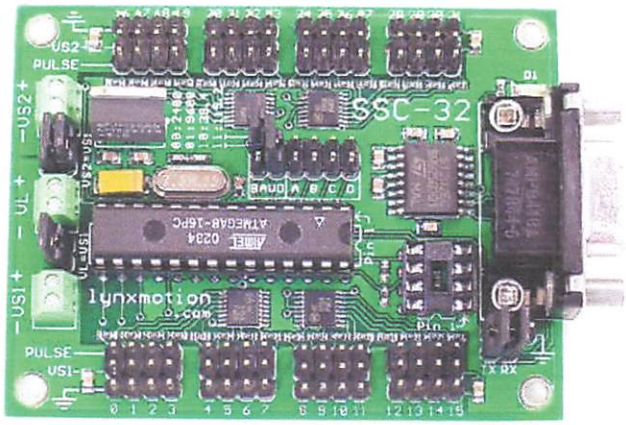
- *8-Bit Microcontroller*

- *Advanced RISC Architecture*
- *High Endurance Non-volatile Memory segments*
  - *8K Bytes of In-System Self-programmable Flash program memory*
  - *4K Bytes EEPROM*
  - *8K Bytes Internal SRAM*
  - *Write/Erase cyles: 10,000 Flash/100,000 EEPROM*
  - *Data retention: 20 years at 85°C/100 years at 25°C*
  - *Programming Lock for Software Security*
- *Peripheral Features*
  - *16-channel ADC*
  - *Programmable Serial USART*
  - *Master/Slave SPI Serial Interface*
  - *Byte-oriented 2-wire Serial Interface (Philips IC compatible)*
  - *Interrupt and Wake-up on Pin Change*



SSC-32 Ver 2.0

Manual written for firmware
version SSC32-1.06XE
Range is 0.50mS to 2.50mS



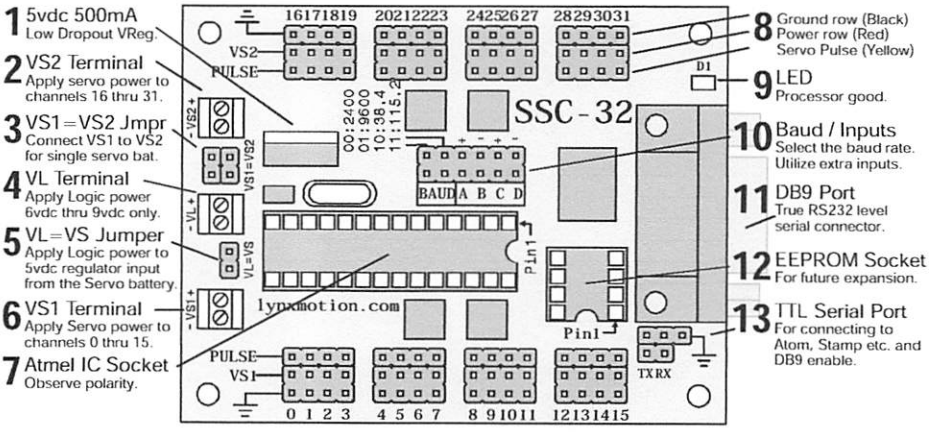
Lynxmotion, Inc.
PO Box 818
Pekin, IL 61555-0818
Tel: 309-382-1816 (Sales)
Tel: 309-382-2760 (Support)
Fax: 309-382-1254
E-m: sales@lynxmotion.com
E-m: tech@lynxmotion.com
Web: http://www.lynxmotion.com

Things that go Boom!

Caution! Read this quick start guide completely before wiring and applying power to the board! Errors in wiring can damage the SSC-32 board, Atmel or EEPROM Chip, and any attached servos or peripherals.

Caution! Never reverse the power coming in to the board. Make sure the black wire goes to (-) ground, and the red wire goes to (+) Vlogic, or Vservo. Never connect peripherals when the board is powered on.

Caution! The onboard regulator can provide 250mA total. This includes the microcontroller chip, the onboard LEDs, and any attached peripherals. Drawing too much current can cause the regulator to overheat.



1 The Low Dropout regulator will provide 5vdc out with as little as 5.5vdc coming in. This is important when operating your robot from a battery. It can accept a maximum of 9vdc in. The regulator is rated for 500mA, but we are de-rating it to 250mA to prevent the regulator from getting too hot.

2 This terminal connects power to servo channels 16 thru 31. Apply 4.8vdc to 7.2vdc for normal servos. Apply 4.8vdc to 6.0vdc when using micro servos. Do not exceed 7.4vdc (measure it) when using HSR-5995TG servos!

Board	Input
VS2+	RED
VS2 -	BLACK

3 These jumpers are used to connect VS1 to VS2. Use this option when you are powering all servos from the same battery. Use both jumpers.

4 This is the Electronics Power Input. It is also referred to as the Logic Voltage, or VL. This input is normally used with a 9vdc battery connector to provide power to the ICs and anything connected to the 5vdc lines on the board. This input is used to isolate the logic from the Servo Power Input.

Board	Input
VL+	RED
VL -	BLACK

5 This jumper allows powering the microcontroller and support circuitry from the servo power supply. This requires at least 6vdc to operate correctly. If the microcontroller resets when many servos are moving it may be necessary to power the microcontroller separately using the VL input. A 9vdc battery works nicely for this.

6 This terminal connects power to servo channels 0 thru 15. Apply 4.8vdc to 7.2vdc for normal servos. Apply 4.8vdc to 6.0vdc when using micro servos. Do not exceed 7.4vdc (measure it) when using HSR-5995TG servos!

Board	Input
VL1+	RED
VL1 -	BLACK

7 This is where the Atmel IC chip goes. Be careful to insert it with Pin 1 in the upper right corner as pictured. Take care not to bend the pins.

8 This is where you connect the servos or other output devices. Use caution and remove power when connecting anything to the I/O bus.

9 This is the Processor Good LED. It will light steady when power is applied and will remain lit until the processor has received a valid serial command. It will then go out and then blink whenever it is receiving serial data. Note, this feature may not be used on user-submitted firmware for the SSC-32.

10 The two BAUD inputs allow configuring the baud rate. Please see the examples below. The ABCD inputs have both static and latching support. The inputs have internal weak (50k) pullups that are used when a Read Digital Input command is used. A normally open switch connected from the input to ground will work fine. These features may not be used on user-submitted firmware for the SSC-32.

Jumpers	Baud Rate
0 0	2400
0 1	9600
1 0	38.4k
1 1	115.2k

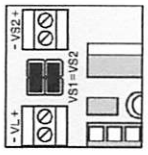
11 Simply plug a straight-through M/F DB9 cable from this plug to a free 9 pin serial port on your PC for receiving servo positioning data. Alternately a USB-to-serial adaptor will work well.

12 This is an 8 pin EEPROM socket. It is not used in this version of the firmware, although it will be used in future versions.

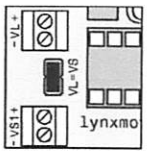
13 This is the TTL serial port or DB9 serial port enable. Install two jumpers as illustrated below to enable the DB9 port. Install wire connectors to utilize TTL serial communication from a host microcontroller.

Shorting Bar Jumpers and Connectors at a glance

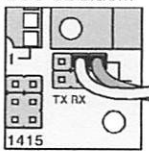
Applies VS1 to VS2.



Applies VS to VL.

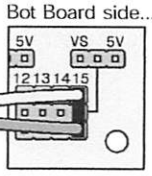


TTL Serial com. SSC-32 side...

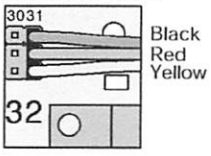


DC-01

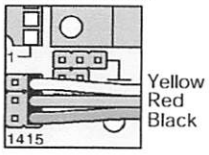
TTL Serial com. Bot Board side...



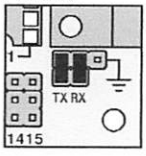
Example servo connection 16-31.



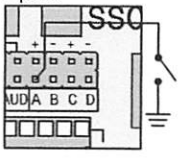
Example servo connection 0-15.



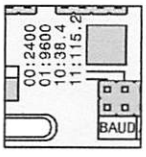
DB9 enable for PC use.



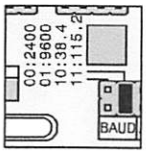
ABCD auxiliary inputs.



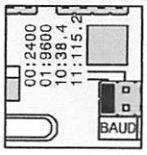
Baud rate 2400.



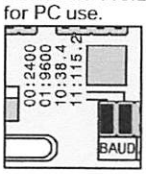
Baud rate 9600.



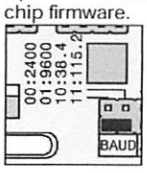
Baud rate 38.4k for Basic Atom use.



Baud rate 115.2k for PC use.



Update the Atmel chip firmware.



