TUGAS AKHIR – TF 141581

RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE *FUZZY LOGIC*

AMALIA PURUHITA
NRP. 02311440000119

Dosen Pembimbing:
Dr. Purwadi Agus Darwito, S.T., M.T.
Andi Rahmadiansah, S.T., M.T.

DEPARTEMEN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TF 141581

DESIGN OF PROTOTYPE AUTONOMOUS VEHICLE STEERING CONTROL USING FUZZY LOGIC

AMALIA PURUHITA
NRP. 02311440000119

Supervisors:
Dr. Purwadi Agus Darwito, S.T., M.T.
Andi Rahmadiansah, S.T., M.T.

ENGINEERING PHYSICS DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan dibawah ini:

Nama : Amalia Puruhita
NRP : 02311440000119
Departemen/ Prodi : Teknik Fisika/ S1 Teknik Fisika
Fakultas : Fakultas Teknologi Industri
Perguruan Tinggi : Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Rancang Bangun Pengendali Kemudi Prototipe Kendaraan *Autonomous* dengan Metode Fuzzy Logic" adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, 19 Juli 2018
Yang membuat pernyataan,

Amalia Puruhita
NRP. 02311440000119

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

TUGAS AKHIR

RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE FUZZY LOGIC

Oleh:
Amalia Puruhita
NRP. 02311440000119

Surabaya, 19 Juli 2018

Menyetujui, Dosen Pembimbing I	Menyetujui, Dosen Pembimbing II
Dr.Purwadi Agus Darwito, S.T., M.T.	**Andi Rahmadiansah, S.T., M.T.**
NIPN. 19620822 198803 1 001	**NIPN. 19790517 20031 1 002**

Mengetahui,
Kepala Departemen
Teknik Fisika, FTI-ITS

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI INDUSTRI DEPARTEMEN TEKNIK FISIKA

Agus M. Hatta, S.E., M.Si., Ph.D
NIPN. 19780902 200312 1 002

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN II
RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE FUZZY LOGIC

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Instrumentasi
Program Studi S-1 Departemen Teknik Fisika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:
AMALIA PURUHITA
NRP. 02311440000119

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Purwadi Agus Darwito, S.T., M.T ..... (Pembimbing I)
2. Andi Rahmadiansah, S.T., M.T ..... (Pembimbing II)
3. Dr. Ir. Totok Soehartanto, DEA ..... (Ketua Penguji)
4. Dr. Suyanto, M.T ..... (Penguji I)
5. Ir. Roekmono, M.T ..... (Penguji II)

SURABAYA
JULI, 2018

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena rahmat dan hikmat-Nya serta sholawat dan salam kepada Nabi Muhammad SAW sehingga penulis diberikan kesehatan, kemudahan, dan kelancaran dalam menyusun laporan Tugas Akhir yang berjudul:

"RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE FUZZY LOGIC"

Penulis telah banyak memperoleh bantuan dari berbagai pihak dalam penyelesaian Tugas Akhir dan laporan Tugas Akhir ini. Penulis mengucapkan terima kasih kepada :

1. Bapak Agus Muhamad Hatta, ST, MSi, Ph.D selaku Ketua Departemen Teknik Fisika ITS.
2. Bapak Dr. Purwadi Agus Darwito, S.T., M.T. dan Bapak Andi Rahmadiansah, S.T., M.T. selaku dosen pembimbing yang telah dengan sabar memberikan petunjuk, ilmu, serta bimbingan yang sangat bermanfaat kepada penulis.
3. Kedua orang tua serta saudara penulis atas segala cinta, kasih sayang, doa, perhatian, serta dukungan moril dan materiil yang telah diberikan untuk penulis.
4. Seluruh dosen, karyawan dan civitas akademik Teknik Fisika FTI-ITS atas segala bantuan dan kerjasamanya.
5. Ergi Qibar atas bantuan dan dukungannya selama mengerjakan tugas akhir ini.
6. Teman – teman Laboratorium Simulasi dan Komputasi E205 (Niken, Deni, Juniar, Munir, Fathur, Dyah, Mba Novia) atas segala dukungan, bantuan, dan ilmu – ilmu yang dipelajari bersama sejak semester 3.
7. Teman – teman F49 atas dukungannya selama mengerjakan tugas akhir ini.

Penulis menyadari bahwa mungkin masih ada kekurangan dalam laporan ini, sehingga kritik dan saran penulis terima.

Semoga laporan ini dapat berguna dan bermanfaat bagi penulis dan pihak yang membacanya.

Surabaya, 26 Juli 2018

Penulis

RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE *FUZZY LOGIC*

Nama Mahasiswa : Amalia Puruhita
NRP : 0231144000119
Jurusan : Teknik Fisika FTI-ITS
Dosen Pembimbing I : Dr. Purwadi Agus Darwito, S.T., M.T.
Dosen Pembimbing II : Andi Rahmadiansah, S.T., M.T.

ABSTRAK

Penelitian ini bertujuan untuk merancang sistem kendali serta mengimplementasikannya pada *steering* prototipe *autonomous vehicle* menggunakan *fuzzy logic controller* agar mampu menghindari halangan serta berada pada lintasan yang benar. Prototipe ini dapat mengetahui jarak dari *obstacle* pada lintasannya dengan menggunakan 3 sensor ultrasonik yang dipasang pada posisi yang telah ditentukan (kanan mobil, depan mobil, kiri mobil), motor dc merupakan aktuator dari penggerak dan kemudi dari prototipe. Tujuan dari penelitian ini yaitu mendapatkan parameter dan spesifikasi agar prototipe dapat bermanuver belok dan mampu bekerja secara terintegrasi dengan hasil rancangan prototipe. Parameter sistem kendali hasil rancangan memiliki 4 *input* dan 2 *output*. *Input* pada sistem kendali ini terdiri dari jarak dari *RightUltrasonic*(s), *FrontUltrasonic*(s), dan *LeftUltrasonic*(s) memiliki *range* 0 cm – 200 cm, *input* keempat merupakan nilai kecepatan pada motor penggerak (*RPMMot* (ωs) dengan *range* 3765 RPM – 12000 RPM. *Output* pada sistem kendali ini terdiri dari *SteerAngle* (θ) dan kecepatan motor penggerak *RPMMot*(ωf) dengan range secara berurutan yaitu -450 - 450 dan 3765 RPM – 12000 RPM. Spesifikasi yang ditentukan pada prototipe agar dapat terintegrasi dan menghasilkan performa baik antara lain menetapkan nilai pembacaan minimum hingga maksimum dari prototipe. Prototipe diberikan pembacaan minimal 10 cm agar dapat mencegah *crash* antara prototipe dan *obstacle* serta dapat membaca *obstacle* dengan tinggi

yang kurang dari tinggi peletakkan sensor pada prototipe sebesar 2.1 cm, dan pembacaan maksimal sebesar 200 cm. Kecepatan RPM motor penggerak diberi kecepatan awal ωs 3765 RPM berdasarkan kemampuan prototipe dapat berjalan dengan beban 275 gram dan diberikan kecepatan maksimum sebesar 12000 RPM. Rancang bangun pengendali kemudi prototipe *autonomous vehicle* ini telah berhasil dirancang dengan algoritma *fuzzy logic* dengan melakukan validasi pengujian dengan lintasan yang telah ditentukan kondisinya.

Kata kunci: Sistem Kendali, Pengendalian Kemudi Otomatis, Logika Fuzzy, Kendaraan Otonom.

DESIGN OF STEERING CONTROL PROTOTYPE AUTONOMOUS VEHICLE USING FUZZY LOGIC

Name	**: Amalia Puruhita**
NRP	**: 0231144000119**
Department	**: Teknik Fisika FTI-ITS**
Supervisor I	**: Dr. Purwadi Agus Darwito, S.T., M.T.**
Supervisor II	**: Andi Rahmadiansah, S.T., M.T.**

ABSTRACT

This study aims to design a control system and implement it on autonomous vehicle prototype steering using fuzzy logic controller to avoid obstacles and be on the right track. This prototype can detect the distance from the obstacle on its path by using 3 ultrasonic sensors mounted at a predetermined position (right car, front car, left car), dc motor is the actuator of the driving and steering of the prototype. The purpose of this research is to get the parameters and specifications so that the prototype can maneuver turn and able to work in an integrated with the design of the prototype. The design result control parameter has 4 inputs and 2 outputs. Input on this control system consists of the distance from RightUltrasonic (s), FrontUltrasonic (s), and LeftUltrasonic (s) has a range of 0 cm - 200 cm, the fourth input is the speed value of the driving force (RPMMot (ωs) with a range of 3765 RPM - 12000 RPM. The output of this control system consists of SteerAngle (θ) and RPMMot drive speed (ωf) with sequence ranges of -450 - 450 and 3765 RPM - 12000 RPM. Specifications are specified on the prototype to be integrated and produce good performance among others, sets a minimum to maximum reading value of the prototype. The prototype is given a minimum reading of 10 cm in order to prevent crashes between prototype and obstacle and can read obstacles with a height less than the height of placing the sensor on a prototype of 2.1 cm, and a maximum reading of 200 cm. Speed RPM driven motor is given initial speed ωs 3765 RPM based on prototype capability can run with load 275 g ram and is given a maximum speed of 12000 RPM. The design of autonomous vehicle prototype steering wheel control system has

been successfully designed with fuzzy logic algorithm by performing validation testing with predetermined path.

Keywords**:** *Control System, Automatic Steering Control, Fuzzy Logic, Autonomous Vehicle.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR NOTASI

Notasi	Satuan	Keterangan
ω	$[RPM]$	Kecepatan Motor DC penggerak *Prototype*
ω_s	$[RPM]$	Kecepatan awal (start) Motor DC Penggerak *Prototype*
ω_f	$[RPM]$	Kecepatan akhir (final) Motor DC Penggerak *Prototype*
θ	$[degree°]$	Sudut belok kemudi *Prototype*
s	[cm]	Jarak

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Teknologi di dunia industri ditandai dengan adanya kecenderungan pemanfaatan otomatisasi, termasuk di dalamnya adalah penggunaan teknologi robotika. Teknologi robotika tersebut bertujuan untuk membantu menyelesaikan tugas – tugas manusia, karena manusia memiliki keterbatasan kemampuan seperti: cepat lelah, kurangnya konsentrasi, tidak dapat mempertahankan presisi, dan dapat menghindari resiko kerja yang membahayakan manusia. Teknologi ini dirancang untuk dapat membantu kekurangan manusia tersebut dan dapat melakukan pekerjaan(Ingrand & Ghallab, n.d.). Teknologi robotik yang diperlukan salah satunya adalah otomatisasi. Otomatisasi bertujuan untuk membantu kekurangan – kekurangan yang dimiliki oleh manusia. Otomatisasi dalam dunia otomotif salah satunya adalah *Autonomous Vehicle* atau mobil tanpa pengemudi. Kendaraan *autonomous* ini adalah kendaraan yang dapat menjelajahi lingkungannya dan bergerak tanpa penggerakan dari manusia(Litman, 2018).

Kendaraan *autonomous* dapat mengetahui informasi dari lingkungannya serta dapat memetakan lokasi *obstacle* dengan menggunakan alat deteksi *obstacle* sepert LiDAR, ultrasonik, inframerah, dan sebagainya (Hyoven & Huhtala, 2012) (Cui, Li & Liu, 2016). Sensor ultrasonik yang terpasang pada kendaraan dapat menedeksi jarak kendaraan pada *obstacle.* Jarak yang didapatkan merupakan informasi yang diproses agar kendaraan *autonomous* ini dapat bermanuver menghindari *obstacle*. Informasi tersebut akan diproses oleh *microcontroller* yang telah diberikan algoritma *fuzzy logic controller* didalamnya, agar dapat mengendalikan kemudi dari kendaraan untuk bergerak menghindari *obstacle*.

Fuzzy Logic Controller merupakan suatu metode pengendalian yang dapat mengendalikan sistem yang

nonlinearitas, ketidakpastian, dan kompleks. Kendaraan *autonomous* memiliki sistem yang kompleks karena memiliki nilai masukan yang tidak pasti tergantung dengan lingkungan yang dilaluinya (Woon, 2014) (Sailan, Kuhnert & Hardt, 2015). *Fuzzy logic controller* terdiri dari 3 poin yaitu fuzzifikasi, *rule base*/*inference system*, dan defuzzifikasi. Pengendalian yang dilakukan dengan metode *fuzzy logic controller* pada rancang bangun *prototype* sistem kemudi memberikan perintah pada motor dc sebagai penggerak dan ultrasonik sebagai navigasi kemudi.

Penelitian ini akan merancang sistem kendali pada *steering autonomous vehicle* metode Fuzzy Logic dengan mobil *remote control* yang memiliki 2 motor dc, masing – masing memiliki fungsi sendiri yaitu sebagai motor penggerak *prototype* dan motor penggerak kemudi terhadap *obstacle* suatu lingkungan. *Prototype* tersebut memiliki 3 sensor ultrasonik yang terpasang pada chassis depan mobil untuk dapat membaca jarak pada lingkungan yang dilintasi. Penelitian ini bertujuan untuk mengetahui nilai parameter sistem kendali *fuzzy logic* para *prototype* dan spesifikasi dari *prototype* agar menghasilkan performa baik pada saat terintegrasi dengan sistem kendali.

1.2 Rumusan Masalah

Adapun permasalahan yang muncul dari latar belakang tersebut adalah:

1. Parameter apa saja untuk sistem kendali yang sesuai untuk menggerakkan sistem kemudi agar memenuhi performansi yang diinginkan?
2. Bagaimana merancang bangun sistem pengendali kemudi *prototype autonomous vehicle* dengan metode *fuzzy logic*?
3. Komponen apa saja yang dapat digunakan dalam rancang bangun sistem pengendali kemudi *prototype autonomous vehicle*?

4. Apakah spesifikasi dari sistem kemudi agar mampu bekerja secara terintegrasi dengan sistem kendali hasil rancangan?

1.3 Tujuan

Tujuan utama dari penelitian ini adalah:

1. Menentukan parameter sistem kendali yang sesuai untuk menggerakkan sistem kemudi agar memenuhi performansi yang diinginkan.
2. Melakukan rancang bangun sistem pengendali kemudi *prototype autonomous vehicle* dengan metode *fuzzy logic.*
3. Menentukan komponen yang digunakan dalam rancang bangun sistem pengendali kemudi *prototype autonomous vehicle*
4. Mengetahui apakah spesifikasi dari sistem kemudi agar mampu bekerja secara terintegrasi dengan sistem kendali hasil rancangan.

1.4 Batasan Masalah

Adapun Beberapa batasan masalah pada penelitian ini adalah:

1. Rancang bngun sistem kendali dilakukan pada kemudi dai mobil *remote control*.
2. Kemudi mobil *remote control* akan dirancang dengan jenis kemudi yang digerakkan oleh motor dc menggunakan *gear* untuk kemudi roda.
3. Uji performansi dari sistem kendali mobil *remote control* dilakukan pada lintasan yang telah ditentukan yaitu menggunakan jalan satu jalur, pengujian menggunakan 4 kondisi lintasan (tanpa *obstacle*, dengan *obstacle* 10 cm X 10 cm, *obtsacle* 20 cm X 20 cm, dan *obstacle* 35cm X 20 cm) dan jalan tidak berlubang.
4. Sistem kendali yang digunakan pada penelitian ini yaitu sistem kendali berbasis *Fuzzy Logic* dengan

Microcontroller ARM (*Advanced RISC Machine*) *Teensy* 3.6.

5. Simulasi dilakukan menggunakan *software* MATLAB R2014a.

1.5 Sistematika Laporan

Secara sistematis, penyusunan laporan tugas akhir ini tersusun dalam lima bab dengan penjelasan sebagai berikut:

BAB I Pendahuluan

Bab ini berisi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, dan sistematika laporan.

BAB II Tinjauan Pustaka

Bab ini berisi mengenai teori-teori penunjang yang terkait dalam penulisan tugas akhir.

BAB III Metodologi Penelitian

Bab ini akan dijelaskan mengenai langkah-langkah yang telah dilakukan dalam penelitian.

BAB IV Analisis Data dan Pembahasan

Bab ini akan ditampilkan data dan analisa hasil simulasi beserta pembahasannya.

BAB V Kesimpulan dan Saran

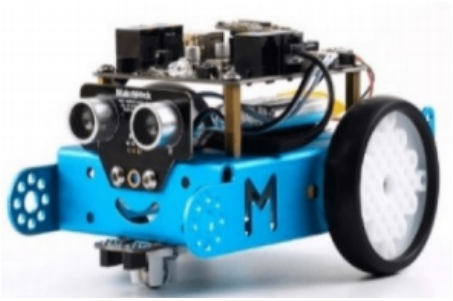
Bab ini berisi tentang kesimpulan pokok dari seluruh rangakaian penelitian yang telah dilakukan dan saran yang dapat dijadikan sebagai pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 *Autonomous Vehicle*

Autonomous Vehicle merupakan kendaraan yang mampu mengenal kondisi jalan dan kondisi jalurnya sehingga dapat mengarahkan dirinya ke sebuah tujuan tertentu tanpa bantuan manusia. Kemampuan untuk dapat mengenal jalan dan kondisi jalurnya, *autonomous vehicle* dapat dikombinasikan dengan pemasangan sensor ultrasonik, LiDAR (*Light Detection and Ranging)*, radar, kamera, dan sebagainya. Pada penelitian ini, perancangan *prototype autonomous vehicle* menggunakan *chassis* dari mobil *remote control* dan dipasang sensor ultrasonik untuk mendeteksi jarak mobil terhadap *obstacle*. Informasi jarak yang diterima dari sensor tersebut menjadi masukan dari *fuzzy logic controller* sebagai pengendali kemudi pada kendaraan agar dapat bermanuver sesuai dengan keluaran yang diberikan *fuzzy logic controller*.

Kendaraan *Autonomous* yang dikembangkan oleh (Ismail, Naim, Ayob & Amat, 2017) bernama *MBot*. *Mbot* menggunakan metode *fuzzy logic* untuk memberikan navigasi pada Mbot serta menggunakkan sensor ultrasonik sebagai pembacaan jarak terhadap *obstacle* yang ditemui.



Gambar 2.1 *Prototype Autonomomus Vehicle* dengan Sensor Ultrasonik (Mbot) (Ismail, Naim, Ayob & Amat, 2017)

2.2 Motor DC (*Direct Current*)

Motor dc memerlukan suplai tegangan yang searah pada kumparan medan untuk diubah menjadi energi mekanik. Kumparan medan pada motor dc tersebut disebut stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). tegangan yang berubah – ubah arah pada setiap setengah putaran ditimbulkan oleh terjadinya putaran pada kumparan jangkar pada medan magnet. Prinsip kerja dari arus searah adalah membalik phasa tegangan dari gelombang yang mempunyai nilai poisitif dengan menggunakan komutator, dengan demikian arus yang berbalik arah dengan kumparan jangkar yang berputar dalam medan magnet. Bentuk motor paling sederhana memiliki kumparan satu lilitan yang bisa berputar bebas di antara kutub – kutub magnet.

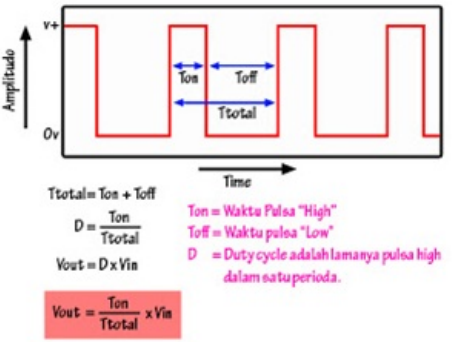
Gambar 2.2 Motor DC ("Miniature Motors", 2012)

Kumparan medan pada motor dc dialiiri arus listrik dan menghasilkan medan magnet yang melingkupi kumparan jangkar dengan arah tertentu. Arah putaran motor DC dapat diubah arahnya dengan cara dengan membalikkan tegangan *input* positif dan negatif dari motor dc tersebut (Bathia, n.d.).

2.2.1 PWM

Pulse Width Modulation (PWM) merupakan suatu cara memanipulasi lebar sinyal yang dinyatakan dengan pulsa dalam satu periode untuk mendapatkan tegangan rata-rata.

Sinyal PWM memiliki amplitudo dan frekuensi dasar yang tetap, tetapi memiliki lebar pulsa yang bervariasi dengan memvariasikan besar *duty cycle* dari 0% hingga 100%.



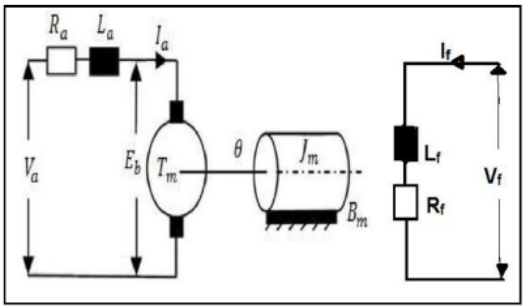
Gambar 2.3 Sinyal PWM dan Rumus Perhitungan (Sharma & Mohapatra, 2011)

PWM merupakan salah satu teknik untuk mendapatkan sinyal analog dari sebuah piranti digital. Perubahan sinyal PWM dipengaruhi oleh resolusi dari PWM. Resolusi (jumlah variasi perubahan nilai) dari PWM sebesar 8 bit ($2^8 = 256$) yaitu memiliki variasi mulai dari 0 – 255, perubahan nilai tersebut mewakili nilai *duty cycle* 0% - 100% dari keluaran PWM tersebut. Pengaturan lebar pulsa *On* dan *Off* dalam satu periode gelombang melalui pemberian besar sinyal referensi *output* dari suatu PWM maka akan didapatkan besar *duty cycle* yang diinginkan.

Nilai tegangan keluaran didapat dari berapa persentasi dari *duty cycle* yang diberikan, jika diberikan nilai *duty cycle* sebesar 100% maka tegangan akan dilewatkan seluruhnya (Sharma & Mohapatra, 2011).

2.2.2 Pemodelan Motor DC

Motor DC bekerja berdasarkan prinsip gaya elektromagnetik sehingga apabila motor tersebut diberi catu daya, arus akan mengalir ke dalam motor kemudian meghasilkan torsi yang sebanding dengan arus tersebut. Pemodelan rangkaian internal Motor DC dengan beban dan analisisnya adalah sebagai berikut:



Gambar 2.4 Motor DC (Andika, 2018)

Persamaan torsi yang dibangkitkan oleh Motor DC dapat didekati secara linear menurut persamaan berikut ini:

$T = K_a i$.....(2.1)

Dimana K_a adalah konstanta angkar motor yang bergantung pada banyaknya lilitan pada jangkar, jumlah kutub medan, tipe belitan dan penampang jangkarnya. Adapun besarnya tegangan ggl induksi lawan yang dibangkitkan motor ketika berputar adalah sebanding dengan konstanta motor K_b dan kecepatan sudut putaran motor ω.

$e = K_b \omega$.....(2.2)

Dengan menggunakan hukum newton, bahwa persaman torsi yang terkait dengan momen inersia dan rasio redaman dari motor:

$T = J \frac{d\omega}{dt} + b\omega$.....(2.3)

Dari persamaan (2.1) dan (2.3) diperoleh:

$$i = \frac{J\frac{d\omega}{dt}+b\omega}{K_b} \quad \text{.....(2.4)}$$

Sedangkan besarnya tegangan V menurut hukum kirchoff adalah:

$$V = iR + L\frac{di}{dt} + e \quad \text{.....(2.5)}$$

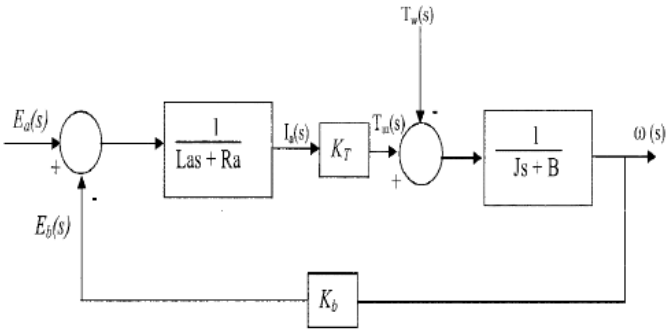
$$V = iR + L\frac{di}{dt} + K_b\omega \quad \text{.....(2.6)}$$

Dengan dilakukan laplace pada persamaan (2.3) dan (2.6) dan mensubtitusikannya, maka diperoleh fungsi transfer antara kecepatan sudut motor ω terhadap tegangan armature V dimana:

$K_a = K_b$

$$\frac{\omega(s)}{V(s)} = \frac{K}{s^2LJ+s(JR+Lb)+bR+K^2} \quad \text{.....(2.7)}$$

Motor DC memiliki nilai torsi, dimana nilai torsi tersebut dapat mempengaruhi kecepatan dari motor dc. Torsi adalah kemampuan putar suatu benda hingga benda tersebut dapat bergerak/berputar. Torsi dapat di rumuskan seperti pada persamaan 2.3, dimana nilai torsi tersebut dipengaruhi oleh gaya (F).



