TUGAS AKHIR - TF 141581

PERANCANGAN SISTEM *ACTIVE FAULT TOLERANT CONTROL* PADA PENGENDALIAN KECEPATAN SISTEM SERVO MODULAR MS150 DC DENGAN KESALAHAN PADA AKTUATOR DAN SENSOR

VIQI BHAGASKARA KRESNA
NRP. 2413 100 002

Dosen Pembimbing :
Dr. Katherin Indriawati, S.T., M.T.

DEPARTEMEN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2017

“Halaman ini memang dikosongkan”

FINAL PROJECT – TF 141581

DESIGN ACTIVE FAULT TOLERANT CONTROL SYSTEM FOR VELOCITY CONTROL OF MODULAR SERVO SYSTEM MS150 DC WITH FAULT AT ACTUATOR AND SENSOR

VIQI BHAGASKARA KRESNA
NRP. 2413 100 002

Supervisors :
Dr. Katherin Indriawati, S.T., M.T.

ENGINEERING PHYSICS DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

“Halaman ini memang dikosongkan”

PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini

Nama	: Viqi Bhagaskara Kresna
NRP	: 2413100002
Departemen/ Prodi	: Teknik Fisika/ S1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "Perancangan Sistem *Active Fault Tolerant Control* Pada Pengendalian Kecepatan Sistem Servo Modular MS150 DC Dengan Kesalahan Pada Aktuator dan Sensor" adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada tugas akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, 25 Juli 2017
Yang membuat pernyataan,

Viqi Bhagaskara Kresna
NRP. 2413100002

“Halaman ini memang dikosongkan”

LEMBAR PENGESAHAN
TUGAS AKHIR

PERANCANGAN SISTEM *ACTIVE FAULT TOLERANT CONTROL* PADA PENGENDALIAN KECEPATAN SISTEM SERVO MODULAR MS150 DC DENGAN KESALAHAN PADA AKTUATOR DAN SENSOR

Oleh:
Viqi Bhagaskara Kresna
NRP. 2413 100 002

Surabaya, 25 Juli 2017

Menyetujui,
Dosen Pembimbing

Dr. Katherin Indriawati, S.T., M.T.
NIPN. 197605232000122001

Mengetahui,
Ketua Departemen
Teknik Fisika FTI-ITS

Agus Muhamad Hatta, S.T., M.Si., Ph.D.
NIPN. 197809022003121002

“Halaman ini memang dikosongkan”

PERANCANGAN SISTEM *ACTIVE FAULT TOLERANT CONTROL* PADA PENGENDALIAN KECEPATAN SISTEM SERVO MODULAR MS150 DC DENGAN KESALAHAN PADA AKTUATOR DAN SENSOR

TUGAS AKHIR

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

Oleh:

VIQI BHAGASKARA KRESNA
NRP. 2413 100 002

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Katherin Indriawati, S.T., M.T. ..... (Pembimbing)
2. Hendra Cordova, S.T., M.T. ..... (Ketua Penguji)
3. Detak Yan Pratama, S.T., M.Sc. ..... (Penguji)
4. Andi Rahmadiansah, S.T., M.T. ..... (Penguji)
5. Herry Sufyan Hadi, S.T., M.T. ..... (Penguji)

SURABAYA
Juli, 2017

“Halaman ini memang dikosongkan”

PERANCANGAN SISTEM *ACTIVE FAULT TOLERANT CONTROL* PADA PENGENDALIAN KECEPATAN SISTEM SERVO MODULAR MS150 DC DENGAN KESALAHAN PADA AKTUATOR DAN SENSOR

Nama : Viqi Bhagaskara Kresna
NRP : 2413 100 002
Departemen : Teknik Fisika FTI-ITS
Dosen Pembimbing : Dr. Katherin Indriawati, S.T., M.T.