Caution! Don't do this if you don't know what you're doing. Connecting this jumper can overwrite the Atmel chip's firmware.

Getting Started

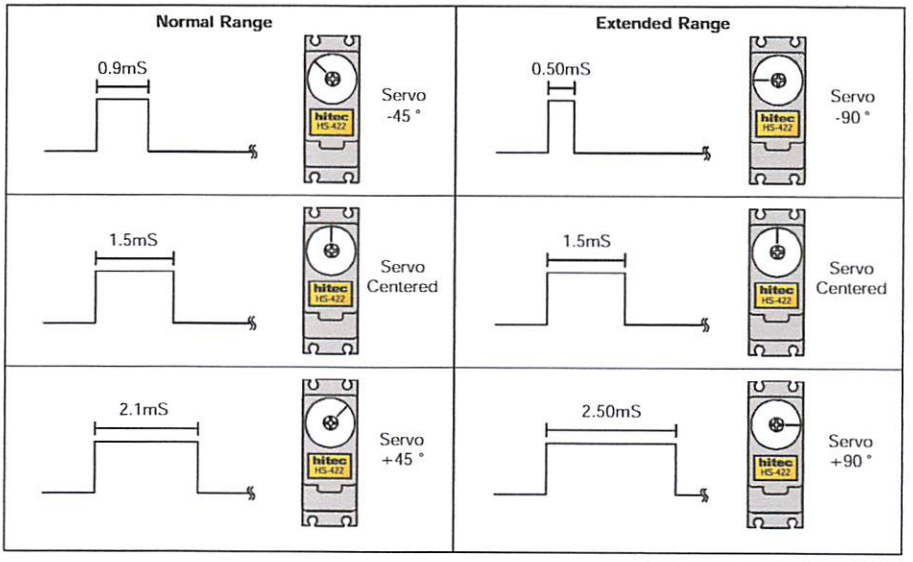
What is a Servo?

Before we illustrate how to use the servo controller we need to explain what a servo is, and define the control methodology.

Pulse-proportional servos are designed for use in radio-controlled (R/C) cars, boats and planes. They provide precise control for steering, throttle, rudder, etc. using a signal that is easy to transmit and receive. The signal consists of positive going pulses ranging from 0.9 to 2.1mS (milliseconds) long, repeated 50 times a second (every 20mS). The servo positions its output shaft in proportion to the width of the pulse, as shown below.

In radio-control applications, a servo needs no more than a 90° range of motion, since it is usually driving a crank mechanism that can't move more than 90°. So when you send pulses within the manufacturer-specified range of 0.9 to 2.1mS, you get around 90° range of motion.

Most servos have more than 90° of mechanical range. In fact, most servos can move up to 180° of rotation. However, some servos can be damaged when commanded past their mechanical limitations. The SSC-32 lets you use this extra range. A position value of 500 corresponds to 0.50mS pulse, and a position value of 2500 corresponds to a 2.50mS pulse. A one unit change in position value produces a 1uS (microsecond) change in pulse width. The positioning resolution is 0.09°/unit (180°/2000). From here on, the term pulse width and position are the same.



Remember that some servos may not be able to move the entire 180° range. Use care when testing servos. Move to the extreme left or right slowly, looking for a point when additional positioning values no longer result in additional servo output shaft movement. When this value is found, put it as a limit in your program to prevent damaging the servo. Generally, micro servos are not able to move the entire 180° range.

Command Formatting for the SSC-32

Command Types and Groups

1) Servo Movement.
2) Discrete Output.
3) Byte Output.
4) Query Movement Status.
5) Query Pulse Width.
6) Read Digital Inputs.
7) Read Analog Inputs.
8) 12 Servo Hexapod Gait Sequencer.
9) Query Hex Sequencer.
10) Get Version.
11) Go To Boot.
12) MiniSSC-II Compatibility.

With the exception of MiniSSC-II mode, all SSC-32 commands must end with a carriage return character (ASCII 13). Multiple commands of the same type can be issued simultaneously in a *Command Group*. All of the commands in a command group will be executed after the final carriage return is received. Commands of different types cannot be mixed in the same command group. In addition, numeric arguments to all SSC-32 commands must be ASCII strings of decimal numbers, e.g. "1234". Some commands accept negative numbers, e.g. "-5678". Programming examples will be provided. ASCII format is not case sensitive. Use as many bytes as required. Spaces, tabs, and line feeds are ignored.

Servo Move or Group Move:

<ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr>

<ch> = Channel number in decimal, 0 - 31.
<pw> = Pulse width in microseconds, 500 - 2500.
<spd> = Movement speed in uS per second for one channel. (Optional)
<time> = Time in mS for the entire move, affects all channels, 65535 max. (Optional)
<cr> = Carriage return character, ASCII 13. (Required to initiate action)
<esc> = Cancel the current command, ASCII 27.

Servo Move Example: "#5 P1600 S750 <cr>"

The example will move the servo on channel 5 to position 1600. It will move from its current position at a rate of 750uS per second until it reaches its commanded destination. For a better understanding of the speed argument consider that 1000uS of travel will result in around 90° of rotation. A speed value of 100uS per second means the servo will take 10 seconds to move 90°. Alternately a speed value of 2000uS per second equates to 500mS (half a second) to move 90°.

Servo Move Example: "#5 P1600 T1000 <cr>"

The example will move servo 5 to position 1600. It will take 1 second to complete the move regardless of how far the servo has to travel to reach the destination.

Servo Group Move Example: "#5 P1600 #10 P750 T2500 <cr>"

The example will move servo 5 to position 1600 and servo 10 to position 750. It will take 2.5 seconds to complete the move, even if one servo has farther to travel than another. The servos will both start and stop moving at the same time. This is a very powerful command. By commanding all of the legs in a walking robot with the Group Move it is easy to synchronize complex gaits. The same synchronized motion can benefit the control of a robotic arm as well.

You can combine the speed and time commands if desired. The speed for each servo will be calculated according to the following rules:

1. All channels will start and end the move simultaneously.
2. If a speed is specified for a servo, it will not move any faster than the speed specified (but it might move slower if the time command requires).
3. If a time is specified for the move, then the move will take at least the amount of time specified (but might take longer if the speed command requires).

Servo Move Example: "#5 P1600 #17 P750 S500 #2 P2250 T2000 <cr>"

The example provides 1600uS on ch5, 750uS on ch17, and 2250uS on ch2. The entire move will take at least 2 seconds, but ch17 will not move faster than 500uS per second. The actual time for the move will depend on the initial pulse width for ch17. Suppose ch17 starts at position 2000. Then it has to move 1250uS. Since it is limited to 500uS per second, it will require at least 2.5 seconds, so the entire move will take 2.5 seconds. On the other hand, if ch17 starts at position 1000, it only needs to move 250uS, which it can do in 0.5 seconds, so the entire move will take 2 seconds.

Important! Don't issue a speed or time command to the servo controller as the first instruction. It will assume it needs to start at 500uS and will zip there as quickly as possible. The first positioning command should be a normal "# <ch> P <pw>" command. Because the controller doesn't know where the servo is positioned on power up it has to be this way.

Pulse Offset:
<ch> PO <offset value> ... # <ch> PO <offset value> <cr>

<ch> = Channel number in decimal, 0 - 31.
<offset value> = 100 to -100 in uSeconds.
<cr> = Carriage return character, ASCII 13.

The servo channel will be offset by the amount indicated in offset value. This makes it easy to setup legs in a robot that do not allow mechanical calibration.

Discrete Output:
<ch> <lvl> ... # <ch> <lvl> <cr>

<ch> = Channel number in decimal, 0 - 31.
<lvl> = Logic level for the channel, either 'H' for High or 'L' for Low.
<cr> = Carriage return character, ASCII 13.

The channel will go to the level indicated within 20mS of receiving the carriage return.

Discrete Output Example: "#3H #4L <cr>"

This example will output a High (+5v) on channel 3 and a Low (0v) on channel 4.

Byte Output:
<bank> : <value> <cr>

<bank> = (0 = Pins 0-7, 1 = Pins 8-15, 2 = Pins 16-23, 3 = Pins 24-31.)
<value> = Decimal value to output to the selected bank (0-255). Bit 0 = LSB of bank.

This command allows 8 bits of binary data to be written at once. All pins of the bank are updated simultaneously. The banks will be updated within 20mS of receiving the CR.

Bank Output Example: "#3:123 <cr>"

This example will output the value 123 (decimal) to bank 3. 123 (dec) = 01111011 (bin), and bank 3 is pins 24-31. So this command will output a "0" to pins 26 and 31, and will output a "1" to all other pins.

Query Movement Status:
Q <cr>

This will return a "." if the previous move is complete, or a "+" if it is still in progress.

There will be a delay of 50uS to 5mS before the response is sent.