Gambar 2.5 Blok Diagram Sistem Motor DC (Khuswah and Wadhwani, 2013)

Diagram blok motor dc pada **Gambar 2.5** terdapat *disturbance* variabel torsi. Variabel torsi tersebut dihasilkan dari nilai gaya (F) dikalikan dengan jari – jari roda motor dc (r) dengan persamaan sebai berikut:

$T = F \times r$.....(2.8)

T = Torsi (N.m)

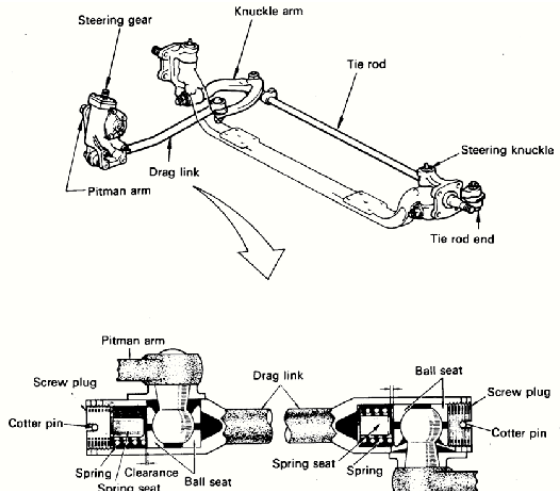
F = Gaya (N)

r = Jari – jari Motor DC (m)

Gaya (F) pada motor dc merupakan nilai yang didapatkan dari massa (m) beban di kali dengan percepatan gravitasi bumi (g) sebesar 9.81 m/s^2, sehingga massa (m) beban pada motor dc mempengaruhi nilai torsi motor dc. ((Khuswah and Wadhwani, 2013)

2.3 Sistem Kemudi *Steering Linkage*

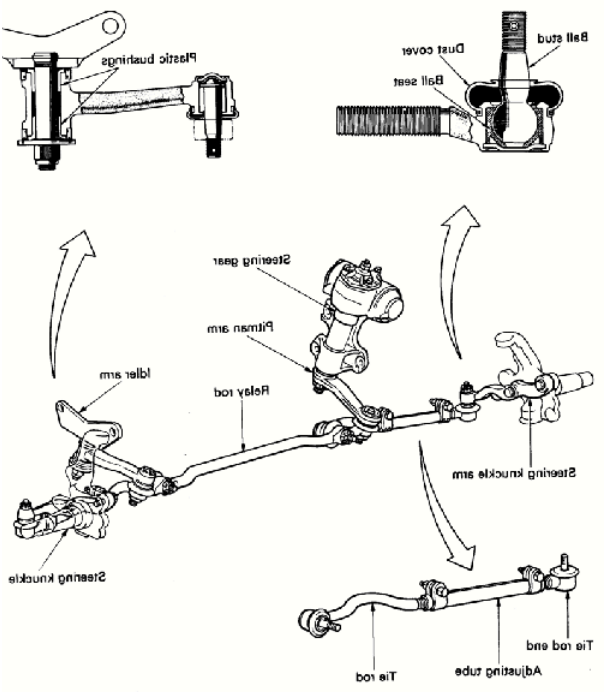
Sistem kemudi *steering linkage* terdiri dari *tie rod* dan *arm*, dan berfungsi untuk meneruskan tenaga gerak dari *steering gear* ke roda depan. Sistem kemudi ini terdiri dari dua



Gambar 2.6 *Steering Linkage* untuk Suspensi Rigid (Yanuary, 2010)

tipe, yaitu *steering linkage* untuk suspensi rigid dan *steering linkage* untuk suspensi *independent*. *Steering linkage* untuk suspensi rigid terdiri dari *pitman arm*, *drag link*, *knuckle arm*, *tie rod*, dan *tie rod end*. *Tie rod* mempunyai pipa yang berfungsi untuk menyetel panjang *tie rod.*

Steering linkage untuk suspensi *independent* terdiri dari sepasang *tie rod* yang dihubungkan oleh *relay rod* sebuah pipa yang dipasang di antara *tie rod end* untuk menyetel panjang *rod*.



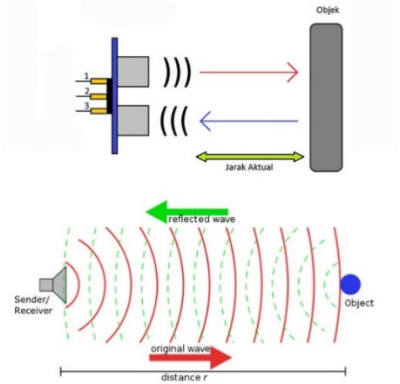
Gambar 2.7 *Steering Linkage* untuk Suspensi *Independent* (Yanuary, 2010)

2.4 Sensor Ultrasonik HC-SR05

Sensor ultrasonik merupakan sensor yang bekerja berdasarkan prinsip pantulan gelombang suara dan digunakan untuk mendeteksi keberadaan suatu objek tertentu didepannya.

Gambar 2.8 Sensor Ultrasonik HC-SR05 (Lim, Keoh & Thing, 2018)

Gelombang ultrasonik merupakan gelombang yang memiliki frekuensi diatas frekuensi gelombang suara 20 kHz. Sensor ultrasonik terdiri dari rangkaian pemancar ultrasonik (*transmitter*) dan rangkaian penerima ultrasonik (*receiver*). Sinyal ultrasonik akan dipancarkan dari *transmitter* ultrasonik, pada saat sinyal tersebut mengenai benda penghalang maka sinyal akan dipantulkan dan diterima oleh *receiver* ultrasonik. Sinyal yang diterima *receiver* akan dikirimkan ke rangkaian mikrokontroler untuk menghitung jarak terhadap *obstacle* yang berada di depannya (bidang pantul).



Gambar 2.9 Prinsip Kerja Sensor Ultrasonik (Lim, Keoh & Thing, 2018)

2.5 Mikrokontroler ARM (*Advanced RISC Machine*)

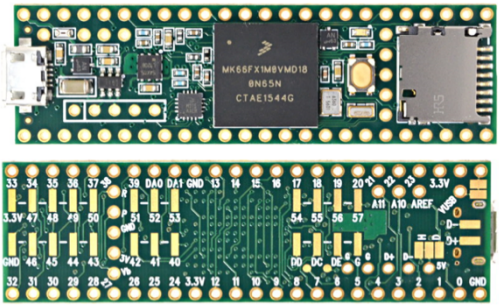
ARM adalah prosesor dengan arsitektur set instruksi 32bit RISC (*Reduced Instruction Set Computer*) yang dikembangkaan oleh ARM *Holdings*. Prosesor ARM digunakan di berbagai bidang seperti elektronik umum, termasuk *PDA*, *mobile phone*, *media player*, *music player*, *game console* genggam, kalkulator dan periperal komputer seperti *hard disk driver* dan *router*. Arsitektur ARM memiliki fitur *register file* yang berkapasitas besar. Arsitektur ARM memiliki arsitektur *load/store*, dimana operasi pengolahan data hanya beroperasi pada konten register dan tidak secara langsung pada konten memori. *Addressing mode* untuk *load/store* pada ARM ditentukan dari konten *register* dan *field* instruksi saja.

2.5.1 ARM *Teensy* 3.6

Teensy USB *Development Board* merupakan perkembengan sistem mikrokontroler berbasis USB (*Unit Serial Bus*). Berikut merupakan spesifikasi dari *Teensy* 3.6:

- 180 MHz ARM Cortex-M4 dengan *Floating Unit Point*
- 1M Flash, 256K, 4K EEPROM
- *Microcontroller Chip* MK66FX1M0VMD18
- USB *High Speed* (480Mbit/sec) *Port*
- 2 CAN *Bus Ports*
- 32 *General Purpose DMA Channels*
- *22* PWM *Outputs*
- 4 I^2C *Ports*
- 11 *Touch-Sensing Input*
- 62 I/O *Pins* (42 *breadboard friendly*)
- 25 *Analog Inputs to 2* ADCs *with 13-bit resolution*
- 2 *Analog Outputs* (DACs) *with 12-bit resolution*
- USB *full speed* (12Mbit/sec) *Port*
- *Ethernet mac, capable of full 100*Mbit/sec *speed*
- *Native* (4-bit SDIO) *micro* SD *card port*

- I^2S *Audio Port,* 4-*Channel Digital Audio Input & Output*
- 14 *Hardware Timers*
- *Cryptographic Acceleration Unit*
- *Random Number Generator*
- CRC *Computation Unit*
- 6 *Serial Ports* (2 *with* FIFO *and Fast Baud Rates*)
- 3 SPI *Ports* (1 *with* FIFO)
- *Real-Time Clock*
- 62.3 mm x 18.0 mm x 4.2mm



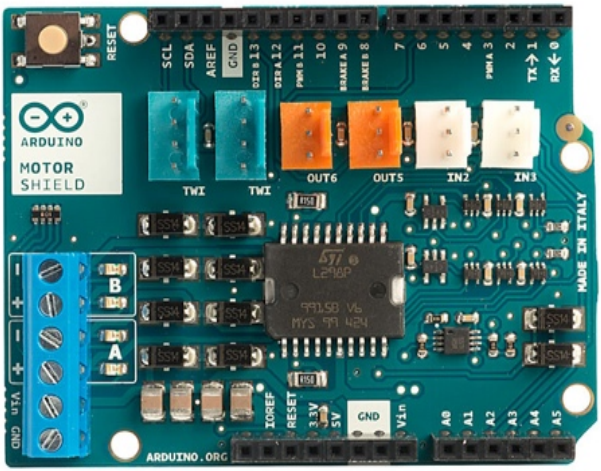
Gambar 2.10 *Teensy 3.6* dengan Mikrokontroler ARM Cortex-M4 (Sultana & Manzke, 2018)

2.6 *Arduino Motor Shield*

Arduino Motor Shield berfungsi sebagai *motor driver* untuk motor dc. *Arduino Motor Shield* berdasarkan *datasheet* dari L298 (IC *driver motor*). *Arduino Motor Shield* memiliki *dual ful bridge driver* yang dirancang untuk menjalankan beban induksi seperti *relay*, solenoid, motor dc maupun motor *stepper*. *Arduino Motor Shield* dapat menjalankan dua motor dc yang terhubung dengan *board* mikrokontroler. *Motor Shield* ini dapat mengatur kecepatan dan arah dari setiap motor yang dihubungkan. Spesifikasi dari *Arduino Motor Shield* adalah sebagai berikut:

Tabel 2.1 Spesifikasi *Arduino Motor Shield*

Operating Voltage	5 Volt – 12 Volt
Motor Controller	L298P, *Drives* 2 *DC Motor or* 1 *stepper motor*
Arus Maximum	2A *per channel* or 4A *max* (*with external power supply*)
Current Sensing	1.65V/A
Free running stop and brake function	



Gambar 2.11 *Arduino Motor Shield Rev3* ("Arduino Motor Shield Rev3", 2018)

2.7 *Fuzzy Logic*

Fuzzy logic merupakan tingkatan logika *Boolean* yang mendeskripsikan konsep kebenaran sebagian. Logika *fuzzy* dapat menentukan keputusan berbasis sebab dan akibat (*if-then*). Sistem *fuzzy* dapat menghasilkan keluaran yang kompleks melalui sistem yang sederhana(sumber). Sistem *fuzzy* pada penelitian ini memiliki 4 jenis *input* yaitu jarak yang didapat dari 3 sensor ultrasonik dan kecepatan PWM motor dc

penggerak mobil *remote control* dan 2 jenis *output* yaitu sudut putar roda dan kecepatan PWM motor dc mobil *remote control*.

Himpunan *fuzzy* merupakan rentang nilai – nilai, dimana nilai tersebut memiliki derajat keanggotaan antara 0 hingga . suatu himpunan *fuzzy* A dalam semesta U dinyatakan dengan fungsi keanggotaan μA, dimana nilainya berada dalam interval [0,1], dapat dinyatakan dengan:

$$\mu A = U \rightarrow [0,1] \quad \text{.....(2.8)}$$

Himpunan *fuzzy* A dalam semesta U dinyatakan sebagai sekumpulan pasangan elemen u (u anggota U) dan derajat keanggotaannya dinyatakan sebagai berikut:

$$A = \{(u, \mu A\ (u)) | u\ \in U\} \quad \text{.....(2.9)}$$

Himpunan *fuzzy* dinotasikan dengan cara menulis pasangan elemen secara berurutan, dengan elemen pertama menunjukkan nama elemen dan elemen kedua menunjukkan nilai keanggotaannya. Semesta X didapatkan himpunan diskret, maka himpunan *fuzzy* A dapat dinotasikan sebagai:

$$A = \sum_{i=1}^{n} \frac{\mu A(xi)}{xi} \quad \text{.....(2.10)}$$

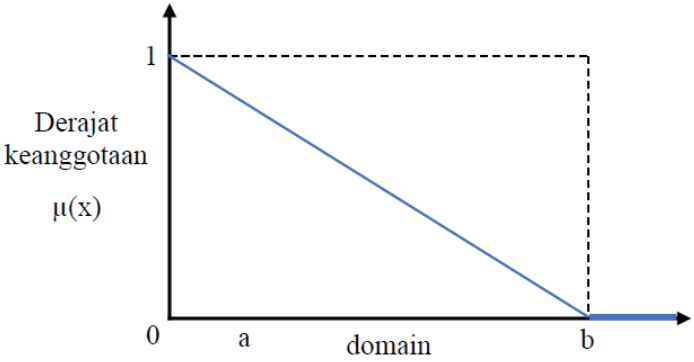
Semesta X didapatkan himpunan yang kontinu, maka himpunan *fuzzy* A dapat dinotasikan sebagai:

$$A = \int x \frac{\mu A(x)}{x} \quad \text{.....(2.11)}$$

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik – titik *input* data kedalam nilai keanggotaan yang memiliki interval antara 0 – 1. Representasi fungsi keanggotaan yang dapat digunakan yaitu:

a. Representasi Linear Turun:

   Garis linear yang dimulai dari nilai domain dengan derajat keanggotaan tertinggi atau satu, kemudian bergerak turun ke nilai domain yang memiliki derajat keanggotaan lebih rendah.



Gambar 2.12 Representasi Linear Turun (Firdaus, Syaryadhi and Rahman, 2017)

Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} \frac{(b-x)}{(b-a)}, & a \leq x < b \\ 0, & x \geq b \end{cases} \quad \text{.....(2.12)}$$

Keterangan :

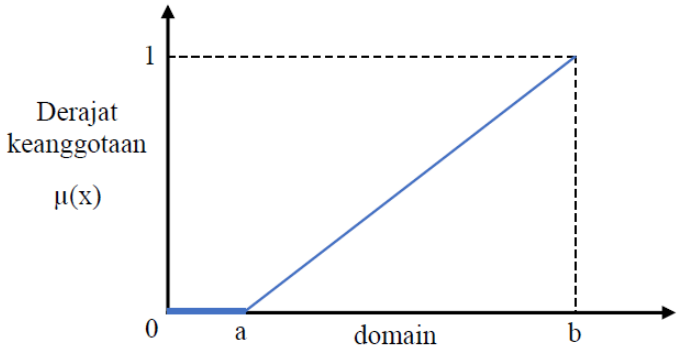
a = nilai domain yang mempunyai derajat keanggotan satu

b = nilai domain yang mempunyai derajat keanggotan nol

x = nilai input yang akan diubah kedalam bilangan Fuzzy

b. Representasi Linear Naik

Kenaikan himpunan dimulai dari nilai domain yang memiliki nilai keanggotaan rendah atau nol bergerak kekanan menuju ke nilai domain yang memiliki derajat keanggotaan yang lebih tinggi.



Gambar 2.43 Representasi Linear Naik (Firdaus, Syaryadhi and Rahman, 2017)

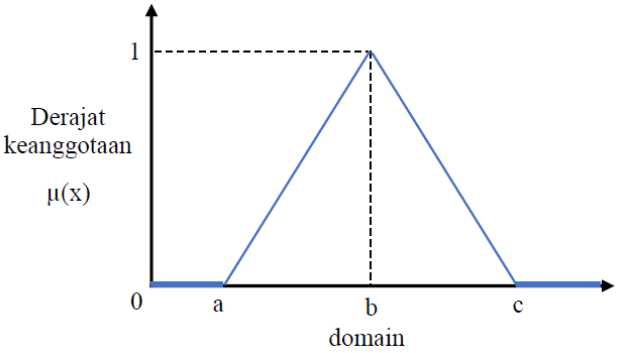
Fungsi keanggotaan:

$$\mu(x) = \begin{cases} 0, & x \leq a \\ \frac{(x-a)}{(b-a)}, & a < x \leq b \end{cases} \quad \text{.....(2.13)}$$

Keterangan :
a = nilai domain yang mempunyai derajat keanggotan nol
b = nilai domain yang mempunyai derajat keanggotan satu

x = nilai input yang akan diubah kedalam bilangan Fuzzy

c. Representasi Kurva Segitiga
Representasi kurva segitiga adalah gabungan antara dua representasi linear (linear naik dan linear turun)



Gambar 2.54 Representasi Kurva Segitiga (Firdaus, Syaryadhi and Rahman, 2017)

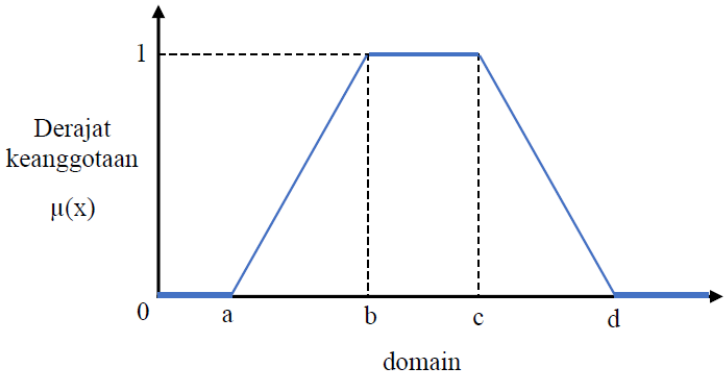
Fungsi Keanggotaan:

$$\mu(x) = \begin{cases} 0, & x \leq a \; dan \; x \geq c \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x < c \end{cases} \qquad \text{.....(2.14)}$$

Keterangan :
a = nilai domain terkecil memiliki derajat keanggotan nol
b = nilai domain memiliki derajat keanggotan satu
c = nilai domain terbesar memiliki derajat keanggotaan nol
x = nilai input yang akan diubah kedalam bilangan Fuzzy

d. Representasi Kurva Trapesium

Representasi kurva trapesium terdiri dari bentuk kurva segitga dan memiliki beberapa titik yang mempunyai nilai keanggotaan 1 (satu).



Gambar 2.65 Representasi Kurva Trapesium (Firdaus, Syaryadhi and Rahman, 2017)

Fungsi Keanggotaan:

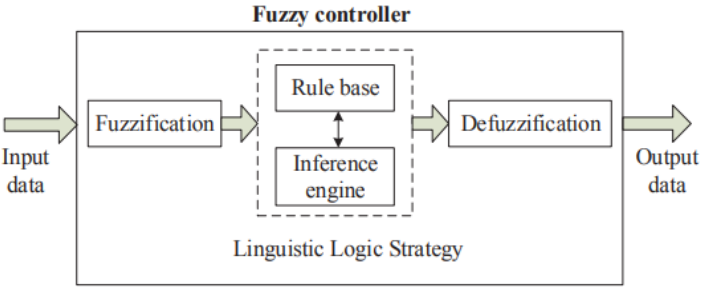
$$\mu(x) = \begin{cases} 0, & x \leq a \; dan \; x \geq d \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & b < x \leq c \\ \frac{d-x}{d-c}, & c < x < d \end{cases} \quad \text{.....(2.15)}$$

Keterangan :
a = nilai domain terkecil memiliki derajat keanggotan nol
b = nilai domain terkecil memiliki derajat keanggotan satu
c = nilai domain terbesar memiliki derajat keanggotaan satu
d = nilai domain terbesar memiliki derajat keanggotaan nol

x = nilai input yang akan diubah kedalam bilangan Fuzzy

Sistem inferensi *fuzzy* (FIS) yang digunakan untuk pengendalian kemudi *autonomous vehicle* ini menggunakan Metode Mamdani. Metode mamdani dapat bekerja berdasarkan

aturan – aturan linguistik. Proses – proses di dalam FIS adalah sebagai berikut:



Gambar 2.76 Arsitektur Sistem *Fuzzy*. (Rojas, Ponce & Molina, 2014)

2.7.1. *Fuzzification*

Fuzzyfication adalah proses pemetaan nilai *crisp input* (numerik) ke dalam himpunan *fuzzy* dengan *domain* yang telah ditentukan dan dapat dinyatakan dengan suatu *membership function input*. Setelah mendapatkan *membership function* dari *input* yang diberikan, maka dapat diketahui nilai derajat keanggotaannya.

2.7.2. *Rule Evaluation*

Rule Evaluation merupakan suatu proses penentuan *fuzzy output* dari *fuzzy input*. Nilai *fuzzy input* dari *fuzzification* dimasukkan kedalam sebuah *rule* yang telah di tentukan agar dapat menghasilkan *fuzzy output*. *Rule evaluation* ini diolah dalam sebuah *fuzzy inference engine* terdapat *rule base* yang menggunakan metode *fuzzy* Mamdani. *Crisp input* pada *fuzzification* tersebut akan diolah oleh *inference engine* untuk menghasilkan keluaran *fuzzy* dengan komposisi *if-then* yang ada di dalam *rule base.*

2.7.3. *Defuzzification*

Defuzzification berfungsi untuk menentukan nilai *crisp output* yang berasal dari *rule evaluation*. *Output* dari *rule evaluation* tersebut dimasukkan kedalam *membership function output*. Proses terjadinya *defuzzification* yaitu keluaran variabel rule (linguistik) yang berada pada *inference engine* diubah menjadi *crip* output sesuai dengan *membership function* dari sistem *fuzzy* tersebut. Ada lima metode dalam *defuzzifikasi*, yaitu metode max, metode titik tengah, metode rata-rata, metode penjumlahan titik tengah, dan metode titik tengah terbesar(Yerubandi, Reddy & Reddy, 2015).

Perancangan *fuzzy logic controller* pada sistem pengendali kemudi (Woon, 2014) dan melakukan simulasi pada *software simulink MATLAB*. Perancangan *fuzzy logic controller* pada penelitian tersebut terdiri dari 3 *input* dari jarak yang terdeteksi oleh 3 ultrasonik dan 2 *output* untuk pengendalian kecepatan pada kedua motor penggerak.

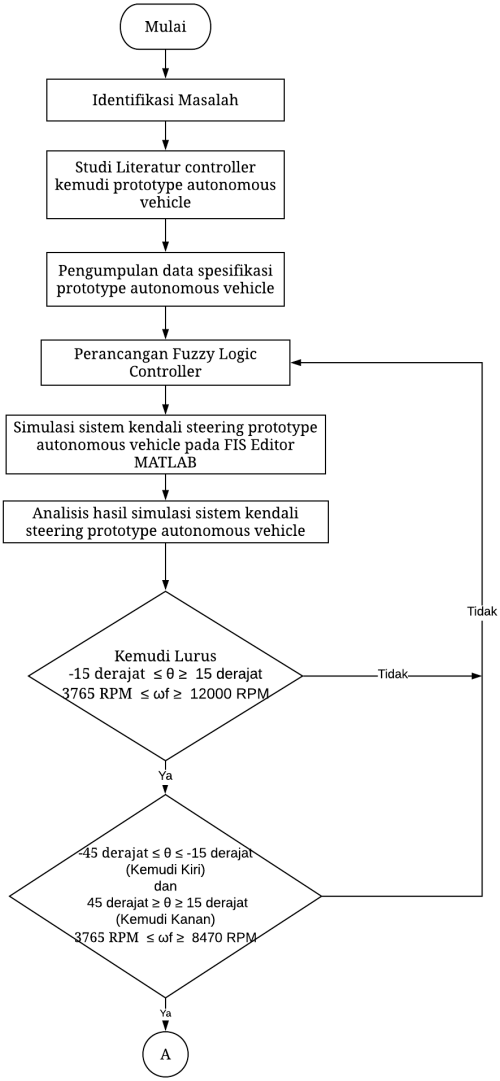
BAB III
METODOLOGI PENELITIAN

3.1 Alur Penelitian

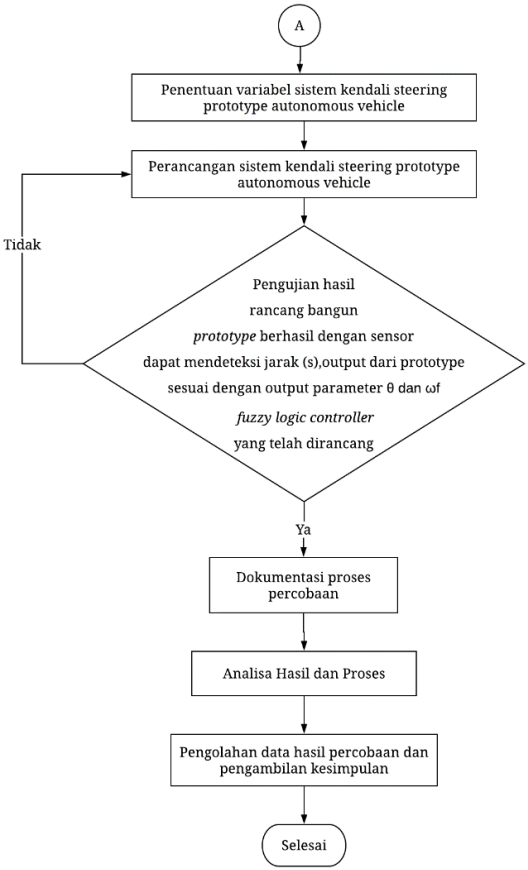
Pada penelitian ini dilakukan rancang bangun sistem pengendalian kemudi *prototype* kendaraan *autonomous* dengan metode *fuzzy logic* dengan simulasi pada *software Fis Editor* MATLAB kemudian melakukan perancangan *prototype autonomous vehicle*. Uji simulasi dilakukan menggunakan *software Fis Editor* MATLAB R2014a dengan terlebih dahulu pengendali *fuzzy logic* yang memiliki 4 input digunakan pada *Fuzzy Inference System*, yaitu 3 jarak (s) dari ketiga sensor ultrasonik yang dipasang pada posisi yang telah ditentukan dan kecepatan RPM (ω_s) awal (*start*) dari motor dc penggerak *prototyoe* tersebut, sementara *output* yang dihasilkan ada 2, yaitu sudut belok kemudi (θ) dan kecepatan RPM (ω_f) akhir (*final)* dari motor dc penggerak *prototype*. Pengujian siste kemudi hasil rancangan *fuzzy logic controller* dan perancangan *prototype autonomous vehicle* bertujuan untuk mendapatkan nilai parameter sistem pengendali kemudi agar dapat berjalan sesuai performansi serta spesifikasi dari sistem kemudi *prototype* yang telah berhasil terintegrasi dengan hasil rancangan sistem pengendali *fuzzy logic*.