Abstrak

Kegagalan sistem motor DC pada mobil listrik akibat kesalahan pada komponen seperti aktuator dan sensor dapat berakibat fatal karena dapat membahayakan pengendara,penumpang maupun masyarakat sekitar karena dapat menyebabkan terjadinya kecelakaan. Pada Tugas akhir ini dilakukan perancangan sistem *Active Fault Tolerant Control* (AFTC) yang mampu bekerja ketika terdapat kesalahan pada komponen seperti aktuator dan sensor pada pengendalian kecepatan sistem servo modular MS150 DC sehingga performansi sistem tetap terjaga. Langkah pertama yang dilakukan adalah *set-up* dan identifikasi sistem servo modular MS150 DC. Langkah kedua adalah merancang sistem pengendali PI. Langkah ketiga adalah merancang sistem AFTC secara simulasi dan langkah terakhir adalah merancang sistem AFTC untuk aplikasi *real time*. Penerapan AFTC menghasilkan nilai *maximum overshoot* lebih kecil dibandingkan dengan sistem tanpa AFTC, selain itu penerapan AFTC menyebabkan nilai *settling time* yang dihasilkan lebih kecil dibandingkan dengan sistem tanpa AFTC untuk kesalahan bias pada aktuator dan pada sensor dengan nilai bias sebesar 4 – 6 volt, kemudian penerapan AFTC mempengaruhi nilai *error* rata-rata, dimana ketika terjadi kesalahan pada sensor, nilai *error* rata-rata untuk sistem dengan AFTC lebih kecil dibandingkan dengan sistem tanpa AFTC, begitu juga ketika terjadi kesalahan bias pada aktuator, sistem dengan AFTC memiliki nilai *error* rata-rata lebih kecil dibandingkan dengan sistem tanpa AFTC dengan nilai kesalahan bias yang terjadi bernilai 4-6 volt

Kata Kunci: Aktuator, PI, Sensor, Sistem AFTC, Sistem servo modular MS150 DC.

"Halaman ini memang dikosongkan"

DESIGN ACTIVE FAULT TOLERANT CONTROL SYSTEM FOR VELOCITY CONTROL OF MODULAR SERVO SYSTEM MS150 DC WITH FAULT AT ACTUATOR AND SENSOR

Name	***: Viqi Bhagaskara Kresna***
NRP	***: 2413 100 002***
Department	***: Teknik Fisika FTI-ITS***
Supervisors	***: Dr. Katherin Indriawati, S.T., M.T.***

Abstract

Failure of DC motor in electric cars due to errors in actuator and sensor can harm the driver, the passengers and people because accident can happened. On this final project, designing active fault tolerant control (AFTC) systems is able to work when there are errors in actuator and sensor components for speed control of modular servo system MS150 DC, so AFTC can maintain its performance. The first step is set-up and identification system of the modular servo system MS150 DC, the second step is designing the PI control system, the third step is designing simulation of AFTC system and the last step is designing AFCT system for realtime. Results of AFTC's perfromance for realtime shows that the value of maximum overshoot for AFTC system is smaller than without using AFTC system, then the value of settling time for AFTC system is smaller too than without using AFTC system for sensor fault and actuator fault when the bias fault of actuator 4-6 volt, and the value of error for AFTC system is smaller than without using AFTC system for sensor fault and actuator fault when the bias fault of actuator 4-6 volt.

Keywords : Actuator, PI, Sensor, AFTC system, Modular servo system MS150

"Halaman ini memang dikosongkan"

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang senantiasa melimpahkan rahmat serta hidayah-Nya, serta shalawat serta salam kepada Nabi Muhammad SAW, hingga terselesaikannya tugas akhir beserta laporan tugas akhir yang berjudul

PERANCANGAN SISTEM *ACTIVE FAULT TOLERANT CONTROL* PADA PENGENDALIAN KECEPATAN SISTEM SERVO MODULAR MS150 DC DENGAN KESALAHAN PADA AKTUATOR DAN SENSOR