Query Pulse Width:
QP <arg> <cr>

This will return a single byte (in binary format) indicating the pulse width of the selected servo with a resolution of 10uS. For example, if the pulse width is 1500uS, the returned byte would be 150 (binary).

Multiple servos may be queried in the same command. The return value will be one byte per servo. There will be a delay of least 50uS to 5mS before the response is sent. Typically the response will be started within 100uS.

Read Digital Inputs:
A B C D AL BL CL DL <cr>

A, B, C, or D reads the value on the input as a binary value. It returns ASCII "0" if the input is a low (0v) or an ASCII "1" if the input is a high (+5v).

AL, BL, CL, or DL returns the value on the input as an ASCII "0" if the input is a low (0v) or if it has been low since the last *L command. It returns a high (+5v) if the input is a high and never went low since the last *L command. Simply stated it will return a low if the input ever goes low. Reading the status automatically resets the latch.

The ABCD inputs have a weak pullup (~50k) that is enabled when used as inputs. They are checked approximately every 1mS, and are debounced for approximately 15mS. The logic value for the read commands will not be changed until the input has been at the new logic level continuously for 15mS. The Read Digital Input Commands can be grouped in a single read, up to 8 values per read. They will return a string with one character per input with no spaces.

Read Digital Input Example: "A B C DL <cr>"

This example returns 4 characters with the values of A, B, C, and D-Latch. If A=0, B=1, C=1, and DL=0, the return value will be "0110".

Read Analog Inputs:
VA VB VC VD <cr>

VA, VB, VC, VD reads the value on the input as analog. It returns a single byte with the 8-bit (binary) value for the voltage on the pin.

When the ABCD inputs are used as analog inputs the internal pullup is disabled. The inputs are digitally filtered to reduce the effect of noise. The filtered values will settle to their final values within 8mS of a change. A return value of 0 represents 0vdc. A return value of 255 represents +4.98vdc. To convert the return value to a voltage multiply by 5/256. At power-up the ABCD inputs are configured for digital input with pullup. The first time a V* command is used, the pin will be converted to analog without pullup. The result of this first read will not return valid data.

Read Analog Input Example: "VA VB <cr>"

This example will return 2 bytes with the analog values of A and B. For example if the voltage on Pin A is 2vdc and Pin B is 3.5vdc, the return value will be the bytes 102 (binary) and 179 (binary).

12 Servo Hexapod Sequencer Commands:
LH <arg>, LM <arg>, LL <arg>
Set the value for the vertical servos on the left side of the hexapod. LH sets the high value, i.e. the pulse width to raise the leg to its maximum height; LM sets the mid value; and LL sets the low value. The valid range for the arguments is 500 to 2500uS.

RH <arg>, RM <arg>, RL <arg>
Set the value for the vertical servos on the right side of the hexapod. RH sets the high value, i.e. the pulse width to raise the leg to its maximum height; RM sets the mid value; and RL sets the low value. The valid range for the arguments is 500 to 2500uS.

VS <arg>
Sets the speed for movement of vertical servos. All vertical servo moves use this speed. Valid range is 0 to 65535uS/Sec.

LF <arg>, LR <arg>
Set the value for the horizontal servos on the left side of the robot. LF sets the front value, i.e. the pulse width to move the leg to the maximum forward position; LR sets the rear value. The valid range for the arguments is 500 to 2500uS.

RF <arg>, RR <arg>
Set the value for the horizontal servos on the right side of the robot. RF sets the front value, i.e. the pulse width to move the leg to the maximum forward position; RR sets the rear value. The valid range for the arguments is 500 to 2500uS.

HT <arg>
Sets the time to move between horizontal front and rear positions. The valid range for the argument is 1 to 65535uS.

XL <arg>, XR <arg>
Set the travel percentage for left and right legs. The valid range is -100% to 100%. Negative values cause the legs on the side to move in reverse. With a value of 100%, the legs will move between the front and rear positions. Lower values cause the travel to be proportionally less, but always centered. The speed for horizontal moves is adjusted based on the XL and XR commands, so the move time remains the same.

XS <arg>
Set the horizontal speed percentage for all legs. The valid range is 0% to 200%. With a value of 100%, the horizontal travel time will be the value programmed using the HT command. Higher values proportionally reduce the travel time; lower values increase it. A value of 0 will stop the robot in place. The hex sequencer will not be started until the XS command is received.

XSTOP
Stop the hex sequencer. Return all servos to normal operation.

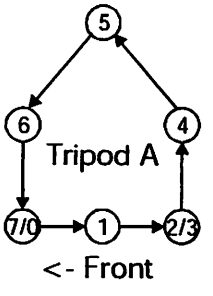
Notes on Hex Sequencer:

1) The following servo channels are used for the Hex Sequencer

0 = Right Rear Vertical	16 = Left Rear Vertical
1 = Right Rear Horizontal	17 = Left Rear Horizontal
2 = Right Center Vertical	18 = Left Center Vertical
3 = Right Center Horizontal	19 = Left Center Horizontal
4 = Right Front Vertical	20 = Left Front Vertical
5 = Right Front Horizontal	21 = Left Front Horizontal

2) The Hexapod walking gait is an alternating tripod. The tripods are labeled Tripod A and Tripod B. Tripod A consists of {Left Front, Left Rear, Right Center}, and Tripod B consists of {Left Center, Right Front, Right Rear}.
3) While walking, the legs pass through 6 points: (Low Front), (Low Center), (Low Rear), (Mid Rear), (High Center), and (Mid Front). 'Center "refers to the mid-point between the Front and Rear pulse widths.
4) The walking sequence consists of 8 states, numbered 0 –7. They are defined below:

State	Tripod A		Tripod B	
	Vertical	**Horizontal**	**Vertical**	**Horizontal**
0	Low	Front to Center	Mid to High	Rear to Center
1	Low	Center to Rear	High to Mid	Center to Front
2	Low	Rear	Mid to Low	Front
3	Low to Mid	Rear	Low	Front
4	Mid to High	Rear to Center	Low	Front to Center
5	High to Mid	Center to Front	Low	Center to Rear
6	Mid to Low	Front	Low	Rear
7	Low	Front	Low to Mid	Rear



In this table, 'Front "and 'Rear "are modified by the XL and XR commands. A value of 100% results in the movement in the table. Between 0 and 100%, the Front/Rear positions are moved closer to Center. For negative values, Front and Rear are exchanged. For example, with an XL of -100%, in State 0, Tripod A on the left side would be moving Rear to Center, and Tripod B would be moving Front to Center.

5) When a horizontal servo is moving, its speed will be adjusted based on the Front/Rear pulse widths, the XL/XR percentage, and the XS percentage. Regardless of the travel distance from front to rear (adjusted by XL/XR), the total move time will be the HT divided by the XS percentage.
6) When a vertical servo is moving from Low to Mid or from Mid to Low, it will move at the speed specified by the VS command. When a vertical servo is moving from Mid to High or High to Mid, the vertical speed will be adjusted so that the horizontal and vertical movements end at the same time.
7) Any of the Hex Sequencer commands can be issued while the sequencer is operating. They will take effect immediately.

Hex Sequencer Examples:

'LH1000 LM1400 LL1800 RH2000 RM1600 RL1200 VS3000 <cr> "
Sets the vertical servo parameters.

'LF1700 LR1300 RF1300 RR1700 HT1500 <cr> "
Sets the horizontal servo parameters.

'XL50 XR100 XS100 <cr> "
Causes the gradual left turn at 100% speed (and starts the sequencer if it is not already started).

'XL -100 XR 100 XS 50 <cr> "
Causes a left rotate in place at 50% speed.

'XSTOP <cr> "
Stops the sequencer and allows servo channels 0-5, 16-21 to be controlled using the normal servo movement commands.

Query Hex Sequencer State:
XQ <vr>
Returns 1 digit representing the state of the hex sequencer, and the approximate percentage of movement in the state. The high nibble will be 0 'to 7', and the low nibble will be 0 'to 9'. For example, if the sequencer is 80% of the way through state 5, it will return the value 58 hex.

Get Software Version:
VER <cr>
Returns the software version number as an ASCII string.

Transfer to Boot:
GOBOOT <cr>
Starts the bootloader running for software updates. To exit the bootloader and start running the application, power cycle the control or enter (case sensitive, no spaces).

g0000<cr>

SSC Emulation: Binary format, 3-bytes.
Byte 1: 255, the sync byte
Byte 2: 0 - 31, the servo number
Byte 3: 0 - 250, the pulse width, 0=500uS, 125=1500uS, 250=2500uS

Testing the Controller

The easiest way to test the controller is to use the Lynx SSC-32 Terminal. It's a free download from the website. Once installed click on the Port Drop Down and select your com port. This will work with USB to serial port adaptors. Install the jumpers for 115.2k baud and the two DB9 serial port enable jumpers. Plug a straight through DB9 M/F cable from the PC to the controller.

Install two servos, one on channel 0 and one on channel 1.