Penentuan *membership function* i*nput* dan *output* dilakukan setelah mengumpulkan spesifikasi – spesifikasi dari alat (motor dc pada **Tabel 2.1** dan sensor ultrasonik pada **Tabel 2.2**) yang akan dirancang. Keberhasilan simulasi dapat dikatakan berhasil ketika *output* sudah sesuai dengan *rule base* yang ditelah ditentukan. Perancangan alat dilakukan dengan mengumpulkan spesifikasi dari alat yang dipergunakan. Proses perancangan alat dimulai dengan merakit motor dc dengan *arduino motor shield rev3* dan mikrokontroler *teensy* 3.6. *Prototype* menggunakan 2 motor dc, yaitu sebagai penggerak *prototype* dan membelokkan kemudi *prototype*. Pada motor dc penggerak (2 roda belakang *prototype*) diberikan kecepatan

RPM (ω) minimum sesuai dari beban yang dibawa oleh *prototype*, dimana kecepatan RPM (ω_s) tersebut dianggap sebagai kecepatan awal dari *prototype*. *Protoype* dapat berbelok karena arah putaran yang berbeda dari motor dc pembelok (2 roda depan *prototype*) dimana dalam program kodingan diinisialisasikan dengan keadaan *LOW* untuk belok kearah kiri dan *HIGH* untuk belok kearah kanan. Sensor ultrasonik digunakan sebagai alat *sensing obstacle* pada jalur yang ditentukan. *Prototype* menggunakan 3 ultrasonik.



Gambar 3.1 Diagram Alir Penelitian



Gambar 3.1 Diagram Alir Penelitian. (Lanjutan)

Agar mendapatkan hasil yang optimal, maka sebelum penelitian ini dimulai, dilakukan terlebih dahulu studi literatur mengenai sistem pengendalian kemudi *prototype* kendaraan *autonomous* dengan metode *fuzzy logic*. Setelah itu mencari spesifikasi, pemodelan posisi motor dc dan pemodelan

kecepatan motor dc. Lalu merancang sistem kontrol *Fuzzy Logic* berdasarkan pemodelan matematis yang telah di dapatkan sebelumnya. Uji perancangan *fuzy logic controller* ini diperlukan untuk mengetahui apakah sistem pengendalian yang telah dirancang telah berhasil terintegrasi dengan *prototype* agar dapat berjalan sesuai simulasi perancangan *fuzzy logic controller*. Untuk alur selengkapnya dapat di lihat pada diagram alir yang ditunjukkan **Gambar 3.1**.

3.2 Studi Literatur

Studi literatur pada tugas akhir ini dilakukan untuk memahami perancangan sistem kendali pada *prototype autonomous vehicle* dengan pengujiannya. Materi didapatkan dari beberapa jurnal penelitian yang telah dilakukan sebelumnya. Materi yang dipelajari merupakan materi penelitian yang bersangkutan pada sistem kendali untuk *steering* dari kendaraan dengan *fuzzy logic controller*.

3.3 Spesifikasi *Prototype Autonomous Vehicle*

Spesifikasi dari *prototype* yang akan digunakan dalam perancangan pengendali kemudi *autonomous* adalah sebagai berikut:

1. *Chassis* dari *prototype* dilengkapi dengan empat roda, dengan mekanisme *steering* menggunakan *gearbox*
2. *Prototype* terdiri dari dua motor dc untuk mekanisme *steering* dan untuk penggerak roda.
3. *Prototype* menggunakan 3 buah sensor *ultrasonic* yang dipasang pada bagian depan mobil, serong kiri depan mobil, dan serong kanan depan mobil
4. *Arduino Motor Shield* sebagai *driver* dari kedua motor dc yang meneruskan sinyal analog menuju masing – masing motor.

5. *Teensy* 3.6 prosesor dari rangkaian *prototype* sebagai *interface* yang menghubungkan komputer dengan *driver*.
6. Berat dari *prototype* 275 gram dengan panjang *prototype* 52 cm

3.3.1 Spesifikasi Motor DC pada *Prototype Autonomous Vehicle*

Pada penelitian ini, motor DC yang digunakan merupakan motor DC dari mobil *remote control* yang digunakkan sebagai *prototype* dengan spesifikasi sebagai berikut:

Tabel 3.1 Spesifikasi Motor DC *Prototype Autonomous Vehicle*

Tegangan	3 Volt – 9 Volt
Arus	0.2 Ampere
Kecepatan (Tanpa Beban), ω	12000 RPM
Torsi	65 g.cm atau 0.00065 kg.m
Dimensi	27 mm (p) x 20 mm (l) x 15 mm (t)
Diameter *Shaft*	2 mm
Total Panjang	28 mm
Berat	17 gram

3.3.2 Spesifikasi Ultrasonik HC-SR05

HC-SR05 merupakan sensor ultrasonik yang digunakan sebagai alat deteksi *obstacle* pada penelitian ini. HC-SR05 memiliki spesifikasi sebagai berikut:

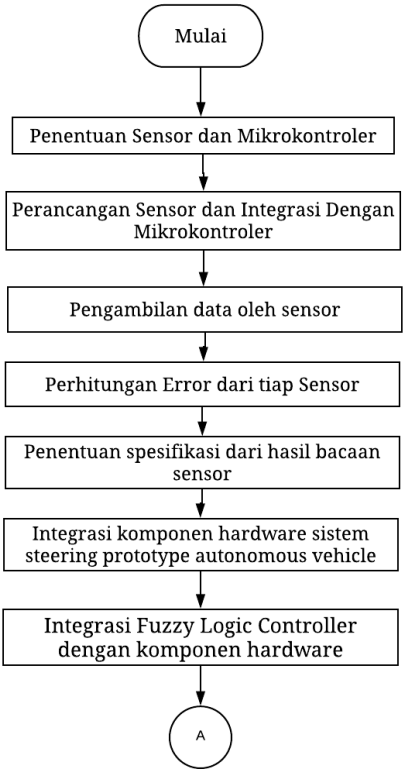
Tabel 3.2 Spesifikasi HC-SR05

Tegangan	**4.5 Volt – 5.5 Volt**
Frekuensi Suara	40 kHz
Resolusi Pengukuran	0.3 cm

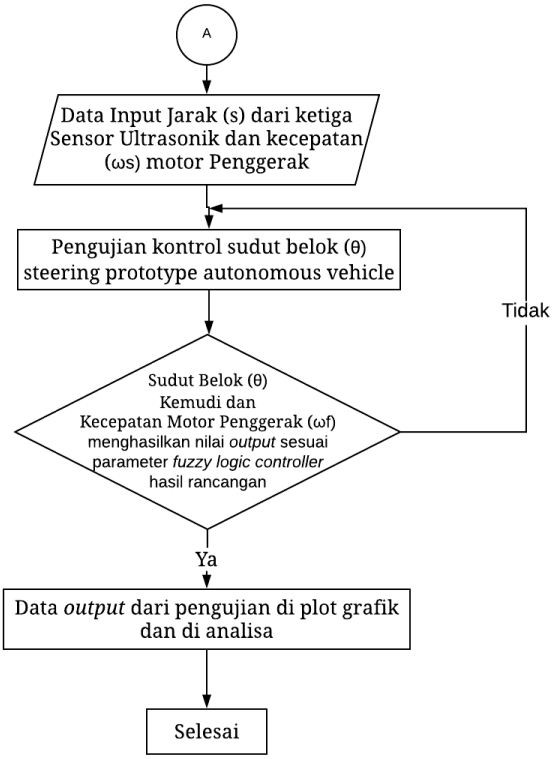
Tabel 3.2 (lanjutan)

Sudut Pengukuran	$< 15^0$
Konsumsi Arus	10 mA – 40 mA
Trigger Pin Format	10 uS *digital pulse*
Connector	5-pin *male connector*
Jangkauan	2 cm – 4500 cm
Dimensi	24 mm (p) x 20 mm (l) x 17 mm (t)

3.4 Perancangan Sistem



Gambar 3.2 Diagram Alir Perancangan *Hardware*



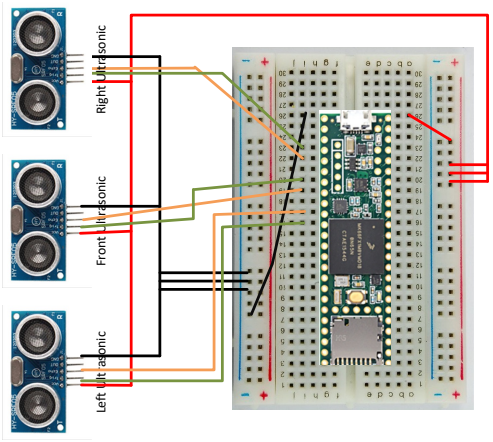
Gambar 3.2 Diagram Alir Perancangan *Hardware* (lanjutan)

Perancangan *hardware* dengan diagramm alir pada **Gambar 3.2** ini dilakukan penentuan komponen *hardware* terlebih dahulu, setelah penentuan tersebut dilakukan pemasangan sensor *ultrasonic* pada *microcontroller teensy 3.6*. Pemasangan sensor tersebut telah ditentukan posisinya setelah membaca referensi agar didapatkan hasil bacaan sensor yang baik. Sensor - sensor tersebut dihitung nilai *error* dan

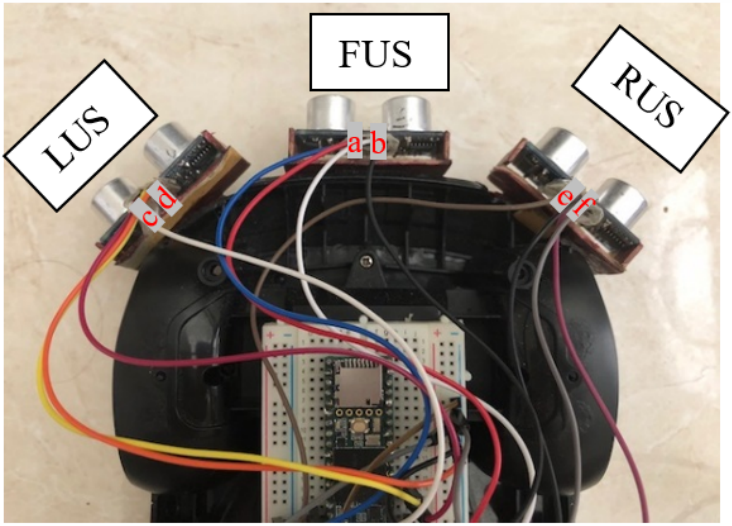
memastikan bahwa data yang diterima sensor memiliki nilai *error* dibawah 2%. Tujuan dari perhitungan *error* dari tiap sensor adalah untuk mendapatkan spesifikasi *obstacle* serta jarak yang dapat terbaca oleh sensor, sehingga dapat menentukan parameter apa sajakah yang harus terpenuhi agar *steering* pada *hardware* dapat berjalan dengan baik. Setelah itu dilakukan perancangan seluruh komponen *hardware* dan mengintegrasikannya dengan *fuzzy logic controller* yang telah dirancang.

3.4.1 Perancangan Sensor Ultrasonik pada Mikrokontroler *Teensy 3.6*

Tahap ini menghubungkan 3 buah sensor *ultrasonic* pada *microcontroller Teensy 3.6.* Sensor dipasang pada bagian depan mobil dengan posisi depan, serong kiri, dan serong kanan. Sensor – sensor tersebut dipisahkan dengan jarak 3 mm agar hasil bacaan dari *beam* sensor tidak saling *crash. Wiring* dari integrasi sensor ultrasonik dan mikrokontroler *teensy 3.6* terdapat pada **Gambar 3.3**.



Gambar 3.3 *Wiring* pada 3 Sensor Ultrasonic



Gambar 3.4 Posisi Penempatan Sensor *Ulrasonic* pada *Prototype*

Ultrasonik yang dipasang pada *prototype* sebanyak 3 buah dengan penempatan ultrasonik pada bagian kiri yaitu *Left Ultrasonic* (LUS), bagian kanan yaitu *Right Ultrasonic*, (RUS) dan bagian depan yaitu *Front Ultrasonic* (FUS) seperti pada **Gambar 3.4**. Masing – masing ultrasonik dihubungkan pada *teensy* 3.6.

Sensor ultrasonik yang digunakan pada *prototype* ini yaitu HC-SR05, dimana sensro tersebut memiliki 5 *male pin* yaitu Vcc, Trig, Echo, Output, GND. Sensor ultrasonik ini hanya digunakan sebagai *input* dari pengendali kemudi *prototype* ini. Pada *software programming* Arduino IDE diinisialisasikan pin sensor tersebut sebagai berikut:

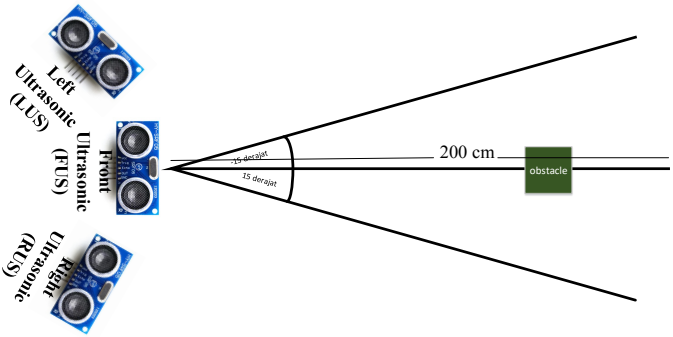
```
const int frontEchoPin = 6;……(a)
const int frontTriggerPin = 5;……(b)
const int leftEchoPin = 3;……..(c)
```

```
const int leftTriggerPin = 2;……..(d)
const int rightEchoPin = 9;……(e)
const int rightTriggerPin = 8;……(f)
```

3.4.2 Pengambilan Data dan Perhitungan *Error* pada Sensor Ultrasonik

Penggunaan sensor ultrasonik pada rancangan harus dihitung *error* dan melakukan pengukuran terbacanya jarak dari sensor seperti pada **Gambar 3.5**. Metode pengukuran tersebut, memastikan sensor dapat membaca dan menghasilkan *output* yang sesuai dengan perintah yang diberikan.

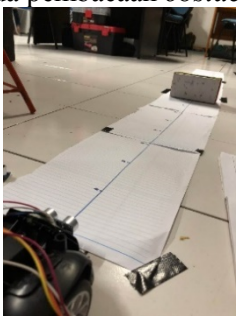
Perancangan *prototype* ini menggunakan 3 buah sensor ultrasonik HC-SR05 yang dipasang seri pada bagian serong kanan *prototype*, bagian depan *prototype*, dan bagian serong kiri dari *prototype*. Tiap sensor dicari *error*nya, dan melihat respon *error* tersebut dengan memberikan kondisi yang berbeda – beda. Pengukuran ini dilakukan dengan ketentuan *datasheet* yang ada.



Gambar 3.5 Metode Pengukuran *Error* Pada Sensor Ultrasonik

Pengukuran untuk mencari nilai *error* dari sensor tersebut dilakukan dengan mengukur jarak ultrasonik dari *obstacle* (yang tidak ditentukan) sejauh 200 cm dengan kondisi lurus didepan sensor, mengukur jarak terhadap *obstacle (*yang tidak

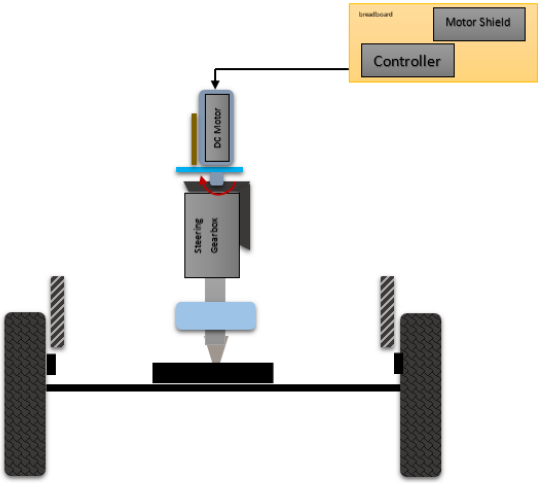
ditentukan) dengan kemiringan jarak dari sensor sebesar 15^0 ke kanan dan ke kiri, mengukur jarak maksimal pada *obstacle* dengan tinggi *obstacle* yang telah ditentukan (*small, medium, big*), setelah itu mengukur jarak deteksi *obstacle* tersebut hingga 200 cm. Tujuan dari pengambilan data serta perhitungan *error* dari ketiga (*Left Ultrasonic (LUS)*, *Front Ultrasonic (FUS)*, dan *Right Ultrasonic (RUS)*) sensor ultrasonik adalah agar mendapatkan spesifikasi dari *prototype autonomous vehicle* pada pembacaan *obstacle*.



Gambar 3.6 Pengukuran Jarak untuk Menentukan *Error* pada Sensor Ultrasonik

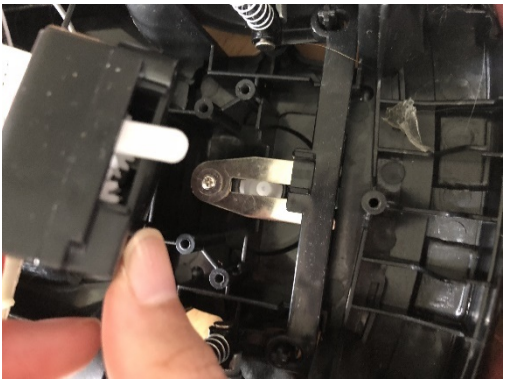
3.4.3Sistem Kemudi *Prototype Autonomous Vehicle*

Perancangan kemudi *Autonomous Vehicle* pada mobil *remote control* menggunakan 4 buah roda yang berpenggerak roda belakang dan sistem kemudi pada mobil *remote control* ini menggunakan motor dc yang dihubungkan dengan *gear box* sebagai penggerak kemudi dari mobil *remote control* tersebut.



Gambar 3.7 Rancangan Sistem Kemudi pada *Prototype Autonomous Vehicle*

Prinsip kerja sistem kemudi *autonomous vehicle* pada mobil *remote control* ini terdiri dari motor dc untuk 2 roda kemudi yang dihubungkan dengan sebuah *gearbox*. *Gearbox* dihubungkan dengan tuas yang disangkutkan pada *steering linkage* sehingga dapat membelokkan mobil sesuai dengan output yang diberikan. Motor dc digerakkan oleh *motor shield*, dimana *motor shield* tersebut terdapat fitur *motor driver*. *Motor driver* pada motor dc berfungsi untuk mengatur posisi motor dc serta kecepatan dari PWM motor dc tersebut agar mobil dapat berbelok sesuai dengan *output* yang diberikan. *Output* yang diterima berasal dari keluaran *fuzzy logic controller* pada *microcontroller* Teensy 3.6.



Gambar 3.8 Sistem Kemudi *Prototype Autonomous Vehicle*

3.4.4Integrasi Komponen *Hardware* dengan *Fuzzy Logic Controller*

Integrasi sistem pengendalian kemudi pada *prototype autonomous vehicle* ini dilakukan dengan menyiapkan motor dc sebanyak 2 buah (sebagai penggerak dan *steering*), sensor ultrasonik sebanyak 3 buah, *Arduino motor shiel* sebagai motor driver untuk 2 motor dc yang digunakan, dan *Teensy 3.6*.

Perancangan *prototype* dirancang dengan menggunakan 2 motor dc sebagai motor penggerak dan motor pembelok roda (*steering*). Motor dc penggerak dihubungkan ke *motor shield arduino* pada pin motor A, dan motor dc pembelok (*steering*) dihubungkan pada pin motor B. Pin – pin untuk mengatur PWM, *direction*, dan *brake*. 2 buah motor dc disambungkan pada *Arduino Motor Shield.* Kedua motor dc tersebut disambungan pada pin motor A dan B pada *Arduino Motor Shield*. Pin motor A disambungkan pada motor dc penggerak, dan pin motor B disambungkan pada motor dc untuk kemudi. Untuk melakukan pengendalian pada motor dc, *arduino motor*

shield disambungkan pada *ARM Teensy* 3.6. Pin yang digunakan pada *teensy 3.6* terdapat pada **Tabel 3.1**:

Tabel 3.3 *Pin Connection*

Arduino Motor Shield pin	***Teensy 3.6 pin***
5V	Vin (3.6 to 6 V)
GND	GND
DIRB	19 (A5)
DIRA	22 (PWM)
PWMB	20 (PWM)
BRAKEA	21 (PWM)
BRAKEB	18 (A4)
PWMA	23 (PWM)

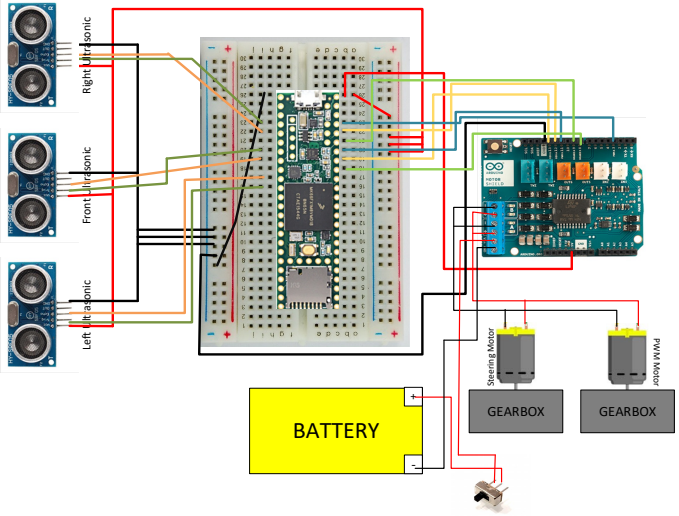
Pin yang terhubung pada *teensy* 3.6 di inisialisasikan pada *programming software Arduino IDE* dengan *script* sebagai berikut:

```
#define Mot_DIR 22
#define Mot_PWM 23
#define Steer_DIR 19
#define Steer_PWM 20
#define Mot_brake 21
#define Steer_brake 18
```

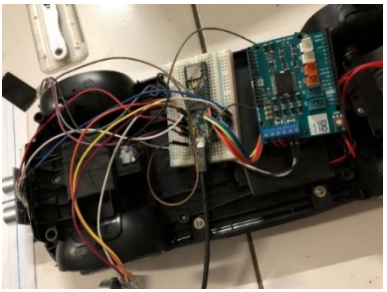
Dari, *script* diatas inisialisasi untuk motor penggerak adalah Mot_, dimana motor dc penggerak tersebut diatur *direction* (Mot_DIR), *brake* (Mot_BRAKE), serta PWM (Mot_PWM). Insialisasi pada motor dc kemudi adalah Steer_, dimana motor dc yang digunakan untuk membelokkan kemudi diatur *direction* (Steer_DIR), *brake* (Steer_BRAKE), serta PWM (Steer_PWM).

Integrasi semua komponen tersebut ke *teensy 3.6* sesuai pin yang digunakan kemudian melakukan pemrograman pada *software porgram* Arduino IDE seperti pada **Gambar 3.9**. Setelah dilakukan pemrograman tersebut, kemudian

memeriksa hasil pengambilan dan penyimpanan data pada *serial monitor* Arduino IDE. Apabila terjadi *error* pada pembacaan maka dilakukan pengecekan ulang pada *wiring* dan *program* sensor yang terhubung pada mikrokontroler.



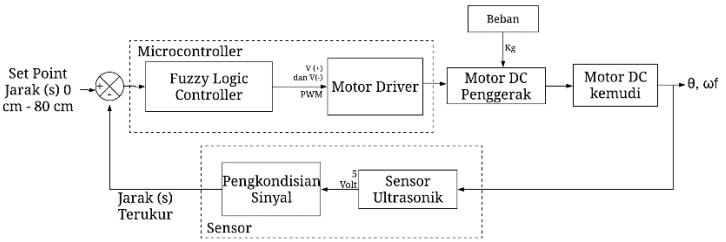
Gambar 3.9 *Wiring* Integrasi *Prototype*



Gambar 3.10 Rangkaian Sistem Kemudi pada *Prototype*

3.5 Perancangan *Sistem Kendali Steering Prototype Autonomous Vehicle*

Perancangan sistem kendali *steering prototype autonomous vehicle* merupakan sistem kendali *closed loop* atau sistem kendali lup tertutup. Sistem pengendalian *closed loop* terdiri dari *controller*, aktuator, *plant* dan *sensor.* Rancangan sistem pengendalian pada *steering prototype autonomous vehicle* ditunjukkan oleh **Gambar 3.11**:



Gambar 3.11 Diagram Blok Sistem Pengendalian *Closed Loop* pada *Steering Prototype Autonomous Vehicle*

Diagram blok sistem pengendalian pada **Gambar 3.11** menunjukkan nilai set point jarak (s) bernilai 0 cm hingga 80 cm, nilai set point tersebut merupakan batas jarak yang diterima oleh *prototype* terhadap *obstacle* agar *prototype* dapat mengatur sudut belok kemudi u ntuk menghindari *obstacle*. Masukan nilai jarak (s) yang diberikan dari 3 sensor ultrasonik kemudian menjadi nilai *crisp input* pada *fuzzy logic controller*, melalui *rule base* yang telah dirancang dan menghasilkan *crisp output* yaitu sudut belok kemudi *SteerAngle (θ)* dan kecepatan motor penggerak *RPMMot* (ω_f).

Nilai digital keluaran *fuzzy logic controller* pada mikrokontroler akan menjadi perintah untuk *motor driver* agar dapat menggerakkan Motor DC Kemudi dan Motor DC Penggerak sesuai perintah dari *fuzzy logic controller*. *Input*

yang diterima oleh *motor driver* merupakan variable manipulasi dimana untuk dapat membelokkan kemudi dengan memanipulasi *input* tegangan positif dan negative, dan untuk dapat menjalankan motor dc penggerak maka yang dimanipulasi atau diubah adalah nilai pwm dari motor dc penggerak. *Motor driver* memberikan perintah untuk mengatur kemudi dengan mengatur arah putaran dari motor DC dengan cara membalikkan tegangan input positif dan negatif dari motor tersebut (Sharma & Mohapatra, 2011), pengaturan tersebut sudah dideklarasi pada program dengan memberikan tegangan *input* negatif atau *low* untuk membelokkan ke kiri dan tegangan *input* positif atau *high* untuk membelokkan ke kanan. Baris ke 32 pada **lampiran** menyatakan *syntax* tersebut mendeklarasi variable untuk sudut belok motor dc kemudi *prototype*. Baris ke 81 pada **lampiran**, *syntax* dari line tersebut menyatakan nilai dari *array* g_fisOutput[0] merupakan nilai *output* dari *fuzzy* yang telah dirancang untuk sudut belok dari kemudi, diberi *delay* 10 ms agar sudut belok kemudi berbelok sesuai dengan kondisi lintasan yang diberikan dan pembacaan yang baik.