Penulis telah banyak memperoleh bantuan dari berbagai pihak dalam penyelesaian tugas akhir dan laporan Tugas Akhir ini. Penulis mengucapkan terimakasih kepada :

1. Bapak Agus Muhamad Hatta, S.T., M.Si., Ph.D. selaku ketua departemen teknik fisika yang telah memberikan petunjuk, ilmu, serta bimbingan selama menempuh pendidikan di teknik fisika.
2. Ibu Dr. Katherin Indriawati, S.T., M.T. selaku dosen pembimbing yang telah dengan sabar memberikan petunjuk, ilmu, serta bimbingan yang sangat bermanfaat.
3. Bapak Totok Ruki Biyanto, S.T., M.T., Ph.D. Selaku kepala Laboratorium Rekayasa Instrumensi yang telah memberikan ilmu, petunjuk, nasihat, serta kemudahan perizinan.
4. Bapak Dr. Ir. Totok Soehartanto, DEA. selaku dosen wali yang telah membimbing penulis selama perkuliahan.
5. Kedua orang tua (Bapak Teguh Yuli Witanto dan Ibu Sri Andayani) serta kaka (Novika Widyasri). Terimakasih atas segala cinta, kasih sayang, doa, perhatian, serta dukungan moril dan materiil yang telah diberikan.
6. Seluruh teman Tugas Akhir Ivan Taufik Akbar, Alief Ghazi, dan Alif Helmi, terima kasih untuk semuanya.
7. Seluruh teman – teman Departemen Teknik Fisika angkatan 2013, terima kasih untuk semuanya.

8. Seluruh dosen, karyawan dan civitas akademik Teknik Fisika, terimakasih atas segala bantuan dan kerjasamanya.
9. Semua pihak yang tidak dapat disebutkan satu persatu, terimakasih atas bantuannya.

Penulis sadar bahwa penulisan laporan tugas akhir ini tidak sempurna, namun semoga laporan ini dapat memberikan kontribusi yang berarti dan menambah wawasan yang bermanfaat bagi pembaca, keluarga besar Teknik Fisika khususnya, dan civitas akademik ITS pada umumnya. Semoga laporan tugas akhir ini dapat bermanfaat sebagai referensi pengerjaan laporan tugas akhir bagi mahasiswa yang lain.

Surabaya, 25 Juli 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

“Halaman ini memang dikosongkan”

DAFTAR TABEL

“Halaman ini memang dikosongkan”

DAFTAR NOTASI

Ψ	=	Fluks
K_f	=	Konstanta Fluks
i_f	=	Arus fluks (ampere)
T	=	Torsi (N-m)
K_t	=	Konstanta torsi motor
i_a	=	Arus kumparan jangkar (ampere)
e_b	=	Gaya gerak listrik balik (volt)
K_b	=	Konstanta emf balik
ω	=	Kecepatan rotasi (rad/s)
J	=	Momen inersia eqivalen dari motor (Kg-m^2)
b	=	Koefisien gesekan viskos ekivalen dari motor atau beban pada poros motor (N-m/rad/s)
K	=	Konstanta
L	=	Induktansi kumparan jangkar (henry)
R	=	Tahanan kumparan jangkar (ohm)
e_a	=	Tegangan kumparan jangkar (volt)
V	=	Tegangan input (volt)
$P(z)$	=	Pengendali P dalam bidang diskrit domain z
$I(z)$	=	Pengendali I dalam bidang diskrit domain z
$u(z)$	=	Sinyal kendali dalam domain z
$e(z)$	=	Sinyal error dalam bidang diskrit domain z
Kp	=	Konstanta proposal
Ki	=	Konstanta integral
T_s	=	Waktu cuplik
τ_i	=	Konstanta waktu integral
$x(k)$	=	Vektor keadaan
$y(k)$	=	Vektor keluaran
$u(k)$	=	Vektor kendali
A	=	Matriks keadaan
B	=	Matriks masukan
C	=	Matriks keluaran
D	=	Matriks gangguan
$x(k+1)$	=	Turunan vektor keadaan