Power up the SSC-32 (Logic and Servo) and notice the green LED is illuminated.

Then click on the terminal window so you can type the following into it.

#0 P1500 #1 P1500 <cr> <- (This means hit Enter.)

You should notice both servos are holding position in the center of their range. The LED is also no longer illuminated. It will now only light when the controller is receiving data. Type the following:

#0 P750 #1 P1000 T3000 <cr>

You should notice servo 0 moving CW slowly and servo 1 moving CCW a bit faster. They will arrive at their destination at the exact same time even though they are moving different distances.

Now to test the Query Movement Status. Type the following:

#0 P750 <cr>

Then type the following line. This will make the servo move full range in 10 seconds.

#0 P2250 T10000 <cr>

While the servo is moving type the following:

Q <cr>

When the servo is in motion the controller will return a "+". It will return a "." when it has reached its destination.

To experiment with the speed argument try the following:

#0 P750 S1000 <cr>

This will move the servo from 2250 to 750 (around 170°) in 1.5 seconds.

$$\frac{\text{2250uS-750uS (travel distance)}}{\text{1000uS/Sec. (speed value)}} = 1.5 \text{ Sec.}$$

Next try typing the following:

#0 P2250 S750 <cr>

This will move the servo from 750 to 2250 (around 170°) in 2.0 seconds.

$$\frac{\text{2250uS-750uS (travel distance)}}{\text{750uS/Sec. (speed value)}} = \text{2.0 Sec.}$$

Speed values above around 3500 will move the servo as quickly as the servo can move.

Updating the SSC-32 firmware
From the SSC-32 Terminal main screen click on Firmware. This will show the current version of the firmware at the top, and allow you to browse and open the new *.abl firmware file. Click Begin Update to finish the update process.

Don't forget to check the website for the latest tutorials for the servo controller.

Troubleshooting Information
If you notice the servos turn off, or stop holding position when moving several servos at one time. This indicates the SSC-32 has reset. This can be verified by noticing if the green LED is on steady after the servos are instructed to move. The green LED is not a power indicator, but a status indicator. When the SSC-32 is turned on the LED will be on steady. It will remain on until it has received a valid serial command, then it will go out and only blink when receiving serial data.

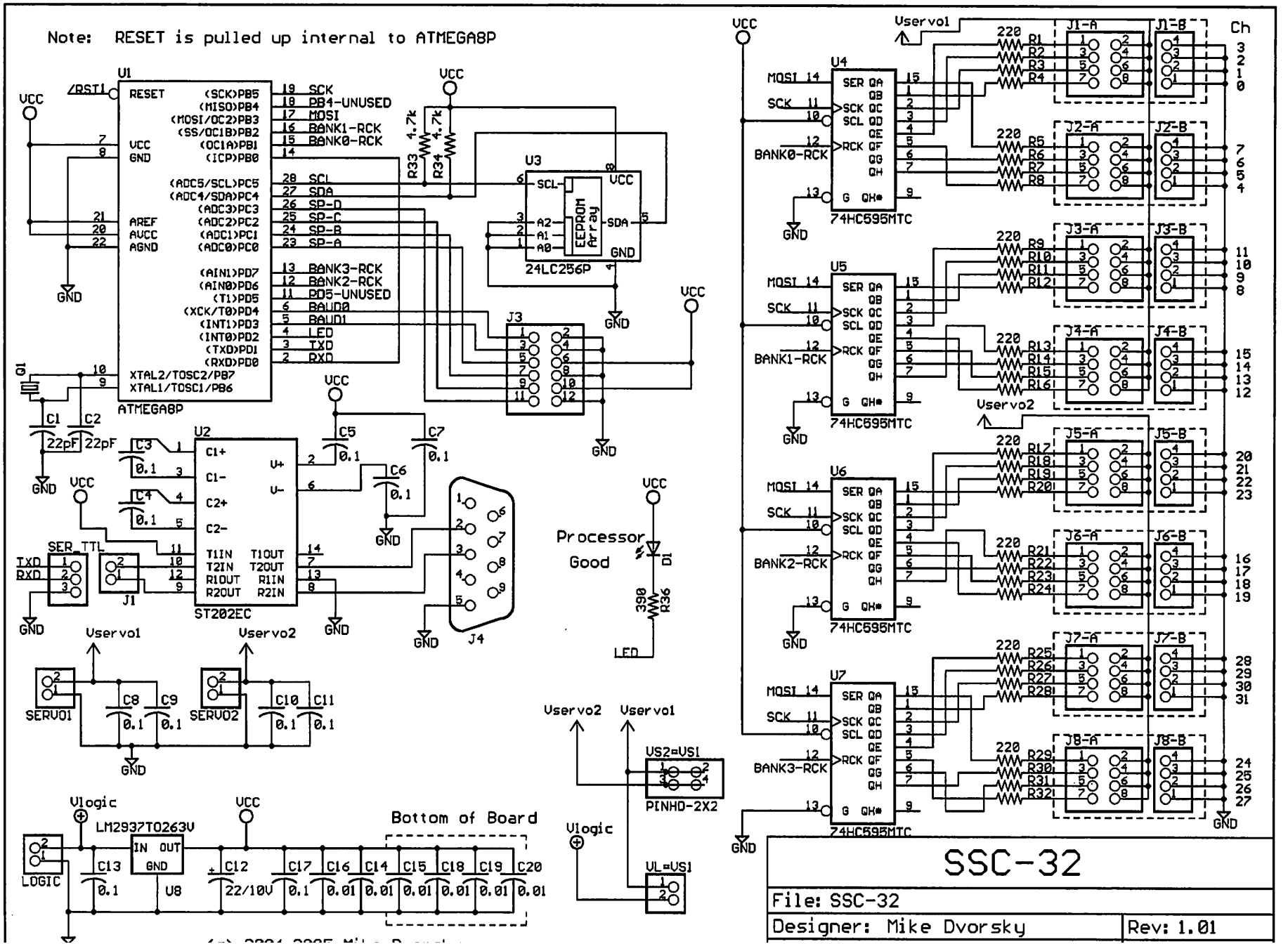
The SSC-32 has two power supply inputs. The logic supply (VL) powers the microcontroller and it's support circuitry through a 5vdc regulator. The servo supply (VS) powers the servos directly. In single supply mode (default) the jumper VS1=VL will provide power to the VL 5vdc regulator from the VS terminal. This works great for battery use, and with most wall pack use, as long as the voltage does not drop too much. However if it does drop, the voltage to the microcontroller is interrupted and the SSC-32 resets. To fix this you remove the VS1=VL jumper and connect a 9vdc battery clip to the VL input. This isolates the servo and logic supplies so one cannot effect the other.

Using the single supply mode is generally safe for the following conditions:
- VS of 7.2vdc 2800mAh NiCad or NiMH battery packs for up to 24 servos.
- VS of 7.4vdc 2800mAh LiPo battery packs for up to 24 servos.
- VS of 6.0vdc 1600mAh NiCad or NiMH battery packs for up to 18 servos.
- VS of 6.0vdc 2.0amp wall pack for up to 8 servos.

Note, these are just general guidelines and some exceptions may exist. The only other thing that can cause this effect is a poor power delivery system. If the wires carrying the current are too small, or connections are made with stripped and twisted wire, or cheap plastic battery holders are used, the same problem may occur. 99% of customers problems with the SSC-32 are power supply related. If you are noticing erratic or unstable servo movements, look at the power delivery system.