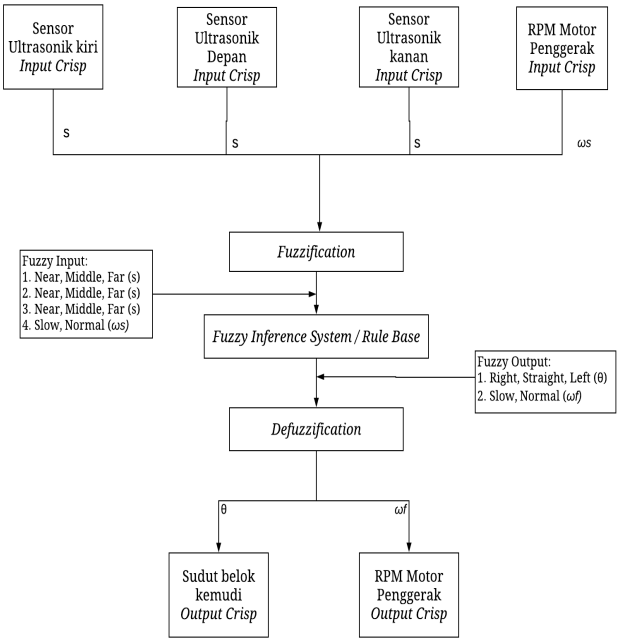
Pengaturan kecepatan pada motor dc dapat diatur dengan mengatur PWM dari motor tersebut dengan mengubah nilai pwm dari pulsa yang dibentuk. Perubahan nilai pwm dapat dilakukan dengan menggunakan mikrokontroler yang telah diberi program (Sharma & Mohapatra, 2011). Pengaturan kecepatan pada motor dc dilakukan dengan memvariasikan tegangan rata – rata yang diberi pada motor. PWM merupakan salah satu cara pengaturan kecepatan dengan mendapatkan variasi tegangan sesuai dengan pwm yang dikirim ke motor. Baris ke 31 pada **lampiran** merupakan deklarasi variable pwm untuk kecepatan awal motor dc penggerak *prototype*. *Syntax* bari ke 80 pada **lampiran** menyatakan nilai dari *array* g_fisOutput[1] merupakan nilai *output* untuk variable pwm dengan *delay* 1 ms.

Syntax dari baris ke 89 hingga baris ke 109 pada **lampiran** merupakan *syntax* yang berisi pengaturan pwm motor dc penggerak dan berisi pengaturan untuk *input* tegangan pada motor dc kemudi agar dapat menghasilkan belok ke kanan dan ke kiri sesuai dengan besar derajat belok keudi dari *fuzzy* yang telah dirancang.

Motor DC kemudi dan motor DC penggerak akan dijalankan oleh *motor driver* sesuai dengan *fuzzy output* dari *microcontroller*. *Fuzzy output* yang dikeluarkan dari *microcontroller* berupa kecepatan motor dc pada motor dc penggerak (RPM) dan besar sudut belok untuk motor dc kemudi (θ). Kecepatan yang dihasilkan pada motor dc penggerak juga dapat dipengaruhi oleh beban yang berada pada motor dc tersebut, untuk mendapatkan kecepatan motor tersebut dengan mencari nilai torsi dari motor dc yang di beri beban yang dibahas pada persamaan 2.8.

3.5.1 Perancangan *Fuzzy Logic Controller*

Fuzzy Logic Controller dirancang menggunakan FIS Editor pada *Software* MATLAB. *Fuzzy Logic Controller* pada sistem *steering prototype autonomous vehicle* menggunakan *fuzzy* tipe Mamdani dan memiliki 4 *input* dan 2 *output*. Perancangan *fuzzy logic controller* dilakukan melalui beberapa tahap, yaitu penentuan fungsi keanggotaan masing – masing *input* dan *output* dari *fuzzy logic controller* dan kemudian penentuan aturan atau *rule base* yang akan menjadi prinsip kerja dari sistem *fuzzy* yang akan dirancang.

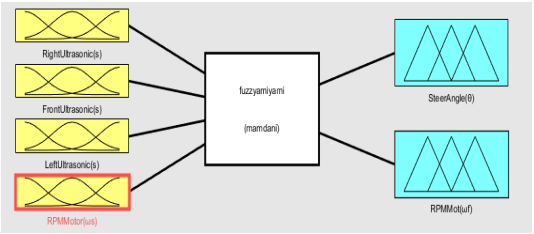


Gambar 3.12 Blok Diagram Sistem *Fuzzy Prototype Autonomous Vehicle*

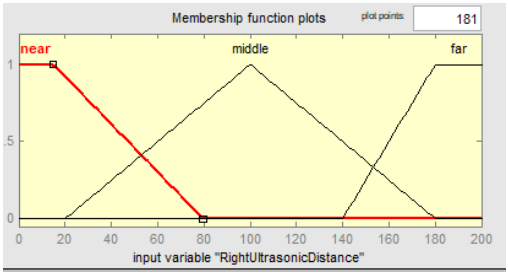
- **Fungsi Keanggotaan**

Fungsi keanggotaan yang digunakan pada *fuzzy logic controller* ini sebanyak 4 *input* dan 2 *output* seperti pada **Gambar 3.13**. *Input* tersebut terdiri dari jarak (s) pada *FrontUltrasonic(s)* pada **Gambar 3.16**, *LeftUltrasonic(s)* pada **Gambar 3.15**, *RightUltrasonic(s)* pada **Gambar 3.14** memiliki tipe fungsi gabungan trapesium dan segitiga dengan *range* sebesar 0 cm hingga 200 cm dengan fungsi keanggotaan antara lain *Near*, *Middle*, dan *Far*, dan *Input RPMMot (ωs)* pada **Gambar 3.17** memiliki tipe fungsi trapesium dengan *range* 3765 RPM hingga 12000 RPM, memiliki fungsi keanggotaan *slow* dan *normal*. *Output* terdiri dari *SteerAngle (θ)* pada **Gambar 3.18** dengan tipe fungsi gabungan trapesium

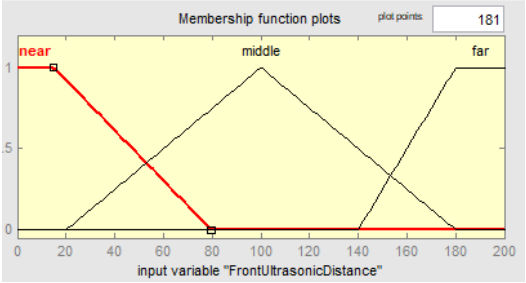
dan segitiga dengan *range* -45^0 hingga 45^0 yang memiliki fungsi keanggotaan *right* sebesar $15^0 < \theta < 45^0$, *straight* sebesar $-15^0 < \theta < 15^0$, dan *left* sebesar $-15^0 > \theta > -45^0$. *Output* RPMMot(*ωf)* pada **Gambar 3.19** memiliki tipe fungsi trapesium dengan *range* 3765 RPM hingga 12000 RPM dan fungsi keanggotaan *slow* dan *normal*. *Range* dari fungsi keanggotaan pada setiap *input* dan *output* didapatkan berdasarkan referensi penelitian serupa dan uji coba *trial and error* agar didapatkan *range* yang sesuai dengan sistem *prototype hardware*. *Input* yang berupa numerik kemudian akan melalui proses fuzzifikasi sehingga berubah menjadi *crisp input* untuk diolah dalam sebuah *inference system*.



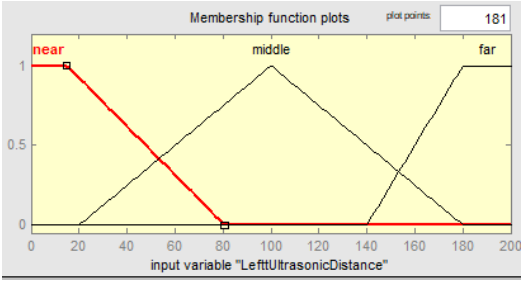
Gambar 3.13 FIS pada Sistem *Steering Prototype Autonomous Vehicle*



Gambar 3.14 Fungsi Keanggotaan *Input RightUltrasonic(s)*

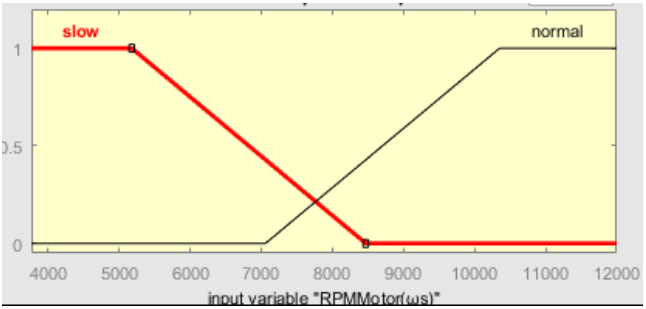


Gambar 3.15 Fungsi Keanggotaan *Input* F*rontUltrasonic(s)*



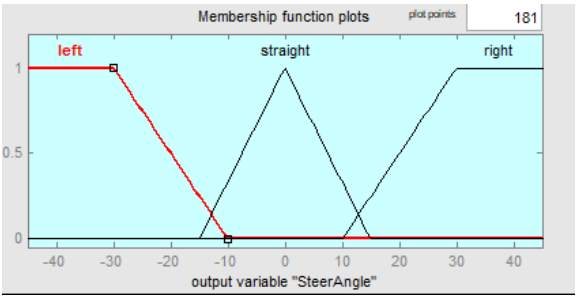
Gambar 3. 16 Fungsi Keanggotaan *Input LeftUltrasonic(s)*

Input membership function pada **Gambar 3.14**, **Gambar 3.15**, dan **Gambar 3.16** menggunakan gabungan dari trapesium dan segitiga. *Range* dari *input* sebesar 0 cm hingga 200 cm, *membership function* "near" dan far" menggunakan bentuk trapesium dengan nilai *input* masing – masing sebesar [-inf, -15, 15, 80] dan [140, 180, 208, inf]. *Membership function* "middle" menggunakan bentuk segitiga dengan nilai [20, 100, 180].



Gambar 3.17 Fungsi Keanggotaan *Input RPMMot(ωs)*

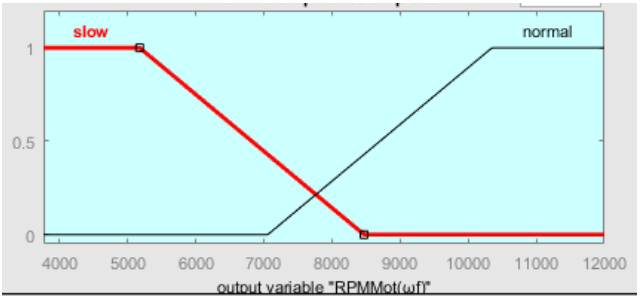
Input membership function pada **Gambar 3.17** memiliki 2 *membership function* dengan bentuk trapesium pada keduanya. *Range* dari *input* sebesar 3765 RPM hingga 12000 RPM, *membership function* "slow" dan "near" menggunakan bentuk trapesium dengan nilai *input* masing – masing sebesar [-inf, 3294, 5176, 8471] dan [7059, 10353, 12706, inf].



Gambar 3.18 Fungsi Keanggotaan *Output SteerAngle(θ)*

Input membership function pada **Gambar 3.18** menggunakan gabungan dari trapesium dan segitiga. *Range* dari *input* sebesar -45 derajat hingga 45 derajat, pada *membership function* "right" berbelok ke kanan dan "left"

berbelok ke kiri menggunakan bentuk trapesium dengan nilai *output* masing – masing sebesar [-inf, --50, -30, -10] dan [10, 30, 50, inf]. *Membership function* "straight" atau tidak berbelok menggunakan bentuk segitiga dengan nilai [-15, 0, 15].

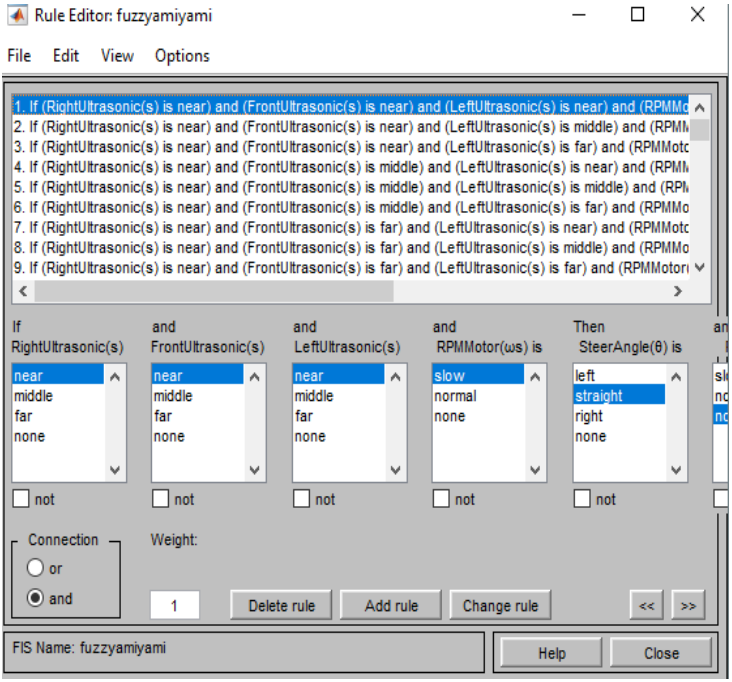


Gambar 3.19 Fungsi Keanggotaan *Output RPMMot(ωf)*

Input membership function pada **Gambar 3.19** memiliki 2 *membership function* dengan bentuk trapesium pada keduanya. *Range* dari *output* sebesar 3765 RPM hingga 12000 RPM, *membership function* "slow" dan "near" menggunakan bentuk trapesium dengan nilai *input* masing – masing sebesar [-inf, 3294, 5176, 8471] dan [7059, 10353, 12706, inf].

- ***Fuzzy Inference System / Rule Base***

Bilangan *crisp* yang dinyatakan dalam fungsi keanggotaan akan diolah dalam *inference system* yang memiliki prinsip sebab akibat (*If....Then*). Dalam *inference system* terdapat *rule base* tang mengatur *output fuzzy* sesuai *input* yang diterima sistem. *Rule base* merupakan aturan sebab akibat yang dirancang untuk mengambil keputusan akhir. Dasar perancangan *rule base* pada *fuzzy logic* berdasarkan referensi penelitian dan *trial and error*. Pada penelitian ini *rule base* yang dirancang sebanyak 54 buah *rule* seperti pada **Gambar 3.20**.

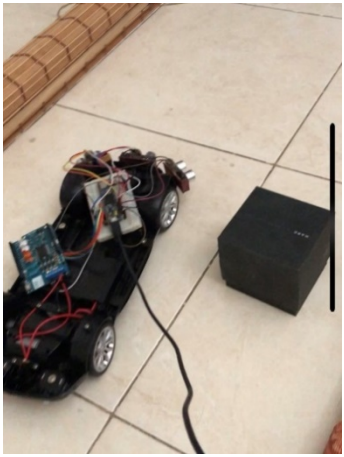


Gambar 3.20 *Rule Base* FIS terhadap *output SteerAngle* dan *PWMMot*.

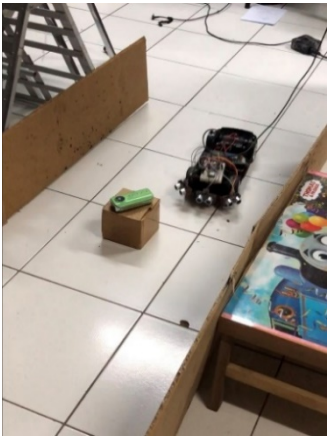
3.5.2 Pengujian Sistem Kendali *Fuzzy Logic* pada *Steering Prototype Autonomous Vehicle*

Pada tahap ini melakukan pengujian pada *hardware* yang telah di *upload* program *fuzzy logic controller* yang telah dirancang. Dilakukan pengujian dengan lintasan yang telah ditentukan, lintasan tersebut berbentuk lurus dan memiliki jarak sekitar 150 cm dengan pembatas. Pengujian tersebut dilakukan dengan 4 kondisi yang berbeda, yaitu tanpa *obstacle* seperti pada **Gambar 3.21**, *obstacle* dengan ukuran 10cm X 10cm seperti pada **Gambar 3.22**, *obstacle* dengan ukuran 20cm X 20cm seperti pada **Gambar 3.23**, dan *obstacle* dengan ukuran 35cm X 20 cm seperti pada **Gambar 3.24**.

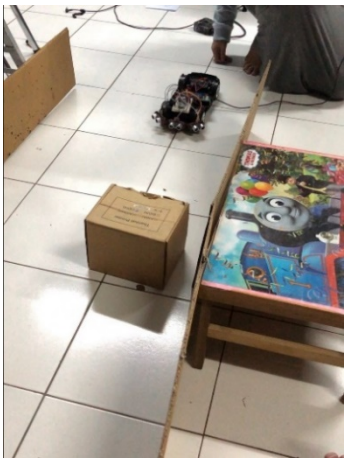
Gambar 3.21 Pengujian *Hardware* pada Lintasan tanpa *Obstacle*



Gambar 3.22 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 10cm X 10cm



Gambar 3.23 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 20cm X 20cm



Gambar 3.24 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 35cm X 20cm

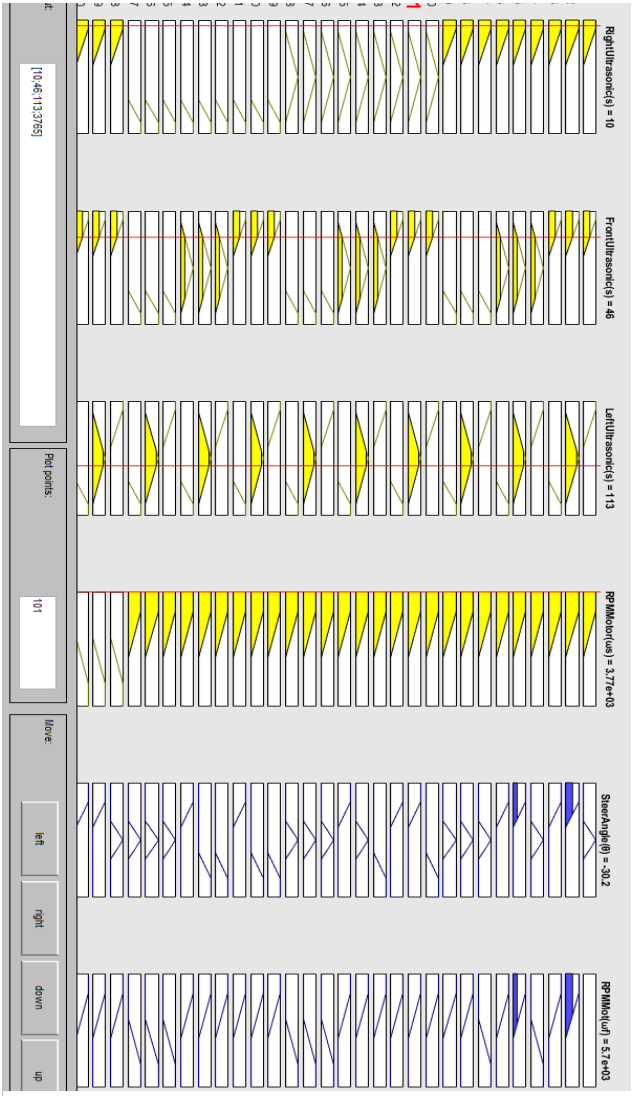
Halaman ini sengaja dikosongkan

BAB IV
ANALISA DATA DAN PEMBAHASAN

4.1 Hasil Simulasi Rancangan Sistem Steering Prototype Autonomous Vehicle dengan Fuzzy Logic Controller

Simulasi motor dc untuk sistem kendali *steering prototype* dengan *fuzzy logic controller* pada matlab. Simulasi dilakukan untuk mengetahui respon dari *rule base fuzzy* yang telah dirancang dengan parameter yang sesuai pada rangkaian *prototype* yang digunakan. Hasil dari pengujian model kemudian dijadikan acuan untuk merancang sistem kendali untuk sistem *steering prototype autonomous vehicle*. Pada penelitian dilakukan untuk mengetahui respon sistem kendali pada sudut belok kemudi dan kecepatan RPM *prototype*.

Pada *FIS Editor*, diberikan 4 *input* (jarak (s) yang didapat dari ultrasonik bagian serong kanan *prototype*¸ultrasonik bagian depan *prototype*, ultrasonik bagian serong kiri *prototype* dan RPM *(ωs)* Motor dc penggerak) dengan nilai secara berurutan yaitu 10 cm, 46 cm, 113 cm, dan 3765 RPM. *Output* dari FIS pada simulink diatas, didapatkan sudut belok dari kemudi dan RPM *(ωf)* motor dc penggerak. Sudut belok (θ) kemudi yang dihasilkan yaitu sebesar -30.2 derajat dan RPM *(ωf)* motor dc penggerak meningkat menjadi 57000 RPM. Nilai *output* dari sistem pengendali ini sesuai dengan *rule base* yang telah ditetapkan dapat dilihat pada **Gambar 4.1**.



Gambar 4.1 *FIS rule viewer*

4.2 Perhitungan *Error* Sensor Ultrasonik

Sensor ultrasonik yang digunakan pada *prototype* sebanyak 3 buah sensor. Sensor tersebut diletakan pada serong kanan *chassis prototype* sebesar x derajat, pada bagian depan *chassis prototype*, dan pada bagian serong kiri *chassis prototype* sebesar x derajat. Perhitungan *error* pada sensor ini perlu dilakukan agar dapat mengetahui seberapa besar *error* dari tiap ultrasonik agar pembacaan data dari sensor tidak berbeda jauh. Pada Tabel 2.1 diketahui *datasheet* atau spesifikasi dari sensor ultrasonik HC-SR05. Sensor tersebut dapat membaca jarak dengan jangkauan 2 cm hingga 4500 cm. Pada *datasheet* tersebut juga diketahui sudut *beam* dari ultrasonik tersebut sebesar $<15^0$. Langkah yang dilakukan dalam perhitungan *error* sensor ultrasonik ini dengan cara mengukur jarak dari tiap ultrasonik ke *obstacle* dengan jarak 0 cm hingga 200 cm, tiap kelipatan 10. Perhitungan *error* pada sudut *beam* ultrasonik bagian kanan dan bagian kiri juga dilakukan dengan jarak pengukuran yang sama. Perhitungan *error* juga dilakukan dengan menggunakan 3 ukuran *obstacle* yang berbeda pada tiap ultrasonik.

4.2.1. Perhitungan *Error* Ultrasonik Terhadap *Obstacle*

Perhitungan *error* ultrasonik yang pertama dilakukan dengan menggunakan *obstacle* yang acak pada 0^0 (garis lurus) dengan jarak 0 cm – 200 cm. Pengukuran deteksi jarak tersebut dilakukan tiap kelipatan 10 cm, tiap jarak dilakukan 5 kali pengambilan data.

Dari pengukuran yang dilakukan didapatkan tabel perhitungan sebagai berikut:

Tabel 4.1 Perhitungan E*rror* Ultrasonik *Right*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	2	-2
2	10	9.2	0.8
3	20	18.6	1.4
4	30	28.6	1.4

Tabel 4.1 (Lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
5	40	38.8	1.2
6	50	48	2
7	60	58	2
8	70	67.4	2.6
9	80	77.8	2.2
10	90	87.6	2.4
11	100	97.4	2.6
12	110	107.2	2.8
13	120	117.4	2.6
14	130	127.8	2.2
15	140	138	2
16	150	148	2
17	160	158.2	1.8
18	170	167.6	2.4
19	180	178.2	1.8
20	190	189.6	0.4
21	200	198	2
Total Error			36.6
Error Rata-Rata			1.742857

Tabel 4.2 Perhitungan E*rror* Ultrasonik *Front*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	2	-2
2	10	9	1
3	20	19.4	0.6
4	30	29	1
5	40	39	1
6	50	48.8	1.2
7	60	58.6	1.4
8	70	68.2	1.8
9	80	79.2	0.8
10	90	89	1
11	100	97.6	2.4
12	110	107.6	2.4
13	120	117.6	2.4

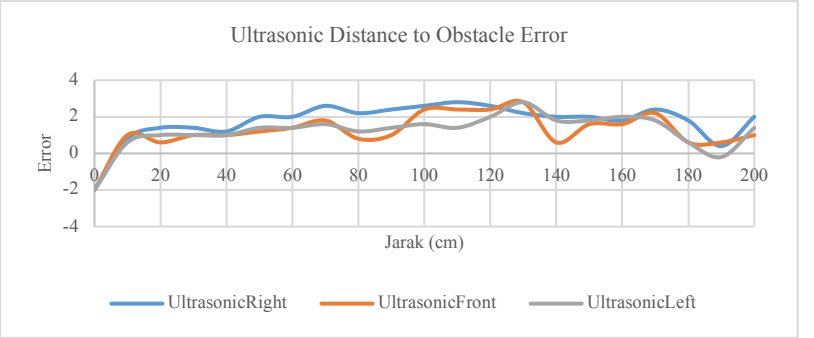
Tabel 4.2 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
14	130	127.2	2.8
15	140	139.4	0.6
16	150	148.4	1.6
17	160	158.4	1.6
18	170	167.8	2.2
19	180	179.4	0.6
20	190	189.4	0.6
21	200	199	1
Total Error			26
Error Rata-Rata			1.23809523

Tabel 4.3 Perhitungan *Error* Ultrasonik *Left*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	2	-2
2	10	9.4	0.6
3	20	19	1
4	30	29	1
5	40	39	1
6	50	48.6	1.4
7	60	58.6	1.4
8	70	68.4	1.6
9	80	78.8	1.2
10	90	88.6	1.4
11	100	98.4	1.6
12	110	108.6	1.4
13	120	118	2
14	130	127.2	2.8
15	140	138.2	1.8
16	150	148.2	1.8
17	160	158	2
18	170	168.2	1.8
19	180	179.4	0.6
20	190	190.2	-0.2
21	200	198.6	1.4
Total Error			25.6
Error Rata-Rata			1.21904761

Dari ketiga tabel *error* diatas didapatkan hasil *error* pada ketiga ultrasonik bernilai 1.74, 1.24, dan 1.22, dimana nilai *error* tersebut masih berada dibawah 2% toleransi *error*. Plot dari ketiga perhitungan *error* dihasilkan grafik pada **Gambar 4.3**:



Gambar 4.2 Grafik *Error* Ketiga Ultrasonik

Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 2.8 pada jarak 110 cm, Ultrasonik *Front* didapatkan *peak error* 2.8 pada jarak 130 cm, dan Ultrasonik *Left* didapatkan *peak error* 2.8 pada jarak 130 cm.

