TUGAS AKHIR – TE 141599

PERANCANGAN KONTROLER *NEURO-FUZZY* UNTUK *ELECTROMAGNETIC ANTI-LOCK BRAKING SYSTEM* (ABS) PADA KENDARAAN LISTRIK

Rian Lukito
NRP 2211 100 180

Dosen Pembimbing
Ir. Josaphat Pramudijanto, M.Eng.
Andri Ashfahani, S.T., M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

DESIGN NEURO-FUZZY CONTROLLER FOR ELECTROMAGNETIC ANTI-LOCK BRAKING SYSTEM (ABS) ON ELECTRIC VEHICLE

Rian Lukito
NRP 2211 100 180

Supervisors
Ir. Josaphat Pramudijanto, M.Eng.
Andri Ashfahani, S.T., M.T.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

PERANCANGAN KONTROLER *NEURO-FUZZY* UNTUK *ELECTROMAGNETIC ANTI-LOCK BRAKING SYSTEM* (ABS) PADA KENDARAAN LISTRIK

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Josaphat Pramudijanto, M.Eng.	**Andri Ashfahani, S.T., M.T.**
NIP:196210051990031003	NIP: 2200201405003

SURABAYA
JUNI, 2015

PERANCANGAN KONTROLER *NEURO-FUZZY* UNTUK *ELECTROMAGNETIC ANTI-LOCK BRAKING SYSTEM* (ABS) PADA KENDARAAN LISTRIK

Nama : Rian Lukito
Pembimbing I : Ir. Josaphat Pramudijanto, M.Eng.
Pembimbing II : Andri Ashfahani, S.T., M.T.

ABSTRAK

Anti-lock braking system (ABS) digunakan pada kendaraan untuk menjaga agar roda tidak terkunci saat pengereman mendadak (pengereman dalam) dan meminimalkan jarak berhenti kendaraan. Permasalahan yang terjadi saat pengereman mendadak adalah roda terkunci sehingga *steering* kendaraan tidak dapat dikendalikan. Sistem ABS yang dirancang akan diterapkan pada simulator ABS dengan menggunakan pengereman elektromagnetik. Dalam kondisi normal atau dalam keadaan tanpa pengereman, kecepatan *longitudinal* kendaraan akan sama dengan kecepatan putaran roda sehingga slip ratio bernilai 0 (0%)dan apabila kecepatan putaran roda bernilai 0 (dalam keadaan terkunci) maka roda bernilai slip 1 (100%). Sistem ABS akan menjaga agar nilai rasio slip berada di nilai 0,2 (20%). Pada tugas akhir ini digunakan metode *neuro-fuzzy* untuk mengontrol nilai slip pada roda. Masukkannya adalah slip yang diharapkan dan keluarannya adalah slip dari *plant.* Algoritma *learning* yg digunakan adalah *BackPropagation* yang akan bekerja secara *feedforward* untuk mendapatkan *ouput* aktual dan bekerja secara *feedback* setelah mendapatkan nilai *error* dengan *output* target. Jaringan yang dibuat berdasarkan mekanisme *fuzzy* yakni fuzzifikasi, inferensi dan defuzzifikasi. Kontroler *neuro-fuzzy* dapat mengurangi *overshoot* respon *plant* hingga 43,2% dibandingkan respon *plant* tanpa kontroler secara *loop* terbuka.

Kata kunci: ABS, *BackPropagation*, *Neuro-Fuzzy,* Simulator ABS

--Halaman ini sengaja dikosongkan--

DESIGN NEURO-FUZZY CONTROLLER FOR ELECTROMAGNETIC ANTI-LOCK BRAKING SYSTEM (ABS) ON ELECTRIC VEHICLE

Name **: Rian Lukito**
Supervisor I **: Ir. Josaphat Pramudijanto, M.Eng.**
Supervisor II **: Andri Ashfahani, S.T., M.T.**

ABSTRACT

Anti-lock braking system (ABS) is used on the vehicle to keep the wheels are not locked during sudden braking (braking in) and minimize the distance termination. Problems that occur when sudden braking is locked so that the will be oversteering. ABS system will be applied using electromagnetic braking on the wheels of ABS simulator. Under normal conditions, the vehicle longitudinal speed will be the same as the rotation speed of the wheel so that the slip ratio is 0 (0%) and when the wheel rotation speed is 0 (locked), the wheel slip ratio is 1 (100%). ABS system will keep the value of the slip ratio is at a value of 0.2 (20%). Neuro-Fuzzy method used to control the value of the wheel slip ratio in my final project. Slip ratio input is expected and output is the slip ratio of the plant. Learning algorithms use Back Propagation which will work to get the actual output with feedforward and feedback work after getting the error value with a target output. Network created by the fuzzy mechanism fuzzification, inference and defuzzification. Neuro-Fuzzy controller can reduced overshoot of plant response until 43.2% than plant response without controller in open loop system.

Key words: ABS, ABS *Simulator, BackPropagation, Neuro-Fuzzy*

--Halaman ini sengaja dikosongkan--

KATA PENGANTAR

Segala puji dan syukur penulis ucapkan kepada Tuhan YME, yang telah memberikan nikmat dan hidayah-Nya, sehingga laporan Tugas Akhir yang berjudul: "**Perancangan Kontroler *Neuro-Fuzzy* untuk *Electromagnetic Anti-lock Braking System* (ABS) pada Kendaraan Listrik"**ini dapat terselesaikan dengan baik. Shalawat dan salam semoga tetap terlimpahkan kepada Rasulullah SAW yang telah memberikan suri tauladan kepada kita semua.

Penulis menyadari bahwa dalam pengerjaan Tugas Akhir ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak, baik bantuan berupa tenaga, pikiran, maupun moril untuk penulis. Untuk itu, penulis mengucapkan terima kasih kepada semua pihak yang telah turut serta berperan dalam penyelesaian Tugas Akhir ini, khususnya kepada dosen pembimbing, yakni Ir. Josaphat Pramudijanto, M.Eng., dan Andri Ashfahani, ST., MT. yang selalu membimbing dan memberikan motivasi kepada penulis, kedua orang tua yang selalu mendoakan kesuksesan penulis, Saudara Sondang Sentosa O, Saudara Hendra Antomy, Saudara Tito Luthfan Ramadhan, Tim ABS, dan seluruh teman-teman yang memberikan dukungan penuh kepada penulis serta kepada semua pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa Tugas Akhir ini masih banyak kekurangan. Oleh karena itu, penulis akan menerima kritik dan saran dari para pembaca. Semoga Tugas Akhir ini dapat bermanfaat khususnya untuk penulis, dan umumnya untuk semua pembaca laporan Tugas Akhir ini.

Surabaya, Juni 2015

Penulis

--Halaman ini sengaja dikosongkan--

DAFTAR ISI

DAFTAR TABEL

--Halaman ini sengaja dikosongkan--

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Pendahuluan mencakup latar belakang, perumusan masalah, batasan masalah, tujuan, sistematika penulisan, dan relevansi.

1.1 Latar Belakang

Pengereman merupakan salah satu bagian penting dari sistem kendaraan. Pengereman yang benar adalah dengan tidak melakukan pengereman dalam (mendadak) karena dapat membahayakan kendaraan dan kendaraan lainnya di belakang. Akan tetapi apabila terjadi kondisi darurat yang dapat menghalangi laju kendaaran dan membahayakan pengemudi maka pengemudi diharuskan melakukan pengereman dalam (mendadak). Saat pengereman mendadak roda dapat mengunci sehingga steering kendaraan tidak dapat dikendalikan dan menyebabkan kendaraan tergelincir. Salah satu sistem yang dapat menjaga atau mengendalikan pengereman tersebut adalah *Anti Lock Braking System* (ABS). Sistem ABS akan mengendalikan slip dari roda kendaraan sehingga roda tidak sampai terkunci saat dilakukan pengereman. Slip pada roda bergantung dari perbandingan antara selisih kecepatan *longitudinal* kendaraan (*longitudinal velocity*) dan kecepatan putaran roda (*wheel speed*) dengan kecepatan *longitudinal* kendaraan *(longitudinal velocity*). Pengereman Elektromagnetik adalah sistem pengereman yang menggunakan gaya elektromagnetik untuk memperlambat gerakan poros pada kendaraan. Pengereman Elektromagnetik yang digunakan adalah tipe rem *microeletromagnetic clutch brake* memanfaatkan medan elektromagnetik yang dihasilkan oleh kumparan karena adanya arus yang melewati kumparan tersebut. Besarnya medan magnet yang dihasilkan berbanding lurus dengan besarnya arus yang melewati kumparan. Gaya medan magnet yang dihasilkan berlawanan dengan arah putaran poros roda sehingga dapat memperlambat laju kendaraan.

Kontroler tradisional seperti PID tidak dapat memenuhi kebutuhan sistem ABS karena tidak dapat menentukan parameter kontroler secara dinamik dan tidak dapat mengurangi efek gangguan pada sistem. Kontroler *neuro-fuzzy* adalah kontrol cerdas yang merupakan kombinasi dari algoritma *neural network* dan aturan *fuzzy*. Algoritma *neural network* adalah algoritma s*upervised learning* yang menggunakan *neural network* sebagai jaringan pembelajaran. Jaringan akan dilatih secara terus menerus

dengan memperbaharui nilai pembobotan pada masing- masing neuron pada *hidden layer* secara *feedforward* dan *feedback*. Jadi saat diberikan *input*, jaringan akan bekerja secara *feedforward* untuk menghasilkan *output* aktual, kemudian selisih antara *ouput* aktual dengan *output* target akan diumpan balikkan menuju *input* melalui *hidden layer* dengan tujuan memperbaharui nilai bobot pada setiap lapisannya. Jaringan dibuat menyerupai dengan mekanisme *fuzzy* yakni fuzzifikasi, mekanisme inferensi *fuzzy*, dan defuzzifikasi. Dengan kontroler *neuro-fuzzy* dapat mengatasi efek gangguan pada *plant* non linear dan dapat membuat respon *plant* memiliki *error steady state* seminimal mungkin.

1.2 Perumusan Masalah

Adapun rumusan masalah yang akan dibahas dalam tugas akhir adalah mengendalikan nilai slip sebesar 20% pada roda untuk kondisi jalan kering, lurus, dan tidak bergelombang saat proses pengereman menggunakan kontroler *neuro-fuzzy*.

1.3 Batasan Masalah

Terdapat berberapa batasan masalah dalam pelaksanaan tugas akhir ini yakni :

1. Nilai slip pada roda sebesar 20% mengacu pada nilai slip saat koefisien gesek maksimal pada berberapa kondisi jalan seperti jalan kering, dan jalan basah.
2. Pada Tugas Akhir kali ini, tujuan utama adalah mengendalikan slip sebesar 20% dan tidak membahas mengenai penurunan kecepatan (deselarasi) kendaraan yang terjadi saat nilai slip dikendalikan sebesar 20%.

1.4 Tujuan

Tujuan dari pelaksanaan tugas akhir antara lain merancang dan mengimplementasikan *plant* simulator ABS menggunakan kontroler neuro-fuzzy. Tujuan utama pada sistem *Anti-locking Braking* adalah menjaga nilai slip antara kecepatan *longitudinal* (roda bawah) dan kecepatan putar (roda atas) pada saat nilai koefisien gesek maksimal pada seluruh kondisi jalan sebesar 20%.

1.5 Sistematika Penulisan

Sistematika penulisan buku laporan Tugas Akhir ini akan dibagi menjadi lima bab, yakni :

Bab I: Pendahuluan

Bab Pendahuluan meliputi latar belakang pelaksanaan topik Tugas Akhir, permasalahan topik Tugas Akhir, tujuan penelitian, sistematika penulisan laporan, dan relevansi topik yang dikerjakan.

Bab 2: Teori Penunjang

Bab teori penunjang membahas tinjauan pustaka yang membantu pengerjaan Tugas Akhir, diantaranya adalah konsep tentang dinamika kendaraan, konsep sistem ABS, Cara kerja simulator ABS, teori rem elektromagnetik, dan teori tentang algoritma *neuro-fuzzy*.

Bab 3: Perancangan Sistem

Bab perancangan sistem membahas tentang perancangan sistem simulator ABS menggunakan rem elektromagnetik dan perancangan algoritma kontroler *neuro-fuzzy* berdasarkan teori pada Bab 2.

Bab 4: Pengujian dan Analisis

Bab pengujian dan analisis mencakup hasil simulasi dan analisa kontroler pada sistem ABS dan implementasinya pada real *plant* yang digunakan, yakni simulator ABS.

Bab 5: Penutup

Bab penutup berisi tentang kesimpulan dan saran dari hasil pembahasan Tugas Akhir yang diperoleh.

1.6 Relevansi

Hasil yang diperoleh dari pelaksanaan Tugas Akhir ini diharapkan mampu menjadi bahan referensi dalam perancangan kontroler *neuro-fuzzy* untuk sistem *Anti-lock Braking System* (ABS).

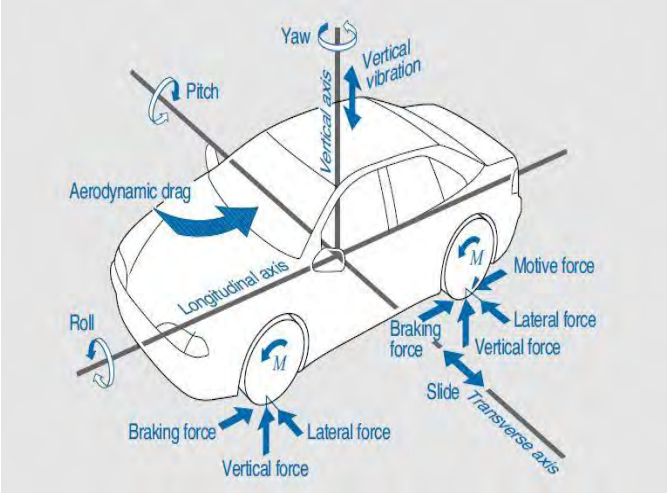
--Halaman ini sengaja dikosongkan--

BAB 2
TEORI PENUNJANG

Teori penunjang mencakup pemahaman mengenai dinamika gerak sebuah kendaraan, konsep kerja simulator ABS, Komponen yang digunakan, penjabaran singkat mengenai identifikasi *plant*, skema kontroler *fuzzy* dan algoritma *neuro-fuzzy* FMN Tipe II.

2.1 Dinamika Kendaraan [1]

Dinamika gerak sebuah kendaraan mempunyai 3 sumbu gerak yakni *axis* l*ongitudinal*, *axis vertical*, dan *axis transverse*. Masing-masing *axis* mempunyai sumbu putar yakni *roll*, *pitch* dan *yaw*. Gambar 2.1 berikut akan merepresentasikan dinamika gerak dari sebuah kendaraan:



Gambar 2. 1 Dinamika Gerak Kendaraan [1]

Salah satu gaya yang bekerja pada kendaraan dari keadaan awal bergerak hingga berhenti adalah gaya pengereman yang berlawanan arah dengan arah laju kendaraan. Ketika gaya pengereman bekerja maka dinamika kendaraan akan berubah pada 3 sumbu *axis*. Pada bahasan kali ini, dinamika kendaraan yang akan dibahas adalah pada sumbu *axis longitudinal*. Pada *axis longitudinal*, efek pengereman akan mengakibatkan perubahan pada dua parameter, yakni kecepatan

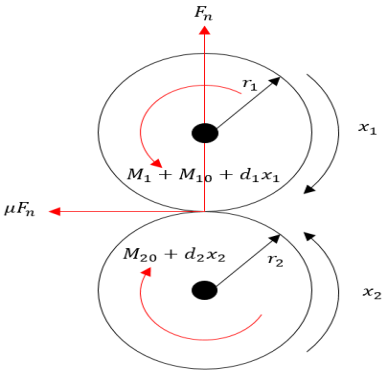
longitudinal kendaraan dan kecepatan putar roda. Kecepatan putar dan kecepatan *longitudinal* akan mengalami slip atau *brake* slip. Nilai slip ini perlu dikendalikan pada nilai 0%-20% agar kendaraan dapat dikemudikan pada saat pengereman dalam (mendadak). Slip terjadi pada masing-masing roda pada sebuah kendaraan. Dinamika gaya yang terjadi pada saat pengereman pada sebuah roda adalah gaya-gaya yang bekerja pada sebuah roda. Saat pengereman gaya yang bekerja adalah *braking force*, gaya normal kendaraan dan gaya gesek jalan.

2.2 Simulator ABS [2] [3]

Salah satu cara untuk menguji ABS ialah dengan menggunakan simulator ABS. Simulator ABS merupakan suatu simulator yang memiliki cara kerja menyerupai dinamika kendaraan saat dilakukan pengereman. Komponen utama dari simulator ABS adalah dua buah roda yang permukaannya saling bersinggungan, dimana roda sebelah atas merupakan roda bebas dan roda bawah diputar dengan motor DC. Gambar 2.2 berikut merupakan contoh simulator ABS produksi INTECO:

Gambar 2. 2 Simulator ABS Produksi INTECO [2] [3]

Simulator ABS hanya memperhitungkan dinamika *longitudinal* kendaraan dan tidak mempertimbangan dinamika *vertikal* dan *lateral* (samping) kendaraan. Kecepatan putar roda bebas mewakili kecepatan putar roda mobil (ω) dan kecepatan putar roda bawah yang diputar dengan motor DC mewakili kecepatan *longitudinal* mobil (v). Gambar 2.3 berikut merupakan gaya-gaya yang bekerja pada Simulator ABS:



Gambar 2. 3 Skema Momen Gaya pada Simulator ABS [2]

Torsi rem ditambahkan pada roda bagian atas yang mewakili roda kendaraan, sehingga menyebabkan peralambatan pada roda atas. Persamaan matematis momen gaya yang bekerja pada roda bagian atas simulator ABS adalah sebagai berikut:

$$J_1\dot{x}_1 = F_n r_1 s\mu(\lambda) - d_1x_1 - s_1M_{10} - s_1M_1 \quad (2.1)$$

dimana J_1 adalah momen inersia pada roda atas. Nilai momen inersia dipengaruhi dari bentuk dan besarnya massa dari roda atas. $x_1$adalah kecepatan angular roda atas, r_1 adalah jari-jari roda atas,d_1 adalah koefisien gaya gesek dinamis permukaan roda atas, M_{10} adalah gaya gesek statis permukaan roda atas, F_n merupakan gaya normal, $\mu(\lambda)$ adalah koefisien gesek permukaan jalan.

Sedangkan untuk roda bawah simulator ABS adalah sebagai berikut:

$$J_2\dot{x}_2 = -F_n r_2 s\mu(\lambda) - d_2x_2 - s_2M_{20} \quad (2.2)$$

dimana J_2 adalah momen inersia pada roda bawah. Nilai momen inersia dipengaruhi dari bentuk dan besarnya massa dari roda bawah. $x_2$adalah kecepatan angular roda bawah,r_2 adalah jari-jari roda bawah, d_2 adalah koefisien gaya gesek dinamis permukaan roda bawah, M_{20} adalah gaya gesek statis permukaan roda bawah.

Momen Inersia dari roda atas dan roda bawah berbeda, hal ini didapat dari persamaan roda bagian atas dan roda bagian bawah yang mempunyai nilai torsi yang sama. $\dot{X}_1$ adalah perubahan kecepatan roda atas tiap waktu dan $\dot{X}_2$ adalah perubahan kecepatan roda bawah tiap waktu. Agar kondisi slip terjadi pada simulator ABS, maka $\dot{X}_2$ lebih kecil

dari $\dot{X}_1$sehingga momen inersia pada roda bawah lebih besar dibandingkan momen inersia pada roda atas.

Selain variabel diatas, terdapat pula variabel bantu s yang berguna untuk membedakan arah rotasi roda dan bernilai konstan. Berikut merupakan persamaan matematis variabel s:

$$s = sgn(r_2 x_2 - r_1 x_1) \quad (2.3)$$

$$s_1 = sgn(x_1) \quad (2.4)$$

$$s_2 = sgn(x_2) \quad (2.5)$$

2.3 Motor DC [4]

Motor DC terdiri dari kumparan *stator* dan kumparan *rotor*. Kumparan *stator* terdiri dari kumparan medan dan kumparan *rotor* terdiri dari kumparan jangkar. Kumparan medan akan menghasilkan medan magnet (Φ) dan kumparan jangkar akan menginduksikan tegangan internal(E_a) pada *brush*. *Brush* akan menginduksikan tegangan pada komutator yang terpasang di dalam kumparan medan. Adanya medan magnet di sekitar komutator dan tegangan pada komutator maka akan mucul gaya elektromagnetik yang membuat komutator berotasi. Rotasi inilah yang akan membuat motor DC berputar pada kecepatan tertentu

Untuk mengendalikan kecepatan motor DC dapat dilakukan dengan mengubah hambatan jangkar (Ra) dan mengubah tegangan pada tegangan terminal (Vt). Cara yang kedua banyak digunakan karena tidak perlu merubah konstruksi motor DC, yakni dengan mengubah tegangan terminal motor DC. Metode untuk megubah tegangan terminal motor DC bermacam-macam tergantung pada desain kontroler yang diinginkan.