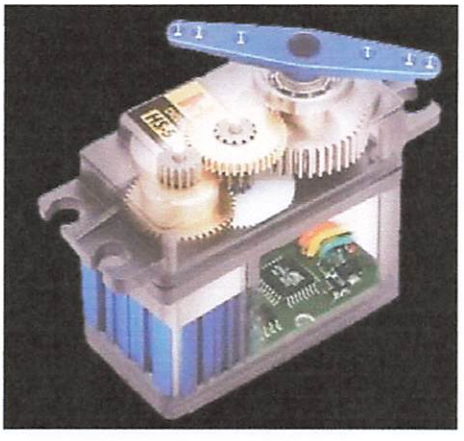
Basic Atom Programming Examples for the SSC-32

```
' Atom / SSC-32 Test
' Configure the SSC-32 for 38.4k baud.

servo0pw  var  word
movetime  var  word

servo0pw = 1000
movetime = 2500

start:
servo0pw = 1000
serout p0,i38400,["#0P",DEC servo0pw,"T",DEC movetime,13]
pause 2500
servo0pw = 2000
serout p0,i38400,["#0P",DEC servo0pw,"T",DEC movetime,13]
pause 2500
goto start

' Biped example program.
aa  var  byte   '<- general purpose variable.
rax  var  word  '<- right ankle side-to-side.   On pin0
ray  var  word  '<- right ankle front-to-back.  On pin1
rkn  var  word  '<- right knee.                 On pin2
rhx  var  word  '<- right hip front-to-back.    On pin3
rhy  var  word  '<- right  hip side-to-side.    On pin4
lax  var  word  '<- left ankle side-to-side.    On pin5
lay  var  word  '<- left ankle front-to-back.   On pin6
lkn  var  word  '<- left knee.                  On pin7
lhx  var  word  '<- left hip front-to-back.     On pin8
lhy  var  word  '<- left hip side-to-side.      On pin9
ttm  var  word  '<- time to take for the current move.

' First command to turn the servos on.
for aa=0 to 9
serout p0,i38400,["#", DEC2 aa\1, "P", DEC 1500, 13]
next

start:
' First position for step sequence, and time to move, put in your values here.
rax=1400: ray=1400: rkn=1400: rhx=1400: rhy=1400
lax=1400: lay=1400: lkn=1400: lhx=1400: lhy=140 0
ttm=1000
gosub send_data
pause ttm

' Second position for step sequence, and time to move, put in your values here.
rax=1600: ray=1600: rkn=1600: rhx=1600: rhy=1600
lax=1600: lay=1600: lkn=1600: lhx=1600: lhy=1600
ttm=1000
gosub send_data
pause ttm

' Third...

' Forth...

' Etc...

goto start

' This sends the data to the SSC-32. The serout is all one line, no Wrap!
send_data:
serout p0,i38400,["#0P",DEC rax,"#1P",DEC ray,"#2P",DEC rkn,"#3P",DEC rhx,"#4P",DEC
rhy,"#5P",DEC lax,"#6P",DEC lay,"#7P",DEC lkn,"#8P",DEC lhx,"#9P",DEC lhy,"T",DEC
ttm,13]
return
```


Note: RESET is pulled up internal to ATMEGA8P
U1
ATMEGA8P
U2
ST202EC
U3
24LC256P
U4
U5
U6
U7
74HC595MTC
J1-A
J1-B
J2-A
J2-B
J3-A
J3-B
J4-A
J4-B
J5-A
J5-B
J6-A
J6-B
J7-A
J7-B
J8-A
J8-B
J1
J3
J4
SERVO1
SERVO2
LDOGIC
U8
LM2937TO263V
PINHD-2X2
VS2=VS1
VL=VS1
Processor Good
D1
LED
SER_TTL
Vservo1
Vservo2
Vlogic
VCC
GND
Bottom of Board
SCK
MOSI
BANK0-RCK
BANK1-RCK
BANK2-RCK
BANK3-RCK
SSC-32
File: SSC-32
Designer: Mike Dvorsky
Rev: 1.01

Spesifikasi Motor Servo HS-645MG

Control System: +Pulse Width Control 1500usec Neutral
Required Pulse: 3-5 Volt Peak to Peak Square Wave
Operating Voltage: 4.8-6.0 Volts
Operating Temperature Range: -20 to +60 Degree C
Operating Speed (4.8V): 0.24sec/60° at no load
Operating Speed (6.0V): 0.20sec/60° at no load
Stall Torque (4.8V): 106.93 oz/in. (7.7kg.cm)
Stall Torque (6.0V): 133.31 oz/in. (9.6kg.cm)
Operating Angle: 45 Deg. one side pulse traveling 400 usec
Continuous Rotation Modifiable: Yes
Direction: Clockwise/Pulse Traveling 1500 to 1900 usec
Current Drain (4.8V): 8.8mA/idle and 350mA no load operating
Current Drain (6.0V): 9.1mA/idle and 450mA no load operating
Dead Band Width: 8usec
Motor Type: 3 Pole Ferrite
Potentiometer Drive: Indirect Drive
Bearing Type: Dual Ball Bearing
Gear Type: 3 Metal Gears and 1 Resin Metal Gear
Connector Wire Length: 11.81" (300mm)
Dimensions: 1.59" x 0.77"x 1.48" (40.6 x 19.8 x 37.8mm)
Weight: 1.94oz. (55.2g)