4.2.2. Perhitungan *Error* Ultrasonik pada Sudut 15^0

Sensor ultrasonik memiliki *angle beam* $<15^0$ dari tegak lurus sensor. Pengukuran jarak pantul dari *obstacle* pada persebaran *beam* ultrasonik tersebut penting dilakukan karena pada saat melakukan seri terhadap 3 ultrasonik sebagai sensor jarak harus diperhitungkan persebarannya agar gelombang ultrasonik tiap sensor yang dipancarkan tidak saling mempengaruhi satu sama lain.

Perhitungan *error* ultrasonik pada sudut $<15^0$ diukur dari jarak 0 cm – 200 cm tiap kelipatan 10 cm. Pengukuran dilakukan sebanyak 5 kali pengulangan tiap kelipatan 10 cm.

a. Perhitungan *error* ultrasonik pada sudut 15^0 kearah kanan

Tabel 4.4 Perhitungan *Error* Ultrasonik *Right* pada Sudut 15^0 Kearah Kanan

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
1	0	3	-3
2	10	10	0
3	20	22.6	-2.6
4	30	31.2	-1.2
5	40	39.4	0.6
6	50	49.6	0.4
7	60	60	0
8	70	68.8	1.2
9	80	78.8	1.2
10	90	88.2	1.8
11	100	98.2	1.8
12	110	108.4	1.6
13	120	117.8	2.2
14	130	128.8	1.2
15	140	137.6	2.4
16	150	149.6	0.4
17	160	158.8	-656.4
18	170	169.2	-22.4
19	180	178.4	-68.2
20	190	187.8	-1236.6
21	200	198.2	-564.4
Total Error (0 cm – 150 cm)			8
Error Rata-Rata			0.5

Tabel 4.5 Perhitungan *Error* Ultrasonik *Front* pada Sudut 15^0 Kearah Kanan

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
1	0	3	-3
2	10	10.4	-0.4
3	20	20.8	-0.8
4	30	31.6	-1.6
5	40	40.4	-0.4
6	50	47.8	2.2
7	60	59.8	0.2
8	70	69.4	0.6

Tabel 4.5 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
9	80	80	0
10	90	90.8	-0.8
11	100	99.6	0.4
12	110	109.2	0.8
13	120	119.8	0.2
14	130	130.4	-0.4
15	140	140.4	-0.4
16	150	149.8	0.2
17	160	867.6	-707.6
18	170	283.4	-113.4
19	180	307.8	-127.8
20	190	339.8	-149.8
21	200	668.2	-468.2
Total Error (0 cm – 150 cm)			3.2
Error Rata-Rata			0.2

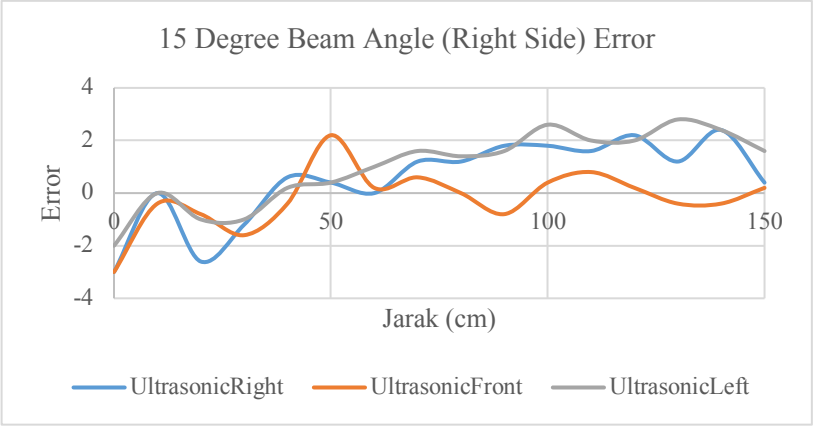
Tabel 4.6 Perhitungan *Error* Ultrasonik *Left* pada Sudut 15^0 Kearah Kanan

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
1	0	2	-2
2	10	10	0
3	20	21	-1
4	30	31	-1
5	40	39.8	0.2
6	50	49.6	0.4
7	60	59	1
8	70	68.4	1.6
9	80	78.6	1.4
10	90	88.4	1.6
11	100	97.4	2.6
12	110	108	2
13	120	118	2
14	130	127.2	2.8
15	140	137.6	2.4
16	150	148.4	1.6
17	160	318.6	-158.6

Tabel 4.6 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	Error
18	170	614.8	-444.8
19	180	1506.6	-1326.6
20	190	311.6	-121.6
21	200	351	-151
Total Error (0 cm – 150 cm)			15.6
Error Rata-Rata			0.975

Dari ketiga tabel *error* diatas didapatkan hasil *error* pada ketiga ultrasonik bernilai 0.5, 0.2, dan 0.975, dimana nilai *error* tersebut masih berada dibawah 2% toleransi *error*. Tulisan berwarna merah tersebut merepresentasikan nilai *error* tinggi pada jarak 160 cm hingga 200 cm. Plot dari ketiga perhitungan *error* dihasilkan grafik pada **Gambar 4.4**:



Gambar 4.3 Grafik *Error* pada 15^0 *Beam Angle* Kearah Kanan.

Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 2.4 pada jarak 140 cm, Ultrasonik *Front* didapatkan *peak error* 2.8 pada jarak 50 cm, dan Ultrasonik *Left*

didapatkan *peak error* 2.8 pada jarak 130 cm. Ketiga ultrasonik didapatkan *error* yang tinggi pada jarak 160 cm hingga 200 cm.

b. Perhitungan *error* ultrasonik pada sudut 15^0 kearah kiri

Tabel 4.7 Perhitungan *Error* Ultrasonik *Right* pada Sudut 15^0 Kearah Kiri

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	3	-3
2	10	9	1
3	20	20.2	-0.2
4	30	30.8	-0.8
5	40	39	1
6	50	49.6	0.4
7	60	58.2	1.8
8	70	68.2	1.8
9	80	77.2	2.8
10	90	86	4
11	100	96.8	3.2
12	110	106.6	3.4
13	120	118.2	1.8
14	130	128	2
15	140	138	2
16	150	147	3
17	160	207.6	-47.6
18	170	547.6	-377.6
19	180	291.8	-111.8
20	190	1045.2	-855.2
21	200	240.2	-40.2
	Total (0 cm – 150 cm)		24.2
	Error Rata-Rata		1.5125

Tabel 4.8 Perhitungan *Error* Ultrasonik *Front* pada Sudut 15^0 Kearah Kiri

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	3	-3
2	10	10	0

Tabel 4.8 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
3	20	19	1
4	30	29.4	0.6
5	40	39.6	0.4
6	50	50	0
7	60	59	1
8	70	67.8	2.2
9	80	80.2	-0.2
10	90	89	1
11	100	99	1
12	110	109.2	0.8
13	120	119.6	0.4
14	130	128.2	1.8
15	140	138	2
16	150	148.8	1.2
17	160	463	-303
18	170	803.8	-633.8
19	180	106.8	73.2
20	190	333	-143
21	200	244.2	-44.2
	Total (0 cm – 150 cm)		10.2
	Error Rata-Rata		0.6375

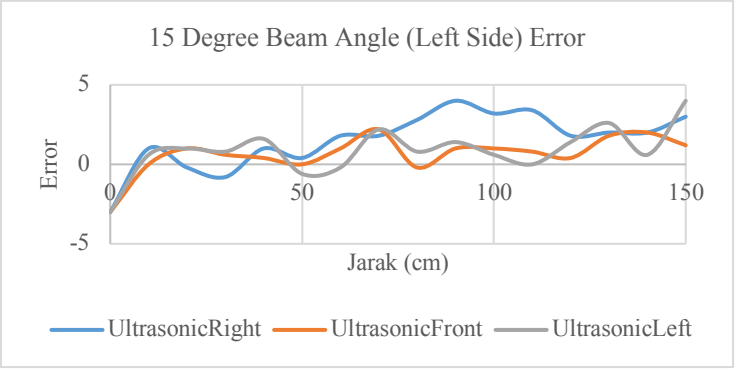
Tabel 4.9 Perhitungan *Error* Ultrasonik *Left* pada Sudut 15^0 Kearah Kiri

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	3	-3
2	10	9.4	0.6
3	20	19	1
4	30	29.2	0.8
5	40	38.4	1.6
6	50	50.6	-0.6
7	60	60.2	-0.2
8	70	67.8	2.2
9	80	79.2	0.8
10	90	88.6	1.4
11	100	99.4	0.6
12	110	110	0
13	120	118.6	1.4

Tabel 4.9 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur	error
14	130	127.4	2.6
15	140	139.4	0.6
16	150	146	4
17	160	1352.2	-1192.2
18	170	118	52
19	180	112	68
20	190	126.6	63.4
21	200	357.6	-157.6
Total (0 cm – 150 cm)			13.8
Error Rata-Rata			0.8625

Dari ketiga tabel *error* diatas didapatkan hasil *error* pada ketiga ultrasonik bernilai 1.51, 0.63, dan 0.86, dimana nilai *error* tersebut masih berada dibawah 2% toleransi *error*. Tulisan berwarna merah tersebut merepresentasikan nilai *error* tinggi pada jarak 150 cm hingga 200 cm. Plot dari ketiga perhitungan *error* dihasilkan grafik pada **Gambar 4.5**:



Gambar 4.4 Grafik *Error* pada 15^0 *Beam Angle* Kearah Kiri.

Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 4 pada jarak 90 cm, Ultrasonik *Front* didapatkan *peak error* 2.2 pada jarak 70 cm, dan Ultrasonik *Left* didapatkan *peak error* 2.6 pada jarak 130 cm. Ketiga ultrasonik didapatkan *error* yang tinggi pada jarak 150 cm hingga 200 cm.

4.2.3. Perhitungan Error Ultrasonik Mendeteksi Tinggi *Obstacle* yang Terukur Terhadap Jarak

Sensor Ultrasonik dipasang pada *chassis prototype* setinggi 2.7 cm dari dasar. Tujuan dari pengukuran ini, agar dapat mengetahui hasil pengukuran sensor ultrasonik terhadap *obstacle* dengan tinggi dan jarak yang berbeda.

Pengukuran ini dilakukan dengan tinggi *obstacle* yang berbeda-beda, *Small Obstacle* berukuran 3cm x 2.8 cm x 2.1 cm, *Medium Obstacle* berukuran 4 cm x 10 cm x 3.8 cm, dan *Big Obstacle* berukuran 13.8 cm x 3.2 cm x 13.8 cm. Pengukuran dimulai dari jarak 0 cm hingga 9 cm agar dapat diketahui jarak minimum dari deteksi *obstacle* berdasarkan tinggi yang berbeda. Pengambilan data dilakukan sebanya 5 kali di tiap jarak yang telah ditentukan.

Tabel 4.10 Perhitungan Error Ultrasonik untuk Mendeteksi Tinggi *Small Obstacle* yang Terukur Terhadap Jarak

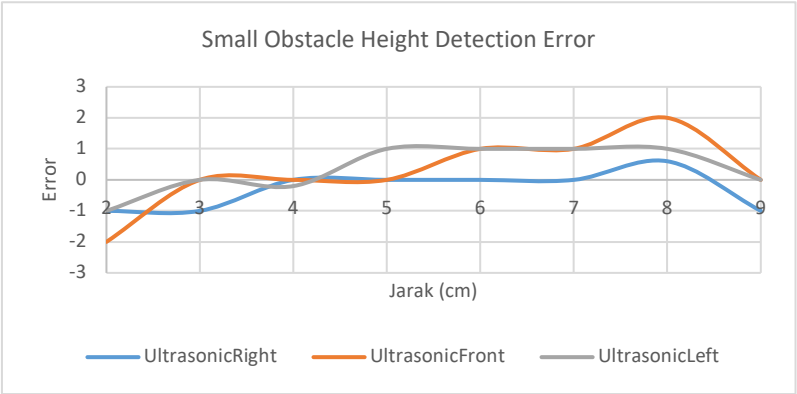
NO	Jarak (cm)	Ultrasonik Right		Ultrasonik Front		Ultrasonik Left	
		Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR
1	0	866	-866	-1064.4	-064.4	924.8	-924.8
2	1	299.4	-298.4	273.6	-272.6	288.8	-287.8
3	2	3	-1	4	-2	3	-1
4	3	4	-1	3	0	3	0
5	4	4	0	4	0	4.2	-0.2
6	5	5	0	5	0	4	1
7	6	6	0	5	1	5	1
8	7	7	0	6	1	6	1
9	8	7.4	0.6	6	2	7	1

Tabel 4.10 (lanjutan)

NO	Jarak (cm)	Ultrasonik Right		Ultrasonik Front		Ultrasonik Left	
		Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR
10	9	10	-1	9	0	9	0
Total (2 cm – 9 cm)			2.4		2		2.8
Error Rata – rata			0.3		0.25		0.35

Nilai *error* pada jarak 0 cm hingga 2 cm belum bisa terukur dengan baik oleh sensor, tinggi dari *obstacle* adalah 2.5 dan tinggi dari sensor sebesar 2.7, maka sensor mengalami kesulitan untuk mengukur *obstacle* tersebut. Pada jarak 2 cm *obstacle* dapat terbaca oleh sensor.

Dari data yang diadapatkan, kemudian di *plot* data tersebut menjadi sebuah grafik sebagai berikut:



Gambar 4.5 Grafik E*rror* Deteksi *Small Obstacle*

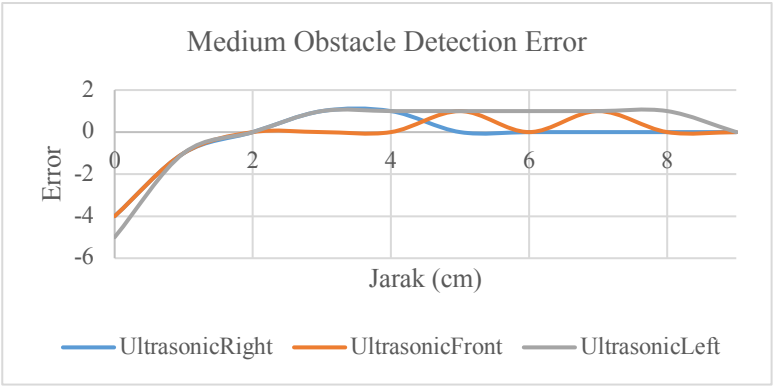
Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 0.6 pada jarak 9 cm, Ultrasonik *Front* didapatkan *peak error* 2 pada jarak 9 cm, dan Ultrasonik *Left* didapatkan *peak error* 1 pada jarak 6 cm, 7 cm, 8 cm, dan 9 cm. Ketiga

ultrasonik didapatkan *error* yang tinggi pada jarak 1 cm hingga 2 cm.

Tabel 4.11 Perhitungan Error Ultrasonik untuk Mendeteksi Tinggi *Medium Obstacle* yang Terukur Terhadap Jarak

NO	Jarak (cm)	Ultrasonik Right		Ultrasonik Front		Ultrasonik Left	
		Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR
1	0	4	-4	4	-4	5	-5
2	1	2	-1	2	-1	2	-1
3	2	2	0	2	0	2	0
4	3	2	1	3	0	2	1
5	4	3	1	4	0	3	1
6	5	5	0	4	1	4	1
7	6	6	0	6	0	5	1
8	7	7	0	6	1	6	1
9	8	8	0	8	0	7	1
10	9	9	0	9	0	9	0
(2 cm – 9 cm)			3		3		0
Error Rata-Rata			0.3		0.3		0

Nilai *error* pada pengukuran *medium obstacle* sangat kecil. Tinggi *obstacle* melebihi tinggi dari peletakan sensor.



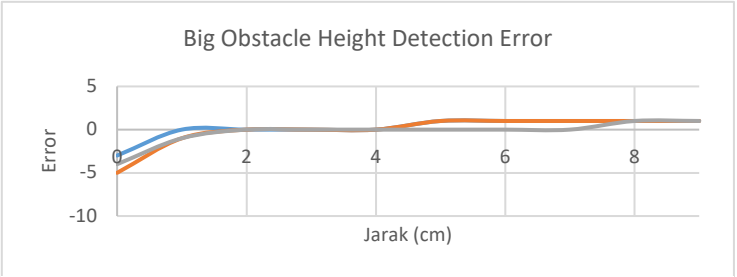
Gambar 4.6 Grafik *error* deteksi *Medium obstacle*

Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 1 pada jarak 4 cm dan 5 cm, Ultrasonik *Front* didapatkan *peak error* 1 pada jarak 6 cm dan 8 cm, dan Ultrasonik *Left* didapatkan *peak error* 1 pada jarak 4 cm, 5 cm, 6 cm, 7 cm, 8 cm, dan 9 cm.

Tabel 4.12 Perhitungan Error Ultrasonik untuk Mendeteksi Tinggi *Big Obstacle* yang Terukur Terhadap Jarak

NO	Jarak (cm)	Ultrasonik Right		Ultrasonik Front		Ultrasonik Left	
		Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR	Jarak terukur (cm)	ERROR
1	0	3	-3	5	-5	4	-4
2	1	1	0	2	-1	2	-1
3	2	2	0	2	0	2	0
4	3	3	0	3	0	3	0
5	4	4	0	4	0	4	0
6	5	4	1	4	1	5	0
7	6	5	1	5	1	6	0
8	7	6	1	6	1	7	0
9	8	7	1	7	1	7	1
10	9	8	1	8	1	8	1
Total (2 cm – 9 cm)			2		1		3
Error Rata-Rata			0.2		0.1		0.3

Nilai *error* pada pengukuran *big obstacle* sangat kecil. Tinggi *obstacle* melebihi tinggi dari peletakan sensor.



Gambar 4.7 Grafik *Error* Deteksi *Big Obstacle*

Pada grafik dapat diketahui *peak error* dari ketiga ultrasonik yang telah diukur pada Ultrasonik *Right* didapatkan *peak error* 1 pada jarak 6 cm hinnga 10 cm, Ultrasonik *Front* didapatkan *peak error* 1 pada jarak 6 cm hingga 10 cm, dan Ultrasonik *Left* didapatkan *peak error* 1 pada jarak 8 cm dan 9 cm.

4.2.3 Perhitungan *Error* Sensor Ultrasonik Terhadap 3 Variasi Ukuran *Obstacle*

Perhitungan *error* kali ini melakukan pengukuran jarak antara sensor dan *obstacle*. *Obstacle* yang digunakan memiliki 3 ukuran yang berbeda. Tujuan dari pengukuran ini agar dapat mengetahui respon ultrasonik terhadap perbedaan ukuran *obstacle* dengan *range* jarak sebesar 0 cm hingga 200 cm. Ultrasonik diletakkan pada 2.7 cm diatas dasar.

Tabel 4.13 Perhitungan *Error* dari *Small Obstacle* untuk *Right Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	866	-866
2	10	11	-1
3	20	20	0
4	30	30	0
5	40	40	0
6	50	50.2	-0.2
7	60	60.2	-0.2
8	70	70.2	-0.2
9	80	80	0
10	90	163.4	-73.4
11	100	101.2	-1.2
12	110	110.4	-0.4
13	120	151.2	-31.2
14	130	330	-200
15	140	140	0
16	150	271.8	-121.8
17	160	272.2	-112.2
18	170	307.6	-137.6

Tabel 4.13 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
19	180	272	-92
20	190	306.8	-116.8
21	200	271.6	-71.6
	Total Error		307.8
	Error Rata-Rata		21.9857

Tabel 4.14 Perhitungan *Error* dari *Small Obstacle* untuk *Front Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	1064.4	-1064.4
2	10	7.5	2.5
3	20	15	5
4	30	22.5	7.5
5	40	30	10
6	50	37.5	12.5
7	60	45	15
8	70	52.5	17.5
9	80	60	20
10	90	67.5	22.5
11	100	75	25
12	110	82.5	27.5
13	120	90	30
14	130	97.5	32.5
15	140	105	35
16	150	112.5	37.5
17	160	120	40
18	170	127.5	42.5
19	180	135	45
20	190	142.5	47.5
21	200	150	50
	Total Error		262.5
	Error Rata-Rata		18.75

Tabel 4.15 Perhitungan *Error* dari *Small Obstacle* untuk *Left Ultrasonic*

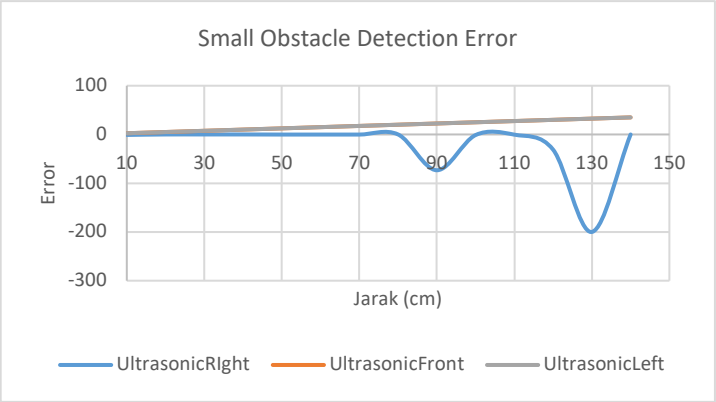
NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	924.8	-866
2	10	7.5	-1

Tabel 4.15 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
3	20	15	0
4	30	22.5	0
5	40	30	0
6	50	37.5	-0.2
7	60	45	-0.2
8	70	52.5	-0.2
9	80	60	0
10	90	67.5	-73.4
11	100	75	-1.2
12	110	82.5	-0.4
13	120	90	-31.2
14	130	97.5	-200
15	140	105	0
16	150	112.5	-121.8
17	160	120	-112.2
18	170	127.5	-137.6
19	180	135	-92
20	190	142.5	-116.8
21	200	150	-71.6
	Total Error		262.5
	Error Rata-Rata		18.75

Dari tabel diatas, dapat diketahui nilai *error* untuk *Small Obstacle* ini tidak dapat terbaca pada jarak 0 cm, kemudian pada jarak 150 cm hingga 200 cm didapatkan hasil pembacaan dari ultrasonik dengan nilai *error* yang jauh dari data – data sebelumnya. Data yang ada pada nilai 0 cm hingga 200 cm tersebut didapatkan karena ukuran *obstacle* kecil hingga gelombang ultrasonik tidak dapat memantulkan gelombang

dengan baik pada *obstacle* tersebut. Nilai *error* dari tabel diatas kemudian di plot menjadi:



Gambar 4.8 Grafik *Error* Deteksi *Small Obstacle*

Tabel 4.16 Perhitungan *Error* dari *Medium Obstacle* untuk *Right Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	4	-4
2	10	9.8	0.2
3	20	19.2	0.8
4	30	28.8	1.2
5	40	39	1
6	50	50	0
7	60	60	0
8	70	67.6	2.4
9	80	79.2	0.8
10	90	89	1
11	100	100.2	-0.2
12	110	110.8	-0.8
13	120	120.4	-0.4
14	130	128.8	1.2

Tabel 4.16 (lanjutan)

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
15	140	139	1
16	150	148.6	1.4
17	160	332.2	-172.2
18	170	293.2	-123.2
19	180	303.6	-123.6
20	190	316.8	-126.8
21	200	232	-32
Total Error			5.6
Error Rata-Rata			0.35

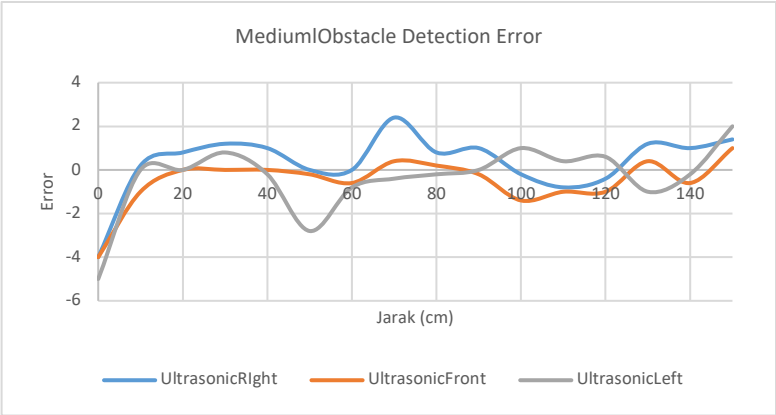
Tabel 4.17 Perhitungan *Error* dari *Medium Obstacle* untuk *Front Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	4	-4
2	10	11	-1
3	20	20	0
4	30	30	0
5	40	40	0
6	50	50.2	-0.2
7	60	60.6	-0.6
8	70	69.6	0.4
9	80	79.8	0.2
10	90	90.2	-0.2
11	100	101.4	-1.4
12	110	111	-1
13	120	121	-1
14	130	129.6	0.4
15	140	140.6	-0.6
16	150	149	1
17	160	271.8	-111.8
18	170	341.6	-171.6
19	180	273.2	-93.2
20	190	272.2	-82.2
21	200	270.6	-70.6
Total Error			-8
Error Rata-Rata			-0.5

Tabel 4.18 Perhitungan *Error* dari *Medium Obstacle* untuk *Left Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	5	-5
2	10	10	0
3	20	20	0
4	30	29.2	0.8
5	40	40.2	-0.2
6	50	52.8	-2.8
7	60	60.8	-0.8
8	70	70.4	-0.4
9	80	80.2	-0.2
10	90	90	0
11	100	99	1
12	110	109.6	0.4
13	120	119.4	0.6
14	130	131	-1
15	140	140.2	-0.2
16	150	148	2
17	160	271.2	-111.2
18	170	272	-102
19	180	270.6	-90.6
20	190	270.6	-80.6
21	200	273	-73
	Total Error		-5.8
	Error Rata-Rata		-0.3625

Dari tabel diatas, dapat diketahui nilai *error* untuk *Medium Obstacle* ini tidak dapat terbaca pada jarak 160 cm hingga 200 cm didapatkan hasil pembacaan dari ultrasonik dengan nilai *error* yang jauh dari data – data sebelumnya.