2.4 Arduino UNO [5]

Aduino adalah pengendali mikro *project-board* yang bersifat *open-source*, dan dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. Arduino memiliki prosesor Atmel AVR (AT Mega 328) dan *compiler*nya merupakan pengembangan dari bahasa pemrograman java. Bahasa pemrograman yang digunakan dalam *compiler* adalah C++ yang sudah dilengkapi oleh berberapa *library* dasar. Arduino UNO mempunyai 14 *pin digital input/output* (6 diantaranya dapat digunakan sebagai keluaran PWM), 6 *input analog,* sebuah *osilator* kristal 16 MHz, sebuah koneksi USB, sebuah power jack, sebuah ICSP *header*, dan sebuah tombol *reset. Pin* yang dapat digunakan sebagai keluaran *Pulse width Modulation* (PWM) yakni *pin* 3,5,6,9,10, dan 11. *Pin* 3,9,10, dan 11 mempunyai frekuensi *default* 490Hz dan *pin* 5,6 mempunyai frekuensi

default 980 Hz. *Input* analog pada *pin* A4 dan A5 dapat digunakan sebagai komunikasi *inter IC (*I2C*)*. Komunikasi I2C menggunakan *pin* A4 sebagai *Serial Data* (SDA) dan *Serial Clock* (SCL) untuk berkomunikasi dengan *device* lain dengan topologi jaringan *bus*. Gambar 2.4 berikut merupakan tampilan dari *project board* arduino UNO:

Gambar 2. 4 Arduino UNO Tampak Depan dan Belakang [5]

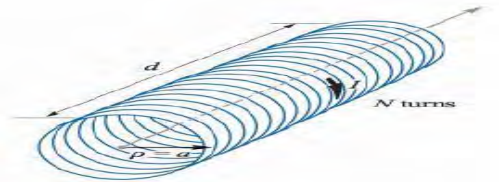
Tabel 2.1 berikut merupakan spesifikasi yang dimiliki oleh Arduino UNO:

Tabel 2. 1 Spesifikasi Arduino UNO [5]

Mikrokontroler	ATmega328
Tegangan pengoprasian	5 V
Tegangan input yang disarankan	7-12 V
Batas tegangan input	12 V
Jumlah *pin* I/O *digital*	14 (6 di antaranya menyediakan keluaran PWM)
Jumlah *pin input analog*	6
Arus DC tiap *pin* I/O	40 mA
Arus DC untuk *pin* 3,3 V	50 mA
Memori *Flash*	32 kb (ATmega328), sekitar 0,5 kb digunakan oleh *bootloader*
SRAM	2 kb (ATmega328)
EEPROM	1 kb (ATmega328)
Kecepatan *Clock*	16 MHz

2.5 Rem Elektromagnetik [6]

Rem elektromagnetik adalah komponen elektro-mekanik yang berfungsi untuk mengehentikan laju kendaraan menggunakan medan elektromagnetik. Rem elektromagnetik memanfaatkan medan listrik yang dihasilkan dari kumparan yang dialiri arus listrik, dengan arah gaya yang berlawanan dengan arah putaran ban, sehingga kendaraan akan terhenti. Gambar 2.5 berikut merupakan ilustrasi medan listrik yang dibangkitkan oleh kumparan yang dialiri arus listrik:



Gambar 2. 5 Cara Kerja Rem Elektromagnetik [6]

Rem elektromagnetik terdiri dari kumparan dan sebuah piringan besi didepan kumparan yang berputar sesuai putraran roda. Sesuai dengan Hukum Ampere, hubungan antara arus dan kuat medan magnet pada rem elektromagetik dapat dirumuskan dalam persamaan:

$$B = \mu_0 H = \frac{\mu_0 NI}{d} a_z \quad (2.6)$$

dimana μ_0 adalah konstanta permeabilitias ruang hampa, N merupakan jumlah lilitan, d adalah panjang lililtan, dan I adalah arus yang mengalir pada lilitan.

2.6 Modul Sensor Kecepatan dan Piringan *Encoder*

Sensor kecepatan digunakan untuk mengukur kecepatan putar roda dan estimasi kecepatan *longitudinal* simulator ABS. Sensor akan mendeteksi tiap lubang pada piringan *encoder* dan mengubahnya dalam bentuk pulsa digital. Pulsa digital yang dikeluarkan oleh sensor kecepatan hasil pembacaan piringan *encoder* berupa sinyal *pulse width modulation* (PWM). Gambar 2.6 berikut merupakan modul sensor kecepatan dengan komparator LM 393:

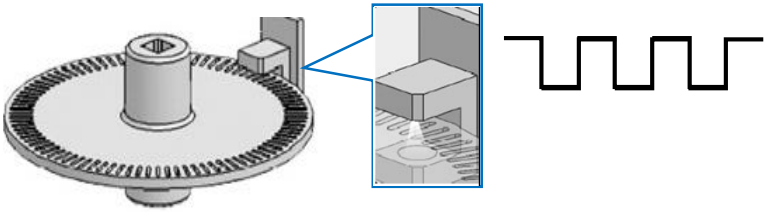
Gambar 2. 6 Modul Sensor Kecepatan dengan Komparator LM393 [7]

Tabel 2.2 berikut merupakan spesifikasi yang dimiliki oleh Modul sensor kecepatan:

Tabel 2.2 Spesifikasi pada Modul Sensor Kecepatan: [7]

Komparator yang digunakan	LM393
Tegangan pengoprasian	3,3 - 12 V
Format keluaran	*Digital switching*
Lebar celah *optocoupler*	5 mm
Sinyal keluaran komparator	Lebih dari 15 mA
Dimensi modul	3,2 cm x 1,4 cm

Piringan *encoder* memiliki sejumlah lubang yang mengitari sisinya dengan jarak tertentu. Sensor mendeteksi lubang pada *encoder* dengan menggunakan *optocoupler* yang mentransmisikan cahaya diantara celah piringan, bila cahaya melewati lubang piringan *encoder* dan diterima *receiver*, maka sensor akan mengeluarkan pulsa *high*, sedangkan saat cahaya tidak melewati lubang piringan *encoder*, maka sensor akan mengelurakan pulsa *low*. Gambar 2.7 berikut merupakan ilustrasi dari prinsip kerja sebuah sensor kecepatan dan piringan *encoder*:



Gambar 2. 7 Prinsip Kerja Sensor Kecepatan dan Piringan *Encoder* [8]

2.7 Identifikasi Sistem [9]

Struktur model pendekatan parameter *plant* dapat dinyatakan dalam berberapa model salah satunya adalah model pendekatan stokastik *Auto regressive external* (ARX). Identifikasi parameter dapat dilakukan dengan berbagai metode yakni metode penyelesaian persamaan linear simultan, metode *gradient*, dan metode *least square*. Dengan mengetahui data *input plant* (u) dan data *output plant* (y) dapat diketahui model *plant* yang ingin diketahui. Identifikasi dilakukan secara *offline*, yakni dengan mendapatkan parameter *plant* terlebih dahulu tanpa langsung dihubungkan dengan kontroler. Pemodelan ARX adalah sebagai berikut :

$$\text{y(k)}=-a_1\ y(k-1)-a_2\ y(k-2)-a_3\text{y(k-3)-...}\ a_{na}\ y(k-na)+b_0\ u(k-1)+\cdots b_{nb}u(k-nb-d) \quad (2.7)$$

2.8 Kontroler *Fuzzy* [10]

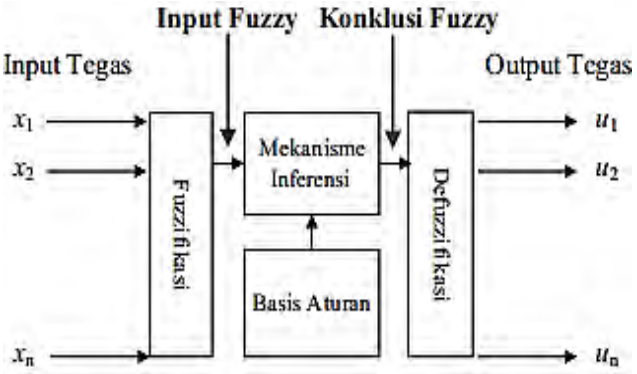
Kontroler *fuzzy* merupakan salah satu kontrol cerdas, yang memiliki algoritma seperti cara berfikir manusia. Logika *fuzzy* pertama kali dikembangkan oleh Lofti A. Zadeh dalam papernya yang berjudul "*Fuzzy Set"* pada tahun 1965.

Dalam algoritma kontrol *fuzzy*, dibuat aturan-aturan yang sesuai dengan kebiasaaan atau pengalaman manusia saat mengatur proses suatu objek kontrol atau *plant* secara manual. Aturan-aturan dasar yang sesuai dengan kebiasaan manusia tersebut memiliki kualifikasi bahasa yang samar dan tidak kaku *(linguistic variable)* seperti yang digunakan oleh manusia. Aturan dasar *fuzzy* dapat ditulis dengan menggunakan format sebagai berikut:

If premis ***then*** konsekuen (2.8)

dimana premis merupakan variabel masukan *fuzzy* dan konsekuen merupakan variabel keluaran *fuzzy*. Variabel *fuzzy* memiliki derajat keanggotaan yang bernilai antara satu dan nol dan terhimpun dalam suatu himpunan *fuzzy*.

Penggunaan logika *fuzzy* pada kontrol cerdas memilki lima bagian utama, seperti pada Gambar 2.8, yakni fuzzifikasi (*fuzzyfication*), aturan dasar *fuzzy(fuzzy basic rules*), mekanisme inferensi atau operasi himpunan *fuzzy*, dan defuzzifikasi (*defuzzification*). Gambar 2.8 berikut merupakan skema kontroler *fuzzy* atau mekanisme kontroler *fuzzy*:



Gambar 2. 8 Skema Kontroler *Fuzzy*

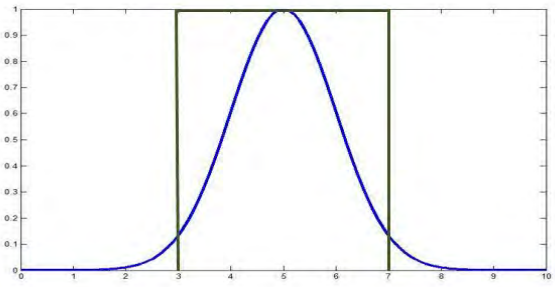
Selain lima bagian utama kontroler *fuzzy* pada Gambar 2.8, terdapat beberapa istilah penting yang perlu diketahui dalam logika *fuzzy*, antara lain himpunan *fuzzy* dan fungsi keanggotaan *fuzzy*.

2.8.1 Himpunan *Fuzzy* [10]

Himpunan *fuzzy* merupakan sebuah himpunan dengan elemen-elemen yang memiliki derajat parsial, tanpa batasan yang kaku. Pada himpunan *fuzzy*, elemen-elemen pada semesta pembicaraan dapat menjadi anggota himpunan sekaligus tidak menjadi anggota himpunan.

Keanggotaan elemen-elemen pada semesta pembicaraan dipengaruhi oleh derajat keanggotaannya. Hal ini berbeda dengan himpunan tegas (klasik) yang elemen-elemen pada semesta pembicaraan hanya dapat digolongkan termasuk atau tidak termasuk dalam suatu himpunan.

Gambar 2.9 menunjukkan perbedaan antara himpunan tegas dengan himpunan *fuzzy*. Pada himpunan tegas, (garis hijau) tidak terdapat derajat keanggotaan dan elemen pada semesta pembicaraan hanya digolongkan dalam himpunan ataupun tidak. Sedangkan himpunan *fuzzy* (garis biru) memiliki derajat keanggotaan yang memungkinkan elemen untuk masuk diantara dua himpunan keanggotaan.



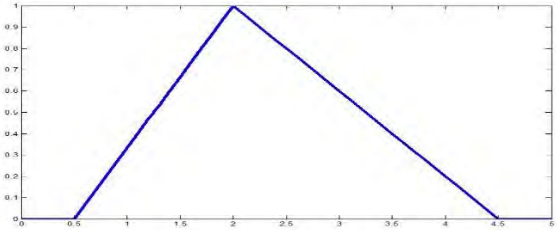
Gambar 2. 9 Himpunan Tegas (Hijau) dan Himpunan *Fuzzy* (Biru)

2.8.2 Fungsi Keanggotaan

Fungsi keanggotaan adalah suatu fungsi yang memetakan letak tiap masukan pada nilai keanggotaan atau derajat keanggotaan, yang bernilai antara satu dan nol. Berikut merupakan macam-macam fungsi keanggotaan pada logika *fuzzy*:

a) Fungsi Segitiga

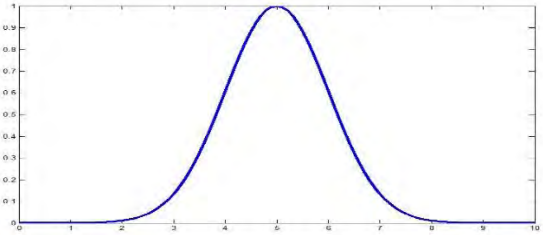
Gambar 2.10 berikut menunjukkan bentuk fungsi segitiga:



Gambar 2. 10 Fungsi Keanggotaan Segitiga

b) Fungsi *Gaussian*

Gambar 2.11 berikut menunjukkan bentuk fungsi *Gaussian*:



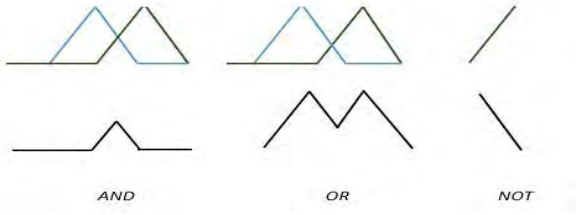
Gambar 2. 11 Fungsi Keanggotaan *Gaussian*

2.8.3 Operasi Himpunan *Fuzzy* [10]

Operasi himpunan *fuzzy* merupakan operasi logika terhadap dua buah fungsi keanggotaan atau lebih yang saling bersinggungan satu sama lain. Ada beberapa operasi yang dapat digunakan pada himpunan *fuzzy*, yakni:

a) Komplemen *(NOT)*
   Operasi *NOT* pada himpunan *fuzzy* A dengan fungsi keanggotaan μ_A dapat dinyatakan dengan persamaan:
   $$\mu_{A'}(x) = 1 - \mu_A(x) \quad (2.9)$$
b) Interseksi *(AND)*
   Interseksi dari himpunan *fuzzy* A dan B dengan himpunan semesta X dinotasikan dengan $A \cap B$ dapat dinyatakan dalam dua metode berikut:
   1) Minimum
      $$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, x \in X \quad (2.10)$$
   2) *Algebraic product*
      $$\mu_{A\cap B}(x) = \{\mu_A(x) \cdot \mu_B(x)\}, x \in X \quad (2.11)$$
c) *Union (OR)*
   Union dari himpunan *fuzzy* A dan B dengan himpunan semesta X dinotasikan dengan $A \cup B$ dapat dinyatakan dalam dua metode berikut:
   1) Maksimum
      $$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}, x \in X \quad (2.12)$$
   2) *Algebraic sum*
      $$\mu_{A\cup B}(x) = \{\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)\}, x \in X \quad (2.13)$$

Gambar 2.12 berikut merupakan ilustrasi dari operasi-operasi himpunan *fuzzy*:



Gambar 2. 12 Operasi Himpunan *Fuzzy*

Adapun langkah atau mekanisme inferensi *fuzzy* dalam penentuan keputusan seperti yang tertera pada Gambar 2.12 adalah sebagai berikut:

a) Fuzzifikasi masukan, yakni pemeteaan masukan tegas kedalam derajat keanggotaan dalam himpunan *fuzzy*.
b) Penerapan operator *fuzzy*, yakni penggunaan operasi logika semisal *AND* atau *OR* untuk menentukan hasil perpotongan tiap fungsi keanggotaan.
c) Penerapan metode implikasi, yakni pencarian konklusi atau hasil dari aturan *if-then* berdasarkan derajat fungsi keanggotaan premis masukan.
d) Agregasi, yakni pengkombinasian semua hasil dari aturan *if-then* menjadi satu.
e) Defuzzifikasi, yakni perubahan hasil agregasi menjadi keluaran tegas.

2.8.4 Defuzzifikasi [10]

Defuzzifikasi merupakan proses perubahan keluaran *fuzzy* menjadi keluaran tegas (*crisp*). Terdapat dua macam metode defuzzifikasi yang umum digunakan,yakni:

a. COG*(Center of Gravity)*

Metode COG menghitung keluaran *fuzzy* menggunakan titik tengah dari setiap hasil implikasi *fuzzy*. Metode COG dapat dituliskan dalam perasamaan:

$$u^{tegas} = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \tag{2.14}$$

dimana b_i metupakan titik tengah fungsi keanggotaan dan $\int \mu_{(i)}$adalah luas area dibawah fungsi keanggotaan $\mu_{(i)}$.

b. *Center Average*

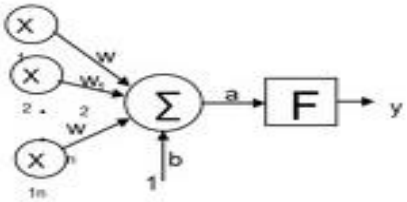
Metode *center average* menghitung rata-rata dari nilai tengah fungsi keanggotaan keluaran implikasi operator *fuzzy*. Metode *center average* dapat dituliskan dalam persamaan berikut:

$$u^{tegas} = \frac{\sum_i b_i \mu_{(i)}}{\sum_i \mu_{(i)}} \tag{2.15}$$

dimana b_i metupakan titik tengah fungsi keanggotaan dan $\mu_{(i)}$ adalah nilai bobot hasil implikasi operator *fuzzy* yang telah digunakan sebelumnya.

2.9 Jaringan Syaraf Tiruan [11]

Jaringan syaraf tiruan terdiri dari berberapa neuron yang saling terhubung. Neuron tersebut mentransformasikan informasi yang diterima melalui sambungan keluar menuju neuron-neuron lain. Pada jaringan syaraf tiruan hubungan ini dikenal sebagai bobot. Bobot dapat dikatakan sebagai nilai dari *arc* sebuah jaringan sedangkan neuron dapat dikatakan sebagai *node* dari sebuah jaringan. Nilai total dari semua bobot yang berhubungan dengan neuron masing-masing akan dijumlahkan dan akan dibandingkan dengan suatu nilai ambang tertentu (*threshold*) melalui fungsi aktivasi. Apabila neuron tersebut diaktifkan, maka neuron tersebut akan mengirimkan nilai *ouput* melalui bobot *output* ke semua neuron yang berhubungan. Pada jaringan syaraf, neuron akan dikumpulkan dalam lapisan-lapisan (*layer*). Neuron akan dihubungkan dengan *layer* sebelumnya dan *layer* sesudahnya dengan pola tertentu. Berberapa lapisan neuron yang saling terhubung dikatakan sebagai lapisan tersembunyi (*hidden layer*). Lapisan tersembunyi (*hidden layer)* akan merambatkan informasi dari lapisan *input* hingga lapisan *output* bergantung dari algoritma *learning* yang digunakan.Untuk sebuah jaringan syaraf tiruan sederhana yang terdiri dari neuron *input,* fungsi aktivasi dan *output* maka nilai dari masing-masing neuron *input* akan dikalikan dengan bobot masing-masing *arc* yang bersesuaian. Nilai dari total penjumlahan akan diaktifkan melalui fungsi aktivasi tertentu menuju *output*. Gambar 2.13 merupakan arsitektur jaringan syaraf tiruan sederhana yang terdiri dari lapisan *input,* lapisan tersembunyi dan lapisan *output*:



Gambar 2. 13 Jaringan Syaraf Tiruan Sederhana [11]

Dari Gambar 2.13 neuron akan menerima nilai *input* sebanyak N, kemudian akan dikalikan dengan bobot yang bersesuaian. Kadang kala jaringan syaraf tiruan tidak memakai bias (b) dalam arsitektur

jaringannya. Sehingga dari gambar 2.9.1 dapat dirumuskan sebagai berikut :

$$a = \sum_i^N X_i W_i \quad (2.16)$$

Kemudian fungsi aktivasi akan mengaktivasi a menjadi *output* y.
Untuk jaringan syaraf dengan jumlah neuron pada lapisan *output* sebanyak k buah maka :

$$a_k = \sum_i^N X_i \; ; \; \text{k= 1,2, 3…..,m} \quad (2.17)$$

Arsitektur jaringan syaraf terdiri dari jaringan syaraf lapisan tunggal, jaringan syaraf dengan banyak lapisan, dan jaringan syaraf dengan lapisan kompetitif. Jaringan syaraf dengan banyak lapisan (*layer*) mempunyai fungsi aktivasi sebanyak lapisan (*layer*) syaraf dan jumlah neuron pada lapisan *output* setiap lapisan.