$\hat{x}(k+1)$	=	Turunan estimasi x_{k+1}
Ke	=	*Gain observer*
$\hat{x}(k)$	=	Estimasi vektor keadaan
$\hat{y}(k)$	=	Estimasi vektor keluaran
I	=	Matriks identitas
z	=	*poles*
X_t	=	Data aktual pada periode t
F_t	=	Data pemodelan pada periode t
n	=	Jumlah data
$f_a(k)$	=	Vektor kesalahan aktuator
$f_s(k)$	=	Vektor kesalahan sensor
F_a	=	Matriks kesalahan aktuator
F_s	=	Matriks kesalahan sensor
$\overline{F}$	=	Matriks kesalahan
$f(k)$	=	Vektor kesalahan
$\hat{f}(k+1)$	=	turunan vektor kesalahan
L	=	*Gain* kesalahan aktuator dan sensor
$y(k)$	=	Nilai pengukuran sebenarnya
$y_m(k)$	=	Nilai pengukuran mengandung kesalahan sensor
$u(k)$	=	Nilai kendali sebenarnya
$u_m(k)$	=	Nilai kendali smengandung kesalahan aktuator

BAB I
PENDAHULUAN

1.1 Latar Belakang

Motor DC (*Direct Current*) atau motor arus searah merupakan sebuah perangkat elektromagnetis yang berfungsi untuk mengubah energi listrik menjadi energi mekanik. Motor DC termasuk dalam kategori jenis motor yang sering digunakan dalam bidang industri, peralatan rumah tangga, mainan anak-anak atau sebagai piranti pendukung sistem instrumen elektronik. Saat ini, penggunaan motor DC sudah merambah ke dunia otomotif berupa penggunaan motor DC untuk mobil listrik.

Sistem motor DC diharapkan tidak pernah mengalami gangguan dalam penerapannya. Namun, gangguan dapat terjadi secara mendadak dan tidak dapat dihindarkan. Hal ini dapat mempengaruhi jalannya suatu sistem yang menyebabkan berbagai macam kerugian. Sampai saat ini gangguan yang terjadi pada motor DC dapat ditangani dengan menggunakan sistem pengendali PID, tetapi jika gangguan tersebut berupa kesalahan yang terjadi pada komponen-komponen pendukung suatu sistem maka sistem pengendali PID tidak mampu dalam mengatasi gangguan tersebut.

Gangguan berupa kesalahan komponen yang sering terjadi di bidang industri terdapat pada komponen aktuator dan sensor (Zhang & Jiang, 2008). Aktuator dan sensor merupakan komponen yang paling mudah rusak karena terletak di lapangan dan langsung berhubungan dengan lingkungan. Kesalahan pada komponen menyebabkan jalannya produksi dari suatu perusahaan tidak dapat berjalan secara lancar yang berdampak pada kerugian bagi perusahaan. Kesalahan komponen aktuator dan sensor tidak hanya terjadi di bidang industri, tetapi dapat terjadi pada sistem motor DC yang digunakan pada mobil listrik. Kesalahan tersebut memiliki dampak yang dapat menyebabkan sistem gagal bekerja, sehingga menyebabkan keselamatan pengemudi dan masyarakat sekitar terancam. Kesalahan pada komponen dapat diatasi dengan penggunaan sistem pengendali yang mampu menoleransi

kesalahan secara otomatis, sehingga kegagalan sistem untuk bekerja dapat diatasi.

Sistem kendali yang mampu menoleransi gangguan berupa kesalahan pada komponen secara otomatis disebut *Fault Tolerant Control System* (FTCS). FTCS merupakan suatu perkembangan teknologi untuk kebutuhan keamanan dan perbaikan unjuk kerja. FTCS dapat diklasifikasikan pada 2 tipe yaitu *Passive Fault Tolerance Control System* (PFTCS) dan *Active Fault Tolerance Control System* (AFTCS). AFTCS terdiri dari 2 tahapan, yaitu *Fault Detection and Indentification* (FDI) dan *Reconfigurable Control* (Zhang & Jiang, 2008).