Gambar 4.9 Grafik *Error* Deteksi *Medium Obstacle*

Tabel 4.19 Perhitungan *Error* dari *Big Obstacle* untuk *Right Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	3	-3
2	10	9	1
3	20	19.2	0.8
4	30	29	1
5	40	39	1
6	50	48.8	1.2
7	60	58.6	1.4
8	70	68.4	1.6
9	80	79.2	0.8
10	90	89.2	0.8
11	100	97.6	2.4
12	110	107.4	2.6
13	120	117.8	2.2
14	130	127.2	2.8
15	140	139.2	0.8
16	150	148.6	1.4
17	160	158.4	1.6
18	170	167.8	2.2

Tabel 4.19 (lanjutan)

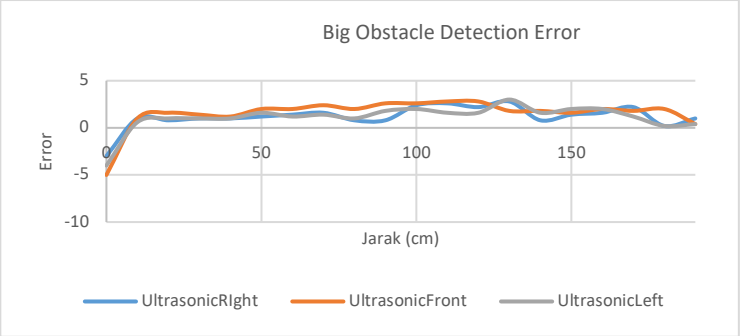
NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
19	180	179.8	0.2
20	190	189	1
21	200	258	-58
Total Error			23.8
Error Rata-Rata			1.19

Tabel 4.20 Perhitungan *Error* dari *Big Obstacle* untuk *Front Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	5	-5
2	10	9	1
3	20	18.4	1.6
4	30	28.6	1.4
5	40	38.8	1.2
6	50	48	2
7	60	58	2
8	70	67.6	2.4
9	80	78	2
10	90	87.4	2.6
11	100	97.4	2.6
12	110	107.2	2.8
13	120	117.2	2.8
14	130	128.2	1.8
15	140	138.2	1.8
16	150	148.4	1.6
17	160	158	2
18	170	168.2	1.8
19	180	178	2
20	190	189.6	0.4
21	200	239	-39
Total Error			30.8
Error Rata-Rata			1.54

Tabel 4.21 Perhitungan *Error* dari *Big Obstacle* untuk *Left Ultrasonic*

NO	Jarak (cm)	Rata-rata jarak terukur (cm)	error
1	0	4	-4
2	10	9.4	0.6
3	20	19	1
4	30	29	1
5	40	39	1
6	50	48.4	1.6
7	60	58.8	1.2
8	70	68.6	1.4
9	80	79	1
10	90	88.2	1.8
11	100	98	2
12	110	108.4	1.6
13	120	118.4	1.6
14	130	127	3
15	140	138.4	1.6
16	150	148	2
17	160	158	2
18	170	168.8	1.2
19	180	179.8	0.2
20	190	189.6	0.4
21	200	227.4	-27.4
Total Error			22.2
Error Rata-Rata			1.11



Gambar 4.10 Grafik *Error* Deteksi *Big Obstacle*

Dari tabel diatas, dapat diketahui nilai *error* untuk *Big Obstacle* ini tidak dapat terbaca pada jarak 200 cm saja.

4.3 Pengujian *Hardware* Pada Lintasan dengan 4 Kondisi yang Berbeda

Pengujian *hardware* ini dilakukan dengan *fuzzy logic controller* yang dirancang dan diubah menjadi bahasa pemrograman, kemudian di *compile* pada *software* pemrograman *Arduino IDE*. Pada program, pembacaan jarak pada sensor diberikan *delay* sebesar 10 milisekon dengan respon yang baik. Pembacaan jarak (s) dari sensor menghasilkan keluaran sudut belok (θ) kemudi dan kecepatan (ω) RPM motor penggerak. Pengujian ini merupakan validasi hasil perancangan *fuzzy logic controller* pada *prototype autonomous vehicle.*

Pengujian dilakukan pada lintasan lurus berjarak kurang lebih 150 cm dan lebar lintasan 60 cm, pinggiran lintasan diberikan penghalang setinggi kurang lebih 50 cm. Pengujian dilakukan dengan kondisi yang berbeda yaitu lintasan tanpa *obstacle*, lintasan dengan *obstacle* 10 cm X 10 cm, lintasan dengan *obstacle* 20 cm X 20 cm, dan lintasan dengan *obstacle* 35cm X 20cm. Pengujian dilakukan dengan kecepatan RPM Motor penggerak sebesar 3765 RPM dan dan kondisi kemudi

yaitu lurus pada lintasan. Pengambilan data didapatkan dari hasil monitoring dari *programming software* Arduino IDE, didapatkan hasil sebagai berikut

4.3.1 Pengujian *hardware* pada lintasan sepanjang 150 cm dengan lebar lintasan 60 cm, dan lintasan tidak diberi *obstacle.*

Tabel 4.22 Pengujian *Hardware* pada Lintasan Tanpa *Obstacle*

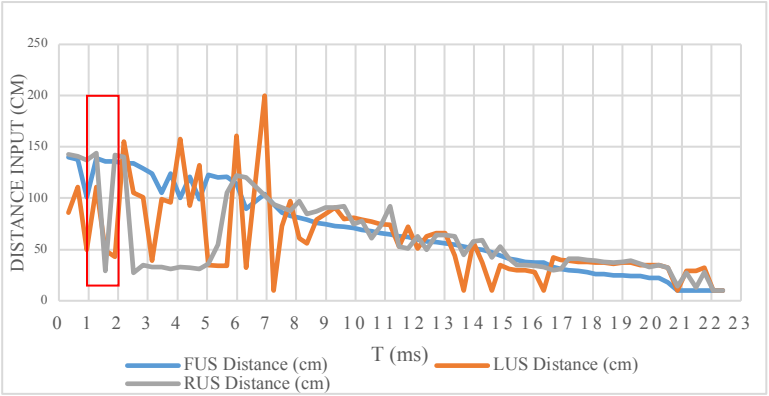
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_R(RPM)	Angle Output, θ(derajat)
0.322	140	86	143	3764.705882	5.36
0.635	138	111	141	3764.705882	1.95
0.947	101	50	137	3764.705882	18.39
1.26	139	111	144	3764.705882	6.85
1.573	136	49	29	4847.058824	-8.98
1.885	136	43	142	3764.705882	20.01
2.197	134	155	140	3811.764706	-17.82
2.51	134	105	27	3717.647059	-26.73
2.838	129	101	35	3670.588235	-24.24
3.15	124	39	33	5270.588235	-2.73
3.463	105	99	33	3670.588235	-25.61
3.776	124	96	31	3670.588235	-26.14
4.088	100	158	33	3764.705882	-22.78
4.417	121	93	32	3670.588235	-26.07
4.729	99	132	31	3717.647059	-25.28
5.041	123	35	36	5364.705882	0.46
5.354	120	34	55	4611.764706	8.44
5.666	121	34	105	3670.588235	25.21
5.995	114	161	122	3717.647059	-21.78

Tabel 4.22 (lanjutan)

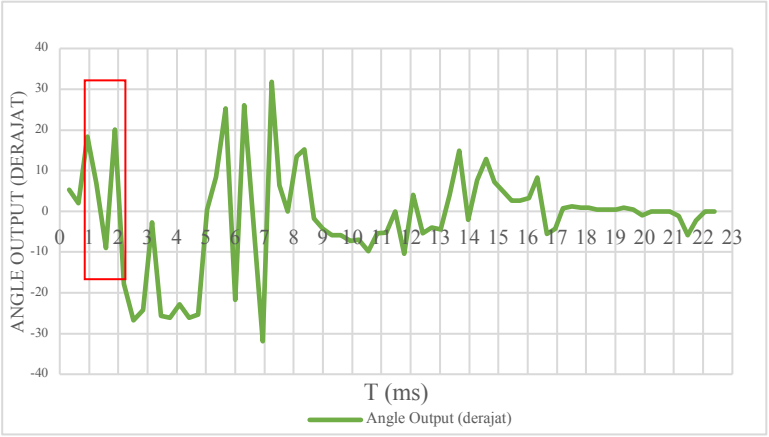
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_f(RPM)	Angle Output, θ(derajat)
6.307	90	32	120	3623.529412	26.1
6.932	104	200	103	3576.470588	-31.9
7.245	94	10	94	3576.470588	31.81
7.511	86	73	91	3670.588235	6.44
7.792	83	97	88	3623.529412	0
8.104	81	61	97	3717.647059	13.42
8.37	79	56	85	3764.705882	15.15
8.682	76	79	87	3670.588235	-1.75
8.995	75	85	91	3670.588235	-4.22
9.307	73	91	91	3670.588235	-5.79
9.62	72	80	92	3670.588235	-5.85
9.948	71	81	75	3670.588235	-7.24
10.229	69	79	78	3764.705882	-6.96
10.542	68	77	61	3858.823529	-9.77
10.854	66	75	73	3905.882353	-5.41
11.167	65	74	92	3717.647059	-5.22
11.464	63	53	53	4658.823529	0.01
11.776	62	72	51	4094.117647	-10.47
12.089	60	51	63	4376.470588	4.01
12.402	58	63	50	4376.470588	-5.32
12.714	57	66	64	4282.352941	-4.02
13.026	56	66	64	4329.411765	-4.47
13.339	55	44	63	4376.470588	4.01
13.652	53	10	45	4847.058824	14.86
13.964	51	57	58	4517.647059	-2.12
14.277	50	37	59	4470.588235	7.68
14.589	47	10	42	4800	12.87

Tabel 4.22 (lanjutan)

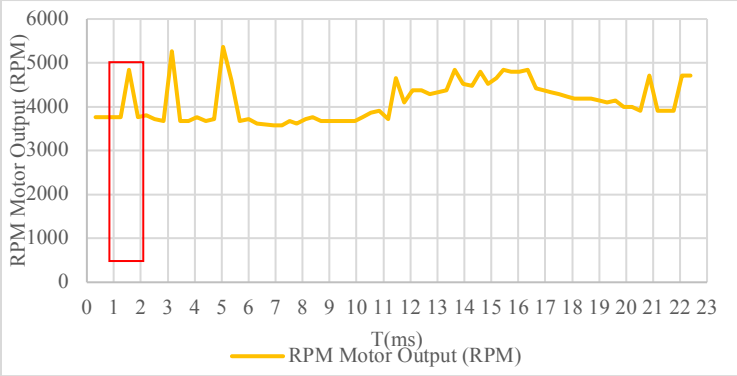
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_n(RPM)	Angle Output, θ(derajat)
14.87	44	35	53	4517.647059	7.18
15.183	41	31	41	4658.823529	4.64
15.449	40	30	35	4847.058824	2.61
15.73	38	30	35	4800	2.57
16.042	37	28	34	4800	3.24
16.339	37	10	33	4847.058824	8.27
16.667	33	42	30	4423.529412	-5.6
16.933	31	40	31	4376.470588	-4.21
17.199	30	39	41	4329.411765	0.81
17.511	29	38	41	4282.352941	1.24
17.824	28	38	40	4235.294118	0.83
18.089	26	37	39	4188.235294	0.85
18.371	26	37	38	4188.235294	0.43
18.699	25	36	37	4188.235294	0.44
19.011	25	37	38	4141.176471	0.43
19.293	24	37	39	4094.117647	0.85
19.605	24	35	36	4141.176471	0.46
19.918	22	35	33	4000	-0.95
20.23	22	35	35	4000	0
20.527	18	32	32	3905.882353	0
20.855	10	10	14	4705.882353	0
21.168	10	29	27	3905.882353	-1.15
21.48	10	29	13	3905.882353	-5.88
21.762	10	32	28	3905.882353	-2.17
22.074	10	10	10	4705.882353	0
22.387	10	10	10	4705.882353	0



Gambar 4.11 Grafik *Input* Jarak pada Lintasan Tanpa *Obstacle*



Gambar 4.12 Grafik *Output* Sudut Kemudi pada Lintasan Tanpa *Obstacle*



Gambar 4.13 Grafik *Output* RPM Motor pada Lintasan Tanpa *Obstacle*

Respon *output* yang didapatkan pada pengujian *hardware* dengan lintasan tanpa *obstacle* ini sesuai dengan *fuzzy logic controller* yang telah dirancang. Respon dari program dan respon gerak aktuator sudah sesuai dengan lintasan dengan kondisi yang diberikan. Hasil yang diharapkan pada percobaan ini yaitu *hardware* dapat berjalan lurus sesuai lintasan yang diberikan dengan sudut -15^0 hingga 15^0. Kemudi mencapai destinasi dengan dapat berjalan lurus, tetapi mengalami beberapa belokan seperti pada detik ke 1 ms hingga 2 ms, detik tersebut menyatakan pada **Gambar 4.11** jarak yang dideteksi oleh sensor ultrasonik mengalami beberapa pembacaan dari sensor ultrasonic bagiand epan (FUS) terbaca jarak sebesar 139 cm, 136 cm, 136 cm, 134 cm, kemudian yang terbaca dari ultrasonic bagian kiri (LUS) terbaca 111 cm, 49 cm, 43 cm, 155 cm, dan yang terbaca sensor ultrasonic kanan (RUS) terbaca 144 cm, 29 cm, 142 cm, 140 cm. Perbedaan pembacaan jarak dari tiap sensor menghasilkan sudut belok θ pada **Gambar 4.12** dan kecepatan motor ω_f pada **Gambar 4.13**

sebesar 6.85^0 berjalan lurus dengan kecepatan 3764.70 RPM, -8.98^0 berjalan lurus dengan kecepatan 4847.05 RPM, saat sensor ultrasonic LUS terbaca jarak 43 cm dengan jarak RUS 142 cm FUS 136 cm, maka sudut kemudi berbelok sebesar 20.01^0 berbelok ke kanan dan menurunkan kecepatannya sebesar 3764.70 RPM, -17.82^0 berbelok ke kiri dengan kecepatan sebesar 3811.76 RPM. Nilai sudut belok kemudi saat $-15^0 < \theta < 15^0$ menyatakan kemudi berjalan lurus, $15^0 < \theta < 45^0$ menyatakan kemudi berbelok kekanan, $-45^0 < \theta < -15^0$ menyatakan kemudi berbelok ke kiri.

4.3.2 Pengujian *hardware* pada lintasan sepanjang 150 cm dengan lebar lintasan 60 cm, dan lintasan diberikan *obstacle* 10 cm X 10 cm.

Tabel 4.23 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 10 cm X 10 cm

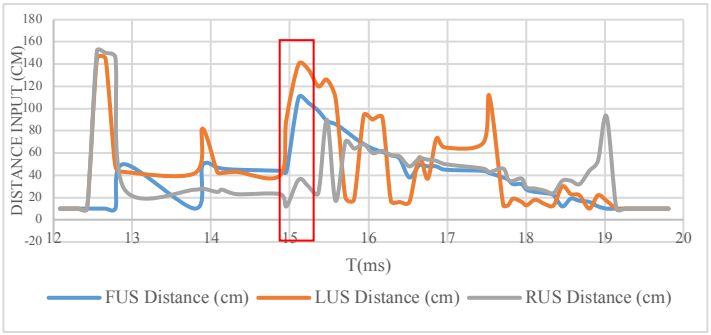
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_f(RPM)	Angle Output, θ(derajat)
12.082	10	10	10	4705.882353	0
12.207	10	10	10	4705.882353	0
12.316	10	10	10	4705.882353	0
12.426	10	10	10	4705.882353	0
12.551	10	145	152	3811.764706	-2.14
12.66	10	145	150	3811.764706	-5.59
12.785	10	48	146	3764.705882	2.68
13.891	49	82	28	3764.706	-25.29
14.079	47	43	25	4752.941	-9.44
14.141	46	42	27	4800	-7.66
14.329	45	43	23	4752.941	-10.65
14.879	44	40	23	4800	-9.49

Tabel 4.23 (lanjutan)

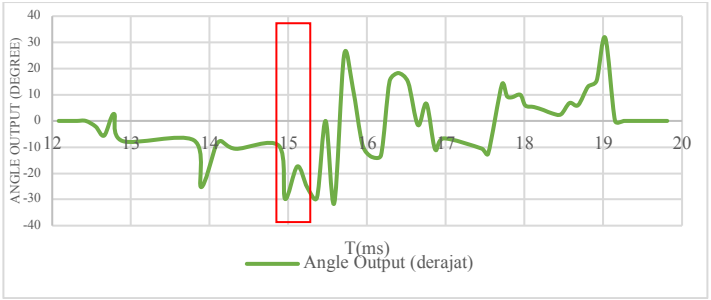
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_f(RPM)	Angle Output, θ(derajat)
14.957	43	90	12	3717.647	-29.78
15.113	110	140	36	3764.706	-22.38
15.238	105	135	30	3717.647	-25.32
15.363	98	120	24	3623.529	-29.26
15.472	89	126	90	3670.588	0
15.582	86	109	17	3623.529	-31.43
15.707	80	20	70	4000	25.16
15.816	74	18	64	4235.294	12.99
15.941	68	94	68	3717.647	-9.22
16.051	64	90	60	3764.706	-13.64
16.176	61	92	62	3717.647	-13.13
16.285	58	17	58	4470.588	19.15
16.395	55	16	57	3764.706	18.29
16.52	38	16	48	4752.941	14.8
16.645	49	56	55	4800	-1.58
16.754	48	37	54	4752.941	6.54
16.863	48	73	53	4800	-10.85
16.973	45	65	50	3717.647	-6.67
17.457	44	69	46	3764.706	-10.59
17.537	42	111	43	3717.647	-12.49
17.712	38	13	46	3623.529	13.94
17.774	36	13	36	4658.824	9.5
17.837	32	19	35	4564.706	9.01
17.946	32	16	37	4517.647	9.97
18.009	27	13	29	4564.706	5.89
18.118	25	18	28	4470.588	5.31

Tabel 4.23 (lanjutan)

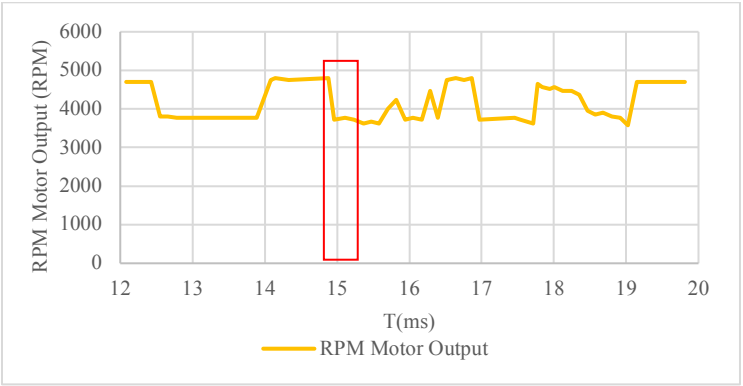
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_r(RPM)	Angle Output, θ(derajat)
18.352	22	13	24	4376.471	2.85
18.462	12	30	35	3952.941	2.52
18.571	19	23	35	3858.824	6.84
18.68	17	22	32	3905.882	6.03
18.805	16	10	44	3811.765	13.09
18.915	12	22	53	3764.706	15.39
19.024	10	17	93	3576.471	31.76
19.149	10	10	10	4705.882	0
19.259	10	10	10	4705.882	0
19.368	10	10	10	4705.882	0
19.477	10	10	10	4705.882	0
19.602	10	10	10	4705.882	0
19.712	10	10	10	4705.882	0
19.812	10	10	10	4705.882	0



Gambar 4.14 Grafik *Input* Jarak pada Lintasan dengan *Obstacle* 10 cm X 10 cm



Gambar 4.15 Grafik *Output* Sudut Kemudi pada Lintasan dengan *Obstacle* 10 cm X 10 cm



Gambar 4.16 Grafik *Output* RPM Motor Penggerak pada Lintasan dengan *Obstacle* 10 cm X 10 cm

Pengujian lintasan dengan diberikan *obstacle* sebesar 10 cm X 10 cm di sisi kanan lintasan dapat dilihat bahwa *prototype* memberikan respon yang sesuai terhadap pengendali pada *obstacle* tersebut. Kemudi menghindari *obstacle* yang berada pada sisi kanan dengan bergerak berbelok kekiri dengan kecepatan yang lebih rendah dari sebelumnya. **Gambar 4.14**, **Gambar 4.15**, dan **Gambar 4.16** menunjukkan grafik jarak yang terbaca oleh ketiga ultrasonic, sudut belok θ, dan kecepatan motor penggerak ω_f. Detik ke 14.897 ms hingga

15.363 ms dimana proses terjadinya belok ke kiri dari *prototype* untuk menghindari *obstacle.* Detik ke 14.897 ms hingga 15.363 ms terbaca jarak yang dideteksi oleh sensor ultrasonik mengalami beberapa pembacaan dari sensor ultrasonic bagian depan (FUS) terbaca jarak sebesar 44 cm, 43 cm, 110 cm, 15 cm, 98 cm. Jarak dari ultrasonic bagian kiri (LUS) terbaca 40 cm, 90 cm, 140 cm, 135 cm, 120 cm. Jarak yang terbaca sensor ultrasonic kanan (RUS) terbaca 23 cm, 12 cm, 36 cm, 30 cm, 24 cm. Perbedaan pembacaan jarak dari tiap sensor menghasilkan sudut belok θ pada **Gambar 4.15** dan kecepatan motor ω_f pada **Gambar 4.16** sebesar -9.49^0 berjalan lurus karena belum terbaca *obstacle* dengan kecepatan 4800 RPM, kemudian berbelok -29.78^0 berbelok kekiri karena mendeteksi *obstacle* pada sisi kanan dengan kecepatan 3717.64 RPM, karena sensor ultrasonic RUS masih membaca jarak dekat terhadap *obstacle* maka sudut kemudi masih berbelok sebesar -22.38^0 berbelok ke kiri dan menurunkan kecepatannya sebesar 3764.70 RPM, -25.32^0 berbelok ke kiri dengan kecepatan sebesar 3717.64 RPM, -29.26^0 berbelok ke kiri dengan kecepatan 3626.52 RPM. Nilai sudut belok kemudi saat $-150 < \theta < 150$ menyatakan kemudi berjalan lurus, $150 < \theta < 450$ menyatakan kemudi berbelok kekanan, $-450 < \theta < -150$ menyatakan kemudi berbelok ke kiri.

4.3.3 Pengujian *hardware* pada lintasan sepanjang 150 cm dengan lebar lintasan 60 cm, dan lintasan diberikan *obstacle* 20 cm X 20 cm.