2.9.1 Fungsi Aktivasi [11]

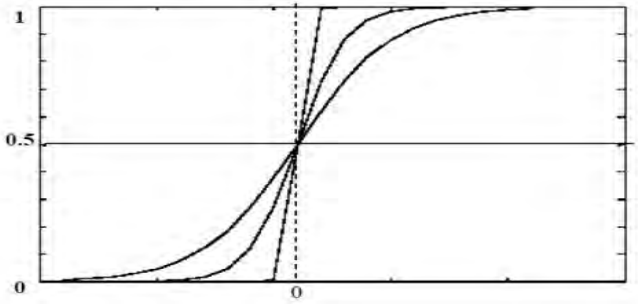
Fungsi aktivasi yang digunakan dalam jaringan syaraf tiruan antara lain :

Fungsi aktivasi sigmoid biner

Fungsi ini digunakan untuk jaringan syaraf yang dilatih dengan menggunakan algoritma *learning backpropagation.* Fungsi ini akan mengkonversikan nilai *input* menjadi nilai pada interval 0 hingga 1. Fungsi aktivasi ini banyak digunakan untuk algoritma *learning* yang membutuhkan nilai *ouput* pada interval 0 hingga 1

$$y = \frac{1}{1+e^{-ax}}. \quad (2.18)$$

nilai a mempengaruhi *gradient* atau *slope* dari nilai *output,* semakin besar a maka *slope* akan semakin besar. Gambar 2.14 menujukkan bentuk fungsi aktivasi sigmoid biner dengan variasi nilai parameter a atau parameter yang mempengaruhi *gradient* dari nilai *output*:



Gambar 2. 14 Sigmoid Biner [13]

2.9.2 Algoritma *BackPropagation* [11]

Algoritma *backpropagation* adalah algoritma *learning* yang berbasis arsitektur jaringan syaraf tiruan Algoritma *backpropagation* bekerja secara *feedfoward* untuk mendapatkan *error* yang merupakan selisih antara target *output* dengan *output learning*. Nilai *error* yang didapat akan dipropagasikan mundur (*back propagation*) untuk merevisi bobot pada jaringan syaraf tiruan. Data kedua kemudian juga dilatih secara *feedforward* dan *backpropagation* untuk mendapatkan nilai *error* pada iterasi pertama data kedua. Satu iterasi terus berlangsung hingga *array* data yang tersedia sudah dilatih semua. Setiap iterasi itu akan mencetak nilai *error*, dan ketika nilai *error* tersebut masih lebih kecil dari *error* target, maka iterasi terus berlangsung hingga *error* lebih besar sama dengan dari *error* target. Inisialisai bobot dilakukan secara acak dengan nilai acak yang cukup kecil. Algoritma back propagation juga dapat dibatasi maksimum iterasinya dengan membatasi iterasi dan mengabaikan *error* target minimum yang ingin dicapai.

2.10. *Fuzzy Modelling Network* [11] [12]

Fuzzy modelling network (FMN) adalah algoritma *learning* yang menggunakan arsitektur jaringan syaraf tiruan sebagai modelnya. Jaringan yang dibuat mengikuti mekanisme inferensi *fuzzy* atau direpresentasikan dengan sekelompok aturan-aturan *fuzzy*. FMN merupakan arsitektur jaringan yang dirancang untuk memproses data-data *fuzzy*. Pada FMN, unsur utama pada jaringan syaraf tidak lagi menggunakan bentuk klasik seperti jaringan syaraf pada umumnya, namun telah diganti dengan pendekatan logika *fuzzy*. Secara umum ada 3 komponen utama jaringan syaraf yang sering menjadi fokus pada FMN,

yaitu neuron *fuzzy*, model jaringan syaraf *fuzzy*, algoritma *learning*. FMN disusun berdasarkan logika *fuzzy* yakni mekanisme inferensi *fuzzy* yang berdasarkan aturan *rule base (if- then*) sehingga ada pemetaan aturan dalam bentuk *if-then* ke dalam jaringan syaraf.

FMN dirancang untuk mengatasi kelemahan dari sistem inferensi *fuzzy* yakni, penentuan fungsi keanggotaan dan pembangkitan fungsi pembelajaran pada aturan *rule base*. Pada kebanyakan aplikasi pada logika *fuzzy*, fungsi keanggotaan biasanya dibentuk menggunakan pendekatan terhadap berberapa fungsi tertentu, seperti linear, segitiga, trapezium,dan sebagainya. Pada FMN fungsi aktivasi sigmoid biner dapat digunakan untuk merepresentasikan fungsi keanggotaan. *neuro- fuzzy* berdasarkan model *input* dan bobot terbagi menjadi 3 tipe yakni tipe 1, tipe 2, dan tipe 3.

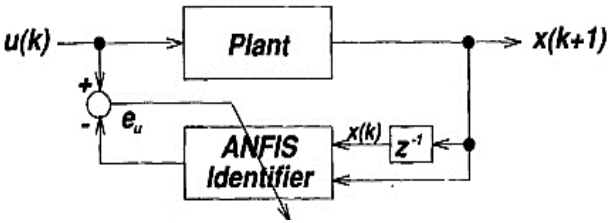
Pada FMN tipe II, aturan *fuzzy* memiliki konsekuen berupa konstanta (orde 0). Arsitektur dari jaringan FMN tipe II adalah sebagai berikut :

Jika X_1 adalah A_{i1} dan X_2 adalah A_{i2} maka y = C_i (2.19)
$i = 1,2,3, \dots, n$

Premis pada FMN tipe II berupa fungsi keanggotaan *fuzzy* dari *input crisp* yang diaktifkan melalui fungsi aktivasi sigmoid.

2.11 *General Learning* [13]

General Learning atau *inverse learning* adalah salah satu metode untuk merancang kontroler *neuro-fuzzy*. *General learning* terdiri dari dua tahapan yakni, tahap *offline* dan tahap *online*. Urutan tahapan adalah tahapan *offline* kemudian tahapan *online*. Tahap *offline* digunakan untuk mendapatkan bobot- bobot yang bersuaian dengan jaringan *neuro-fuzzy*. Tahap *offline* juga merepresentasikan dinamika *invers* dari sebuah *plant*. Gambar 2.15 berikut merupakan diagram blok saat tahapan *offline*:



Gambar 2. 15 Diagram Blok Tahap *Offline* [13]

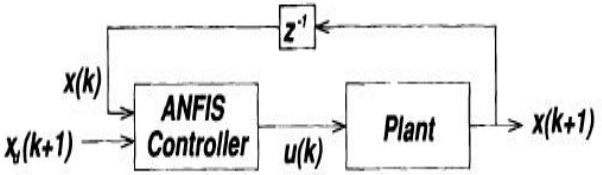
Dari Gambar 2.15 sinyal masukkan ke *plant* adalah sinyal kontrol (u(k)), dan respon keluaran *plant* adalah sinyal respon x(k+1), secara matematis hubungan antara sinyal kontrol dengan sinyal respon adalah sebagai berikut :

$$x(k+1) = f(x(k), u(k)) \quad (2.20)$$

Untuk tujuan pembelajaran, nilai sinyal kontrol dapat diasumsikan sebagai sebuah besaran skalar. Dalam aljabar matriks nilai skalar berarti matriks dengan orde 1×1. Keluaran dari *identifier* adalah sinyal kontrol estimasi (U) yang dapat dinyatakan dengan persamaan berikut:

$$U = G(x(k), x(k+n) \quad (2.21)$$

Fungsi G adalah jaringan *neuro-fuzzy* yang dirancang dan menghasilkan sinyal kontrol estimasi (U). Setelah melakukan pelatihan (*training)* data pada tahapan *offline,* maka bobot yang dihasilkan akan dimasukkan ke jaringan *neuro fuzzy* untuk tahapan *online*. Gambar 2.16 berikut merupakan diagram blok saat tahapan *online*:



Gambar 2. 16 Diagram Blok Tahap *Online* [13]

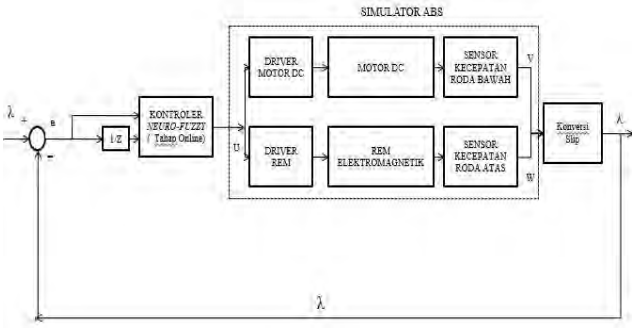
--Halaman ini sengaja dikosongkan--

BAB 3
PERANCANGAN SISTEM

Perancangan sistem mencakup perancangan mekanik, perancangan elektronik, perancangan program, identifikasi *plant,* dan perancangan kontroler *neuro-fuzzy.*

3.1 Gambaran Umum Sistem

Simulator ABS merupakan suatu alat yang digunakan untuk menguji slip pada roda tanpa harus menjalankan roda pada permukaan jalan. Pada simulator ABS, roda kendaraan akan dipasang diatas roda lain (saling bersinggungan) yang diputar dengan kecepatan tertentu. Kecepatan roda kendaraan akan diwakili oleh roda atas dan kecepatan roda bawah akan mewakili kecepatan *longitudinal* kendaraan. Gambar 3.1 merupakan diagram blok dari perancangan sistem simulator ABS yang dibuat adalah seperti berikut:



Gambar 3. 1 Diagram Blok Sistem ABS

Berdasarkan Gambar 3.1 sistem ABS terdiri dari *input* berupa slip yang diharapkan, blok kontroler berupa komputer, *interface* arduino uno, simulator ABS, dan *output* berupa slip hasil kalkulasi di arduino uno dimana data yang diolah didapat dari sensor kecepatan roda atas dan roda bawah. Alur kerja diagram blok sistem ABS yakni Arduino uno akan mengirimkan nilai slip melalui komunikasi serial ke komputer, kemudian

komputer akan mengolah sinyal *error* menggunakan algoritma *Neuro-fuzzy* pada program matlab. Proses pengolahan sinyal *error* dijalankan secara *offline* melalui *general learning*. Setelah *training offline* dilakukan maka bobot bersesuaian yang sudah didapat dimasukkan ke dalam program untuk tahap *online*.Pada tahap *online* sinyal *error* masuk ke kontroler dan menghasilkan sinyal kontrol ke *plant* simulator ABS melalui arduino uno sebagai *interface.*

3.2 Perancangan Perangkat Keras

Perancangan perangkat keras terdiri dari perancangan mekanik, perancangan *driver* motor DC, pemasangan rem elektromagnetik, perancangan *driver* rem elektromagnetik, perancangan sensor kecepatan, dan *project board* arduino uno.

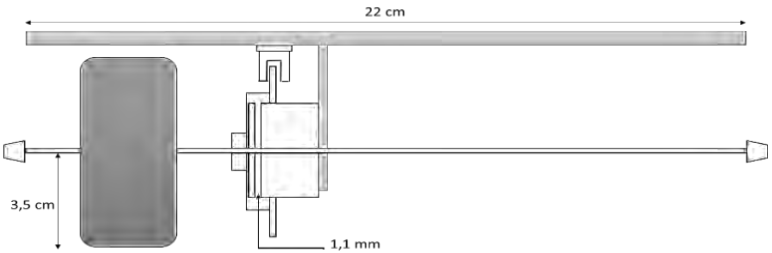
3.2.1 Perancangan Mekanik

Simulator ABS memiliki beberapa komponen mekanik yang memiliki spesifikasi dan fungsi masing-masing untuk menunjang performa simulator. Beberapa komponen mekanik yang perlu dirancang pada simulator ABS adalah roda atas, roda bawah, motor DC, dan rem elektromagnetik

Roda Bebas Atas

Roda bebas pada simulator ABS mewakili roda mobil yang berputar pada kendaraan. Roda bebas memiliki jari-jari roda sebesar 3,5 cm dengan panjang shaf poros 22 cm.

Pada roda bebas, dipasang sebuah sensor kecepatan dan rem elektromagnetik pada shaft penyangga roda atas. Sensor kecepatan digunakan untuk mengetahui kecepatan putar roda, sedangkan rem elektromagnetik digunakan untuk mengatur slip yang terjadi pada roda. Gambar 3.2 berikut merupakan sketsa dari roda bebas atas:



Gambar 3. 2 Sketsa Roda Bebas Atas

Roda Bawah dan Motor DC

Motor DC digunakan sebagai penggerak roda bawah yang berperan sebagai kecepatan *longitudinal* kendaraan. Roda bawah dipasang satu shaft dengan motor DC. Motor DC yang digunakan adalah motor DC *High Torque*. Motor DC yang digunakan mempunyai tegangan kerja 24 Volt DC. Jari-jari roda bawah sama dengan roda atas, yakni 3,4 cm. Roda bawah dipasang pada sumbu yang sama dengan sumbu putar motor DC. Momen inersia roda bawah dibuat lebih besar dibandingkan momen inersia roda atas dengan cara menambah massa roda bawah. Cara yang dilakukan untuk menambah massa roda bawah adalah dengan mengecor ruang diantara jejari roda. Momen Inersia roda bawah dibuat lebih besar agar roda bawah masih meluncur ketika roda atas dilakukan pengereman. Hal ini didasarkan pada konsep torsi yakni torsi roda atas sama dengan torsi roda bawah. Dimana torsi adalah hubungan perkalian antara perlambatan roda dengan momen inersia roda.

Rem Elektromagnetik

Rem elektromagnetik yang digunakan pada tugas akhir ini adalah rem mikro elektromagnetik jenis kering dengan satu piringan (*dry-type single-plate*) produksi *Shinko Electric*. Tabel 3.1 berikut merupakan spesifikasi rem yang digunakan:

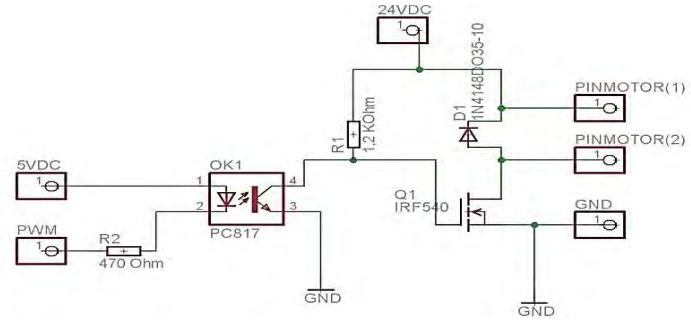
Tabel 3.1 Spesifikasi Rem Elektromagnetik

Tipe rem	BB-2.6-N01
Tegangan pengoprasian rem	12 Volt
Daya pengoperasian rem	3,3 Watt
Produksi	Wincoelectric.co.ltd
Ukuran diameter rem	luar: 2,7 cm dalam: 1,1cm
Tinggi rem	1,5 cm

3.2.2 Perancangan *Driver* Motor DC

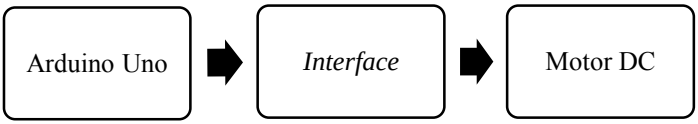
Driver motor digunakan untuk menjalankan motor DC dengan suplai arus yang sesuai. Tegangan Kerja Motor DC yang digunakan adalah 24 Volt DC. Motor DC yang digunakan adalah motor DC *high torque* sehingga arus saat start sebesar 10 A dan arus nominal sekitar 2 A. Oleh karena itu transistor *switching* yang digunakan adalah transistor

FET 540 (NPN) yang mampu menerima arus Id (*drain source*) hingga maksimum 30 A. Sinyal *Pulse Width Modulation* (PWM) dihasilkan oleh arduino uno melalui *pin* PWM dan Vcc arduino menggunakan 5 Volt DC. *Optocoupler* berfungsi sebagai isolasi atau *interface* antara kedua komponen yakni arduino dan transistor NPN. *Pin* konektor yang dipasang pada papan pcb adalah konektor ptr-500. Penggunaan dioda *clamp* yang dipasang paralel dengan beban berfungsi untuk mengurangi beban induksi yang ditimbulkan oleh sinyal PWM. Gambar 3.3 berikut merupakan skematik dari rangkaian *driver* motor DC:



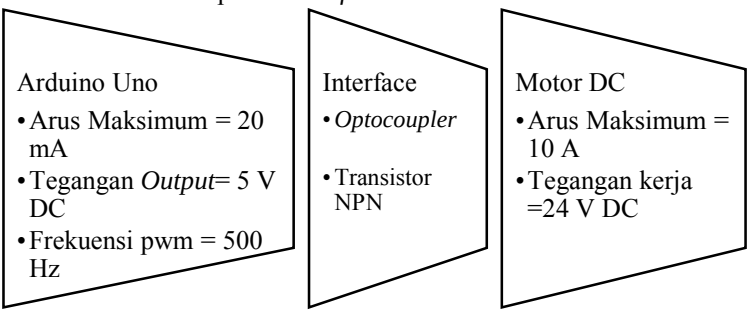
Gambar 3. 3 Skematik *Driver* Motor DC

Skematik rangkaian *driver* motor DC dibuat pada *software* eagle. Rangkaian menggunakan berberapa komponen yakni, konektor ptr 500, ic *optocopler* pc 817, resistor 1,2 KOhm, resistor 470 Ohm dioda *clamp*, dan transistor FET IRF 540. Gambar 3.4 menjelaskan konsep untuk membuat *driver* motor DC adalah sebagai berikut:



Gambar 3. 4 Konsep *Driver* Motor DC

Untuk dapat mengatur kecepatan pada motor DC dari arduino uno dibutuhkan suatu *interface* untuk mensinkronkan spesifikasi listrik (tegangan, arus, dan frekuensi) dari masing-masing komponen. Gambar 3.5 berikut merupakan spesifikasi tegangan dan arus pada I/O arduino dan motor DC serta komponen *interface*:



Gambar 3. 5 Spesifikasi Komponen

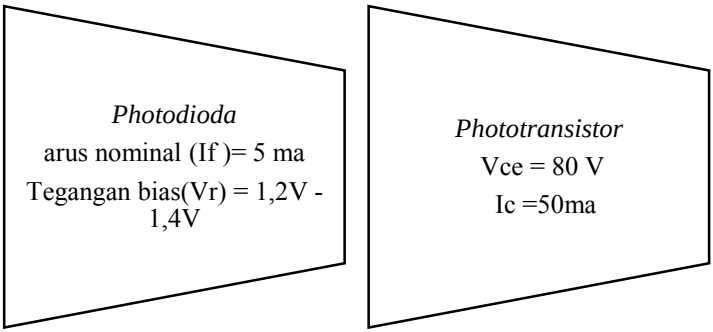
Optocoupler merupakan komponen elektronika yang terdiri dari *photodioda* dan *phototransistor* yang terpisah secara elektrik. Saat *photodioda* diberikan tegangan bias maju (*forward bias*), maka kaki basis pada *phototransistor* akan mempunyai logic *high*, sehingga arus kolektor akan mengalir dari kaki kolektor menuju kaki emitor. Saat arus mengalir (Ic) maka tegangan pada kaki kolektor (Vc) mempunyai logic *low*. Dari pola tersebut, maka *optocoupler* berfungsi sebagai gerbang logika *inverter (*berkebalikan*)* Pada perancangan *driver* motor DC, *optocoupler* digunakan sebagai komponen isolasi antara dua komponen yang memiliki spesifikasi tegangan dan arus yang berbeda. Oleh karena itu posisi *ground* pada kaki *photodioda* dengan posisi *ground* pada kaki *emitor* (*phototransistor*) harus dipisah agar tidak terjadi *short circuit.* Nilai arus dan tegangan pada rangkaian harus diperhitungkan terlebih dahulu. Akan dijelaskan perhitungan nilai komponen pada rangkaian *optocoupler*. Secara garis besar perhitungan dibagi menjadi dua bagian yakni rangkaian keluaran arduino uno dengan *photodioda optocoupler* dan rangkaian *phototransistor optocoupler* dengan transistor FET. Bagian pertama dapat dijelaskan sebagai berikut :

Sebagaimana yang telah dijelaskan bahwa komponen *optocoupler* berfungsi sebagai gerbang logika *inveter* (berkebalikan). Agar bekerja tidak berkebalikan maka kaki *ground photodioda* dihubungkan dengan

pin PWM Arduino dan tegangan 5 Volt DC yang berasal dari Arduino Uno dihubungkan dengan kaki anoda (positif) untuk menghasilkan tegangan bias maju.