PROGRAM

```
/*
* Dedicated for Tito

*/

#include "FuzzyESadewa.h"
#include "InitSRF.h"
#include "Kaki.h"
#include "Ngecek.h"
#include "InitPin.h"
#include "SSC32.h"
#include "WallFollowing.h"

/*
1. program untuk membaca jarak (serial monitor)
    a. depan trigger 29 echo 28
    b. kiri pojok trigger 25 echo 24
    c. kanan pojok trigger 31 echo 30
    dengan data sensor srf depan

2. program untuk fazzy maju
dengan prinsip kerja robot dinyalakan powernya posisi diam lalu kita memilih antara
tombol following kanan atau kiri dengan tombol input pin following kanan( pin arduino 5)
dan following kiri (pin arduino 4) baru setelah memilih kita bisa memencet tombol start
dan robot berjalan sesuai folowing yang sudah di pilih di awal tadi
tombol start nya pin 6 arduino     (BY tito dinti koshars)*/



SSC32 sscku;// pemanggilan ssc.h

void setup()
{
    //untuk pengiriman data melalui serial harus diletakkan paling atas
    Serial.begin(9600);//komunikasi arduino ke pc
    Serial1.begin(115200);//komunikasi arduino ke ssc32
    //====================//
    InitSRF.init(); //inisialisasi semua SRF pin
    InitPin.init(); //inisialisasi semua PIN yang digunakan
    FuzzyESadewa.initFuzzyESadewa(); //inisialisasi Fuzzy yang digunakan
    Kaki.setPijakBiasa(); //robot berpijak
}


void loop()
{
    //=============================================================================================//
    //================================EKSEKUSI SEKALI==============================================//
    //=============================================================================================//

    if (!Ngecek.getSekali()){
        Serial.println("======== eksekusi sekali mulai ==========");
        //selama menunggu terompet, robot diem
        while (Ngecek.isMenungguPerintah()){
            Serial.println("Menunggu Perintah");
            cekSRFku();

            if (digitalRead(tombolWFKanan) == 1){
                Serial.println("tombol WF Kanan");
                Ngecek.setWFKanan(true);
                Ngecek.setWFKiri(false);
                digitalWrite(ledWFKanan, HIGH);
                digitalWrite(ledWFKiri, LOW);
            }
            else if (digitalRead(tombolWFKiri) == 1){
                Serial.println("tombol WF Kiri");
                Ngecek.setWFKanan(false);
                Ngecek.setWFKiri(true);
                digitalWrite(ledWFKanan, LOW);
                digitalWrite(ledWFKiri, HIGH);
            }
        }
```

```
        Ngecek.setSekali(true); //untuk sekali dieksekusi
        Serial.println("======== eksekusi sekali selesai ==========");
    }

    //================================================================================//
    //==============================EKSEKUSI BERULANG=================================//
    //================================================================================//

    if (Ngecek.isWFKanan()){
        WallFollowing.wfKananMaju();
        Serial.println("WF kanan");
    }
    else if (Ngecek.isWFKiri()){
        WallFollowing.wfKiriMaju();
        Serial.println("WF kiri");
    }

    Serial.print("kaki kiri : "); Serial.print(Kaki.getKakiKi());
    Serial.print(", kaki kanan : "); Serial.println(Kaki.getKakiKa());
    Serial.println("========================");
    Serial.println("");
}

//////////////////////

void cekSRFku(){
    Serial.print("SRF DEPAN : ");
    Serial.println(InitSRF.getSrfDepan());
    Serial.print("SRF BELAKANG : ");
    Serial.println(InitSRF.getSrfBelakang());
    Serial.print("SRF KANAN POJOK : ");
    Serial.println(InitSRF.getSrfKananPojok());
    Serial.print("SRF KIRI POJOK : ");
    Serial.println(InitSRF.getSrfKiriPojok());
    Serial.println("====================================");
    delay(100);
}
```

```
//
//
//

#include "WallFollowing.h"
#include "FuzzyESadewa.h"
#include "Kaki.h"
#include "InitSRF.h"

void WallFollowingClass::init(){}

//wall folowing master
void WallFollowingClass::wf(double _srfKiriPojok, double _srfDepan, double _srfKananPojok){
    //================ 1 input ==================//
    FuzzyESadewa.setInput(1, _srfDepan);
    FuzzyESadewa.setInput(2, _srfKananPojok);
    FuzzyESadewa.setInput(3, _srfKiriPojok);
    //================ 2 fuzzifikasi ==================//
    FuzzyESadewa.fuzzifikasi();
    //================ 3 output ==================//
    Kaki.setKakiKi(FuzzyESadewa.getOutput(1));        //////--------------PWM-kaki kiri ------- ↙
    -----
    Kaki.setKakiKa(FuzzyESadewa.getOutput(2));    //////--------------PWM-kaki kanan-------
}

//wall following kanan maju
void WallFollowingClass::wfKananMaju(){
    //print
    Serial.println(">> WF KANAN");
    Serial.print("SRF DEPAN : ");
    Serial.println(srfDe);
    Serial.print("SRF KAPO : ");
    Serial.println(srfKaPo);

    //================ 1 input, rule, output ==================//
    srfDe = InitSRF.getSrfDepan_MinMax(0, 170);
    srfKaPo = InitSRF.getSrfKananPojok_MinMax(0, 150);
    //srfKiPo = InitSRF.getSrfKiriPojok_MinMax(0, 150);
    WallFollowing.wf(-1, srfDe, srfKaPo);

    //================ 2 tindakan ==================//
    if (srfDe <= (18 - 2)){
        Kaki.set90derajatPutar(KIRI, 220);//--------------------untuk halangan------------
    }
    if (srfKaPo <= (13)&&srfKaPo <= (12) ){
        Kaki.set45derajatPutar(KIRI, 220);
    }

    if (srfKaPo >= (16)){
        Kaki.set45derajatPutar(KANAN, 220);
    }
    else{
        Kaki.setJalan2(Kaki.getKakiKi(), Kaki.getKakiKa());//----maju dengan fuzzy----------
    }
}

//wall following kiri maju
void WallFollowingClass::wfKiriMaju(){
    //print
    Serial.println(">> WF KIRI");
    Serial.print("SRF DEPAN : ");
    Serial.println(srfDe);
    Serial.print("SRF KIPO : ");
    Serial.println(srfKiPo);

    //================ 1 input, rule, output ==================//
    srfDe = InitSRF.getSrfDepan_MinMax(0, 170);
    //srfKaPo = InitSRF.getSrfKananPojok_MinMax(0, 150);
    srfKiPo = InitSRF.getSrfKiriPojok_MinMax(0, 150);
    Serial.println(srfDe);
    WallFollowing.wf(srfKiPo, srfDe, -1);

    //================ 2 tindakan ==================//
    if (srfDe <= (18)){
```

```
        Kaki.set90derajatPutar(KANAN, 220);
    }
    else{
        Kaki.setJalan2(Kaki.getKakiKi(), Kaki.getKakiKa());
    }
}

//wallfolowing kanan mundur
void WallFollowingClass::wfKananMundur(){
    //=============== 1 input, rule, output ==================//
    srfDe = InitSRF.getSrfDepan_MinMax(0, 170);
    srfKaPo = InitSRF.getSrfKananPojok_MinMax(0, 15);
    //srfKiPo = InitSRF.getSrfKiriPojok_MinMax(0, 150);
    WallFollowing.wf(-1, srfDe, srfKaPo);

    //=============== 2 tindakan ==================//
    Kaki.setJalan2((-Kaki.getKakiKi()), (-Kaki.getKakiKa()));
}

//walfolowing kiri mundur
void WallFollowingClass::wfKiriMundur(){
    //=============== 1 input, rule, output ==================//
    srfDe = InitSRF.getSrfDepan_MinMax(0, 170);
    //srfKaPo = InitSRF.getSrfKananPojok_MinMax(0, 150);
    srfKiPo = InitSRF.getSrfKiriPojok_MinMax(0, 15);
    WallFollowing.wf(srfKiPo, srfDe, -1);

    //=============== 2 tindakan ==================//
    Kaki.setJalan2((-Kaki.getKakiKi()), (-Kaki.getKakiKa()));
}



WallFollowingClass WallFollowing;
```

```
//
//
//

#include "Kaki.h"
#include "SSC32.h"
#include "Ngecek.h"

//untuk SSC32
SSC32 myssc;

//=====================
float kakiKi;
     float kakiKa;
     //============= INISIALISASI POSISI PIJAK  ===========//
     //====================================================//
     //====================================================//

     //kaki 1
     int PIJAK_POS_1A = 1425;   //-200 itu agar sedikit ke depan
     int PIJAK_POS_1B = 795; //940
     int PIJAK_POS_1C = 1065;

     //kaki 2
     int PIJAK_POS_2A = 1500;
     int PIJAK_POS_2B = 960;
     int PIJAK_POS_2C = 1050;

     //kaki 3
     int PIJAK_POS_3A = 1810;   //+200 itu agar sedikit ke belakang
     int PIJAK_POS_3B = 985; //955
     int PIJAK_POS_3C = 1050;

     //kaki 4
     int PIJAK_POS_4A = 1235;   //-200 itu agar sedikit ke belakang
     int PIJAK_POS_4B = 2130; //2160
     int PIJAK_POS_4C = 1890;

     //kaki 5
     int PIJAK_POS_5A = 1600;
     int PIJAK_POS_5B = 1600;   //2170 mudun //backup 1700
     int PIJAK_POS_5C = 1800;   //1870 munggah

     //kaki 6
     int PIJAK_POS_6A = 1430;   //+200 itu agar sedikit ke depan
     int PIJAK_POS_6B = 2045;
     int PIJAK_POS_6C = 2020;

     //=============================

//================== KAKI PUTAR UNTUK HALANGAN ============================//


void KakiClass::setPutar(int arah, int kakiP){
    // C tetap
    myssc.servoMove(PIN_SERVO_1C, PIJAK_POS_1C);
    myssc.servoMove(PIN_SERVO_2C, PIJAK_POS_2C);
    myssc.servoMove(PIN_SERVO_3C, PIJAK_POS_3C);
    myssc.servoMove(PIN_SERVO_4C, PIJAK_POS_4C);
    myssc.servoMove(PIN_SERVO_5C, PIJAK_POS_5C);
    myssc.servoMove(PIN_SERVO_6C, PIJAK_POS_6C);

    //1 3 5
    //B angkat
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B + derajatAngkat_B);
    //delay(WAKTU_SERVO);//beda dengan method setMaju()

    if (arah == 1){//belok kiri
        //A mundur
        myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A + kakiP);   //15
        myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A + kakiP);
```

```
        // maju
        myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A + kakiP);
    }
    else if (arah == 2){//belok kanan
        //A maju
        myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A - kakiP);  //15
        myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A - kakiP);

        // mundur
        myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A - kakiP);
    }

    delay(WAKTU_SERVO);

    //B turun
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B);

    delay(WAKTU_SERVO);

    //A maju
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A);
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A);
    //mundur
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A);
    delay(WAKTU_SERVO);

    // 2 4 6
    // B angkat
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B + derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B + derajatAngkat_B);

    //delay(WAKTU_SERVO);//beda dengan method setMaju()

    if (arah == 1){//belok kiri
        //A mundur
        myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A + kakiP);

        // maju
        myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A + kakiP);      //15
        myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A + kakiP);
    }
    else if (arah == 2){//belok kanan
        //A maju
        myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A - kakiP);   //15

        //mundur
        myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A - kakiP);
        myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A - kakiP);
    }


    delay(WAKTU_SERVO);
    //B turun
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B);

    delay(WAKTU_SERVO);
    //A maju
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A);
    //mundur
    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A);
}
//------------------------------------------------------------------yang digunakan ------------ ↙
    ---------------------------
//method set90derajatPutar() adalah method kaki melakukan   --- untuk menghindar halangan
//gerakan mundur kaki kiri dulu setelah itu kaki kanan
void KakiClass::set90derajatPutar(int arah, int kakiP){
    for (int x = 1; x <= 5; x++){
        Kaki.setPutar(arah, kakiP);
```

```
    }
}
//-------------------------------------------------------------------------------------------- ↙
    ---------
void KakiClass::set45derajatPutar(int arah, int kakiP){
    for (int x = 1; x <= 1; x++){
        Kaki.setPutar(arah, kakiP);
    }
}
//-------------------------------------------------------------------yang digunakan untuk      ↙
    jalan maju-----------------------------------------------------------
//method setJalan() adalah method kaki melakukan
//gerakan maju kaki kiri dulu setelah itu kaki kanan menggunakan fuzzy
void KakiClass::setJalan2(float kakiKi, float kakiKa){


    myssc.servoMove(PIN_SERVO_1C, PIJAK_POS_1C);
    myssc.servoMove(PIN_SERVO_2C, PIJAK_POS_2C);
    myssc.servoMove(PIN_SERVO_3C, PIJAK_POS_3C);
    myssc.servoMove(PIN_SERVO_4C, PIJAK_POS_4C);
    myssc.servoMove(PIN_SERVO_5C, PIJAK_POS_5C);
    myssc.servoMove(PIN_SERVO_6C, PIJAK_POS_6C);
    //-------------------   maju posisi lingkaran
    //   kaki ------------------------------- 1 3 5-------------------------------------
    //B angkat
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B - derajatAngkat_B); //10
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B + derajatAngkat_B);
    //A geser
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A - 180);      //8
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A - 180);//        -          ----PENAMBAHAN KAKI  ↙
    DENGAN Hasil FUZZY ------
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A + 180);        //13 +
    delay(WAKTU_SERVO);
    //buka
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C+derajatAngkat_C);            //11
    //tutup
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C-derajatAngkat_C);               //7
    //B turun
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B);
    delay(WAKTU_SERVO);
    //tutup
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C);
    //buka
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C);
    //A mundur
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A);
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A);
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A);
    delay(WAKTU_SERVO);


    //-----------------------------------------------------2 4 6-------------------------------- ↙
    -----------------
    //A angkat
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B + derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B + derajatAngkat_B);

    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A + 180);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A + 180); //   +       -----------penambahan Hasil  ↙
    Fuzzy------------
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A - 180);//   -
    delay(WAKTU_SERVO);
    // tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C+derajatAngkat_C);
    // buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C-derajatAngkat_C);

    //B turun
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B);
```

```
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B);
    delay(WAKTU_SERVO);
    //tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C);
    //buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C);

    //B mundur
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A);
    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A);
    //delay(WAKTU_SERVO);

}
//-------------------------------------------------------------------------------sampai ini-- ↙
    ------------------------------------------------
//method setMaju() adalah method kaki melakukan
//gerakan maju kaki kiri dulu setelah itu kaki kanan          ------------    yang lama --------- ↙
    ----
void KakiClass::setMaju(){


    myssc.servoMove(PIN_SERVO_1C, PIJAK_POS_1C);
    myssc.servoMove(PIN_SERVO_2C, PIJAK_POS_2C);
    myssc.servoMove(PIN_SERVO_3C, PIJAK_POS_3C);
    myssc.servoMove(PIN_SERVO_4C, PIJAK_POS_4C);
    myssc.servoMove(PIN_SERVO_5C, PIJAK_POS_5C);
    myssc.servoMove(PIN_SERVO_6C, PIJAK_POS_6C);
    //--------------------   maju posisi lingkaran
    //    1 3 5
    //B angkat
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B - derajatAngkat_B); //10
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B + derajatAngkat_B);
    //A geser
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A - derajatGeserMaju_A);      //8