Tabel 4.24 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 20 cm X 20 cm

T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_f(RPM)	Angle Output, θ(derajat)
03.035	24	99	24	3623.529	-29.83

Tabel 4.24 (lanjutan)

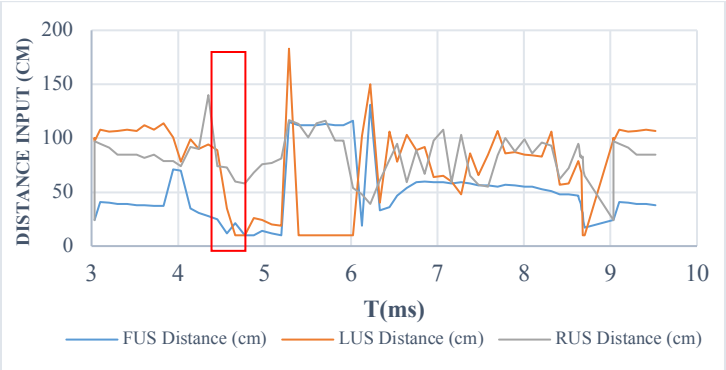
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_r(RPM)	Angle Output, θ(derajat)
03.035	24	100	24	3623.529	-29.83
03.098	41	108	95	3717.647	-22.11
03.207	40	106	91	3717.647	-22.54
03.301	39	107	85	3670.588	-22.97
03.410	39	108	85	3670.588	-22.97
03.520	38	107	85	3670.588	-23.4
03.613	38	112	82	3670.588	-23.4
03.723	37	108	85	3670.588	-23.84
03.832	37	114	79	3670.588	-23.84
03.941	71	101	79	3670.588	-7.24
04.035	70	78	74	3764.706	-5.4
04.144	35	99	92	3670.588	-24.72
04.238	31	90	90	3623.529	-26.53
04.348	28	94	140	3764.706	-25.56
04.457	25	89	74	3670.588	-28.37
04.566	12	35	73	3670.588	14.74
04.660	21	10	60	3764.706	20.07
04.770	10	10	58	3764.706	19.15
04.879	10	26	68	3717.647	18.94
04.973	14	24	76	3670.588	24.8
05.082	12	20	77	3670.588	29.38
05.191	10	19	81	3623.529	31.19
05.285	115	183	117	3623.529	-31.29
05.395	112	10	113	3623.529	31.48
05.504	112	10	101	3623.529	31.53
05.598	112	10	114	3623.529	31.43
05.707	113	10	116	3623.529	31.34

Tabel 4.24 (lanjutan)

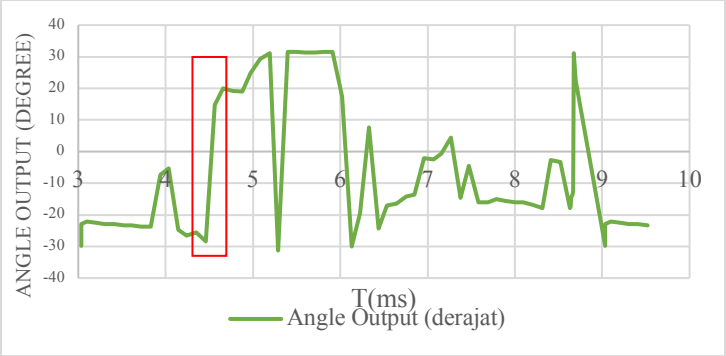
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_r(RPM)	Angle Output, θ(derajat)
05.817	112	10	98	3623.529	31.53
05.910	112	10	98	3623.529	31.53
06.020	116	10	54	4658.824	17.37
06.129	19	102	48	3764.706	-30.04
06.223	131	150	39	3811.765	-19.45
06.332	33	40	61	4141.176	7.55
06.441	36	106	80	3670.588	-24.28
06.535	47	78	95	3717.647	-17.02
06.645	54	103	59	3764.706	-16.48
06.754	59	89	89	3764.706	-14.14
06.848	60	92	67	3764.706	-13.64
06.957	59	64	98	3764.706	-2.04
07.066	59	65	108	3764.706	-2.57
07.160	58	60	59	4470.588	-0.57
07.270	59	48	103	3764.706	4.41
07.379	58	86	65	3764.706	-14.62
07.582	56	85	55	3764.706	-16.03
07.692	55	107	84	3764.706	-16.03
07.785	57	86	100	3764.706	-15.09
07.895	56	87	88	3764.706	-15.56
08.004	55	85	99	3764.706	-16.03
08.098	55	84	86	3764.706	-16.03
08.207	53	83	96	3764.706	-16.93
08.317	51	106	93	3623.529	-17.81
08.410	48	57	62	3623.529	-2.62
08.520	48	58	72	3670.588	-3.34
08.629	47	79	95	3717.647	-17.93

Tabel 4.24 (lanjutan)

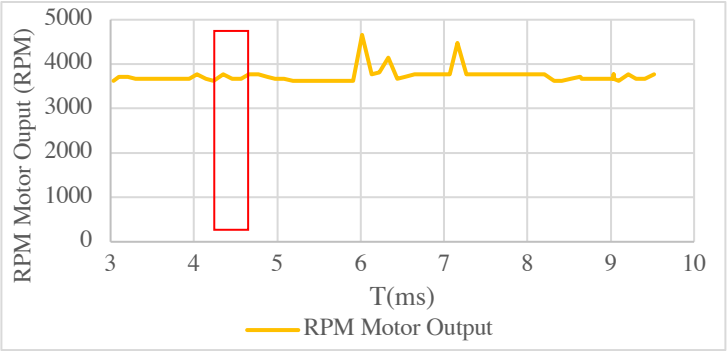
T(ms)	FUS Distance, s (cm)	LUS Distance, s (cm)	RUS Distance, s (cm)	RPM Output, ω_r(RPM)	Angle Output, θ(derajat)
08.638	44	77	93	3717.647	-17.38
08.648	41	72	83	3670.588	-14.48
08.658	39	70	84	3670.588	-13.71
08.667	34	68	82	3670.588	-12.53
08.677	31	10	83	3670.588	31.15
08.686	23	10	70	3670.588	25.16
08.698	17	10	65	3670.588	22.49
09.035	24	99	24	3670.588	-29.83
09.035	24	100	24	3764.706	-29.83
09.035	24	97	97	3670.588	-22.97
09.098	41	108	95	3623.529	-22.11
09.207	40	106	91	3764.706	-22.54
09.301	39	107	85	3670.588	-22.97
09.410	39	108	85	3670.588	-22.97
09.520	38	107	85	3764.706	-23.4



Gambar 4.17 Grafik *Input* Jarak pada Lintasan dengan *Obstacle* 20 cm X 20 cm



Gambar 4.18 Grafik *Output* Sudut Kemudi pada Lintasan dengan *Obstacle* 20 cm X 20 cm



Gambar 4.19 Grafik *Output* RPM Motor Penggerak pada Lintasan dengan *Obstacle* 20 cm X 20 cm

Pengujian *hardware* dengan lintasan lurus memiliki *obstacle* sebesar 20 cm X 20 cm yang ditempatkan pada tengah lintasan memiliki respon yang sesuai seperti pada **Tabel 4.24**, **Gambar 4.17**, **Gambar 4.18**, dan **Gambar 4.19**. Pengujian lintasan dengan diberikan *obstacle* sebesar 20 cm X 20 cm di sisi kanan lintasan dapat dilihat bahwa *prototype* memberikan respon yang sesuai terhadap pengendali pada *obstacle* tersebut. Kemudi menghindari *obstacle* yang berada pada sisi tengah

lintasan dengan bergerak berbelok ke kanan dengan kecepatan yang lebih rendah dari sebelumnya. **Gambar 4.17**, **Gambar 4.18**, dan **Gambar 4.19** menunjukkan grafik jarak yang terbaca oleh ketiga ultrasonic, sudut belok θ, dan kecepatan motor penggerak ω_f. Detik ke 04.457 ms hingga 04.770 ms dimana proses terjadinya belok ke kanan dari *prototype* untuk menghindari *obstacle.* Detik ke 04.457 ms hingga 04.770 ms terbaca jarak yang dideteksi oleh sensor ultrasonik mengalami beberapa pembacaan dari sensor ultrasonic bagian depan (FUS) terbaca jarak sebesar 25 cm, 12 cm, 21 cm, 10 cm. Jarak dari ultrasonic bagian kiri (LUS) terbaca 89 cm, 35 cm, 10 cm, 10 cm. Jarak yang terbaca sensor ultrasonic kanan (RUS) terbaca 74 cm, 73 cm, 60 cm, 58 cm. Perbedaan pembacaan jarak dari tiap sensor menghasilkan sudut belok θ pada **Gambar 4.18** dan kecepatan motor ω_f pada **Gambar 4.19** sebesar -28.37^0 penyesuaian *prototype* berbelok ke kiri terhadap lintasan dengan kecepatan 3760.56 RPM, kemudian 14.74^0 berjalan lurus dengan kecepatan 3670.58 RPM, karena sensor ultrasonic FUS dan LUS membaca jarak dekat terhadap *obstacle* maka sudut kemudi berbelok sebesar 20.07^0 berbelok ke kanan dan menurunkan kecepatannya sebesar 3764.70 RPM, 19.15^0 masih berbelok ke kanan dengan kecepatan sebesar 3764.70 RPM. Nilai sudut belok kemudi saat $-150 < \theta < 150$ menyatakan kemudi berjalan lurus, $150 < \theta < 450$ menyatakan kemudi berbelok kekanan, $-450 < \theta < -150$ menyatakan kemudi berbelok ke kiri.

4.3.4 Pengujian *hardware* pada lintasan sepanjang 150 cm dengan lebar lintasan 60 cm, dan lintasan diberikan *obstacle* 35 cm X 20 cm.

Tabel 4.25 Pengujian *Hardware* pada Lintasan dengan *Obstacle* 35 cm X 20 cm

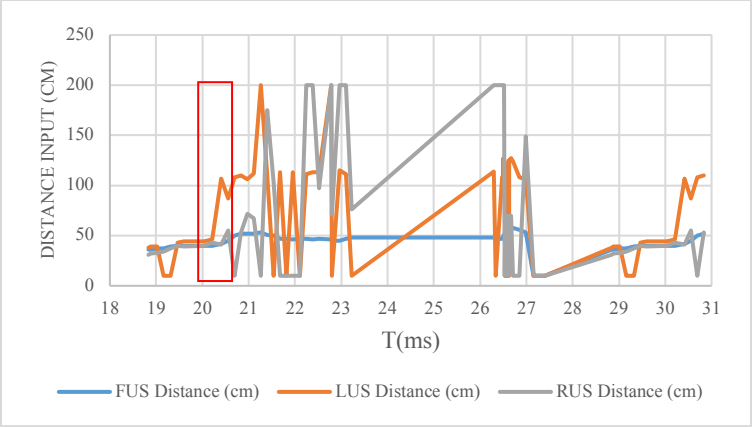
T(Sekon)	FUS Distance (cm)	LUS Distance (cm)	RUS Distance (cm)	PWM Output (PWM)	Angle Output (derajat)
18.833	36	38	31	4611.765	-3.34
18.879	36	39	32	4564.706	-3.25
19.020	37	39	33	4611.765	-2.73
19.161	37	10	34	4800	8.73
19.317	39	10	37	4752.941	10.13
19.458	40	43	40	4611.765	-1.18
19.598	40	44	39	4564.706	-1.98
20.036	40	44	40	4564.706	-1.56
20.208	40	46	43	4517.647	-1.10
20.411	42	107	41	3717.647	-21.94
20.552	45	87	55	3764.706	-18.95
20.692	50	108	10	3764.706	-29.91
20.833	52	110	53	3764.706	-17.10
20.973	52	106	72	3764.706	-17.37
21.114	52	112	67	3764.706	-17.37
21.255	53	200	10	3764.706	-29.70
21.395	51	110	175	3764.706	20.23
21.536	50	10	103	3764.706	29.91
21.677	47	113	10	3717.647	-30.11
21.817	47	10	10	5976.471	-0.00
21.958	46	113	10	3717.647	-30.18
22.099	47	10	10	5976.471	-0.00

Tabel 4.25 (lanjutan)

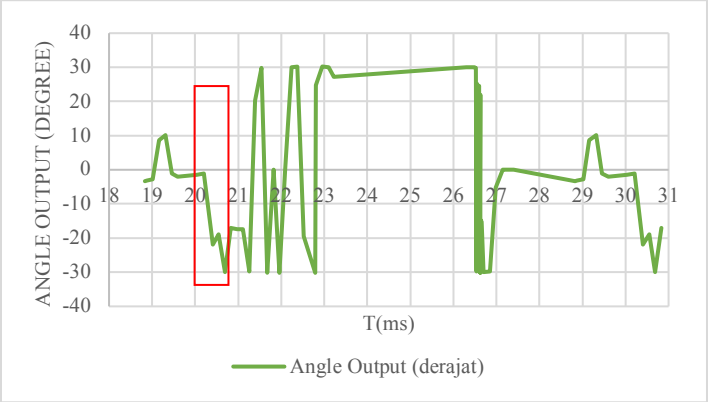
T(Sekon)	FUS Distance (cm)	LUS Distance (cm)	RUS Distance (cm)	PWM Output (PWM)	Angle Output (derajat)
22.239	47	111	200	3717.647	30.11
22.380	46	113	200	3717.647	30.18
22.521	47	113	97	3717.647	-19.54
22.786	46	200	200	3717.647	-30.18
22.802	45	10	71	4047.059	24.77
22.958	45	115	200	3717.647	30.24
23.099	47	111	200	3717.647	30.11
23.224	48	10	76	3905.882	27.27
26.302	48	114	200	3764.706	30.04
26.333	47	10	200	3717.647	30.11
26.474	47	108	200	3717.647	30.11
26.484	48	106	200	3764.706	30.04
26.493	49	105	200	3764.706	29.97
26.502	50	127	200	3764.706	29.91
26.512	50	108	200	3764.706	29.91
26.522	50	108	200	3764.706	29.91
26.532	52	109	10	3764.706	-29.77
26.538	58	10	67	4235.294	22.17
26.546	10	10	70	3670.588	25.16
26.556	60	43	70	4094.118	5.93
26.566	60	51	11	4752.941	-15.97
26.575	10	51	70	3764.706	1.88
26.580	59	10	12	6023.529	0.00
26.590	10	10	69	3717.647	24.60
26.611	58	125	10	3764.706	-29.98
26.621	61	10	65	4235.294	21.87

Tabel 4.25 (lanjutan)

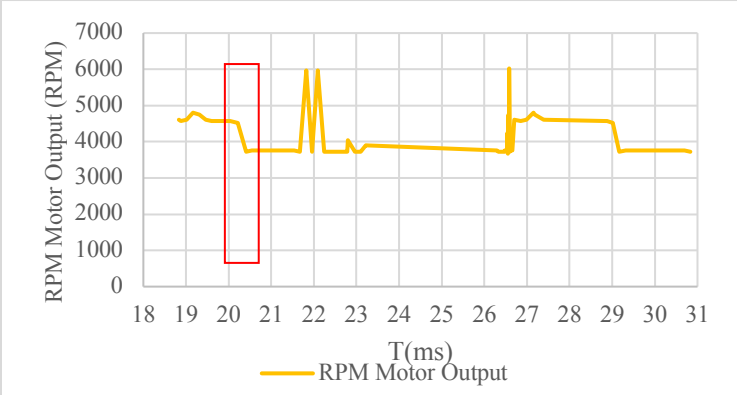
T(Sekon)	FUS Distance (cm)	LUS Distance (cm)	RUS Distance (cm)	PWM Output (PWM)	Angle Output (derajat)
26.628	60	106	12	3764.706	-30.09
26.632	58	124	70	3764.706	-14.62
26.642	57	126	69	3764.706	-15.09
26.654	57	125	70	3764.706	-15.09
26.664	57	127	70	3764.706	-15.09
26.708	58	124	10	4611.765	-29.98
26.849	56	108	10	4564.706	-29.87
26.990	53	106	149	4611.765	-5.17
27.146	10	10	13	4800	-0.00
27.177	10	10	10	4752.941	-0.00
27.396	10	10	10	4611.765	-0.00
28.833	36	38	31	4564.706	-3.34
28.879	36	39	32	4564.706	-3.25
29.020	37	39	33	4517.647	-2.73
29.161	37	10	34	3717.647	8.73
29.317	39	10	37	3764.706	10.13
29.458	40	43	40	3764.706	-1.18
29.598	40	44	39	3764.706	-1.98
30.036	40	44	40	3764.706	-1.56
30.208	40	46	43	3764.706	-1.10
30.411	42	107	41	3764.706	-21.94
30.552	45	87	55	3764.706	-18.95
30.692	50	108	10	3764.706	-29.91
30.833	52	110	53	3717.647	-17.10



Gambar 4.20 Grafik *Input* Jarak pada Lintasan dengan *Obstacle* 35 cm X 20 cm



Gambar 4.21 Grafik *Output* Sudut Kemudi pada Lintasan dengan *Obstacle* 35 cm X 20 cm



Gambar 4.22 Grafik *Output* RPM Motor Penggerak pada Lintasan dengan *Obstacle* 35cm X 20 cm

Pengujian *hardware* dengan lintasan lurus memiliki *obstacle* sebesar 35 cm X 20 cm yang ditempatkan kanan awal lintasan memiliki respon yang sesuai seperti pada **Tabel 4.25**, **Gambar 4.20**, **Gambar 4.21** dan **Gambar 4.22** Pengujian lintasan dengan diberikan *obstacle* sebesar 35 cm X 20 cm di sisi kanan lintasan dapat dilihat bahwa *prototype* memberikan respon yang sesuai terhadap pengendali pada *obstacle* tersebut. Kemudi menghindari *obstacle* yang berada pada sisi tengah lintasan dengan bergerak berbelok ke kiri dengan kecepatan yang lebih rendah dari sebelumnya. **Gambar 4.20**, **Gambar 4.21**, dan **Gambar 4.22** menunjukkan grafik jarak yang terbaca oleh ketiga ultrasonic, sudut belok θ, dan kecepatan motor penggerak ω_f. Detik ke 20.208 ms hingga 20.833 ms dimana proses terjadinya belok ke kanan dari *prototype* untuk menghindari *obstacle.* Detik ke 20.208 ms hingga 20.833 ms terbaca jarak yang dideteksi oleh sensor ultrasonik mengalami beberapa pembacaan dari sensor ultrasonic bagian depan (FUS) terbaca jarak sebesar 40 cm, 42 cm, 45 cm, 50 cm, 52 cm. Jarak

dari ultrasonic bagian kiri (LUS) terbaca 46 cm, 107 cm, 87 cm, 108 cm, 110 cm. Jarak yang terbaca sensor ultrasonic kanan (RUS) terbaca 43 cm, 41 cm, 55 cm, 10 cm, 53 cm. Perbedaan pembacaan jarak dari tiap sensor menghasilkan sudut belok θ pada **Gambar 4.21** dan kecepatan motor ω_f pada **Gambar 4.22** sebesar -1.10^0 penyesuaian *prototype* berjalan lurus terhadap lintasan dengan kecepatan 4517.64 RPM, kemudian berbelok kiri sebesar -21.94^0 dengan kecepatan 3717.64 RPM, karena sensor ultrasonic FUS dan RUS masih membaca jarak dekat terhadap *obstacle* maka sudut kemudi berbelok sebesar -18.95^0 berbelok ke kiri dengan kecepatan sebesar 3764.70 RPM, sensor ultrasonic RUS mendeteksi jarak terhadap *obstacle* sebesar 10 cm sehingga *prototype* berbelok hingga -29.91^0 dengan kecepatan sebesar 3764.70 RPM, kemudian *prototype* menyesuaikan dengan jarak yang terdeksi -17.10^0 dengan kecepatan 3764.70 RPM. Nilai sudut belok kemudi saat $-15^0 < \theta < 15^0$ menyatakan kemudi berjalan lurus, $15^0 < \theta < 45^0$ menyatakan kemudi berbelok kekanan, $-45^0 < \theta < -15^0$ menyatakan kemudi berbelok ke kiri.

Dari hasil pengujian pada Bab 4 ini, didapatkan parameter dari sistem pengendalian *fuzzy logic* yang telah dirancang serta mendapatkan nilai spesifikasi pada hasil rancang *prototype autonomous vehicle* agar *prototype* dapat bekerja sesuai performansi yang diinginkan sehingga dapat dinyatakan penelitian ini berhasil.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisa data yang telah dilakukan maka kesimpulan dari penelitian ini adalah sebagai berikut:

1. Perancangan sistem kendali *steering prototype autonomous vehicle* dengan metode *fuzzy logic* yang diuji pada lintasan yang telah ditentukan teridiri dari 4 *membership function input* dan 2 *membership function output*. *Membership function input* terdiri dari *RightUltrasonic(s)*, *FrontUltrasonic(s)*, *LeftUltrasonic (s)* dengan *range* 0 cm hingga 200 cm, dan kecepatan motor keudi *RPMMot*(ω_s) dengan *range* 3765 RPM hingga 12000 RPM. *Membership function output* terdiri dari sudut belok kemudi *SteerAngle* (θ) dengan *range* -45 derajat hingga 45 derajat, nilai minus pada *range* tersebut menyatakan sudut belok kearah kiri, dan *output* kecepatan motor penggerak *RPMMot*(ω_f) dengan *range* 3765 RPM hingga 12000 RPM. *Rule Base* yang dirancang pada sistem kendali ini sebanya 54 *rules*.
2. Rancang bangun sistem pengendali kemudi *prototype autonomous vehicle* ini telah berhasil dirancang dengan algoritma *fuzzy* yang di *compile* pada *microcontroller teensy* 3.6 kemudian di integrasikan dengan komponen – komponen pada *prototype*, kemudian melakukan pengujian *prototype* pada lintasan yang telah ditentukan sebagai validasi dari keberhasilan rancang bangun *prototype* ini.
3. Komponen yang digunakan pada rancang bangun ini antara lain, *chassis* mobil *remote control* sebagai kerangka dari *prototype*, 2 buah motor DC sebagai aktuator kemudi dan penggerak *prototype*, sensor ultrasonik sebagai sumber informasi jarak untuk

prototype, serta *motor driver* dan *teensy* sebagai *controller* pada *prototype*.

4. Spesifikasi sistem kemudi agar mampu bekerja trintegrasi dengan sistem kendali hasil rancangan didapatkan dengan cara melakukan perhitungan *error* pada pembacaan jarak oleh ketiga sensor ultrasonik yang dapat terbaca dengan baik sebesar 150 cm dan tinggi *obstacle* dengan tinggi minimal 2.1 cm dapat terbaca dengan baik pada jarak 3 cm. Kecepatan motor (ω) penggerak memiliki nilai ω minimal sebesar 3765 RPM dengan beban *hardware* 275gram dan panjang 52 cm.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan maka saran yang dapat diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Sistem pengendalian berbasis Fuzzy untuk penelitian selanjutnya sebaiknya dikaji lagi agar pemberian logika fuzzy memiliki parameter yang lebih baik dan akurat.
2. Optimasi perlu dilakukan pada penelitian ini untuk meningkatkan performa dari sistem kemudi.

DAFTAR PUSTAKA

Ingrand, F., & Ghallab, M. Robotics and Artificial Intelligence: a Perspective on Deliberation Functions. AI Communication, (ISSN 0921-7126), 1-19.

Litman, T. (2018). Autonomous Vehicle Implementation Predictions: Implications for Transport Planning. Victoria Transport Policy Institute, 3-32.

Hyoven, M., & Huhtala, K. (2012). OUTDOOR OBSTACLE DETECTION USING ULTRASONIC SENSORS FOR AN AUTONOMOUS VEHICLE ENSURING SAFE OPERATIONS (M.Sc). Tampere University of Technology.

Cui, H., Li, Y., & Liu, J. (2016). An Obstacle Detection Algorithm Based on Ultrasonic Sensors for Autonomous Land Vehicle. International Conference On Mechatronics, Control And Automation Engineering, MCAE 2016, 0147-0150.

Woon, L. (2014). Obstacle Avoidance Robot Applying Fuzzy Logic Control System (MS). Universiti Tun Hussein Onn Malaysia.

Sailan, K., Kuhnert, K., & Hardt, S. (2015). Obstacle Avoidance Strategy using Fuzzy Logic Steering Control of Amphibious Autonomous Vehicle. International Journal Of Innovative Science, Engineering & Technology, 2(10), 143-148.

Ismail, Z., Naim, S., Ayob, A., & Amat, A. (2017). Mobile Robot Controller Based on Fuzzy Logic System in Uneven Terrain. Discovering Mathematics, 39(2), 70-82.

Miniature Motors. (2012). Retrieved from http://www.farnell.com/datasheets/1662763.pdf?_ga=2.79836311.1699765252.1532024055-456408074.1532024055.

Bathia, A. *DC Generators and Motors* (4th ed., pp. 1-1 - 1-6, 2-8 - 2-9). New York: Continuing Education and

Development, Inc.

Sharma, P., & Mohapatra, P. (2011). *DESIGNING A REAL TIME EMBEDDED CONTROLLER USING DATA ACQUISITION SYSTEM FOR A DC MOTOR SPEED CONTROL* (BS). National Institute of Technology Rourkela.

Andika, D. (2018). MAKALAH MOTOR DC. [online] Academia.edu. Available at: https://www.academia.edu/9091244/MAKALAH_MOTOR_DC?auto=download [Accessed 31 Jan. 2018].

Khuswah, R. and Wadhwani, S. (2013). Speed Control of Separately Excited Dc Motor Using Fuzzy Logic Controller. *International Journal of Engineering Trends and Technology (IJETT)*, 4(6), pp.2518-2519.

Fathoni, K., & Suni, A. (2016). Perancangan Kendali Kecepatan Motor Arus Searah Menggunakan Metode Root LocusPerancangan Kendali Kecepatan Motor Arus Searah Menggunakan Metode Root Locus. Jurnal Teknik Elektro, 8(2), 40.

Lim, B., Keoh, S., & Thing, V. (2018). Autonomous Vehicle Ultrasonic Sensor Vulnerability and Impact Assessment. IEEE, 978-1-4673-9944-9(18), 231-236.

Sultana, Y., & Manzke, R. (2018). Open-Source Audio Platform for Embedded Systems (MS). University of Applied Sciences Kiel.

Yanuary, A. (2010). *SISTEM KEMUDI* (pp. 5-6). Semarang: UNIVERSITAS NEGERI SEMARANG.

Arduino Motor Shield Rev3. (2018). Retrieved from https://store.arduino.cc/usa/arduino-motor-shield-rev3.

Firdaus, M., Syaryadhi, M. and Rahman, A. (2017). Pengendalian Robot Moil Otonom Pemotong Rumput Menggunakan Metode Logika Fuzzy. KTEKTRO: Jurnal Online Teknik Elektro, 2(2), pp.36-43.

Rojas, M., Ponce, P., & Molina, A. (2014). Novel Fuzzy Logic Controller based on Time Delay Inputs for a

Conventional Electric Wheelchair. Revista Mexicana De Ingeniería Biomédica, 3(2), 2.

Yerubandi, V., Reddy, Y., & Reddy, M. (2015). Navigation system for an autonomous robot using fuzzy logic. International Journal Of Scientific And Research Publications, 5(2), 2-3.