Bagian kedua dapat dijelaskan sebagai berikut :

Kaki kolektor *phototransistor* dihubungkan dengan Vcc 24 Volt DCdan dihubungkan juga dengan kaki *Gate* transistor FET. Vcc 24 Volt DC juga terhubung dengan beban motor DC dan diode *clamp*. Setelah *wiring* rangkaian telah dipastikan, maka yang berikutnya menentukan besarnya resistor yang diperlukan antara kedua komponen tersebut.. Gambar 3.6 berikut menjelaskan spesifikasi tegangan dan arus pada *optocoupler*:



Gambar 3. 6 Spesifikasi *Optocoupler*

Untuk *wiring* rangkaian *photodioda optoclouper* dengan arduino uno, arus yang dipilih sebesar 7 ma karena *range* yang tidak jauh dengan arus nominal if dan sekitar 35% dari arus maksimal arduino Uno, sehingga didapatkan

$$R = \frac{Vcc - Vr_{min}}{If} \tag{3.1}$$

$$R = \frac{5 - 1{,}2}{0{,}007} = \frac{3{,}3}{0{,}007} = 470\ Ohm \tag{3.2}$$

Untuk *wiring* rangkaian *optocoupler* dengan Vcc motor DC dan transistor FET. Tegangan Vcc yang digunakan adalah 24 Volt DC sehingga masih dalam *range* aman dibandingkan nilai maksimal Vce. Untuk arus Ic dipilih sebesar 20 ma atau sekitar 40% dari Ic maksimal.

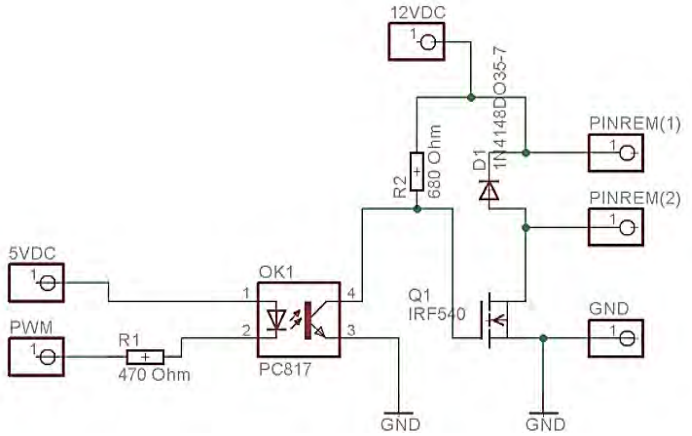
$$R = \frac{Vcc - 0}{Ic} \tag{3.3}$$

$$R = \frac{24 - 0}{0{,}02} = 1{,}2\ K\ Ohm \tag{3.4}$$

Saat Vgs > Vgs_{on} (Vgs_{on} sekitar 3-4 V) maka arus akan mengalir dari kaki kolektor transistor ke kaki emitor pada transistor FET. Cara kerja *driver* motor dc secara keseluruhan adalah saat sinyal PWM arduino *high* maka tidak ada arus yang mengalir pada *photodioda* sehingga tegangan di Vc (tegangan kaki kolektor) *high*. Saat Vc mempunyai tegangan 24 Volt DC maka arus akan mengalir dari kaki kolektor menuju kaki emitor sehingga motor akan berputar sesuai dengan besarnya sinyal PWM yang diberikan. Kecepatan putar motor akan maksimal saat sinyal PWM sebesar 100%. *Driver* motor yang dibuat disesuaikan dengan kebutuhan *plant* yakni arah putaran motor DC hanya 1 arah (CW atau CCW).

3.2.3 *Driver* Rem Elektromagnetik

Driver rem elektromagnetik digunakan untuk menjalankan rem elektromagnetik dengan suplai arus yang sesuai. Tegangan Vcc rem menggunakan 12 Volt DC dan Vcc Arduino menggunakan 5 Volt DC. Rem elektromagnetik yang digunakan memiliki arus nominal sebesar 0,5 A. Dengan arus yang cukup besar maka juga memerlukan transistor *switching High Power* yang dapat melewatkan arus Ic (arus kolektor) untuk transitor BJT atau arus Id (*drain source)* untuk transistor FET diatas nominal 0,5 A. Transistor yang digunakan adalah transistor FET 540 (NPN). Sinyal *Pulse Width Modulation* (PWM) dihasilkan oleh arduino uno melalui *pin* pwm. *Optocoupler* juga berfungsi sebagai rangkaian isolasi dan *interface* antara kedua komponen, yakni arduino dan transistor FET. Penggunaan dioda *clamp* yang dipasang paralel dengan beban berfungsi untuk mengurangi beban induksi yang ditimbulkan oleh sinyal PWM. Gambar 3.7 berikut merupakan skematik dari rangkaian *driver* rem elektromagnetik:

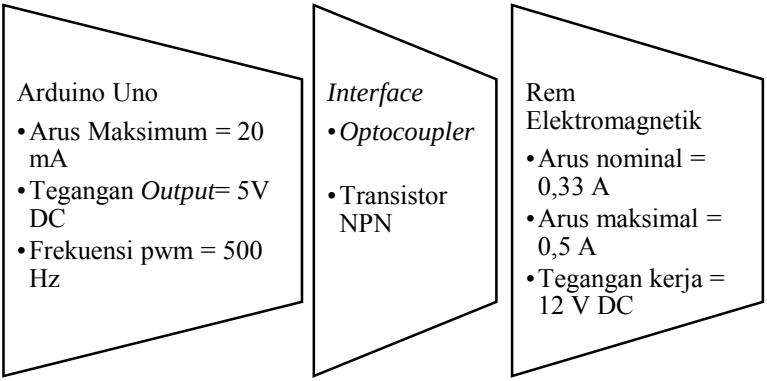


Gambar 3. 7 Skematik *Driver* Rem Elektromagnetik

Skematik rangkaian *drive*r rem dibuat pada *software* eagle. Rangkaian menggunakan berberapa komponen yakni, konektor ptr 500, ic *optocoupler* pc 817, resistor 680 Ohm, resistor 470 Ohm Dioda *clamp,* dan transistor FET IRF 540. Gambar 3.8 berikut menjelaskan konsep untuk membuat rem elektromagnetik:

Gambar 3. 8 Konsep *Driver* Rem Elektromagnetik

Untuk dapat mengatur besarnya pengereman dari arduino uno dibutuhkan suatu *interface* untuk mensinkronkan spesifikasi listrik (tegangan, arus, dan frekuensi) dari masing-masing komponen. Gambar 3.9 berikut merupakan spesifikasi tegangan dan arus pada I/O arduino dan rem elektromagnetik serta komponen *interface*:



Gambar 3. 9 Speifikasi Komponen

Optocoupler merupakan komponen elektronika yang terdiri dari *photodioda* dan *phototransistor* yang terpisah secara elektrik. Saat *photodioda* diberikan tegangan bias maju (*forward bias*), maka kaki basis pada *phototransistor* akan mempunyai logic *high*, sehingga arus kolektor akan mengalir dari kaki kolektor menuju kaki emitor. Saat arus mengalir (Ic) maka tegangan pada kaki kolektor (Vc) mempunyai logic *low*. Dari pola tersebut, maka *optocoupler* berfungsi sebagai gerbang logika *inverter (*berkebalikan*)* pada perancangan *driver* rem, *optocoupler* digunakan sebagai komponen isolasi antara 2 komponen yang memiliki spesifikasi tegangan dan arus yang berbeda. Oleh karena itu posisi *ground* pada kaki *photodioda* dengan posisi *ground* pada kaki emitor *(phototransistor*) harus dipisah agar tidak terjadi *short circuit.* Nilai arus dan tegangan pada rangkaian harus diperhitungkan terlebih dahulu. Akan dijelaskan perhitungan nilai komponen pada rangkaian *optocoupler*. Secara garis besar perhitungan dibagi menjadi 2 bagian yakni rangkaian *output* arduino uno dengan *photodioda optocoupler* dan rangkaian *phototransistor optocoupler* dengan transistor FET. Bagian pertama dapat dijelaskan sebagai berikut :

Sebagaimana yang telah dijelaskan bahwa komponen *optocoupler* berfungsi sebagai gerbang logika *inveter* (berkebalikan). Agar bekerja tidak berkebalikan maka kaki *ground photodioda* dihubungkan dengan *pin* PWM Arduino dan tegangan 5 Volt DC yang berasal dari Arduino Uno dihubungkan dengan kaki anoda (positif) untuk menghasilkan tegangan bias maju. Bagian kedua dapat dijelaskan sebagai berikut :

Kaki kolektor *phototransistor* dihubungkan dengan 12 Volt DC dan dihubungkan juga dengan kaki *gate* transistor fet. Vcc 12 Volt DC juga terhubung dengan beban rem elektromekanik dan dioda *clamp*. Setelah *wiring* rangkaian telah dipastikan, maka yang berikutnya menentukan besarnya resistor yang diperlukan antara kedua komponen tersebut. Spesifikasi tegangan dan arus pada *optocoupler* sudah dijelaskan melalui diagram blok pada bagian *driver* motor DC. Perbedaan mendasar dalam perhitungan *driver* rem dibandingkan dengan *driver* motor DC adalah perbedaan Vcc *supply* beban. Untuk *wiring* rangkaian *photodioda optoclouper* dengan arduino uno, arus yang dipilih sebesar 7 ma karena *range* yang tidak jauh dengan arus nominal if dan sekitar 35% dari arus maksimal arduino Uno, sehingga didapatkan

$$R = \frac{Vcc - Vr_{min}}{If} \tag{3.5}$$

$$R = \frac{5 - 1{,}2}{7\ X\ 10^{-3}} = \frac{3{,}3}{0{,}007} = 470\ Ohm \tag{3.6}$$

Untuk *wiring optocoupler* dengan Vcc rem elektromagnetik dan transistor FET. Tegangan Vcc yang digunakan adalah 12 Volt DC sehingga masih berada dalam tegangan kerja Vce *phototransistor optocoupler*. Arus Ic yang digunakan adalah sebesar 20 ma atau sekitar 40% dari arus Ic maksimal.

$$R = \frac{Vcc - 0}{Ic} \tag{3.7}$$

$$R = \frac{12 - 0}{0{,}02} = 600\ Ohm \tag{3.8}$$

Maka nilai resistor yang dipilih adalah 680 Ohm. Saat $Vgs > Vgs_{on}$ pada transistor FET maka arus *drain* akan mengalir dari kaki kolektor ke kaki emitor. Cara kerja keseluruhan dari *driver* rem elektromagnetik adalah saat sinyal pwm arduino *high* maka tidak ada arus yang mengalir pada *photodioda*, sehingga tegangan pada Vc (tegangan kaki kolektor) *high* dan arus akan mengalir dari kaki kolektor ke kaki emitor pada transistor FET. Besarnya pengaruh kuat medan

elektromagnetik pada rem dipengaruhi dari besarnya sinyal pwm yang diberikan.

3.2.4 Sensor Kecepatan [7]

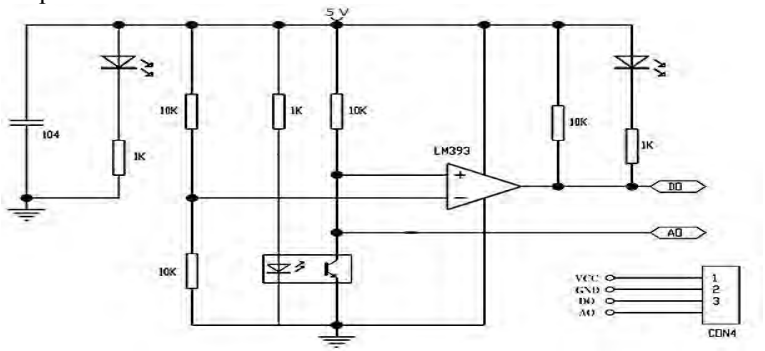
Sensor kecepatan yang digunakan adalah modul sensor kecepatan dengan komparator IC LM 393. Ada empat buah *pin* yang terhubung pada rangkaian sensor, yakni:

- *Pin* Vcc : suplai tegangan 3,3-12 Volt
- *Pin* Gnd : *ground*
- *Pin* D0 : keluaran digital
- *Pin* A0 : keluaran analog

Gambar 3.10 berikut merupakan sensor kecepatan yang digunakan dalam perancangan simulator ABS:

Gambar 3. 10 Sensor Kecepatan [7]

Gambar 3.11 berikut merupakan rangkaian dari sensor kecepatan dengan komparator IC LM 393:



Gambar 3. 11 Skematik Rangkaian Sensor Kecepatan [7]

Pada Praktiknya skema sensor kecepatan telah dimodifikasi sesuai kebutuhan. Keluaran sinyal digital tidak lagi pada *pin* DO melainkan pada *pin* A0, karena komponen kapasitor sebelum *pin* A0 dihilangkan sehingga *pin* A0 merupakan hasil *switcing* dari *phototransistor* di dalam skema. Sensor kecepatan yang digunakan sudah dalam bentuk rangkaian smd dan menggunakan *optocoupler* fc 03.

3.2.5 Arduino UNO

Arduino UNO berfungsi sebagai pengambil data dari sensor kecepatan dan mengirimnya ke Komputer, serta sebagai pengatur PWM motor penggerak roda bawah simulator ABS dan pengatur PWM Rem elektromagnetik. *Pin* PWM yang dihubungkan dengan motor adalah *pin* 10 dan *pin* PWM yang dihubungkan dengan rem elektromagnetik adalah *pin* 9. *Pin* ouput 2 sensor kecepatan dihubungkan dengan arduino pada *pin* I/O digital.

3.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak mencakup perancangan perangkat lunak pada *compiler* arduino dan matlab menggunakan bahasa pemrograman dan t*oolbox* Simulink.

3.3.1 Program Arduino

Program yang dibuat pada arduino diperlukan untuk keperluan pengambilan data, mengatur *pulse width modulation* (pwm) motor DC sebagai penggerak roda bawah, dan mengatur *pulse width modulatio*n (pwm) rem elektromagnetik. Pengaturan pwm pada motor DC dan rem elektromagnetik dilakukan dengan mengatur besarnya lama waktu saat *high* dan lama waktu saat *low* atau yang biasa disebut *duty cycle*. Frekuensi *default* pwm yang dihasilkan oleh pin arduino uno sebesar 500 Hz dan 1000 Hz. Pada Arduino Uno terdapat 6 pin I/O digital yang dapat digunakan sebagai *pin* pwm yakni 3,5,6,9,10,dan 11. *Pin* 3,9,10, dan 11 mempunyai frekuensi *default* sebesar 500 Hz. *Pin* pwm arduino yang dihubungkan dengan motor adalah *pin* 10 dan *pin* pwm yang dihubungkan dengan rem elektromagnetik adalah *pin* 9. Frekuensi 500 Hz berarti dalam 1 detik terdapat 500 pulsa, apabila dikonversi kedalam periode maka didapatkan nilai 2000 µs. Satu periode pulsa pwm yang dihasilkan sebesar 2000 µs yang merupakan jumlah antara waktu *high* dan waktu *low*. Besarnya pwm dikendalikan dari besarnya *duty cycle* dari sinyal pwm, yangmana *duty cycle* dirumuskan sebagai berkut :

$$DC = \frac{T_{high}}{T_{high} + T_{low}} \quad (3,9)$$

Program arduino yang dibuat pada *compiler* arduino menggunakan perintah *delayMicroseconds* pada fungsi *void (loop)* untuk memberikan nilai T_{high} dan T_{low}. Untuk keperluan pengambilan data dari sensor kecepatan, Perintah yang digunakan adalah *Pulsein()*. Data dari sensor kecepatan merupakan sinyal digital yang mempunyai besaran *duty cycle*. *Perintah Pulsein()* akan membaca lama waktu pulsa saat *high* atau *low*, sehingga dapat diketahui berapa pulsa yang dihasilkan dalam waktu satu detik. Informasi mengenai banyaknya pulsa yang dihasilkan dalam 1 detik digunakan untuk mengetahui nilai dari *rotation per seconds* (rps) dengan membagi banyaknya pulsa dalam 1 detik dengan 2 kali jumlah lubang pada piringan *encoder*. Nilai rps kemudian akan dikonversi ke dalam *rotation per minute* (rpm) dengan mengalikan besaran rps dengan 60 karena dalam 1 menit terdapat 60 detik. Nilai kecepatan aktual dari roda atas dan roda bawah dinyatakan dalam rpm. Dalam pengambilan data digunakan 2 arduino untuk sensor kecepatan yang diletakkan di sumbu roda atas dan sensor kecepatan yang diletakkan di sumbu roda bawah. Penggunaan 2 arduino dikarenakan perintah *Pulsein()* pada arduino uno akan berhenti ketika salah satu pembacaan data dari salah satu sensor kecepatan juga berhenti. Arduino kemudian akan mengirimkan data mengenai informasi kecepatan ke komputer melalui komunikasi serial. Data kecepatan putar dan *longitudinal* yang terbaca akan tampil pada *serial monitor compiler* arduino. Pengambilan data dilakukan dengan bantuan aplikasi *coolterm win* yang dapat menyimpan serial data dalam format *txt*.

3.3.2 Program MATLAB

Perancangan kontroler menggunakan bahasa pemrograman dan *toolbox* simulink pada perangkat lunak matlab. Simulink digunakan untuk mensimulasikan kontroler yang dibuat menggunakan bahasa pemrograman pada model matematis yang merepresentasikan *plant Anti Locking Braking System* (ABS). Kontroler yang dibuat menggunakan bahasa pemrograman dihubungkan dengan *toolbox Interpreted Matlab Function* pada simulink. Program matlab dibagi menjadi dua yakni program untuk simulasi dan program untuk implementasi. Untuk Program

simulasi terrdiri dari program saat *training* data *offline* menggunakan *general learning* dan saat simulasi *online.*

3.4 Identifikasi dan Pemodelan *Plant*

Proses Identifikasi bertujuan untuk mencari parameter dari *plant* [14]. Parameter yang sudah didapatkan digunakan untuk membuat model matematis dari *plant* melalui evaluasi data pengukuran *input output*. Identifikasi parameter *plan*t dilakukan secara dinamis yakni dengan mengubah kecepatan roda bawah dan nilai pwm rem pada roda atas secara simultan. Parameter *plant* ABS adalah slip sebagai *output plant* dan besarnya pwm rem sebagai *input plant*. Model matematis *plant* dirumuskan dengan model struktur pendekatan stokastik. Model struktur pendekatan stokastik yang digunakan adalah struktur *Auto Regressive Exogeneus* (ARX).