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A - derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A + derajatGeserMaju_A);        //13
    delay(WAKTU_SERVO);
    //buka
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C+derajatAngkat_C);            //11
    //tutup
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C-derajatAngkat_C);               //7
    //B turun
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B);
    delay(WAKTU_SERVO);
    //tutup
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C);
    //buka
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C);
    //A mundur
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A);
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A);
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A);
    delay(WAKTU_SERVO);

    //2 4 6
    //B angkat
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B + derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B + derajatAngkat_B);

    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A + derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A + derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A - derajatGeserMaju_A);
    delay(WAKTU_SERVO);
    // tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C+derajatAngkat_C);
    // buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C-derajatAngkat_C);

    //B turun
```

```
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B);
    delay(WAKTU_SERVO);
    //tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C);
    //buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C);

    //B mundur
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A);
    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A);
    //delay(WAKTU_SERVO);

}

//method setMundur() adalah method kaki melakukan
//gerakan mundur kaki kiri dulu setelah itu kaki kanan
void KakiClass::setMundur(){

    //myssc.servoMove(PIN_SERVO_2C,PIJAK_POS_2C);
    //myssc.servoMove(PIN_SERVO_5C,PIJAK_POS_5C);
    //--------------------   mundur posisi lingkaran
    //    1 3 5
    //B angkat
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B - derajatAngkat_B); //10
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B + derajatAngkat_B);
    //A geser
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A + derajatGeserMaju_A);      //8
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A + derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A - derajatGeserMaju_A);        //13
    delay(WAKTU_SERVO);
    //buka
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C+derajatAngkat_C);   //-2         //11
    //tutup
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C+derajatAngkat_C);               //7
    //B turun
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B);
    delay(WAKTU_SERVO);
    //tutup
    //myssc.servoMove(PIN_SERVO_1C,PIJAK_POS_1C+derajatAngkat_C);
    //buka
    //myssc.servoMove(PIN_SERVO_3C,PIJAK_POS_3C);
    //A mundur
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A);
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A);
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A);
    delay(WAKTU_SERVO);

    //2 4 6
    //B angkat
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B - derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B + derajatAngkat_B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B + derajatAngkat_B);

    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A - derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A - derajatGeserMaju_A);
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A + derajatGeserMaju_A);
    delay(WAKTU_SERVO);

    // tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C-derajatAngkat_C);
    // buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C-derajatAngkat_C);//+2

    //B turun
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B);
    delay(WAKTU_SERVO);
```

```
    //tutup
    //myssc.servoMove(PIN_SERVO_4C,PIJAK_POS_4C);
    //buka
    //myssc.servoMove(PIN_SERVO_6C,PIJAK_POS_6C-derajatAngkat_C);

    //B mundur
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A);
    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A);
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A);
}

//method setPijakKiri() adalah method kaki kiri melakukan---------- berdiri diam---------- ↙
    ------
//pijak dengan derajat jalan
void KakiClass::setPijakBiasa()
{
    //kaki L1
    myssc.servoMove(PIN_SERVO_1A, PIJAK_POS_1A);
    myssc.servoMove(PIN_SERVO_1B, PIJAK_POS_1B);
    myssc.servoMove(PIN_SERVO_1C, PIJAK_POS_1C);

    //kaki R2
    myssc.servoMove(PIN_SERVO_2A, PIJAK_POS_2A);
    myssc.servoMove(PIN_SERVO_2B, PIJAK_POS_2B);
    myssc.servoMove(PIN_SERVO_2C, PIJAK_POS_2C);

    //kaki L3
    myssc.servoMove(PIN_SERVO_3A, PIJAK_POS_3A);
    myssc.servoMove(PIN_SERVO_3B, PIJAK_POS_3B);
    myssc.servoMove(PIN_SERVO_3C, PIJAK_POS_3C);

    //kaki R4
    myssc.servoMove(PIN_SERVO_4A, PIJAK_POS_4A);
    myssc.servoMove(PIN_SERVO_4B, PIJAK_POS_4B);
    myssc.servoMove(PIN_SERVO_4C, PIJAK_POS_4C);

    //kaki L5
    myssc.servoMove(PIN_SERVO_5A, PIJAK_POS_5A);
    myssc.servoMove(PIN_SERVO_5B, PIJAK_POS_5B);
    myssc.servoMove(PIN_SERVO_5C, PIJAK_POS_5C);

    //kaki R6
    myssc.servoMove(PIN_SERVO_6A, PIJAK_POS_6A);
    myssc.servoMove(PIN_SERVO_6B, PIJAK_POS_6B);
    myssc.servoMove(PIN_SERVO_6C, PIJAK_POS_6C);
}

void KakiClass::setKakiKi(float _kakiKi){
    kakiKi = _kakiKi;
}

void KakiClass::setKakiKa(float _kakiKa){
    kakiKa = _kakiKa;
}

float KakiClass::getKakiKi(){
    return kakiKi;
}

float KakiClass::getKakiKa(){
    return kakiKa;
}


KakiClass Kaki;
```

```
//
//
//

#include "FuzzyESadewa.h"
#include "Kaki.h"
#include "InitSRF.h"

//=========================
      // Pembuatan Objek
     Fuzzy* fuzzy = new Fuzzy();

     //=================== INPUT ====================//
     FuzzySet* srfD_D = new FuzzySet(0, 9, 9, 18);
     FuzzySet* srfD_A = new FuzzySet(15, 107, 107, 170);      //--------membersib SRF depan


     FuzzySet* srfKaPo_min1 = new FuzzySet(-1, -1, -1, -1);
     FuzzySet* srfKaPo_SD = new FuzzySet(0, 6, 6, 12);            //-------- membersip SRF kanan ↙
     Pojok
     FuzzySet* srfKaPo_D = new FuzzySet(11, 12.5, 12.5, 14);
     FuzzySet* srfKaPo_A = new FuzzySet(13, 14, 14, 15);
     FuzzySet* srfKaPo_J = new FuzzySet(14, 15.5, 15.5, 17);
     FuzzySet* srfKaPo_SJ = new FuzzySet(16, 83, 83, 150);

     FuzzySet* srfKiPo_min1 = new FuzzySet(-1, -1, -1, -1);
     FuzzySet* srfKiPo_SD = new FuzzySet(0, 6, 6, 12);            //---------membersip SRF kiri ↙
    Pojok
     FuzzySet* srfKiPo_D = new FuzzySet(11, 12.5, 12.5, 14);
     FuzzySet* srfKiPo_A = new FuzzySet(13, 14, 14, 15);
     FuzzySet* srfKiPo_J = new FuzzySet(14, 15.5, 15.5, 17);
     FuzzySet* srfKiPo_SJ = new FuzzySet(16, 83, 83, 150);

     //=================== OUTPUT ====================//

     // Kaki Kiri
     FuzzySet* kakiKi_SD = new FuzzySet(0, 10, 70, 80);
     FuzzySet* kakiKi_D = new FuzzySet(70, 80, 150, 160);           //-------- kaki kiri
     FuzzySet* kakiKi_S = new FuzzySet(150, 160, 200, 210);
     FuzzySet* kakiKi_B = new FuzzySet(200, 210, 270, 350);
     FuzzySet* kakiKi_SB = new FuzzySet(350, 360, 390, 400);

     // Kaki Kanan
     FuzzySet* kakiKa_SD = new FuzzySet(0, 10, 70, 80);
     FuzzySet* kakiKa_D = new FuzzySet(70, 80, 150, 160);
     FuzzySet* kakiKa_S = new FuzzySet(150, 160, 200, 210);         //-------- kaki kanan
     FuzzySet* kakiKa_B = new FuzzySet(200, 210, 270, 350);
     FuzzySet* kakiKa_SB = new FuzzySet(350, 360, 390, 400);
     //=====================================================

void FuzzyESadewaClass::initFuzzyESadewa()
{
    //=================== INPUT ====================//

    // FuzzyInput untuk SRF Depan
    FuzzyInput* srfD = new FuzzyInput(1); //index 1
    //srfD->addFuzzySet(srfD_SD);
    srfD->addFuzzySet(srfD_D);
    srfD->addFuzzySet(srfD_A);
    //srfD->addFuzzySet(srfD_J);

    fuzzy->addFuzzyInput(srfD);

    // FuzzyInput untuk SRF Kanan Pojok
    FuzzyInput* srfKaPo = new FuzzyInput(2); //index 2
    srfKaPo->addFuzzySet(srfKaPo_min1);
    srfKaPo->addFuzzySet(srfKaPo_SD);
    srfKaPo->addFuzzySet(srfKaPo_D);
    srfKaPo->addFuzzySet(srfKaPo_A);
    srfKaPo->addFuzzySet(srfKaPo_J);
    srfKaPo->addFuzzySet(srfKaPo_SJ);

    fuzzy->addFuzzyInput(srfKaPo);
```

```
    // FuzzyInput untuk SRF Kiri Pojok
    FuzzyInput* srfKiPo = new FuzzyInput(3); //index 3
    srfKiPo->addFuzzySet(srfKiPo_min1);
    srfKiPo->addFuzzySet(srfKiPo_SD);
    srfKiPo->addFuzzySet(srfKiPo_D);
    srfKiPo->addFuzzySet(srfKiPo_A);
    srfKiPo->addFuzzySet(srfKiPo_J);
    srfKiPo->addFuzzySet(srfKiPo_SJ);

    fuzzy->addFuzzyInput(srfKiPo);

    //================== OUTPUT ======================//

    // FuzzyOutput untuk Kaki Kiri
    FuzzyOutput* kakiKi = new FuzzyOutput(1); //---------------hasil fuzzy output---------
    kakiKi->addFuzzySet(kakiKi_SD);
    kakiKi->addFuzzySet(kakiKi_D);
    kakiKi->addFuzzySet(kakiKi_S);
    kakiKi->addFuzzySet(kakiKi_B);
    kakiKi->addFuzzySet(kakiKi_SB);

    fuzzy->addFuzzyOutput(kakiKi);

    // FuzzyOutput untuk Kaki Kanan
    FuzzyOutput* kakiKa = new FuzzyOutput(2);//---------------hasil fuzzy output---------
    kakiKa->addFuzzySet(kakiKa_SD);
    kakiKa->addFuzzySet(kakiKa_D);
    kakiKa->addFuzzySet(kakiKa_S);
    kakiKa->addFuzzySet(kakiKa_B);
    kakiKa->addFuzzySet(kakiKa_SB);

    fuzzy->addFuzzyOutput(kakiKa);

    initRuleWFKi(); initRuleWFKa();
}


void FuzzyESadewaClass::setInput(int index, int crisp){
    fuzzy->setInput(index, crisp);
}

void FuzzyESadewaClass::fuzzifikasi(){
    fuzzy->fuzzify();
}

float FuzzyESadewaClass::getOutput(int index){
    return fuzzy->defuzzify(index);
}

//========================== RULE WF Kanan ================================//

void FuzzyESadewaClass::initRuleWFKa(){
    ruleWFKa_1(); ruleWFKa_2(); ruleWFKa_3(); ruleWFKa_4(); ruleWFKa_5();
    ruleWFKa_6(); ruleWFKa_7(); ruleWFKa_8(); ruleWFKa_9(); ruleWFKa_10();
}

void FuzzyESadewaClass::ruleWFKa_1(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKaPo_SD);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SD);
    maka->addOutput(kakiKa_SB);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(1, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_2(){
    //============== Building FuzzyRule =============//
```

```
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKaPo_D);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_B);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(2, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_3(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKaPo_A);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(3, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_4(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKaPo_J);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_B);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(4, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_5(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKaPo_SJ);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SB);
    maka->addOutput(kakiKa_SD);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(5, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}


void FuzzyESadewaClass::ruleWFKa_6(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKaPo_SD);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SD);
    maka->addOutput(kakiKa_SB);
```

```
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(6, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_7(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKaPo_D);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_B);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(7, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_8(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKaPo_A);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(8, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_9(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKaPo_J);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_B);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(9, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKa_10(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKaPo_SJ);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKiPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SB);
    maka->addOutput(kakiKa_SD);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(10, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}


//======================== RULE WF Kiri ================================//

void FuzzyESadewaClass::initRuleWFKi(){
    ruleWFKi_1(); ruleWFKi_2(); ruleWFKi_3(); ruleWFKi_4(); ruleWFKi_5();
```

```
    ruleWFKi_6(); ruleWFKi_7(); ruleWFKi_8(); ruleWFKi_9(); ruleWFKi_10();
}

void FuzzyESadewaClass::ruleWFKi_1(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKiPo_SD);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SB);
    maka->addOutput(kakiKa_SD);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(11, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_2(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKiPo_D);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_B);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(12, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_3(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKiPo_A);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(13, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_4(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKiPo_J);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_B);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(14, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_5(){
    //============= Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_D, srfKiPo_SJ);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
```

```
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SD);
    maka->addOutput(kakiKa_SB);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(15, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}


void FuzzyESadewaClass::ruleWFKi_6(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKiPo_SD);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SB);
    maka->addOutput(kakiKa_SD);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(16, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_7(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKiPo_D);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_B);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(17, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_8(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKiPo_A);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_S);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(18, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

void FuzzyESadewaClass::ruleWFKi_9(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKiPo_J);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_S);
    maka->addOutput(kakiKa_B);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(19, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}
```

```
void FuzzyESadewaClass::ruleWFKi_10(){
    //============== Building FuzzyRule =============//
    //deklarasi if
    FuzzyRuleAntecedent* jika1 = new FuzzyRuleAntecedent();
    jika1->joinWithAND(srfD_A, srfKiPo_SJ);
    FuzzyRuleAntecedent* jika2 = new FuzzyRuleAntecedent();
    jika2->joinWithAND(jika1, srfKaPo_min1);
    //deklarasi then
    FuzzyRuleConsequent* maka = new FuzzyRuleConsequent();
    maka->addOutput(kakiKi_SD);
    maka->addOutput(kakiKa_SB);
    //fuzzyRule
    FuzzyRule* fuzzyRule = new FuzzyRule(20, jika2, maka);
    fuzzy->addFuzzyRule(fuzzyRule);
}

FuzzyESadewaClass FuzzyESadewa;
```