FDI digunakan untuk memperkirakan kesalahan sedangkan *Reconfigurable Control* digunakan untuk mengkompensasi sinyal kendali secara otomatis sehingga sistem tetap stabil, dengan parameter *maximum overshoot*, *settling time*, dan *error steady state* yang mendekati nol. Pada tugas akhir ini dilakukan perancangan sistem *Active Fault Tolerant Control* (AFTC) pada pengendalian kecepatan sistem servo modular MS150 DC dengan kesalahan pada aktuator dan sensor. Penggunaan sistem AFTC dapat menoleransi kesalahan yang terjadi pada sistem servo modular MS150 DC, sehingga sistem dapat bekerja secara stabil dan performansi tetap terjaga.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, rumusan masalah dari penulisan tugas akhir ini adalah :

a. Bagaimana merancang sistem *Active Fault Tolerant Control* (AFTC) pada pengendalian kecepatan sistem servo modular MS150 DC dengan kesalahan pada aktuator dan sensor ?
b. Mengetahui apakah sistem AFTC mampu menjaga performansi sistem servo modular MS150 DC saat terjadi kesalahan pada aktuator dan sensor, serta mengetahui unjuk kerja sistem tersebut ?

1.3 Tujuan

Berdasarkan latar belakang dan perumusan masalah, tujuan penelitian yang ingin dicapai dalam pengerjaan tugas akhir ini adalah sistem pengendalian kecepatan pada sistem servo modular MS150 DC yang mampu mengatasi kesalahan pada aktuator dan sensor sehingga performansi sistem tetap terjaga.

1.4 Batasan Masalah

Bedasarkan identifikasi masalah maka batasan masalah dari tugas akhir ini adalah :

a. Algoritma pengendalian yang digunakan pada penelitian ini yaitu jenis pengendalian PI.
b. Sistem yang digunakan tanpa penambahan variasi beban (*torque load)*.
c. Variabel yang diukur adalah kecepatan motor dan arus dalam bentuk tegangan dengan kesalahan sensor hanya terjadi pada *tachogenerator* atau pengukuran kecepatan.
d. Komponen yang digunakan dalam menjalankan aksi kendali atau aktuator adalah rangkaian *servoamplifier* dengan kesalahan yang terjadi adalah kesalahan bias.
e. Sistem AFTC yang dirancang diterapkan pada satu kondisi masukan.
f. Kesalahan bias terbesar yang terjadi pada aktuator sebesar 0,2 volt; 0,3 volt; 0,4 volt dan 0,6 volt dan pada sensor -0,2 volt; -0,3 volt; -0,4 volt dan -0,6 volt.

1.5 Sistematika Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

a. BAB I PENDAHULUAN
   Pada bab I ini terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan dan sistematika laporan.
b. BAB II TINJAUAN PUSTAKA
   Pada bab II ini dibahas mengenai teori-teori yang berkaitan dengan penelitian yang akan dilakukan, seperti teori

identifikasi sistem, pengendalian PI, Sistem *Active Fault Tolerant Control.*

c. BAB III METODOLOGI PENELITIAN
   Pada bab III ini berisi mengenai rancangan dari penelitian yang dilakukan, metode dan langkah-langkah dalam penelitian.
d. BAB IV ANALISIS DATA DAN PEMBAHASAN
   Pada bab IV ini berisi tentang analisis hasil perancangan sistem *active fault tolerant kontrol* dibandingkan dengan hasil perancangan kontrol konvensional.
e. BAB V KESIMPULAN DAN SARAN
   Pada bab V ini diberikan kesimpulan tentang tugas akhir yang telah dilakukan berdasarkan data-data yang diperoleh, serta diberikan saran sebagai penunjang maupun pengembangan tugas akhir selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Motor DC

Motor DC adalah sebuah mesin listrik yang prinsip kerjanya mengubah energi listrik menjadi energi mekanik. Energi mekanik yang dihasilkan merupakan energi gerak putar pada poros motor. Konversi energi tersebut dilakukan dalam waktu yang cepat dan merupakan implementasi hukum lorenz, yaitu hukum yang berlaku untuk aliran listrik pada kawat penghantar yang berada didalam medan magnet (Sugiono, 2015).