Proses identifikasi sistem dilakukan dengan mendapatkan data slip pada berberapa titik kecepatan roda bawah dan berberapa titik nilai pwm rem pada roda atas. Pengambilan data dilakukan dengan meningkatkan kecepatan roda bawah secara perlahan-lahan, dan pada setiap kecepatan tertentu diberikan rem dengan berberapa nilai pwm. Kecepatan roda bawah ditingkatkan setiap 10% pwm, jadi terdapat 10 titik kecepatan yang menjadi sampel data. Kecepatan roda bawah ditingkatkan mulai dari 10% hingga 100%, sehingga didapatkan hubungan antara kecepatan roda bawah dan perubahan *duty cycle* (PWM) motor DC yang terhubung satu poros dengan roda bawah yang tertera pada Tabel 3.2 berikut:

Tabel 3.2 Hubungan Kecepatan Roda Bawah dan Pwm Motor DC

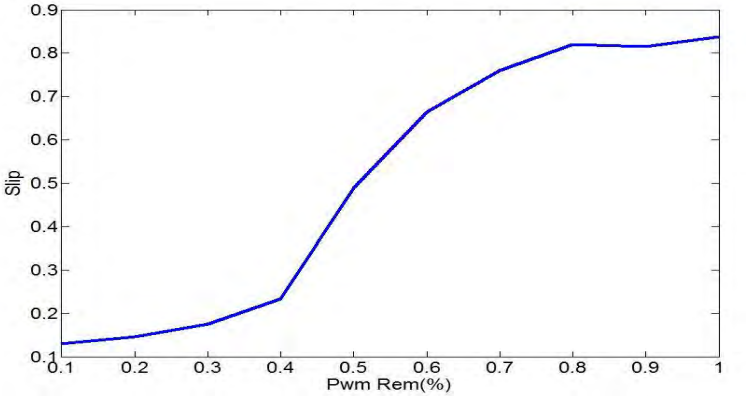
Persentase Pwm Motor DC	Kecepatan Roda Bawah (RPM)
10%	573
20%	1224
30%	1621
40%	1854
50%	1994
60%	2097
70%	2155
80%	2173
90%	2220
100%	2313

Setiap titik kecepatan roda bawah diberikan pwm rem dari 0% hingga 100% (PWM rem ditingkatkan setiap 10%). Saat dilakukan pengereman maka roda bawah harus sampai berhenti, baru kemudian dapat dijalankan kembali. Data yang diperoleh berupa nilai slip pada berberapa *range* kecepatan roda bawah dan *range* nilai pwm rem pada roda atas. Tabel 3.3 berikut merupakan hubungan antara perubahan pwm rem pada roda atas dengan titik kerja slip untuk 3 titik kecepatan roda bawah yang tertera berikut ini:

Tabel 3.3 Hubungan Pwm Rem dan Titik Kerja Slip

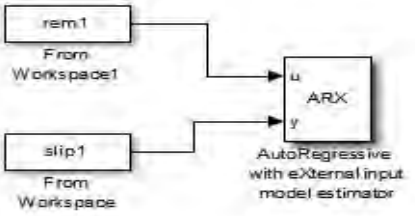
	Kecepatan Roda Bawah (RPM)		
Persentase Pwm Rem Roda Atas	2313 (100%)	2220 (90%)	2193 (80%)
10%	0,12934	0,08476	0,16076
20%	0,14616	0,12543	0,17403
30%	0,17498	0,13771	0,21248
40%	0,23304	0,24226	0,28326
50%	0,48866	0,50346	0,44012
60%	0,66523	0,68928	0,66226
70%	0,76070	0,78347	0,74383
80%	0,82027	0,80893	0,79062
90%	0,81499	0,84319	0,87207
100%	0,83794	0,86017	0,86869

Berdasarkan Tabel 3.3 diatas, titik kerja slip merupakan rata-rata slip yang didapat pada satu titik kerja rem untuk satu titik kecepatan roda bawah. Data yang diolah adalah data slip pada kecepatan roda bawah dan kecepatan roda atas saat mulai mengalami perlambatan. Pada nilai pwm rem tinggi roda bagian atas sempat mengunci (tidak berputar) hingga roda bagian bawah berhenti berputar Dari data yang diperoleh dapat ditentukan titik kerja rem terhadap slip untuk masing-masing kecepatan roda bawah seperti yang dicantumkan pada Tabel 3.3. Dari data pada Tabel 3.3 terlihat bahwa titik kerja slip bernilai lebih dari 0,2 saat persentase pwm rem diatas 40% untuk 3 titik kecepatan roda bawah. Dalam tugas akhir ini, penulis memilih kecepatan 100% (maksimum) untuk kecepatan roda bawah. Gambar 3.12 berikut merupakan *plot* grafik hubungan titik kerja rem terhadap slip pada kecepatan roda bawah 100% pwm:



Gambar 3. 12 Model *Plant* Rem Vs Slip

Berdasarkan Gambar 3.12, data titik kerja rem terhadap slip, akan dimodelkan pendekatan model *plant* dengan struktur pendekatan stokastik yakni Pemodelan dengan auto regressive exougeneus (ARX) menggunakan *toolbox system identification* simulink. Gambar 3.13 berikut merupakan blok simulink untuk mendapatkan pendekatan model ARX:



Gambar 3. 13 Blok Simulink Pemodelan ARX

Data nilai pwm rem dihubungkan dengan *pin* u sebagai masukkan dari sebuah *plant*, dan data nilai slip dihubungkan dengan *pin* y sebagai keluaran *plant*. Data rem1 pada Gambar 3.13 merupakan data *array* yang memiliki karakteristik dua dimensi yakni urutan data titik kerja rem dan perubahan persentase rem. Data slip1 Pada Gambar 3.13 merupakan data

array yang memiliki karakteristik dua dimensi yakni urutan data titik kerja rem dan perubahan titik kerja slip. Tabel 3.4 berikut merupakan nilai setiap elemen untuk masing-masing dimensi berdasarkan data rem1 pada Gambar 3.13:

Tabel 3.4 Data Titik Kerja Rem

Urutan Data Titik Kerja Rem	Persentase PWM Rem
1	10%
2	20%
3	30%
4	40 %
5	50%
6	60%
7	70%
8	80%
9	90%
10	100%

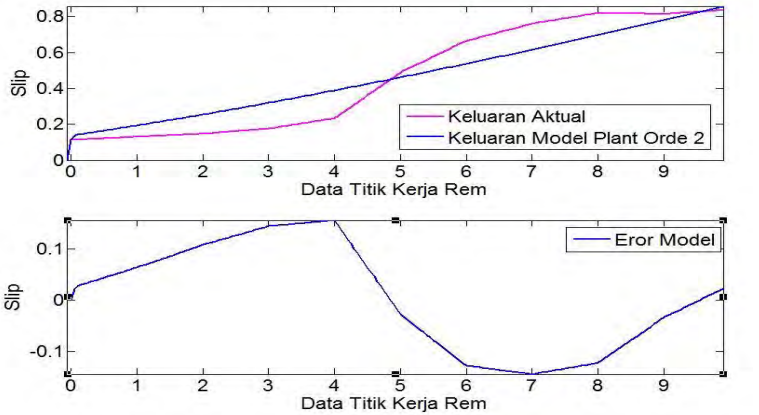
Tabel 3.5 berikut merupakan nilai setiap elemen untuk masing-masing dimensi berdasarkan data slip1 pada Gambar 3.13:

Tabel 3.5 Data Titik Kerja Slip

Urutan Data Titik Kerja Rem	Titik Kerja Slip
1	0,12934
2	0,14616
3	0,17498
4	0,23304
5	0,48866
6	0,66523
7	0,76070
8	0,82027
9	0,81499
10	0,83794

Dalam pemilihan model, na, nb, dan nk (*delay*) harus ditentukan terlebih dahulu untuk menetukan orde model. Pemilihan orde model didasarkan pada estimasi *error* yang dihasilkan setelah memasukkan berberapa orde model. Ada 3 orde model dalam proses ini yakni model orde 2, model orde 3, dan model orde 4.

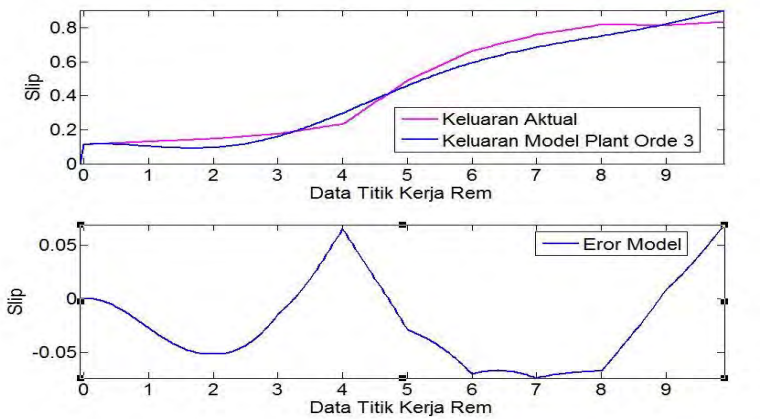
Gambar 3.14 berikut menunjukkan respon *plant* untuk estimasi model *output* dan estimasi model *error* orde 2 (dengan na= 2, nb=2, dan nk=1):



Gambar 3. 14 Estimasi *Output* dan Model *Error* Orde 2

Dari Gambar 3.14 respon aktual *plant* dengan respon *output* estimasi *plant* terlihat bahwa respon *output* estimasi tidak mendekati respon *output* aktual. Hal ini dibuktikan dengan estimasi *error* yang besar dibandingkan model dengan orde yang lebih tinggi.

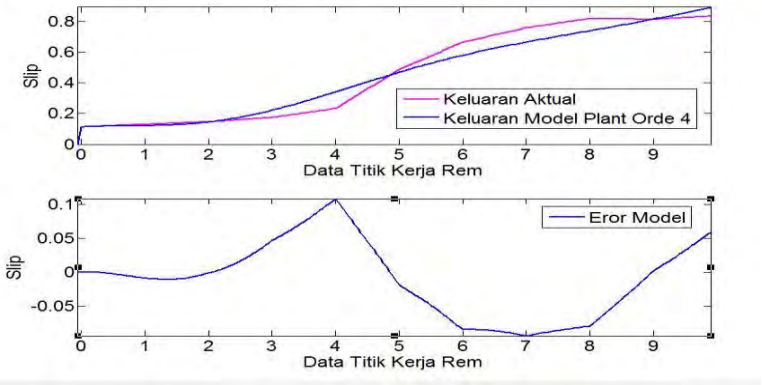
Gambar 3.15 berikut menunjukkan respon *plant* untuk estimasi model *output* dan estimasi model *error* orde 3 (dengan na =3, nb=1, dan nk=1):



Gambar 3. 15 Estimasi *Output* dan Model *Error* Orde 3

Dari Gambar 3.15 respon aktual *plant* dengan respon estimasi *plant* terlihat bahwa respon *output* estimasi mendekati respon *output plant* aktual. Nilai RMSE estimasi pada model *error* ini sebesar 6,64%.

Gambar 3.16 berikut menunjukkan respon *plant* untuk estimasi model *output* dan estimasi model *error* orde 4 (dengan na=4, nb=4, dan nk=1):



Gambar 3. 16 Estimasi *Output* dan Model *Error* Orde 4

Dari Gambar 3.16 respon estimasi *plant* terlihat mendekati dengan respon estimasi *plant* orde 3. Nilai RMSE estimasi model *error* yang dihasilkan sebesar 5,7% . Meskipun estimasi *error* yang dihasilkan lebih kecil bila dibandingkan estimasi *error* model orde 3, akan tetapi model orde 3 sudah merepresentasikan dinamika sebuah *plant* Oleh karena itu pendekatan model *plant* yang digunakan adalah model *plant* orde 3. Model orde 3 yang didapatkan merepresentasikan model *plant* ABS. Model *plant* yang didapatkan sudah dalam bentuk diskrit dengan domain z :

$$G(z) = \frac{0{,}0021184z^2}{z^3 - 1{,}9771z^2 + 0{,}97671z + 0{,}0024989} \qquad (3.10)$$

Tabel 3.6 berikut akan menujukkan perbandingan *root mean square error* (RMSE) dari model *plant* yang didapatkan:

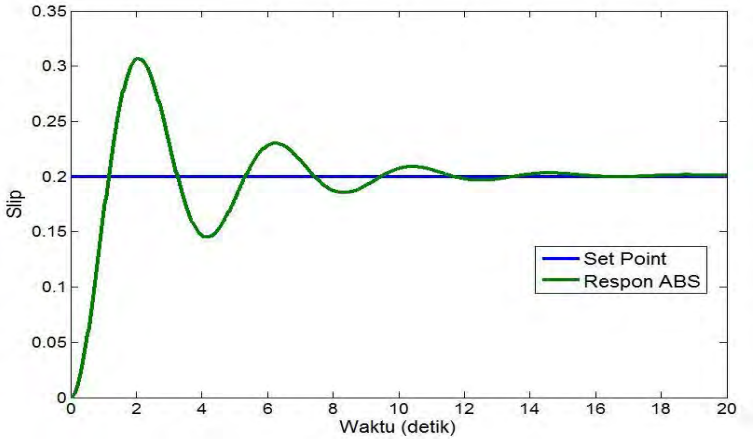
Tabel 3.6 Validitas Model *Plant*

Orde *Plant*	Model *Plant*	RMSE (%)
2	$\frac{0{,}55256z - 0{,}54807}{z^2 - 1{,}1956z + 0{,}199}$	10,4
3	$\frac{0{,}0021184z^2}{z^3 - 1{,}9771z^2 + 0{,}97671z + 0{,}0024989}$	6,64
4	$\frac{0{,}002185Z^3}{Z^4 - 1{,}9534Z^3 + 0{,}93149Z^2 + 0{,}021633Z}$	5,7

Dari data model *plant* pada Tabel 3.6 tersebut, pemilihan model *plant* didasarkan pada perubahan atau selisih nilai *RMSE* estimasi model *erro*r yang cukup signifikan, yakni antara Model *plan*t orde 3 dan orde 2. Perubahan nilai RMSE untuk orde 3 keatas semakin mengecil atau berkurang, akan tetapi perubahannya tidak signifikan sehingga Model *plan*t orde 3 sudah merepresentasikan dinamika dari sebuah *plant* ABS. Model *plant* yang didapat merupakan hubungan titik kerja slip terhadap urutan data titik kerja rem dalam domain *plane* z.

Setelah model *plant* didapat, yang selanjutnya adalah menguji model *plant* secara *open loop* tanpa kontroler untuk melihat karakteristik *transient* dari respon sistem. Sinyal uji yang diberikan adalah sinyal unit *step* sebesar 0,2. Nilai 0,2 adalah nilai slip yang diharapkan pada saat

koefisien gesek maksimal pada berberapa kondisi jalan. Pada Gambar 3.17 akan ditampilkan respon sistem dari hasil pengujian secara *loop* terbuka:



Gambar 3. 17 Respon Sistem ABS *Open Loop* Tanpa Kontroler

Dari respon sistem ABS pada Gambar 3.17 didapatkan data respon *transient* sistem dan respon *steady state* sistem yang akan ditampilkan pada Tabel 3.7:

Tabel 3.7 Data Respon *Transient* Sistem Tanpa Kontroler

Rise Time (detik)	*Settling Time* ±5% (detik)	Maksimum *Overshoot* (%)	*Error Steady State* (%)
1,1 7	16	50	0,75

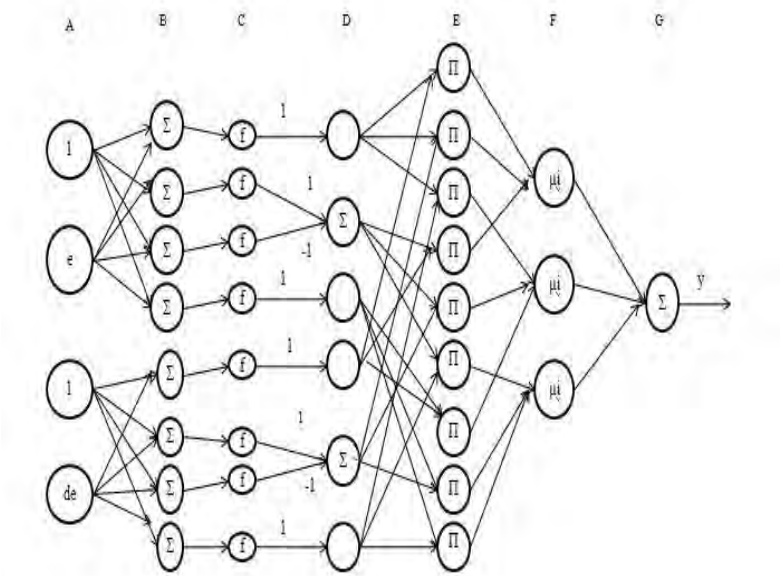
Berdasarkan data respon *transient* sistem pada Tabel 3.7 *error steady state plant* mendekati 0%, artinya kontroler yang dirancang harus mampu memberikan respon dengan *error steady state* yang lebih kecil. Respon sistem tanpa kontroler juga mempunyai *overshoot* yang besar yakni 50% oleh karena itu kontroler yang dirancang juga harus memberikan respon sesuai dengan keinginan yakni *overshoot* dibawah 10%. Waktu yang dibutuhkan respon sistem untuk mecapai *steady state* mencapai 16 detik sehingga kontroler yang dirancang juga harus mampu memberikan *settling time* 2 kali lebih cepat dari *settling time* respon sistem tanpa kontroler. *Rise time* dari sistem ABS kurang dari 2 detik,

artinya nilai slip mampu mencapai 0,2 dalam waktu 1,17 detik dalam sebuah proses pengereman.

3.5 Perancangan Kontroler *Neuro-Fuzzy* [11] [13] [14]

Algoritma *Neuro-fuzzy* adalah algoritma *learning* yang dibuat menggunakan jaringan syaraf tiruan untuk merealisasikan sistem inferensi *fuzzy*. Jaringan syaraf tiruan yang dibuat mengikuti mekanisme inferensi *fuzzy* dan algoritma *learning* yang digunakan adalah *backpropagation*. Algoritma *neuro-fuzzy* digunakan untuk mengatasi kelemahan dari sistem inferensi *fuzzy* dalam hal penentuan fungsi keanggotaan dan penentuan konsekuen pada aturan-aturan inferensi. Algoritma *backpropagation* mempunyai keunggulan sebagai salah satu algoritma *learning* yakni memiliki target pembelajaran sehingga bobot pada *hidden layer* akan mengalami revisi bobot pada setiap iterasinya. Algoritma *backpropagation* yang berbasis pada arsitektur jaringan syaraf tiruan bekerja seperti layaknya sebuah mesin, oleh karena itu dengan algoritma *neuro fuzzy* maka algoritma *backpropagation* bekerja layaknya sebuah pemikiran cerdas manusia dengan merealisasikan sistem inferensi *fuzzy*.

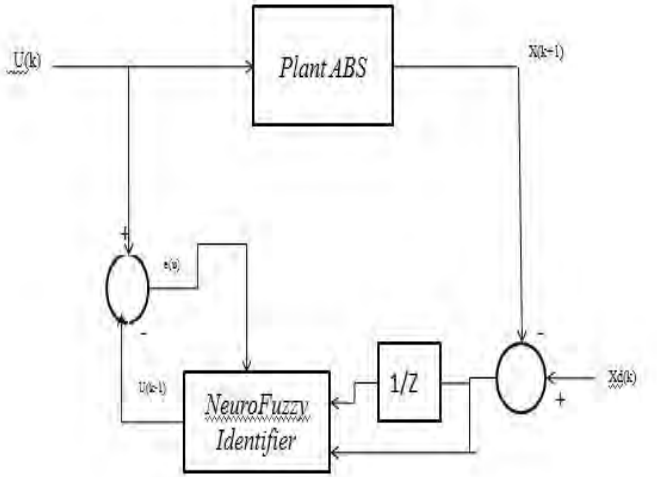
Algoritma *neuro-fuzzy* yang digunakan adalah algoritma *fuzzy modelling network* (FMN). FMN memiliki tiga tipe yang berbeda yakni tipe I, tipe II, dan tipe III. FMN ini akan mengidentifikasi aturan-aturan *fuzzy* dan fungsi keanggotaan secara otomatis dengan cara memodifikasi bobot-bobot jaringan syaraf melalui algoritma *backpropagation.* Pada FMN tipe I digunakan sistem inferensi madani, dimana Konsekuen dari sistem inferensi *fuzzy* berupa fungsi keanggotaan. Pada FMN Tipe II digunakan sistem inferensi madani, dimana konsekuen dari sistem inferensi *fuzzy* berupa nilai tunggal (konstanta). Pada FMN tipe III menggunakan sistem inferensi *fuzzy sugeno*, dimana konsekuen dari sistem inferensi *fuzzy* menggunakan persamaan linier. Pada perancangan kontroller menggunakan FMN tipe II untuk mengatur nilai slip pada *plant* ABS. Perhtungan algoritma *neuro-fuzzy* dibagi menjadi dua tahap yakni tahap *forward propagation* dan tahap *backpropagation*. Gambar 3.18 berikut merupakan arsitektur dari jaringan *neuro-fuzzy* FMN tipe II:



Gambar 3. 18 Jaringan *Neuro-Fuzzy* [11]

Berdasarkan dasar teori pada bab 2 mengenai *general learning*, dalam perancangan kontroler *neuro-fuzzy* dibagi menjadi 2 tahap yakni tahap *training* data *offline* menggunakan *general learning* dan tahap aplikasi atau tahap *online*. Dari model *neuro-fuzzy* yang sudah dirancang, *general learning* merepresentasikan dinamika *invers* dari sebuah *plant.*

Pada tahapan *offline*, dilakukan *training* data untuk mendapatkan bobot minimum yang bersesuaian pada *arc* jaringan *neuro-fuzzy* dengan menetapkan target minimum *error* dan jumlah iterasi maksimum, selama nilai *error* masih lebih besar dari target minimum *error* maka iterasi masih terus berlangsung. Gambar 3.19 berikut merupakan diagram blok dari *training offline* menggunakan *general learning*:



Gambar 3. 19 Diagram Blok *General Learning*

Berdasarkan Gambar 3.19 sinyal target merupakan selisih antara sinyal kontrol (u(k)) dengan sinyal kontrol estimasi (U(k)). Untuk fungsi pembelajaran nilai sinyal kontrol dapat berupa nilai skalar sehingga nilai sinyal kontrol (u(k)) adalah nilai sinyal kontrol saat nilai slip yang diharapkan (set point). Berdasarkan model *plant* pada Gambar 3.12 didapatkan nilai sinyal kontrol pada persamaan berikut:

$X_d(\mathrm{k}) = 0{,}2$ (3.11)

$u(k) = 0{,}345$ (3.12)

Sinyal pelatihan (*training*) data *offline neuro-fuzzy* dilakukan dengan jumlah iterasi maksimum (epoh maksimum) 100 dan target *error* minimum 10^{-6}. Pada tahapan *training* data, perhitungan *neuro-fuzzy* terdiri dari *forward propagation* dan *backpropagation.* Nilai *error* pada setiap iterasi yang terdiri dari proses *forward propagation* dan *backpropagation* akan di kuadratkan kemudian di akarkan. Nilai *error* setiap iterasi akan selalu dibandingkan dengan target *error* minimum hingga nilai *error* lebih kecil dari target *error* minimum.