Bila menginginkan program lebih detail dapat menghubungi email titodintikoshars@gmail.com

Biografi Penulis

Tito Dinti Koshars lahir pada tanggal 16 September 1988 di kabupaten Malang. Merupakan anak tunggal dari bapak Edi Lesriyanto dan ibu Emi Wijayanti. Penulis memulai pendidikan pada sekolahdasar SD Negeri 04 Sumberpucung kabupaten Malang pada tahun 1995 dengan lulus tahun 2001. Setelah menyelesaikan studi sekolah dasar penulis melanjutkan ke jenjang sekolah menengah pertama pada SMP Negeri 02 Sumberpucung pada tahun 2001yang kemudian di selesaikan pada tahun 2004. Setelah menempuh sekolah menengah pertama penulis melanjutkan studi ke tingkat sekolah kejuruan pada SMK Brantas Karangkates pada tahun 2004 yang di selesaikan pada tahun 2007. Setelah menyelesaikan studi SMK penulis melanjutkan bekerja di Perusahaan Engineering sebagai Administrasi Project dari tahun 2007 sampai tahun 2011. Setelah itu penulis melanjutkan studi ke jenjang perkuliahan dengan menempuh S-1 Teknik Elektro pada konsentrasi Teknik Elektronika S-1 Institut Teknologi Nasional Malang. Dengan menempuh studi S-1 selama delapan semester penulis menyelesaikan Sarjana Teknik pada tanggal 26 September 2015.

Robotik Team Institut Teknologi Nasional