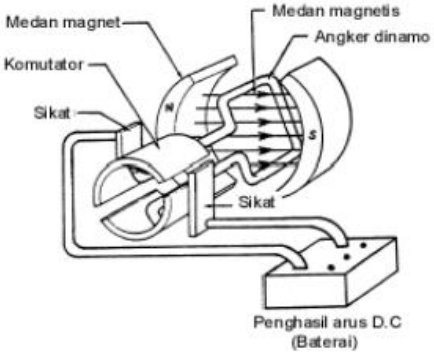
Motor DC dalam aplikasinya sangat luas mulai dari penggerak utama beberapa fungsi mesin industri, penggerak utama pada sistem transportasi mobil maupun kereta listrik bahkan untuk memenuhi kebutuhan penggerak peralatan rumah tangga seperti mesin jahit, kipas angin, peralatan di bidang kedokteran, mainan anak-anak, pompa hingga penggerak peralatan pada sistem komputer. Salah satu jenis motor DC yang sering ditemukan yaitu motor DC jenis magnet permanen, dimana medan magnet dihasilkan oleh magnet permanen dan menghasilkan fluks konstan.

2.1.2 Cara Kerja Motor DC Magnet Permanen

Motor DC magnet permanen seperti pada gambar 2.1 tersusun dari beberapa bagian seperti magnet permanen, kumparan jangkar atau angker dinamo (*armature)*, komutator (*commutator*), dan sikat (*brush).* Terdapat dua prinsip dasar yang mendasari kerja motor DC. Pertama, magnet permanen sebagai stator motor tersusun dari dua magnet dengan kutub yang berbeda (kutub utara dan kutub selatan), yang saling berhadapan dan berfungsi menghasilkan fluks yang nilainya konstan dari kutub utara ke kutub selatan. Komutator yang termasuk bagian dari rotor motor terletak pada kedua ujung kumparan jangkar yang berfungsi mengumpulkan arus induksi dari jangkar dan mengkonversinya menjadi arus searah. Sikat berfungsi menyalurkan arus listrik dari sumber diluar motor ke dalam

kumparan jangkar dan kumparan jangkar (*armature*) sebagai rotor motor digambarkan dalam bentuk sebuah kawat yang memiliki bentuk persegi panjang dan berfungsi untuk merubah energi listrik menjadi energi mekanik dalam bentuk gerak putar.

Kedua, sumber tegangan DC pada gambar 2.1 diilustrasikan dengan baterai yang memiliki kutub positif dan kutub negatif, kemudian baterai dari kutub positif akan mengalirkan arus menuju sikat (*brush*) yang selanjutnya menuju komutator, kemudian arus menuju kumparan jangkar (*armature)*, dan kembali lagi ke komutator, lalu sikat karbon dan terakhir menuju ke kutub negatif baterai. Dibagian stator motor, kumparan medan stator menghasilkan fluks dari kutub utara ke kutub selatan. Dengan adanya arus di dalam kumparan jangkar yang terletak didalam medan magnet yang mengandung fluks, maka akan menghasilkan suatu gaya, yang biasa disebut gaya Lorentz (*F*), selanjutnya gaya tersebut akan menggerakan kumparan jangkar untuk berputar. Komponen komutator yang selalu ikut berputar dengan rotor dan sikat karbon yang selalu diam, menjadi komponen yang akan menjaga arah arus listrik untuk selalu tetap yakni mengalir dari sisi kiri kawat angker ke kanan. Arah arus listrik yang selalu tetap di setiap setengah putaran rotor inilah yang akan membuat rotor motor listrik selalu berputar selama masih ada arus listrik yang mengalir ke kumparan jangkar (Setyaningrum, 2012).