3.5.1. Perancangan Tahap *Forward Propagation Neuro-Fuzzy* [14]

Struktur jaringan FMN tipe II seperti yang dapat dilihat pada Gambar 3.19 terdiri dari 7 lapisan. Tujuh lapisan jaringan tersebut antara lain satu lapisan *input*, lima lapisan tersembunyi, dan satu lapisan *output*. Pada bagian ini akan dijelaskan salah satu tahapan dari perhtungan algoritma *neuro fuzzy* yakni tahap *forward propagation*. Tahapan *forward propagation* bekerja dari lapisan A (lapisan *input*) hingga lapisan G (lapisan *output*). Pada Subbab ini juga akan dijelaskan perhitungan *forward propagation* setiap lapisan jaringan.

a. Lapisan A

Lapisan A merupakan lapisan *input* dari struktur jaringan FMN tipe II. Lapisan ini memiliki empat buah *node* yang akan meneruskan sinyal *error* dan sinyal delta *error*. Bobot dari link ini adalah 1.

$$O_{11} = 1\,;\ O_{2,1} = X_1\,;\ O_{3,1} = 1\,;\ O_{4,1} = X_2 \tag{3.13}$$

Nilai X_1 adalah nilai dari sinyal *error* yang masuk ke kontroler dan nilai X_2 adalah nilai dari sinyal delta *error* yang masuk ke kontroler

b. Lapisan B

Lapisan ini terdiri dari delapan *node* yang merupakan hasil penjumlahan dari nilai *input* yang dikali dengan bobot W_c. Empat *node* awal atau *node* pada bagian atas pada jaringan FMN tipe II merupakan kelompok untuk *input* X_1 dan empat *node* lainnya merupakan kelompok untuk *input* X_2.

$$O_{i,2} = 1(W_c) + X_1 \tag{3.14}$$

$$\mathrm{O_{j,2}} = 1(\mathrm{w_c}) + \mathrm{X_2} \tag{3.15}$$

Untuk i= 1,2,3,4 dan j = 5,6,7,8

Inisialisasi bobot awal dilakukan secara acak pada setiap bobot W_c dengan nilai interval 0 sampai 1 [0,1]. Nilai bobot Wc pertama dibuat positif. Bobot W_c yang kedua, ketiga dan keempat dibuat bernilai negatif untuk mengimbangi nilai bobot yang pertama. Hal ini dikarenakan jika nilai bobot terlalu besar

maka turunan fungsi sigmoidnya akan sangat kecil Nilai bobot W_c yang ketiga dan keempat dibuat dengan inisialisasi sama dengan tujuan menghemat elemen *array* bobot sehingga membantu untuk meningkatkan kecepatan komputasi.

c. Lapisan C

Lapisan ini merupakan fungsi aktivasi dari *output* lapisan B dikali dengan bobot W_g. Bobot $W_g$menunjukkan gradien dari fungsi sigmoid.Bobot awal W_g diinisialisasi dengan fungsi acak dengan nilai interval 0 sampai 1 [0,1]. Fungsi aktivasi yang digunakan pada lapisan ini adalah fungsi aktivasi sigmoid biner. Fungsi aktivasi sigmoid biner digunakan untuk merepresentasikan fungsi keanggotaan. Lapisan ini memiliki delapan buah *node* yang terbagi menjadi dua kelompok. Empat *node* awal merupakan kelompok untuk *input* X_1 dan empat *node* lainnya merupakan kelompok untuk *input* X_2.

$$O_{i,3} = \frac{1}{1 + e^{-wg(O_{i,2})}} \tag{3.16}$$

$$O_{j,3} = \frac{1}{1 + e^{-wg(O_{j,2})}} \tag{3.17}$$

d. Lapisan D

Lapisan ini akan menghasilkan fungsi keanggotaan dari nilai input X_1 dan X_2. Lapisan ini terdiri dari enam *node*, dimana tiga *node* pertama menunjukkan fungsi keanggotaan dari *input* X_1 dan tiga *node* lainnya menunjukkan fungsi keanggotaan dari *input* X_2. Fungsi keanggotaan dari masing-masing *input* dibagi menjadi tiga, yaitu negatif, zero, dan positif.

$$O_{1,4} = O_{i,3} \tag{3.18}$$

$$O_{2,4} = O_{2,3} + (-1)O_{3,3} \tag{3.19}$$

$$O_{3,4} = O_{4,3} \tag{3.20}$$

e. Lapisan E

Lapisan ini terdiri dari Sembilan *node* yang merupakan kombinasi dari aturan inferensi *fuzzy*. Kontroler *neuro-fuzzy* yang dirancang memiliki dua buah *input* (*error* dan delta *error*) dan memiliki 9 basis aturan *fuzzy* dalam bentuk *IF-THEN*. Masing-masing *node* pada lapisan ini akan memiliki dua buah *input* yang berasal dari lapisan sebelumnya. Setiap *node* pada lapisan ini akan melakukan operasi *AND* terhadap kedua *input* tersebut. Operasi yang digunakan pada kontroller NF ini adalah *algebraic product*. Persamaan untuk fungsi operasi tersebut dinyatakan berikut ini:

$$O_{i,5} = O_{m,4} \; X \; O_{n,4} \tag{3.21}$$

Untuk I = 1,2,3,…9 ; m= 1,2,3 ; n= 1,2,3

Kombinasi aturan inferensi *fuzzy* dinyatakan dalam bentuk *IF-THEN*.

f. Lapisan F

Lapisan ini merupakan bagian konsekuen dari sistem inferensi *fuzzy*.

$$O_{i,6} = \sum_{m=1}^{9} O_{m,5} \tag{3.22}$$

Untuk i= 1,2,3

Basis aturan *fuzzy* yang dibuat pada kontroller NF ini direpresentasikan pada Tabel 3.8 berikut:

Tabel 3.8 Basis Aturan Kontroler *Neuro-Fuzzy*

U		Delta *Error*		
		N	Z	P
Error	N	N	N	Z
	Z	N	Z	P
	P	Z	P	P

g. Lapisan G

Lapisan ini merupakan lapisan *output* pada kontroller NF. Pada lapisan ini dilakukan proses defuzzifikasi dengan cara menjumlahkan hasil perkalian dari bobot dengan lapisan sebelumnya yang sudah dinormalisasi. *Output* pada lapisan ini berupa sinyal kontrol tegas.Inisialisasi Bobot awal Wf berupa array 3×1 secara acak. Bobot Wf dalam program ditulis dengan variabel Wout.

$$O_{1,7} = \sum_{i=1}^{3} O_{i,6} \; X \; W_f \qquad (3.23)$$

3.5.2 Perancangan Tahap *Back Propagation Neuro-Fuzzy* [14]

Tahapan kedua dari algoritma *neuro-fuzzy* adalah tahap *back propagation*. Tahap ini bertujuan untuk merevisi nilai bobot pada jaringan *neuro-fuzzy* (W_c, W_g, W_f) agar nilai *output* sesuai dengan target pembelajaran atau target yang diinginkan. Pada subbab ini akan dibahas perhitungan perambatan *error* dan perbaikan bobot pada masing-masing lapisan. Nilai *error* diperoleh dengan menghitung selisih antara target pembelajaran dengan *output* pembelajaran (sinyal kontrol tegas).

a. Lapisan G

Pada lapisan ini terdapat bobot W_f yang diperbaiki untuk memperoleh nilai *output* sesuai dengan target pembelajaran atau target yang diinginkan. Perambatan *error* juga dihitung pada lapisan ini. Nilai *error* pada lapisan ini proporsional terhadap bobot yang ada. Perhitungan bobot W_f dapat ditulis dalam persamaan berikut:

$$W_f(t+1) = W_f(t) + \alpha \left(T_i(t) - O_{i,7}(t) \right) O_{i,6}(t) \qquad (3.24)$$

$$\&_{i,7}(t) = \frac{O_{i,6}}{O_{i,7}} \; (T_i(t) - O_{i,7}(t) \; \text{..(perambatan } error) \qquad (3.25)$$

α merupakan *learning rate* dari *arc* pada lapisan G menuju lapisan F. Nilai dari *learning rate* ini adalah sebesar 0,01.

b. Lapisan F

Pada *arc* lapisan ini tidak ada perubahan bobot, akan tetapi tetap dihitung perambatan *error* dari lapisan F menuju lapisan E.

Berdasarkan struktur kontroler yang dibuat, didapatkan bahwa nilai *error* dari lapisan G menuju lapisan F merupakan penjumlahan dari *error* lapisan F menuju *error* lapisan E. Bobot pada lapisan E menuju lapisan F benilai 1 sehingga perhitungan perambatan *error* dapat ditulis dalam persamaan berikut:

$$\&_{i,6}(t) = \frac{1}{3}\ \&_{i,7}(t) \tag{3.26}$$

c. Lapisan E

Pada *arc* lapisan ini tidak ada perubahan bobot, akan tetapi tetap dihitung perambatan *error* dari lapisan E menuju lapisan D. Berdasarkan struktur kontroler yang dibuat, didapatkan bahwa *output* lapisan E merupakan perkalian dari *output* lapisan D. Perbandingan *error* sama dengan perbandingan *output* pada lapisan D dan lapisan E, sehingga perambatan *error* dapat ditulis sebagai berikut:

$$\&_{i,} = \sqrt{\&_{i,6}}(t) \tag{3.27}$$

d. Lapisan D

Pada *arc* lapisan ini tidak ada perubahan bobot, akan tetapi tetap dihitung perambatan *error* dari lapisan D menuju lapisan C. Berdasarkan struktur kontroler yang dibuat, didapatkan bahwa *output* lapisan C diteruskan ke lapisan D tanpa ada operasi matematika. Akan tetapi pada *node* kedua dan *node* keempat pada lapisan C terdapat operasi penjumlahan sehingga perambatan *error* dapat ditulis sebagai berikut:

$$\&_{i,4}(t) = \ \&_{i,5}(t) \tag{3.28}$$

Perambatan *error* pada *node* kedua dan *node* keempat pada lapisan C proporsional karena berdasarkan struktur kontroler yang dibuat, didapatkan bahwa *output* lapisan C menuju ke lapisan D. Pada *node* kedua dan keempat lapisan C mempunyai bobot yang simetris, yakni 1 dan -1 sehingga perambatan *error* dapat ditulis sebagai berikut:

$$\&_{i,4}\ (t) = \frac{1}{2}\ \&_{i,5}(t);\ \ untuk\ i = 2,3,6,7 \tag{3.29}$$

e. Lapisan C

Pada *arc* lapisan ini terdapat bobot W_g yang diperbaiki untuk memperoleh nilai *output* sesuai target pembelajaran atau target yang diinginkan. Perambatan *error* juga dihitung dalam lapisan ini. Berdasarkan struktur kontroler yang dibuat, didapatkan bahwa pada lapisan ini menggunakan fungsi aktivasi sigmoid biner. Sehingga perhitungan bobot W_g dapat ditulis sebagai berikut:

$$f(x) = \frac{1}{1 + e^{-W_g(X_i + W_c)}} \tag{3.30}$$

$$f'(x) = \left(\frac{1}{1+e^{-W_g(x_i+W_c)}}\right) \left(1 - \frac{1}{1+e^{-W_g(x_i+W_c)}}\right) \tag{3.31}$$

$$W_g(t+1) = W_g(t) + \alpha \, \&_{i,4} \, f'(x)(X_i + W_c) \tag{3.32}$$

$$\&_{i,3}(t) = \alpha(\&_{i,4}\,(t)) \tag{3.33}$$

α merupakan *learning rat*e dari arc pada lapisan C menuju lapisan B. Nilai dari *learning rate* ini adalah sebesar 0,01.

f. Lapisan B

Pada *arc* lapisan ini terdapat bobot W_c yang ingin diperbaiki untuk memperoleh nilai *output* sesuai target pembelajaran atau target yang diinginkan. Perambatan *error* juga dihitung dalam lapisan ini. Pada lapisan ini digunakan fungsi aktivasi sigmoid sehingga perhitungan bobot W_c dapat ditulis sebagai berikut:

$$W_c\,(t+1) = W_c\,(t) + \alpha(\,\&_{i,3}(t) f'(\mathrm{x})) \tag{3.34}$$

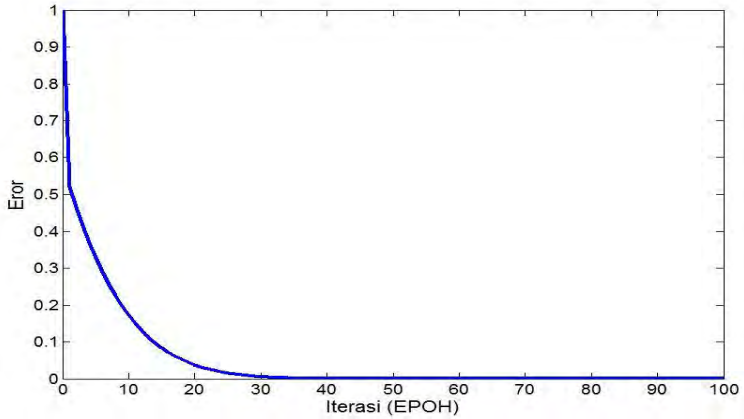
$$\&_{i,2}(t) = \alpha\,(\&_{i,3}(t)) \tag{3.35}$$

α merupakan *learning rat*e dari *arc* pada lapisan B menuju lapisan A. Nilai dari *learning rate* ini adalah sebesar 0,01.

g. Lapisan A

Tidak ada parameter yang perlu diubah pada lapisan ini. Hal ini dikarenakan lapisan ini hanya sebagai penerus dari *input*.

Setelah dilakukan pelatihan data secara *offline* dengan tahapan perhitungan *forward propagation* dan *back propagation* dengan parameter target *error* 10^{-6}; *learning rate* 0,01 ; dan maksimum iterasi (epoh maksimum) sebesar 100, maka didapatkan hubungan grafik antara nilai *error* tiap iterasi (epoh) seperti yang ditunjukkan pada Gambar 3.20 berikut:



Gambar 3. 20 Hasil Pelatihan

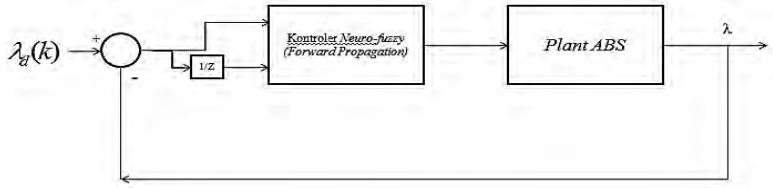
3.5.3 Perancangan Tahap *Online Neuro-Fuzzy*

Tahapan setelah pelatihan data *offline* adalah tahapan simulasi *online*. Bobot yang bersesuaian akan didapatkan setelah proses pelatihan selesai dengan parameter *error* minimum sebesar 10^{-6} dan jumlah iterasi maksimum sebesar 100. *Training* selesai ketika nilai *error* mencapai nilai 0,000005094, padahal nilai ini masih berada diatas 10^{-6}. Akan tetapi pada nilai ini, nilai *error* relatif tidak berubah dan tidak bisa melewati nilai 10^{-6}. Jadi iterasi berhenti pada nilai *error* 0,000005094 pada iterasi (epoh) ke 100. Jadi pada epoh ke 100 didapatkan bobot-bobot yang bersesuaian dengan *arc* jaringan *neuro-fuzzy* yang akan dijadikan bobot tetap saat tahapan simulasi *online*. Tabel 3.9 berikut merupakan bobot hasil pelatihan:

Tabel 3.9 Bobot Hasil Pelatihan *Offline*

Bobot	Nilai Bobot
Wdcde1	0,0957
Wcde2	0,4736
Wcek1	0,5462
Wcek2	0,7070
Wgde	0,0468
Wgek	0,4276
Wout	[0,4895; 0,8396 ; 0,5820]

Setelah bobot dari *arc* jaringan *neuro-fuzzy* didapatkan maka langkah selanjutnya adalah memasukkan nilai bobot tersebut ke dalam algoritma *neuro-fuzzy* tanpa tahapan perhitungan *backpropagation*. Tahapan perhitungan yang digunakan adalah hanya tahapan *forward propagation*. Gambar 3.21 berikut merupakan diagram blok dari tahapan *online* atau tahapan aplikasi:



Gambar 3. 21 Blok Simulink Tahapan *Online*

Hasil dari tahapan *online* adalah respon model *plant* menggunakan kontroler *neuro-fuzzy*. Pada tahapan *online* ini, kontroler sudah tidak membutuhkan waktu pembelajaran saat mengolah sinyal *error* dan delta *error*

BAB 4
PENGUJIAN DAN ANALISIS

Pengujian dan analisis mencakup pengujian model *plant* menggunakan kontroler *neuro-fuzzy* secara simulasi, analisis dari hasil simulasi, dan analisis dari hasil implementasi pada *real plant* ABS menggunakan kontroler *neuro-fuzzy*.

4.1 Gambaran Umum Pengujian Sistem

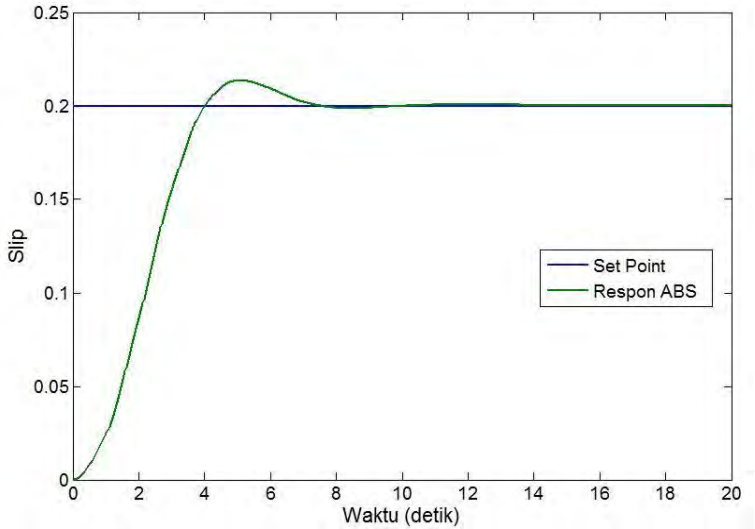
Pada Tugas Akhir ini dilakukan pengujian sistem melalui simulasi dan implementasi. Pengujian Sistem dilakukan untuk mengetahui performansi kontroler yang sudah dirancang dalam menyelesaikan masalah pada sistem. Dalam pengujian berupa simulasi, terdapat aspek pengujian yang diberikan terhadap sistem, yakni simulasi pengujian model *plant* secara *loop* tertutup menggunakan kontroler *neuro-fuzzy* terhadap *set point* berupa sinyal *step* dan membandingkan respon *transient plant* dengan kontroler PID.

4.2 Simulasi Sistem

Simulasi Sistem dilakukan melalui perangkat lunak matlab. Simulasi sistem bertujuan untuk mengetahui data respon *transient* sistem dan respon *steady state* sistem seperti *rise time* (tr), *settling time* (ts) dan Maksimum *Overshoot*, dan *error steady state* (ess).