LAMPIRAN A

```
1.     #include "fis_header.h"
2.
3.     // Number of inputs to the fuzzy inference s
ystem
4.     const int fis_gcI = 4;
5.     // Number of outputs to the fuzzy inference
system
6.     const int fis_gcO = 2;
7.     // Number of rules to the fuzzy inference sy
stem
8.     const int fis_gcR = 54;
9.
10.    FIS_TYPE g_fisInput[fis_gcI];
11.    FIS_TYPE g_fisOutput[fis_gcO];
12.
13.
14.    #define Mot_DIR 22  //IB1
15.    #define Mot_PWM 23   //IA1
16.    #define Steer_DIR 19   //Ib2
17.    #define Steer_PWM 20   //Ia2
18.    #define Mot_brake 21
19.    #define Steer_brake 18
20.    #define inf
21.
22.
23.    const int frontEchoPin = 6;
24.    const int frontTriggerPin = 5;
25.    const int leftEchoPin = 3;
26.    const int leftTriggerPin = 2;
27.    const int rightEchoPin = 9;
28.    const int rightTriggerPin = 8;
29.    int frontDistanceCm,leftDistanceCm, rightDis
tanceCm;
30.
31.    int pwm=80;
32.    int angle;
33.
34.
35.
36.    void setup()
37.    {
```

```
38.       Serial.begin(9600);
39.       Serial.println ("Date & Time, frontDistanc
eCm,leftDistanceCm, rightDistanceCm, pwm, angle");
40.
41.       pinMode(Mot_DIR,OUTPUT);
42.       pinMode(Mot_brake,OUTPUT);
43.       pinMode(Steer_DIR,OUTPUT);
44.       pinMode(Steer_brake,OUTPUT);
45.
46.       pinMode(frontTriggerPin, OUTPUT); //output
 teensy
47.       pinMode(frontEchoPin, INPUT); //input teen
sy
48.       pinMode(leftTriggerPin, OUTPUT);
49.       pinMode(leftEchoPin, INPUT);
50.       pinMode(rightTriggerPin, OUTPUT);
51.       pinMode(rightEchoPin, INPUT);
52.
53.       digitalWrite(Mot_brake,HIGH);
54.       digitalWrite(Steer_brake,HIGH);
55.
56.    }
57.
58.
59.    void loop()
60.    {
61.         checkFrontDistance();
62.          checkRightDistance();
63.          checkLeftDistance();
64.
65.        //Read Input: US1
66.        g_fisInput[0] =leftDistanceCm;
67.        // Read Input: US2
68.        g_fisInput[1] =frontDistanceCm;
69.        // Read Input: US3
70.        g_fisInput[2] = rightDistanceCm;
71.        // Read Input: Vi
72.        g_fisInput[3] = pwm;
73.
74.
75.        g_fisOutput[0] = 0;
76.        g_fisOutput[1] = 0;
77.
78.        fis_evaluate();
```

```
79.
80.         pwm=g_fisOutput[1];
81.         angle=g_fisOutput[0]*10; //10ms adalah d
elay pembacaan
82.         Serial.print (",");
83.         Serial.print(pwm);
84.         Serial.print("\t");
85.         Serial.print (",");
86.         Serial.print(g_fisOutput[0]);1111111;
87.         Serial.println("\t");
88.
89.         digitalWrite(Mot_brake,LOW);
90.         digitalWrite(Mot_DIR,LOW);
91.          // Set output vlaue: Vo
92.         analogWrite(Mot_PWM , g_fisOutput[1]);
93.
94.         if(angle<0)
95.         {
96.           // Set output vlaue: WHEELANGLE
97.           digitalWrite(Steer_brake,LOW);
98.           digitalWrite(Steer_DIR,LOW); //kiri
99.           analogWrite(Steer_PWM,255);
100.          delay(angle*-1);
101.          digitalWrite(Steer_brake,HIGH);
102.        }
103.        else if (angle>0)
104.        {
105.          digitalWrite(Steer_brake,LOW);
106.          digitalWrite(Steer_DIR,HIGH); //kanan
107.          analogWrite(Steer_PWM,255);
108.          delay(angle);
109.          digitalWrite(Steer_brake,HIGH);
110.        }
111.        else
112.        {
113.          delay(100);
114.        }
115.    }
116.
117.
118.
119.    void checkFrontDistance() {
120.      digitalWrite(frontTriggerPin, LOW);
121.      delayMicroseconds(2);
```

```
122.     digitalWrite(frontTriggerPin, HIGH);
123.     delayMicroseconds(10);
124.     digitalWrite(frontTriggerPin, LOW);
125.     const unsigned long frontDuration = pulseI
n(frontEchoPin, HIGH);
126.     frontDistanceCm = frontDuration/29/2;
127.     if(frontDistanceCm>=200)
128.     {
129.       frontDistanceCm=200;
130.
131.     }
132.     if(frontDistanceCm<=10)
133.     {
134.       frontDistanceCm=10;
135.
136.     }
137.     Serial.print (",");
138.     Serial.print(frontDistanceCm);
139.     Serial.print("\t");
140.   }
141.   void checkLeftDistance() {
142.     digitalWrite(leftTriggerPin, LOW);
143.     delayMicroseconds(2);
144.     digitalWrite(leftTriggerPin, HIGH);
145.     delayMicroseconds(10);
146.     digitalWrite(leftTriggerPin, LOW);
147.     const unsigned long leftDuration = pulseIn
(leftEchoPin, HIGH);
148.     leftDistanceCm = leftDuration/29/2;
149.     if(leftDistanceCm>=200)
150.     {
151.       leftDistanceCm=200;
152.     }
153.     if(leftDistanceCm<=10)
154.     {
155.       leftDistanceCm=10;
156.
157.     }
158.     Serial.print (",");
159.     Serial.print(leftDistanceCm);
160.     Serial.print("\t");
161.   }
162.   void checkRightDistance() {
163.     digitalWrite(rightTriggerPin, LOW);
```

```
164.      delayMicroseconds(2);
165.      digitalWrite(rightTriggerPin, HIGH);
166.      delayMicroseconds(10);
167.      digitalWrite(rightTriggerPin, LOW);
168.      const unsigned long rightDuration = pulseI
n(rightEchoPin, HIGH);
169.      rightDistanceCm = rightDuration/29/2;
170.      if(rightDistanceCm>=200)
171.      {
172.        rightDistanceCm=200;
173.      }
174.      if(rightDistanceCm<=10)
175.      {
176.        rightDistanceCm=10;
177.
178.      }
179.      Serial.print (",");
180.      Serial.print(rightDistanceCm);
181.      Serial.print("\t");
182.    }
183.
184.
185.
186.    //*****************************************
*****************************
187.    // Support functions for Fuzzy Inference Sys
tem
188.    //*****************************************
*****************************
189.    // Trapezoidal Member Function
190.    FIS_TYPE fis_trapmf(FIS_TYPE x, FIS_TYPE* p)
191.    {
192.        FIS_TYPE a = p[0], b = p[1], c = p[2], d
 = p[3];
193.        FIS_TYPE t1 = ((x <= c) ? 1 : ((d < x) ?
 0 : ((c != d) ? ((d - x) / (d - c)) : 0)));
194.        FIS_TYPE t2 = ((b <= x) ? 1 : ((x < a) ?
 0 : ((a != b) ? ((x - a) / (b - a)) : 0)));
195.        return (FIS_TYPE) min(t1, t2);
196.    }
197.
198.    // Triangular Member Function
199.    FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
200.    {
```

```
201.        FIS_TYPE a = p[0], b = p[1], c = p[2];
202.        FIS_TYPE t1 = (x - a) / (b - a);
203.        FIS_TYPE t2 = (c - x) / (c - b);
204.        if ((a == b) && (b == c)) return (FIS_TY
PE) (x == a);
205.        if (a == b) return (FIS_TYPE) (t2*(b <=
x)*(x <= c));
206.        if (b == c) return (FIS_TYPE) (t1*(a <=
x)*(x <= b));
207.        t1 = min(t1, t2);
208.        return (FIS_TYPE) max(t1, 0);
209.    }
210.
211.    FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
212.    {
213.        return min(a, b);
214.    }
215.
216.    FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
217.    {
218.        return max(a, b);
219.    }
220.
221.    FIS_TYPE fis_array_operation(FIS_TYPE *array
, int size, _FIS_ARR_OP pfnOp)
222.    {
223.        int i;
224.        FIS_TYPE ret = 0;
225.
226.        if (size == 0) return ret;
227.        if (size == 1) return array[0];
228.
229.        ret = array[0];
230.        for (i = 1; i < size; i++)
231.        {
232.            ret = (*pfnOp)(ret, array[i]);
233.        }
234.
235.        return ret;
236.    }
237.
238.
239.    //*****************************************
*****************************
```

```
240.   // Data for Fuzzy Inference System

241.   //*****************************************
*****************************
242.   // Pointers to the implementations of member
 functions
243.   _FIS_MF fis_gMF[] =
244.   {
245.       fis_trapmf, fis_trimf
246.   };
247.
248.   // Count of member function for each Input
249.   int fis_gIMFCount[] = { 3, 3, 3, 2 };
250.
251.   // Count of member function for each Output
252.   int fis_gOMFCount[] = { 3, 2 };
253.
254.   // Coefficients for the Input Member Functio
ns
255.   FIS_TYPE fis_gMFI0Coeff1[] = { -99999, -
15, 15, 79.6296296296296 };
256.   FIS_TYPE fis_gMFI0Coeff2[] = { 20, 100, 180
};
257.   FIS_TYPE fis_gMFI0Coeff3[] = { 140, 180, 208
, 99999 };
258.   FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coef
f1, fis_gMFI0Coeff2, fis_gMFI0Coeff3 };
259.   FIS_TYPE fis_gMFI1Coeff1[] = { -99999, -
15, 15, 79.6296296296296 };
260.   FIS_TYPE fis_gMFI1Coeff2[] = { 20, 100, 180
};
261.   FIS_TYPE fis_gMFI1Coeff3[] = { 140, 180, 208
, 99999 };
262.   FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coef
f1, fis_gMFI1Coeff2, fis_gMFI1Coeff3 };
263.   FIS_TYPE fis_gMFI2Coeff1[] = { -99999, -
15, 15, 80.6878306878307 };
264.   FIS_TYPE fis_gMFI2Coeff2[] = { 20, 100, 180
};
265.   FIS_TYPE fis_gMFI2Coeff3[] = { 140, 180, 208
, 99999 };
266.   FIS_TYPE* fis_gMFI2Coeff[] = { fis_gMFI2Coef
f1, fis_gMFI2Coeff2, fis_gMFI2Coeff3 };
```

```
267.   FIS_TYPE fis_gMFI3Coeff1[] = { -
99999, 70, 110, 180 };
268.   FIS_TYPE fis_gMFI3Coeff2[] = { 150, 220, 270
, 99999 };
269.   FIS_TYPE* fis_gMFI3Coeff[] = { fis_gMFI3Coef
f1, fis_gMFI3Coeff2 };
270.   FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coef
f, fis_gMFI1Coeff, fis_gMFI2Coeff, fis_gMFI3Coeff }
;
271.
272.   // Coefficients for the Output Member Functi
ons
273.   FIS_TYPE fis_gMFO0Coeff1[] = { -99999, -
50, -30, -10 };
274.   FIS_TYPE fis_gMFO0Coeff2[] = { -15, 0, 15 };
275.   FIS_TYPE fis_gMFO0Coeff3[] = { 10, 30, 50, 9
9999 };
276.   FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coef
f1, fis_gMFO0Coeff2, fis_gMFO0Coeff3 };
277.   FIS_TYPE fis_gMFO1Coeff1[] = { -
70, 50, 80, 120 };
278.   FIS_TYPE fis_gMFO1Coeff2[] = { 100, 130, 160
, 180 };
279.   FIS_TYPE* fis_gMFO1Coeff[] = { fis_gMFO1Coef
f1, fis_gMFO1Coeff2 };
280.   FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coef
f, fis_gMFO1Coeff };
281.
282.   // Input membership function set
283.   int fis_gMFI0[] = { 0, 1, 0 };
284.   int fis_gMFI1[] = { 0, 1, 0 };
285.   int fis_gMFI2[] = { 0, 1, 0 };
286.   int fis_gMFI3[] = { 0, 0 };
287.   int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1, fi
s_gMFI2, fis_gMFI3};
288.
289.   // Output membership function set
290.   int fis_gMFO0[] = { 0, 1, 0 };
291.   int fis_gMFO1[] = { 0, 0 };
292.   int* fis_gMFO[] = { fis_gMFO0, fis_gMFO1};
293.
294.   // Rule Weights
295.   FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
```

```
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
296.
297.   // Rule Type
298.   int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 };
299.
300.   // Rule Inputs
301.   int fis_gRI0[] = { 1, 1, 1, 1 };
302.   int fis_gRI1[] = { 1, 1, 2, 1 };
303.   int fis_gRI2[] = { 1, 1, 3, 1 };
304.   int fis_gRI3[] = { 1, 2, 1, 1 };
305.   int fis_gRI4[] = { 1, 2, 2, 1 };
306.   int fis_gRI5[] = { 1, 2, 3, 1 };
307.   int fis_gRI6[] = { 1, 3, 1, 1 };
308.   int fis_gRI7[] = { 1, 3, 2, 1 };
309.   int fis_gRI8[] = { 1, 3, 3, 1 };
310.   int fis_gRI9[] = { 2, 1, 1, 1 };
311.   int fis_gRI10[] = { 2, 1, 2, 1 };
312.   int fis_gRI11[] = { 2, 1, 3, 1 };
313.   int fis_gRI12[] = { 2, 2, 1, 1 };
314.   int fis_gRI13[] = { 2, 2, 2, 1 };
315.   int fis_gRI14[] = { 2, 2, 3, 1 };
316.   int fis_gRI15[] = { 2, 3, 1, 1 };
317.   int fis_gRI16[] = { 2, 3, 2, 1 };
318.   int fis_gRI17[] = { 2, 3, 3, 1 };
319.   int fis_gRI18[] = { 3, 1, 1, 1 };
320.   int fis_gRI19[] = { 3, 1, 2, 1 };
321.   int fis_gRI20[] = { 3, 1, 3, 1 };
322.   int fis_gRI21[] = { 3, 2, 1, 1 };
323.   int fis_gRI22[] = { 3, 2, 2, 1 };
324.   int fis_gRI23[] = { 3, 2, 3, 1 };
325.   int fis_gRI24[] = { 3, 3, 1, 1 };
326.   int fis_gRI25[] = { 3, 3, 2, 1 };
327.   int fis_gRI26[] = { 3, 3, 3, 1 };
328.   int fis_gRI27[] = { 1, 1, 1, 2 };
329.   int fis_gRI28[] = { 1, 1, 2, 2 };
330.   int fis_gRI29[] = { 1, 1, 3, 2 };
331.   int fis_gRI30[] = { 1, 2, 1, 2 };
332.   int fis_gRI31[] = { 1, 2, 2, 2 };
333.   int fis_gRI32[] = { 1, 2, 3, 2 };
334.   int fis_gRI33[] = { 1, 3, 1, 2 };
```

```
335.   int fis_gRI34[] = { 1, 3, 2, 2 };
336.   int fis_gRI35[] = { 1, 3, 3, 2 };
337.   int fis_gRI36[] = { 2, 1, 1, 2 };
338.   int fis_gRI37[] = { 2, 1, 2, 2 };
339.   int fis_gRI38[] = { 2, 1, 3, 2 };
340.   int fis_gRI39[] = { 2, 2, 1, 2 };
341.   int fis_gRI40[] = { 2, 2, 2, 2 };
342.   int fis_gRI41[] = { 2, 2, 3, 2 };
343.   int fis_gRI42[] = { 2, 3, 1, 2 };
344.   int fis_gRI43[] = { 2, 3, 2, 2 };
345.   int fis_gRI44[] = { 2, 3, 3, 2 };
346.   int fis_gRI45[] = { 3, 1, 1, 2 };
347.   int fis_gRI46[] = { 3, 1, 2, 2 };
348.   int fis_gRI47[] = { 3, 1, 3, 2 };
349.   int fis_gRI48[] = { 3, 2, 1, 2 };
350.   int fis_gRI49[] = { 3, 2, 2, 2 };
351.   int fis_gRI50[] = { 3, 2, 3, 2 };
352.   int fis_gRI51[] = { 3, 3, 1, 2 };
353.   int fis_gRI52[] = { 3, 3, 2, 2 };
354.   int fis_gRI53[] = { 3, 3, 3, 2 };
355.   int* fis_gRI[] = { fis_gRI0, fis_gRI1, fis_g
RI2, fis_gRI3, fis_gRI4, fis_gRI5, fis_gRI6, fis_gR
I7, fis_gRI8, fis_gRI9, fis_gRI10, fis_gRI11, fis_g
RI12, fis_gRI13, fis_gRI14, fis_gRI15, fis_gRI16, f
is_gRI17, fis_gRI18, fis_gRI19, fis_gRI20, fis_gRI2
1, fis_gRI22, fis_gRI23, fis_gRI24, fis_gRI25, fis_
gRI26, fis_gRI27, fis_gRI28, fis_gRI29, fis_gRI30,
fis_gRI31, fis_gRI32, fis_gRI33, fis_gRI34, fis_gRI
35, fis_gRI36, fis_gRI37, fis_gRI38, fis_gRI39, fis
_gRI40, fis_gRI41, fis_gRI42, fis_gRI43, fis_gRI44,
 fis_gRI45, fis_gRI46, fis_gRI47, fis_gRI48, fis_gR
I49, fis_gRI50, fis_gRI51, fis_gRI52, fis_gRI53 };
356.
357.   // Rule Outputs
358.   int fis_gRO0[] = { 2, 0 };
359.   int fis_gRO1[] = { 1, 1 };
360.   int fis_gRO2[] = { 1, 1 };
361.   int fis_gRO3[] = { 2, 2 };
362.   int fis_gRO4[] = { 1, 1 };
363.   int fis_gRO5[] = { 1, 1 };
364.   int fis_gRO6[] = { 2, 2 };
365.   int fis_gRO7[] = { 2, 1 };
366.   int fis_gRO8[] = { 2, 1 };
367.   int fis_gRO9[] = { 3, 1 };
```

```
368.   int fis_gRO10[] = { 1, 1 };
369.   int fis_gRO11[] = { 1, 1 };
370.   int fis_gRO12[] = { 3, 1 };
371.   int fis_gRO13[] = { 2, 1 };
372.   int fis_gRO14[] = { 1, 1 };
373.   int fis_gRO15[] = { 2, 2 };
374.   int fis_gRO16[] = { 2, 2 };
375.   int fis_gRO17[] = { 2, 2 };
376.   int fis_gRO18[] = { 3, 1 };
377.   int fis_gRO19[] = { 3, 1 };
378.   int fis_gRO20[] = { 1, 1 };
379.   int fis_gRO21[] = { 3, 1 };
380.   int fis_gRO22[] = { 3, 1 };
381.   int fis_gRO23[] = { 1, 1 };
382.   int fis_gRO24[] = { 2, 2 };
383.   int fis_gRO25[] = { 2, 2 };
384.   int fis_gRO26[] = { 2, 2 };
385.   int fis_gRO27[] = { 2, 1 };
386.   int fis_gRO28[] = { 1, 1 };
387.   int fis_gRO29[] = { 1, 1 };
388.   int fis_gRO30[] = { 2, 1 };
389.   int fis_gRO31[] = { 2, 1 };
390.   int fis_gRO32[] = { 1, 1 };
391.   int fis_gRO33[] = { 2, 2 };
392.   int fis_gRO34[] = { 2, 2 };
393.   int fis_gRO35[] = { 2, 2 };
394.   int fis_gRO36[] = { 3, 1 };
395.   int fis_gRO37[] = { 1, 1 };
396.   int fis_gRO38[] = { 1, 1 };
397.   int fis_gRO39[] = { 2, 1 };
398.   int fis_gRO40[] = { 2, 1 };
399.   int fis_gRO41[] = { 1, 1 };
400.   int fis_gRO42[] = { 2, 2 };
401.   int fis_gRO43[] = { 2, 2 };
402.   int fis_gRO44[] = { 2, 2 };
403.   int fis_gRO45[] = { 3, 1 };
404.   int fis_gRO46[] = { 3, 1 };
405.   int fis_gRO47[] = { 1, 1 };
406.   int fis_gRO48[] = { 3, 1 };
407.   int fis_gRO49[] = { 3, 1 };
408.   int fis_gRO50[] = { 1, 1 };
409.   int fis_gRO51[] = { 2, 2 };
410.   int fis_gRO52[] = { 2, 2 };
411.   int fis_gRO53[] = { 2, 2 };
```

```
412.   int* fis_gRO[] = { fis_gRO0, fis_gRO1, fis_g
RO2, fis_gRO3, fis_gRO4, fis_gRO5, fis_gRO6, fis_gR
O7, fis_gRO8, fis_gRO9, fis_gRO10, fis_gRO11, fis_g
RO12, fis_gRO13, fis_gRO14, fis_gRO15, fis_gRO16, f
is_gRO17, fis_gRO18, fis_gRO19, fis_gRO20, fis_gRO2
1, fis_gRO22, fis_gRO23, fis_gRO24, fis_gRO25, fis_
gRO26, fis_gRO27, fis_gRO28, fis_gRO29, fis_gRO30,
fis_gRO31, fis_gRO32, fis_gRO33, fis_gRO34, fis_gRO
35, fis_gRO36, fis_gRO37, fis_gRO38, fis_gRO39, fis
_gRO40, fis_gRO41, fis_gRO42, fis_gRO43, fis_gRO44,
 fis_gRO45, fis_gRO46, fis_gRO47, fis_gRO48, fis_gR
O49, fis_gRO50, fis_gRO51, fis_gRO52, fis_gRO53 };
413.
414.   // Input range Min
415.   FIS_TYPE fis_gIMin[] = { 0, 0, 0, 80 };
416.
417.   // Input range Max
418.   FIS_TYPE fis_gIMax[] = { 200, 200, 200, 255
};
419.
420.   // Output range Min
421.   FIS_TYPE fis_gOMin[] = { -45, 50 };
422.
423.   // Output range Max
424.   FIS_TYPE fis_gOMax[] = { 45, 150 };
425.
426.   //****************************************
*****************************
427.   // Data dependent support functions for Fuzz
y Inference System
428.   //****************************************
*****************************
429.   FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet,
 FIS_TYPE x, int o)
430.   {
431.       FIS_TYPE mfOut;
432.       int r;
433.
434.       for (r = 0; r < fis_gcR; ++r)
435.       {
436.           int index = fis_gRO[r][o];
437.           if (index > 0)
438.           {
439.               index = index - 1;
```

```
440.                mfOut = (fis_gMF[fis_gMFO[o][ind
ex]])(x, fis_gMFOCoeff[o][index]);
441.            }
442.            else if (index < 0)
443.            {
444.                index = -index - 1;
445.                mfOut = 1 - (fis_gMF[fis_gMFO[o]
[index]])(x, fis_gMFOCoeff[o][index]);
446.            }
447.            else
448.            {
449.                mfOut = 0;
450.            }
451.
452.            fuzzyRuleSet[0][r] = fis_min(mfOut,
fuzzyRuleSet[1][r]);
453.        }
454.        return fis_array_operation(fuzzyRuleSet[
0], fis_gcR, fis_max);
455.    }
456.
457.    FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzz
yRuleSet, int o)
458.    {
459.        FIS_TYPE step = (fis_gOMax[o] - fis_gOMi
n[o]) / (FIS_RESOLUSION - 1);
460.        FIS_TYPE area = 0;
461.        FIS_TYPE momentum = 0;
462.        FIS_TYPE dist, slice;
463.        int i;
464.
465.        // calculate the area under the curve fo
rmed by the MF outputs
466.        for (i = 0; i < FIS_RESOLUSION; ++i){
467.            dist = fis_gOMin[o] + (step * i);
468.            slice = step * fis_MF_out(fuzzyRuleS
et, dist, o);
469.            area += slice;
470.            momentum += slice*dist;
471.        }
472.
473.        return ((area == 0) ? ((fis_gOMax[o] + f
is_gOMin[o]) / 2) : (momentum / area));
474.    }
```

```
475.
476.   //****************************************
*****************************
477.   // Fuzzy Inference System

478.   //****************************************
*****************************
479.   void fis_evaluate()
480.   {
481.       FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
482.       FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
483.       FIS_TYPE fuzzyInput2[] = { 0, 0, 0 };
484.       FIS_TYPE fuzzyInput3[] = { 0, 0 };
485.       FIS_TYPE* fuzzyInput[fis_gcI] = { fuzzyI
nput0, fuzzyInput1, fuzzyInput2, fuzzyInput3, };
486.       FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
487.       FIS_TYPE fuzzyOutput1[] = { 0, 0 };
488.       FIS_TYPE* fuzzyOutput[fis_gcO] = { fuzzy
Output0, fuzzyOutput1, };
489.       FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
490.       FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
491.       FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules,
 fuzzyFires };
492.       FIS_TYPE sW = 0;
493.
494.       // Transforming input to fuzzy Input
495.       int i, j, r, o;
496.       for (i = 0; i < fis_gcI; ++i)
497.       {
498.           for (j = 0; j < fis_gIMFCount[i]; ++
j)
499.           {
500.               fuzzyInput[i][j] =
501.                   (fis_gMF[fis_gMFI[i][j]])(g_
fisInput[i], fis_gMFICoeff[i][j]);
502.           }
503.       }
504.
505.       int index = 0;
506.       for (r = 0; r < fis_gcR; ++r)
507.       {
508.           if (fis_gRType[r] == 1)
509.           {
510.               fuzzyFires[r] = FIS_MAX;
```

```
511.                for (i = 0; i < fis_gcI; ++i)
512.                {
513.                    index = fis_gRI[r][i];
514.                    if (index > 0)
515.                        fuzzyFires[r] = fis_min(
fuzzyFires[r], fuzzyInput[i][index - 1]);
516.                    else if (index < 0)
517.                        fuzzyFires[r] = fis_min(
fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
518.                    else
519.                        fuzzyFires[r] = fis_min(
fuzzyFires[r], 1);
520.                }
521.            }
522.            else
523.            {
524.                fuzzyFires[r] = FIS_MIN;
525.                for (i = 0; i < fis_gcI; ++i)
526.                {
527.                    index = fis_gRI[r][i];
528.                    if (index > 0)
529.                        fuzzyFires[r] = fis_max(
fuzzyFires[r], fuzzyInput[i][index - 1]);
530.                    else if (index < 0)
531.                        fuzzyFires[r] = fis_max(
fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
532.                    else
533.                        fuzzyFires[r] = fis_max(
fuzzyFires[r], 0);
534.                }
535.            }
536.
537.            fuzzyFires[r] = fis_gRWeight[r] * fu
zzyFires[r];
538.            sW += fuzzyFires[r];
539.        }
540.
541.        if (sW == 0)
542.        {
543.            for (o = 0; o < fis_gcO; ++o)
544.            {
545.                g_fisOutput[o] = ((fis_gOMax[o]
+ fis_gOMin[o]) / 2);
546.            }
```

```
547.        }
548.        else
549.        {
550.            for (o = 0; o < fis_gcO; ++o)
551.            {
552.                g_fisOutput[o] = fis_defuzz_cent
roid(fuzzyRuleSet, o);
553.            }
554.        }
555.
556.
557.    }
```

BIOGRAFI PENULIS

Penulis dengan nama lengkap Amalia Puruhita dilahirkan di Kota Bontang pada tanggal 9 Mei 1996 dari ayah yang bernama Agus Susanto dan Ibu bernama Nunung Dwi Hariyati. Penulis telah menyelesaikan pendidikan formal di SD VIDATRA (Vidya Dahana Patra) Bontang (2002-2008), SMP VIDATRA (Vidya Dahana Patra) Bontang (2008-20011), SMA VIDATRA Bontang (20011-2014). Setelah menyelesaikan pendidikan SMA, penulis diterima di Jurusan Teknik Fisika FTI ITS Surabaya melalui jalur Mandiri Kemitraan tertulis dengan Program Studi S1 Reguler Teknik Fisika dan terdaftar dengan NRP 02311440000119. Selama menempuh pendidikan di Teknik Fisika ITS penulis pernah melakukan Kerja Praktek di PT Badak LNG Bontang, Kalimantan Timur. Penulis mengambil bidang minat Rekayasa Instrumentasi dan Kontrol dalam menyelesaikan Tugas Akhir jenjang S1 dibawah bimbingan bapak Dr. Purwadi Agus Darwito, S.T., M.T, dan bapak Andi Rahmadiansah, S.T., M.T. dengan penelitian yang berjudul "**RANCANG BANGUN PENGENDALI KEMUDI PROTOTIPE KENDARAAN *AUTONOMOUS* DENGAN METODE FUZZY LOGIC**" Bagi pembaca yang memiliki kritik, saran, atau ingin berdiskusi lebih lanjut mengenai Tugas Akhir ini maka dapat menghubungi penulis melalui email amaliapuruhita@gmail.com.