Gambar 2.1 Motor DC (Hudaya, 2013)

2.2 Sistem Servo Modular MS150 DC

Sistem servo modular MS150 DC merupakan blok rangkaian elektronik yang digunakan untuk pengendalian kecepatan dan posisi dari motor DC jenis magnet permanen. Sistem servo modular MS150 DC terdiri dari suatu pemakaian sumber tenaga (*power supply*), *servo amplifier*, unit motor DC, *reduction gear tacho unit.* (Pengaturan, 2015)

Gambar 2.2 Sistem motor servo MS150 DC (Pengaturan, 2015)

2.2.1 Bagian-Bagian Sistem Servo Modular MS150 DC

Bagian-bagian sistem servo modular MS150 secara umum yang terdapat pada gambar 2.2, yaitu :

a. *Power supply* 150 E

Sumber tegangan yang dipakai untuk menyuplai rangkaian pengaturan motor DC adalah modul Feedback tipe PS150E. Alat ini menghasilkan tegangan keluaran sebesar 24V DC 2A dengan 8-*way connector* ke *servo amplifier*. Pada panel depan terdapat dua set 4mm *socket* untuk pasokan tegangan referensi. Terdapat tombol hijau pada panel depan sebagai tombol *power*. Selain itu, juga terdapat ammeter yang digunakan untuk menunjukkan jumlah arus listrik yang

mengalir ke motor, ketika arus melebihi batas 2A, maka lampu akan menyala sebagai peringatan.

b. Motor DC
Motor DC yang digunakan merupakan motor DC tipe DCM150F yang terdiri dari tiga bagian yaitu Motor DC magnet permanen, poros motor atau rotor yang diperpanjang dan piringan inersia atau piringan magnet permanen.

c. *Tachogenerator*
Modular *feedback tachogenerator* GT150X berfungsi untuk mengkonversi besaran mekanik menjadi besaran listrik sehingga putaran motor dapat diukur. Putaran motor dapat dinyatakan baik dalam bentuk tegangan atau rpm. Prinsip kerja dari *tachogenerator* , yaitu bagian rotor pada motor dikopel dan putaran motor tersebut digunakan untuk memutar rotor yang terdapat di *tachogenerator*, dengan prinsip kerja generator maka adanya putaran rotor akan menghasilkan tegangan keluaran.

d. *Servo amplifier*
Modular *feedback servo amplifier* SA150D terdiri dari rangkaian transistor yang dapat menggerakan motor DC dengan dua arah putaran. Komponen *servo amplifier* memiliki *overloading protection* yang berfungsi untuk sistem perlindungan motor dari *overloading* sehingga arus yang mengalir tidak lebih dari 2A. (Setyaningrum, 2012).

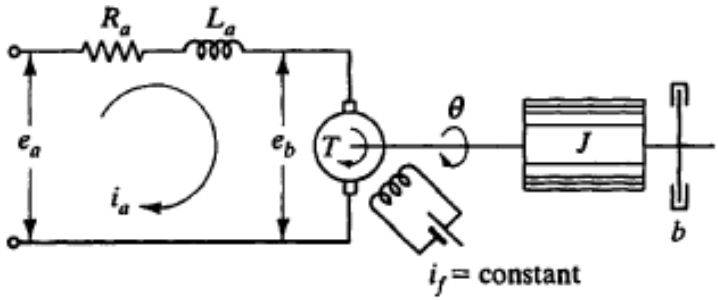
e. *Attenuator*
Attenuator AU150B digunakan sebagai pemberi hambatan dikarenakan terdapat dua potensio dengan kapasitas masing-masing sebesar 10 KΩ yang dapat digunakan secara bersamaan.