4.2.1 Pengujian Model *Plant* Menggunakan Kontroler *Neuro-Fuzzy* Terhadap *Set Point* Berupa Sinyal *Step*

Simulasi dilakukan dengan melakukan pengujian model *plant* secara *loop* tertutup terhadap *set point* berupa sinyal *step.* Seperti yang dijelaskan pada Bab 3 bahwa perancangan kontroler *neuro-fuzzy* dibagi menjadi 2 tahapan yakni tahapan training data *offline* dan tahapan *online*. Pengujian model *plant* yang akan dianalisis adalah saat tahapan *online* yakni bobot-bobot dari hasil training *offline* sudah dimasukkan pada algoritma *neuro-fuzzy* tanpa perhitungan *backpropagation.* Sinyal uji yang diberikan adalah sinyal unit *step* sebesar 0,2 yang merepresentasikan nilai slip yang diharapkan. Pada Gambar 4.1 akan ditampilkan respon sistem menggunakan kontroler *neuro-fuzzy*:



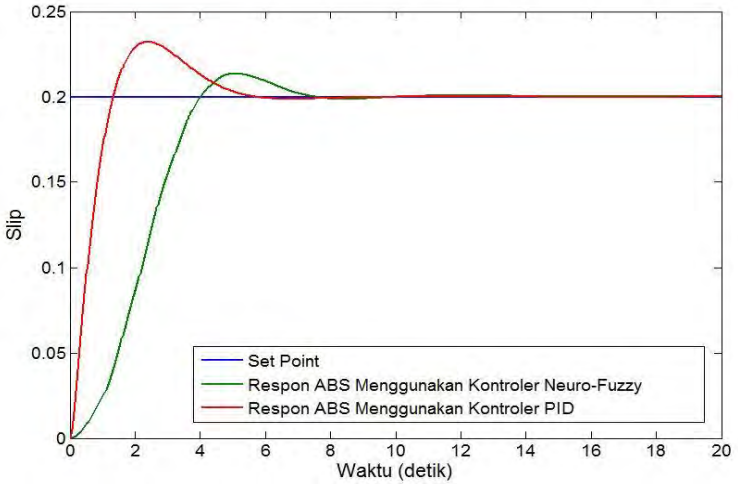
Gambar 4.1 Respon Sistem ABS dengan Kontroler *Neuro-Fuzzy*

Dari respon sistem pada Gambar 4.1 didapatkan data respon *transient* sistem dan respon *steady state* sistem yang akan ditampilkan pada Tabel 4.1:

Tabel 4.1 Data Respon *Transient* Menggunakan Kontroler *Neuro-Fuzzy*

Rise Time (detik)	*Settling Time* ±5% (detik)	Maksimum *Overshoot* (%)	*Error Steady State* (%)
4	8	6,8	0

Respon sistem dengan kontroler *neuro-fuzzy* yang sudah didapat akan dibandingkan dengan respon sistem dengan kontroler PID sebagai pembanding. Kontroler PID yang dirancang berdasarkan mekanisme *trial-error* (coba-coba). Mekanisme coba-coba tersebut dilakukan dengan melakukan penalaan pada parameter *Kp*, *Ki*, dan *Kd*. Pada langkah awal, penalaan dilakukan dengan mengubah parameter *Kp* menjadi lebih besar, kemudian mengubah parameter *Ki* dan parameter *Kd*. Gambar 4.2 berikut merupakan perbandingan respon sistem menggunakan kontroler PID dan kontroler *Neuro-Fuzzy* dengan salah satu hasil penalaan, yakni nilai *Kp* diatur sebesar 5, nilai *Ki* diatur sebesar 4, dan nilai *Kd* diatur sebesar 4:



Gambar 4.2 Perbandingan Respon Sistem ABS dengan Kontroler PID dan Kontroler *Neuro-Fuzzy*

Spesifikasi Respon pada Tabel 4.2 akan menguraikan hasil penalaan parameter *Kp, Ki,* dan *Kd* yang sudah dilakukan terhadap respon *transient* sistem:

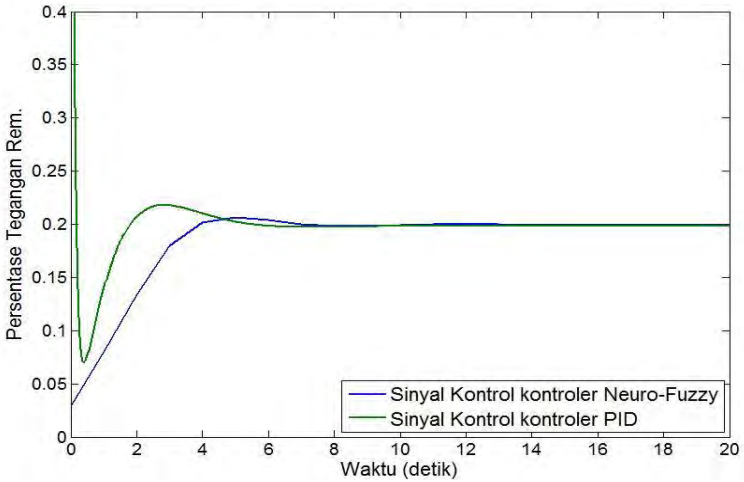
Tabel 4.2 Hasil Penalaan Parameter *Kp,Ki,* dan *Kd*

Kp	*Ki*	*Kd*	*Rise Time* (detik)	*Settling Time* ±5% (detik)	Maksimum *Overshoot* (%)	*Error Steady State* (%)
5	3	4	1,56	8	11,25	0
5	4	4	1,35	8,5	16,1	0
5	4	3	1,11	5,76	13,15	0
5	3	4	1,29	8,25	8,25	0
5	3	1	0,6	6,2	23,5	0

Berdasarkan Tabel 4.2 rata-rata waktu *rise time* kontroler PID jauh lebih baik bila dibandingkan dengan waktu *rise time* kontroler *neuro-fuzzy* artinya kontroler *neuro-fuzzy* tidak mampu memberikan waktu *rise time* yang lebih baik dibandingkan dengan kontroler PID. Kontroler *neuro-fuzzy* mampu memberikan nilai *settling time* 2 kali lebih terhadap

respon sistem tanpa kontroler. Kontroler *neuro-fuzzy* memiliki nilai *settling time* yang tidak jauh berbeda dengan berberapa hasil penalaan parameter kontroler PID, artinya kontroler *neuro-fuzzy* secara umum mampu memberikan nilai *settling time* lebih baik bila dibandingkan dengan kontroler PID. Kontroler *neuro-fuzzy* mampu mengurangi *overshoot* pada respon sistem tanpa kontroler sebesar 43,2 %, artinya kontroler *neuro-fuzzy* mampu untuk mengurangi *overshoot* pada respon sistem secara signifikan. Kontroler *neuro-fuzzy* juga memiliki nilai *overshoot* yang lebih kecil bila dibandingkan kontroler PID. Kontroler *neuro-fuzzy* dan PID dapat memperbaiki sistem karena memiliki *error steady state* mendekati 0%. Pengaruh respon *transient* sistem ABS menggunakan kontroler *neuro-fuzzy* dipengaruhi oleh nilai awal pembobotan dan nilai minimum *error* yang dicapai tidak mencapai minimum global melainkan hanya minimum lokal.

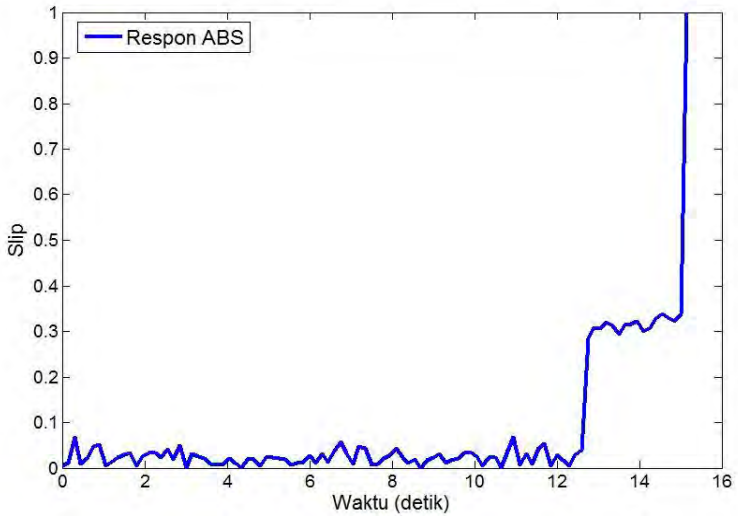
Analisa Perbandingan sinyal kontrol untuk kontroler *neuro-fuzzy* dan kontroler PID perlu dilakukan untuk mengetahui apakah sinyal kontrol yang dikeluarkan kontroler tidak melebihi *range* kerja dari rem elektromagnetik yakni dari persentase 0% hingga 100%. Gambar 4.3 berikut merupakan perbandingan sinyal kontrol antara kontroler *neuro-fuzzy* dan kontroler PID:



Gambar 4.3 Perbandingan Sinyal Kontrol *Neuro-Fuzzy* dan PID

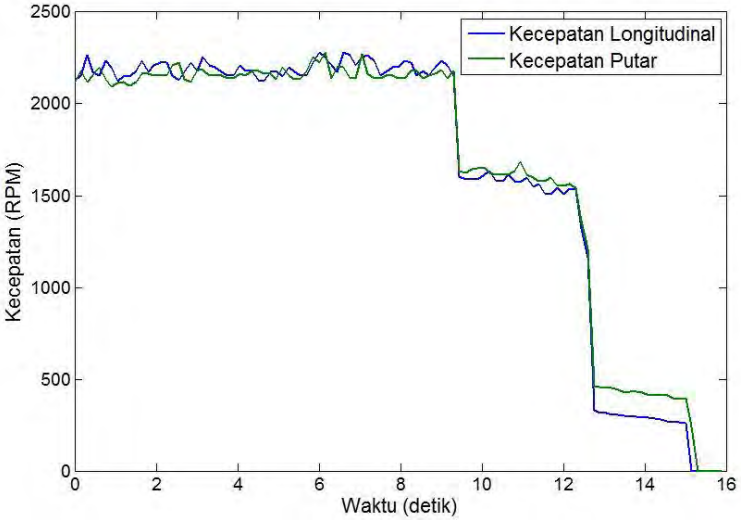
4.3 Implementasi Sistem

Implementasi dilakukan melalui perangkat lunak matlab yang dihubungkan dengan *real plant* ABS. Implementasi sistem bertujuan untuk mengetahui data respon *transient* sistem dan respon *error steady state* saat dihubungkan dengan *real plant* ABS dan membandingkan hasil respon sistem yang didapat dengan hasil respon sistem saat simulasi. Implementasi dilakukan menggunakan dua arduino untuk mengeset kecepatan putar roda bawah yang terhubung dengan motor DC dan sebagai *interface* kontroler dengan *plant*. Program pada arduino dibuat agar dapat berkomunikasi serial secara *half duplex* dengan matlab agar data kecepatan (kecepatan *longitudinal* dan kecepatan putar) dan sinyal kontrol dapat dilewatkan secara bergantian oleh arduino. Hal ini dikarenakan *port* serial yang digunakan untuk berkomunikasi antara PC dan arduino sebagai *interface* hanya 1 *port* (COM) sehingga data yang masuk ke kontroler (data kecepatan *longitudinal* dan putar) dan sinyal kontrol diatur agar tidak tabrakan. Gambar 4.4 berikut merupakan hasil implementasi sistem untuk kecepatan putar roda bawah (kecepatan *longitudinal*) sebesar 2300 rpm (100%):



***Gambar* 4.4** Respon Sistem ABS Hasil Implementasi pada Kecepatan 2300 rpm

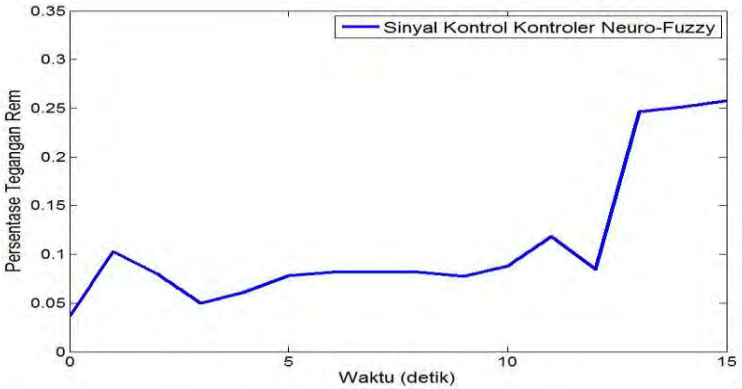
Berdasarkan respon sistem ABS pada Gambar 4.4 menunjukkan bahwa kontroler *neuro-fuzzy* dapat bekerja pada saat dilakukan pengereman. Pada saat pengereman nilai slip tidak dapat mengikuti nilai sinyal referensi sebesar 0,2 tetapi dapat menjaga roda atas tidak mengunci atau menjaga agar slip tidak bernilai 1. Nilai slip yang dihasilkan masih dalam *range* yang diperbolehkan yakni antara 0,15-0,3 [15]. Respon sistem pada Gambar 4.3 memiliki nilai *rise time* sebesar 0,2 detik dan *error steady state* sebesar 50%. Respon Sistem pada saat roda atas dan roda bawah sudah berhenti menuju ke nilai slip 1 (batas atas saturasi) karena pengaruh dari rumus slip yang bernilai 0/0 saat roda atas dan roda bawah sudah berhenti. Saat dilakukan pengereman ada perubahan kecepatan *longitudinal* dan kecepatan putar sebagai efek dari nilai slip yang dapat dikendalikan. Berikut merupakan respon kecepatan pada saat kontroler *neuro-fuzzy* bekerja, yang akan ditampilkan pada Gambar 4.5 berikut ini :



Gambar 4.5 Respon Kecepatan *Longitudinal* dan Kecepatan Putar

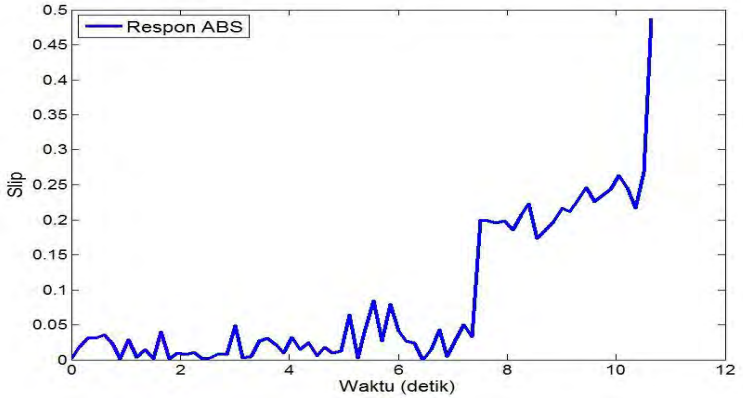
Berdasarkan Gambar 4.5 kecepatan *longitudinal* dan kecepatan putar memiliki selisih nilai yang kecil saat pengereman dilakukan. Penurunan kecepatan *longitudinal* dan kecepatan putar juga memiliki *slope* yang lebih landai dibandingkan penurunan kecepatan tanpa sistem ABS. Sinyak kontrol yang dikeluarkan kontroler masih berada dalam

range kerja dari rem elektromagnetik yakni sebesar 0%-100%. Gambar 4.6 berikut merupakan sinyal kontrol untuk kecepatan roda bawah sebesar 2300 rpm:



Gambar 4.6 Sinyal Kontrol Kontroler *Neuro-Fuzzy*

Selain pada kecepatan 2300 rpm, implementasi juga dilakukan pada kecepatan 2200 rpm. Pada kecepatan 2200 rpm, nilai slip yang dihasilkan dapat mengikuti sinyal referensi sebesar 0,2. Gambar 4.7 berikut merupakan respon sistem ABS pada *real plant* simulator ABS pada kecepatan 2200 rpm:



Gambar 4.7 Respon Sistem ABS Hasil Implementasi pada Kecepatan 2200 rpm

Berdasarkan Gambar 4.7 Respon sistem ABS dapat mengikuti sinyal referensi sebesar 0,2 pada saat dilakukan pengereman. Akan tetapi mempunyai nilai *error steady state* (ess) yang kecenderungannya naik hingga roda atas dan roda bawah berhenti. Kontroler *Neuro-Fuzzy* dapat bekerja untuk kecepatan 2200 rpm karena dapat menjaga roda atas tidak terkunci saat dilakukan pengereman. Respon sistem bernilai 1 saat roda atas dan roda bawah sudah dalam keadaan berhenti, hal ini dikarenakan pengaruh dari rumus slip yang bernilai 0/0 saat roda atas dan roda bawah berhenti.

BAB 5
PENUTUP

Bab penutup mencakup kesimpulan dan saran. Penarikan kesimpulan berdasarkan pengujian model *plant* secara simulasi dan hasil implementasi pada *real plan*t ABS. Saran yang dikemukakan mencakup saran terhadap rancangan mekanik *plant* dan komunikasi data antara matlab pada PC dan arduino sebagai *interface*.

5.1 Kesimpulan

Berdasarkan hasil pengujian model *plant* menggunakan kontroler *neuro-fuzzy* secara simulasi dan hasil implementasi pada *real plant* ABS menggunakan kontroler *neuro-fuzzy* dapat disimpulkan menjadi berberapa point sebagai berikut:

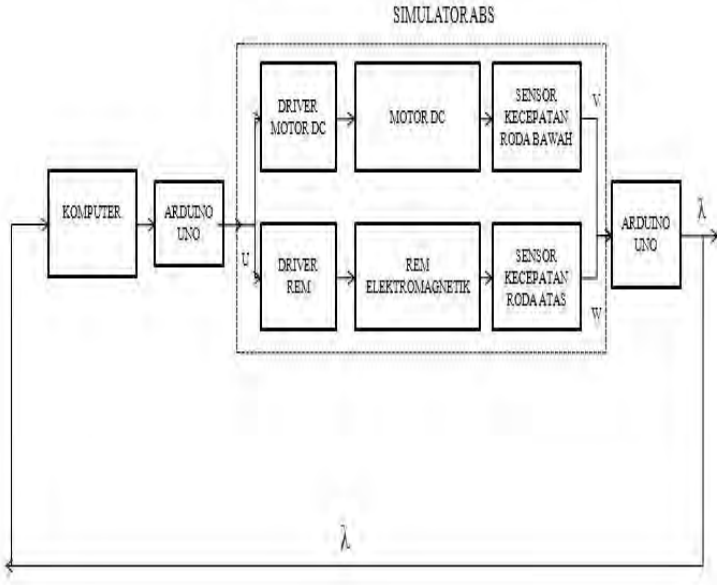
- Kontroler *neuro-fuzzy* mampu mengurangi *overshoot* respon sistem sebesar 43,2% dan mempercepat waktu respon sistem dua kali lebih cepat untuk mencapai kondisi stabil.
- Kontroler *neuro-fuzzy* yang dirancang mampu bekerja saat implementasi untuk kecepatan roda bawah 2200 rpm dan 2300 rpm.
- Untuk kecepatan roda bawah 2300 rpm, respon sistem dengan kontroler *neuro-fuzzy* mempunyai nilai *error steady state* (ess) sebesar 50%.
- Untuk kecepatan 2200 rpm mampu untuk mengikuti sinyal referensi sebesar 0,2 selama $\pm$ 0,5 detik saat awal pengereman.

5.2 Saran

Perancangan mekanik untuk jarak antara celah rem elektromagnetik dan lempengan besi pada piringan *encoder* mempengaruhi besarnya rem sehingga perlu pemasangan yang presisi agar rem dapat bekerja sesuai dengan besarnya sinyal kontrol keluaran kontroler. Komunikasi *Half-duplex* antara Arduino dengan matlab pada PC bekerja dengan basis *sample time*, sehingga diperlukan *sample time* yang sesuai dengan kebutuhan kontroler agar tidak terjadi antrian data kecepatan yang masuk ke kontroler.

--Halaman ini sengaja dikosongkan--

LAMPIRAN A
SKEMA ALAT



--Halaman ini sengaja dikosongkan--

LAMPIRAN B
PROGRAM YANG DIGUNAKAN

a. Program Pwm Motor dan Rem Elektromagnetik

```
int pinrem= 4 ;
int motor = 10 ;
int rem= 9 ;
void setup() {
  // put your setup code here, to run once:
   //pinMode(pinrem,INPUT)
   pinMode(motor,OUTPUT);
  // pinMode(rem,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
 // program utama pwm motor dan rem elektromagnetik
  //if ( digitalRead(pinrem) == HIGH)
  {
     // mengeset nilai pwm rem dengan perintah delayMicroseconds
      //digitalWrite(rem,HIGH);
      //delayMicroseconds(1998);
      //digitalWrite(rem,LOW);
      //delayMicroseconds(2);

  }
  //else
  {

    digitalWrite(motor,HIGH);
    delayMicroseconds(1000);
    digitalWrite(motor,LOW);
    delayMicroseconds(1000);
  }
}
```

b. Program Konversi ke Slip dari Sensor kecepatan

```
 int encoder1  = 4 ;
 int encoder2  = 8 ;
void setup() {
  // put your setup code here, to run once:
   Serial.begin(9600);
   pinMode(encoder1,INPUT);  //longitudinal
   pinMode(encoder2,INPUT);  //putar
   pinMode(13, OUTPUT);
}
void loop() {
    digitalWrite(13, digitalRead(encoder1));
  //if  (digitalRead(encoder2) == HIGH) {
    //while (digitalRead(encoder2) == HIGH)
    {
      //do nothing
    }
   //while (digitalRead(encoder1) == LOW)
    {
     //do nothing
    }
  // kec_putar();
   //}
     if (digitalRead(encoder1) == HIGH )
   {
     while (digitalRead(encoder1) == HIGH)
    {
      //do nothing
    }
    while  (digitalRead(encoder2) == LOW)
    {
      // do nothing
    }
     kec_longitudinal();
   }
}
void kec_longitudinal()
{
```

```
  Serial.print("kecepatan longitudinal =");
  unsigned long waktu = 0;
  float jeda = 0;
  float kecepatan_longitudinal = 0;
  word vel = 0;
  waktu = pulseIn(encoder1, LOW);
  jeda = waktu;
  jeda = jeda / 1000000;
  kecepatan_longitudinal = (1 / (jeda * 110 ))* 60 ;
  vel =  kecepatan_longitudinal;
  Serial.println(vel);
}
void kec_putar()
{
  Serial.print("kecepatan putar =");
  unsigned long waktu = 0;
  float jeda = 0;
  float kecepatan_putar = 0;
  word rpm = 0;
  waktu = pulseIn(encoder2, LOW);
  jeda = waktu;
  jeda = jeda / 1000000;
  kecepatan_putar = (1 / (jeda * 110)) * 60 ;
  rpm =  kecepatan_putar;
  Serial.println(rpm);
}
```

C. Program Matlab Kontroler *Neuro-Fuzzy* Tahap *Training* Data *Offline*

```
    function u = nerofusi(xin)
% deklarasi variabel global
global wcek1 wcek2 wcde1 wcde2 wgek wgde wout alpwc alpwg
alpho iter erout mse
ek   = xin(1);
ekm1 = xin(2);
t    = xin(3);
de   = ek - ekm1;
% Inisialisasi Bobot awal pada Jaringan neuro-fuzzy
```

```
    wcek1 = rand();
    wcek2 = rand();
    wcde1 = rand();
    wcde2 = rand();
    wgek = rand();
    wgde = rand();
    wout = rand(3,1);
    alpwc= 0.01;
    alpwg= 0.01;
    alpho= 0.01;
    iter = 0;
iter = 0;
mse =1;
while  iter < 100 && mse > 10^(-6)
% Forward Propagation (Umpan Maju)
   zer1=wgek*(wcek1-ek);
   zer2=wgek*(ek-wcek1);
   zer3=wgek*(ek-wcek2);
   erf(1)=1/(1+exp(-zer1));
   erf(2)=(1/(1+exp(-zer2)))-(1/(1+exp(-zer3)));
   erf(3)=1/(1+exp(-zer3));
   zde1=wgde*(wcde1-de);
   zde2=wgde*(de-wcde1);
   zde3=wgde*(de-wcde2);
   def(1)=1/(1+exp(-zde1));
   def(2)=(1/(1+exp(-zde2)))-(1/(1+exp(-zde3)));
   def(3)=1/(1+exp(-zde3));
% Kombinasi Aturan Inferensi fuzzy
   k=0;
   for i=1:3
      for j=1:3
         k=k+1;
         zo(k)=erf(j)*def(i);
      end
   end
   zof(1)=((zo(1)+zo(2)+zo(4))*1);
   zof(2)=((zo(3)+zo(5)+zo(7))*0);
   zof(3)=((zo(6)+zo(8)+zo(9))*-1);
   % Normalisasi zof
```

```
jmzof=sum(zof);
zof=zof/jmzof;
u=zof*wout;
erout=t-u;
mse=sqrt(erout*erout);
%pperambatan error
 for i= 1:3
   if u==0
      er1(i) = (1/3)*erout;
   else
      er1(i) = (zof(i)/u)*erout;
   end
 end
 er2(1,1)=sign(er1(1))*sqrt(abs(er1(1))/3);
 er2(1,2)=sign(er1(1))*sqrt(abs(er1(1))/3);
 er2(2,1)=sign(er1(1))*sqrt(abs(er1(1))/3);

 er2(1,3)=sign(er1(2))*sqrt(abs(er1(2))/3);
 er2(2,2)=sign(er1(2))*sqrt(abs(er1(2))/3);
 er2(3,1)=sign(er1(2))*sqrt(abs(er1(2))/3);

 er2(2,3)=sign(er1(3))*sqrt(abs(er1(3))/3);
 er2(3,2)=sign(er1(3))*sqrt(abs(er1(3))/3);
 er2(3,3)=sign(er1(3))*sqrt(abs(er1(3))/3);
 ererf=zeros(1,3);
 erdef=zeros(1,3);
 for i=1:3
    for j=1:3
       ererf(j)=ererf(j)+er2(j,i);
       erdef(i)=erdef(i)+er2(j,i);
    end
 end
 % Revisi bobot
 for i= 1:3
   wout(i) = wout(i)+ alpho*erout*zof(i);
 end
 dwgek1 = ererf(1)*erf(1)*(1-erf(1))*(wcek1-ek);
 dwgek2 = ererf(2)*erf(2)*(1-erf(2))*(ek-wcek1);
 dwgek3 = ererf(3)*erf(3)*(1-erf(3))*(wcek2-ek);
```

```
    wgek = wgek+alpwg*(dwgek1+dwgek2+dwgek3);
    dwgde1 = erdef(1)*def(1)*(1-def(1))*(wcde1-de);
    dwgde2 = erdef(2)*def(2)*(1-def(2))*(de-wcde1);
    dwgde3 = erdef(3)*def(3)*(1-def(3))*(wcde2-de);
    wgde = wgde+alpwg*(dwgde1+dwgde2+dwgde3);
    wcek1=wcek1+alpwc*ererf(1);
    wcek2=wcek2+alpwc*ererf(3);
    wcde1=wcde1+alpwc*erdef(1);
    wcde2=wcde2+alpwc*erdef(3);
    % memasukkan bobot yang dihasilkan ke workspace matlab
    assignin('base','wcek1',wcek1);
    assignin('base','wcek2',wcek2);
    assignin('base','wcde1',wcde1);
    assignin('base','wcde2',wcde2);
    assignin('base','wgek',wgek);
    assignin('base','wgde',wgde);
    assignin('base','wout',wout);
    iter=iter+1;
  % Mencetak nilai error untuk setiap Iterasi (EPOH)
    iter,
    mse,
end
 assignin ('base','mse',mse);
```

D. **Program Matlab Kontroler *Neuro-Fuzzy* Tahap *Online***

```
function u = online(xin)
global wcek1 wcek2 wcde1 wcde2 wgek wgde wout
ek   = xin(1);
ekm1 = xin(2);
%t   = xin(3);
de   = ek - ekm1;
% Inisialisasi Bobot awal jaringan neuro-fuzzy menggunakan bobot
hasil training offline
wcek1 = [0.5462];
wcek2 = [0.7070] ;
wcde1 = [0.0957] ;
wcde2 = [0.4736] ;
wgek = [0.4276] ;
```

```
wgde = [0.0468] ;
wout = [0.4895;0.8396;0.5820];
% forward Propagation ( Umpan Maju)
   zer1=wgek*(wcek1-ek);
   zer2=wgek*(ek-wcek1);
   zer3=wgek*(ek-wcek2);
   erf(1)=1/(1+exp(-zer1));
   erf(2)=(1/(1+exp(-zer2)))-(1/(1+exp(-zer3)));
   erf(3)=1/(1+exp(-zer3));
   zde1=wgde*(wcde1-de);
   zde2=wgde*(de-wcde1);
   zde3=wgde*(de-wcde2);
   def(1)=1/(1+exp(-zde1));
   def(2)=(1/(1+exp(-zde2)))-(1/(1+exp(-zde3)));
   def(3)=1/(1+exp(-zde3));
   % Kombinasi Aturan Inferensi Fuzzy
   k=0;
   for i=1:3
       for j=1:3
         k=k+1;
         zo(k)=erf(j)*def(i);
       end
   end
   zof(1)=((zo(1)+zo(2)+zo(4))*1);
   zof(2)=((zo(3)+zo(5)+zo(7))*0);
   zof(3)=((zo(6)+zo(8)+zo(9))*-1);
   % Normalisasi zof
   jmzof=sum(zof);
   zof=zof/jmzof;
   u=zof*wout;
```

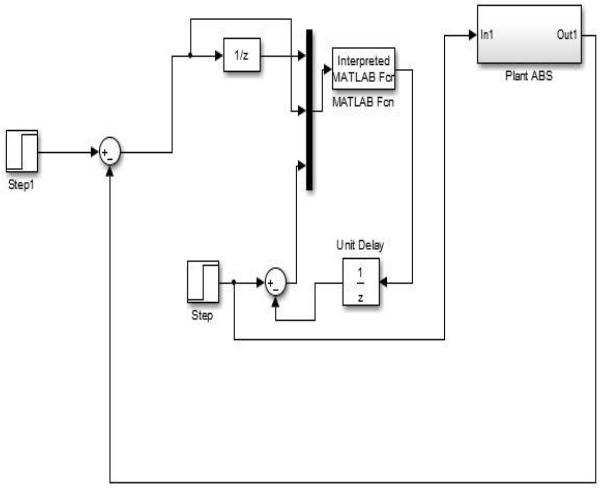
E. Program Arduino Tahap *Online*

```
#include <Wire.h>
int encoder1 = 7;
int encoder2= 5;
float percent ;
float b;
int sinyalkontrol;
char tampilan[1];
```

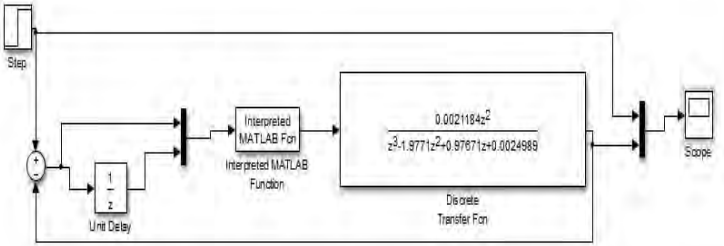
```
char tampilan1[1];
char tampilan2[1];
void setup()
{
  //Wire.begin();
  Serial.begin(9600);
  pinMode(encoder1,INPUT);
  pinMode(encoder2,INPUT);
}
void loop()
{
  //sinyalkontrol=map(percent,0,1,0,255);
  //analogWrite(13,sinyalkontrol);
  if (Serial.available()>0) {
  percent=Serial.parseFloat();
  //sinyalkontrol=map(percent,0,1,0,255);
  //Serial.print(sinyalkontrol);
  b = percent*255;
  analogWrite(9,b);}
  bacaEncoderLong();
  bacaEncoderAng();
 }
void bacaEncoderLong()
{
  //Serial.print("kecepatan longitudinal =");
  unsigned long waktu = 0;
  float jeda = 0;
  float kecepatan_longitudinal = 0;
  word vel = 0;
  waktu = pulseIn(encoder1, LOW, 6884);
  jeda = waktu;
  jeda = jeda / 1000000;
  kecepatan_longitudinal = (1 / (jeda * 110 ))* 60 ;
  vel =  kecepatan_longitudinal + 150;
  sprintf(tampilan,"*%4d",vel);
  Serial.print(tampilan);
}
void bacaEncoderAng()
{
```

```
  //Serial.print("kecepatan putar =");
  unsigned long waktu = 0;
  float jeda = 0;
  float kecepatan_putar = 0;
  word rpm = 0;
  waktu = pulseIn(encoder2, LOW, 6884);
  jeda = waktu;
  jeda = jeda / 1000000;
  kecepatan_putar = (1 / (jeda * 110)) * 60 ;
  rpm =  kecepatan_putar;
  sprintf(tampilan1,"#%4d",rpm);
  Serial.print(tampilan1);
}
```

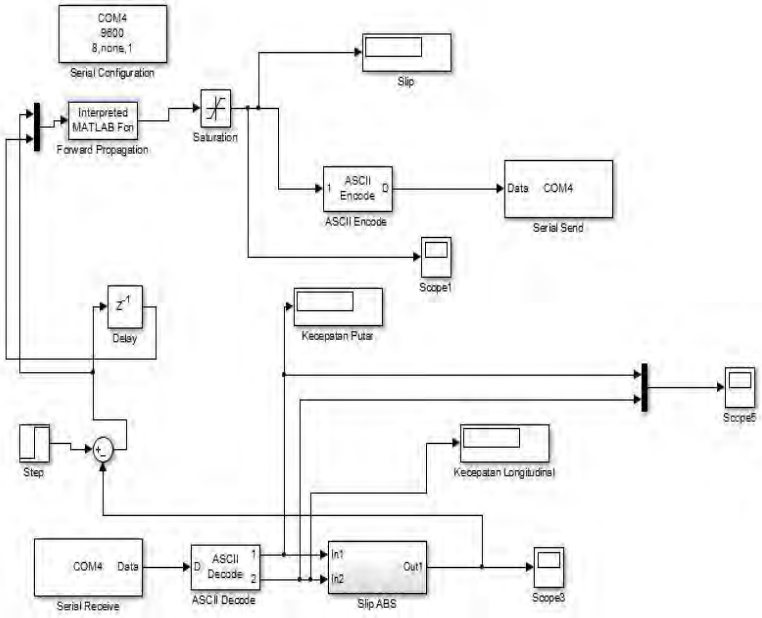
F. Program Blok Simulink Tahap *Offline*



G. Program Blok Simulink Tahap *Online*

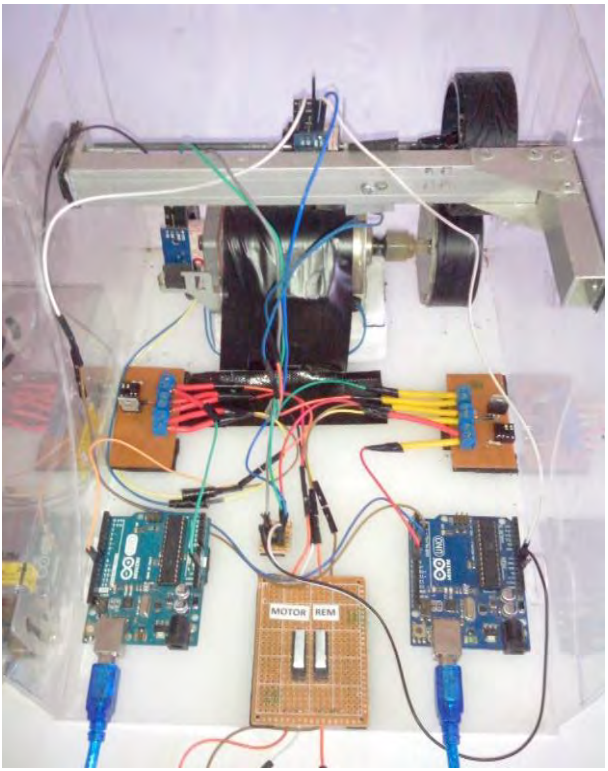


H. Program Blok Simulink Tahap Implementasi

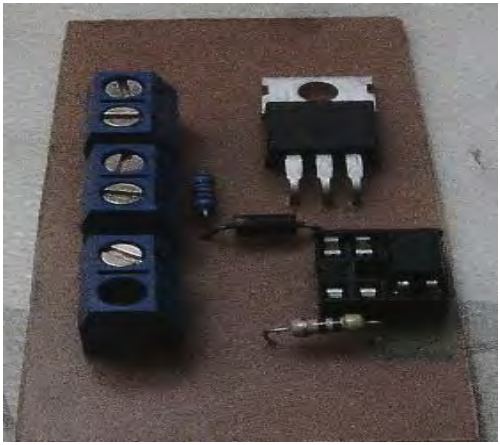


LAMPIRAN C
FOTO ALAT

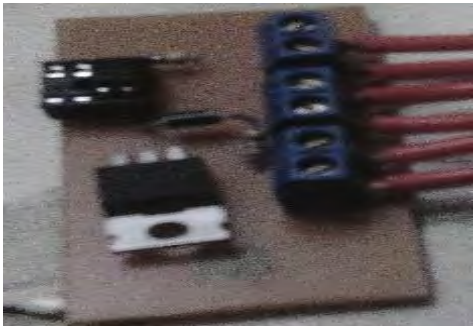
a. Simulator ABS



b. Bentuk Fisik *Driver* Motor DC



C. Bentuk Fisik *Driver* Rem Elektromagnetik



LAMPIRAN D
AKTIFITAS PENGERJAAN

Aktifitas pengerjaan ditampilkan pada Tabel berikut :

Tabel Pengerjaan:

No	Tanggal	Aktifitas Pengerjaan
1	*16-20 Februari 2015*	Menetapkan rancangan Hardware driver PWM untuk rem clutch brake dan encoder
2	23-28 Februari 2015	Membeli motor dc di malang serta melepas rem yang terpasang
3	9-14 Maret 2015	Membuat skema rangkaian encoder dan driver Pwm
4	16-21 Maret 2015	Membantu pemasangan dudukan rem di mobil RC dan mengetching Encoder
5	31 Maret- 4 April 2015	Mensimulasikan rangkaian driver serta troubleshooting Encder dan merancang kotak kayu untuk keperluan dyno test sebagai pengukur

		kecepatan putaran (wx)
6	6-11 April 2015	Mengimplementasikan Rangkaian dan mecoba pembacaan dengan modul sensor kecepatan karena encoder yang dibuat tidak bagus respon pembacaannya
7	13-17 April 2015	Mencoba Pengaruh Rem dengan persentase 25%,50%,75%, dan 100% terhadap Vxd an Wx
8	18-20 April 2015	Memperbaiki dan menyempurnakan program arduino untuk pengambilan 2 data (Vx,Wx) secara bersamaan.
9	22 April 2015	Merubah total rancangan sistem menjadi plant simulator ABS
10	23 April-25 April 2015	Merancang mekanik simulator ABS dan elektronik simulator ABS
		Melakukan Pengambilan data

11	25-28 April 2015	untuk 2 kecepatan (Vx,Wx) untuk dikonversi ke slip
12	28 April -2 Mei 2015	Melakukan pengambilan data titik kerja rem terhadap titik kerja slip
13	2-9 Mei 2015	Merancang kontroler *Neuro-Fuzzy*
14	10-12 Mei 2015	Memodifikasi Plant dengan menmbah massa roda bawah agar data yang diperoleh lebih tepat
15	12-15 Mei 2015	Mengolah data yang diperoleh menjadi berberapa *cluster* titik kerja rem terhadap titik kerja slip untuk 4 titik kecepatan
16	15 -22 Mei 2015	Melanjutkan rancangan kontroler *neuro-fuzzy*
17	22-31 Mei	Mempersiapkan program untuk implementasi dan merevisi buku Tugas Akhir

--Halaman ini sengaja dikosongkan--

DAFTAR PUSTAKA

[1] K. Reif, "Brakes, Brake Control and Driver Assistance Systems Function," *Springer,* vol. VIII, pp. 275-306, 2014.

[2] K. Kogut, "Anti-lock Braking System Modelling and Parameters Identification,"*IEEE International Conference On Control Application,* pp. 342-346, 2014.

[3] M. Smeja, *Modelling and Identificaion of Antilock Braking System* (ABS), Slovakia : AGH University, 2010.

[4] Zuhal, *Dasar Tenaga Listrik*, Bandung: Penerbit Bandung, 1991.

[5] W, Dale, *Arduino Internals*, New York: Appress, 2011.

[6] J.William H.Hayt dan J.A Buck, *Engineering Electromagnetics Sixth Edition*, Boston: McGraw-Hill,2001.

[7] ____, "*Project #11 Infrared Speed Sensor Module For Arduino*", <URL:http://sites.google.com/site/myscratchbooks/home/projects/project-11-infrared-speed-sensing-module>, Maret, 2015

[8] ___ , "*vexrobotics*", <URL:http://vexrobotics.com/wiki/Optical-Shaft-Encoder>, April, 2015

[9] Astrom, K.J, *Adaptive Control Second Edition*, NewYork: Dover Publications inc, 2008.

[10] K.M Passino dan S.Yurkovich, *Fuzzy Control*, California: Addison-Wesley, 1998. K.M Passino dan S.Yurkovich, *Fuzzy Control*, California: Addison-Wesley, 1998.

[11] Kusumadewi, Sri, *Neuro-Fuzzy, Integrasi Sistem Fuzzy dan Jaringan Syaraf*, Yogyakarta: Graha Ilmu, 2006.

[12] C.T, Lin, *Neural Fuzzy Systems*, London: Prentice Hall, 1996.

[13] JSR Jang dan Sun CT, *Neuro-Fuzzy and Soft Computing*, London: Prentice Hall, 1997.

[14] Wibowo, B.P., "Traction Control Pada Parallel Hybrid Electric Vehicle (HEV) Menggunakan Metode Kontrol Neuro-Fuzzy Prediktif," *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, 2014.

[15] Sobottka, Christian, "Optimal Fuzzy Logic Control For an Anti-Lock Braking System,"*IEEE International Conference On Control Application,* New York, 1996.

----Halaman ini sengaja dikosongkan---

RIWAYAT PENULIS

Rian Lukito lahir di Bekasi, 12 April 1992. Saat ini sedang menjalani masa kuliah di Jurusan Teknik Elektro ITS dan mengambil bidang studi Teknik Sistem Pengaturan. Dia Pernah menjadi GA Enginering Team di Control System Service Center Teknik Sistem Pengaturan ITS.