2.3 Model Matematik Motor DC

Sistem merupakan suatu rangkaian yang disusun dari bermacam-macam komponen yang saling berhubungan dalam menjalankan suatu aksi. Berdasarkan komponen yang digunakan, suatu sistem dapat berupa sistem mekanik, pneumatik, elektrik

atau jenis elektro-mekanik. Motor DC termasuk dalam sistem elektro-mekanik.

Suatu prasyarat dasar pada kebanyakan strategi yang dipergunakan dalam membangun suatu sistem kendali adalah kemampuan dalam memodelkan secara matematika dari sistem yaitu sistem yang akan dikendalikan. Model matematika dari suatu sistem dapat berupa persamaan differensial, fungsi transfer atau ruang keadaan (*state space).* Prinsipnya terdapat dua macam pemodelan matematika yang dapat dipilih, yaitu berdasarkan pada teori pengetahuan dengan menggunakan hukum fisika atau dengan proses eksperimen yaitu melakukan pengukuran. Kebanyakan kasus tidak mungkin untuk membuat model yang sempurna hanya dengan menggunakan pengetahuan fisika saja. Beberapa parameter harus ditentukan dari suatu eksperimen. Pendekatan ini disebut dengan identifikasi sistem. Model matematik untuk *plant* motor DC dapat dijabarkan pada bagian elektrik dan mekanik.



Gambar 2.3 Rangkaian motor DC (Ogata, 2004)

Berdasarkan gambar 2.3 secara umum torsi yang dibangkitkan oleh motor DC berbanding lurus dengan hasil kali dari i_a dan ψ yang berbanding lurus dengan arus medan,

$$\psi = K_f i_f \tag{2.1}$$

sehingga torsi dapat dituliskan sebagai berikut,

$$T = K_f i_f Kt. i_a \quad (2.2)$$

Perhatikan bahwa dengan asumsi arus medan konstan, sehingga fluks juga konstan menyebabkan torsi mempunyai arah sesuai arus kumparan jangkar, sehingga

$$T = Kt. i_a \quad (2.3)$$

Sedangkan, ketika gaya dibangkitkan akan timbul gaya gerak listrik balik *eb* yang berbanding lurus dengan kecepatan rotasi motor ω, atau

$$e_b = Kb. \omega \quad (2.4)$$

Dalam unit SI konstanta torsi motor dan gaya gerak listrik balik adalah sama, sehingga *Kt = Kb*; dengan demikian dapat digunakan K untuk menyatakan keduanya.

Di sisi mekanik, rotor (J) dan beban (b) dapat disebut sebagai inersia dan gesekan viskositas, sehingga besarnya torsi dituliskan

$$J\frac{d\omega}{dt} + b\omega = T = K. i_a \quad (2.5)$$

Kecepatan kumparan jangkar servomotor dc dikendalikan oleh tegangan kumparan jangkar e_a, sehingga agar kumparan jangkar dapat bergerak, e_a melewati rangkaian jangkar yang terdiri dari hambatan jangkar (R) dan induktansi jangkar (L), sehingga persamaan diferensial rangkaian kumparan jangkar adalah,

$$L\frac{di_a}{dt} + Ri_a + e_b = e_a \quad (2.6)$$

Jika persamaan (2.4) dimasukan kedalam persamaan (2.6), menjadi

$$L\frac{di_a}{dt} + Ri_a = V - K.\omega \quad (2.7)$$

Dengan menggunakan transformasi laplace dapat dirubah sebagai berikut,

$$Js\omega(s) + b\omega(s) = KI(s) \quad (2.8)$$
$$LsI(s) + RI(s) = V(s) - K\omega(s) \quad (2.9)$$

Dari persamaan (2.9) dapat dituliskan I(s) menjadi,

$$I(s) = \frac{V(S)-K\omega(s)}{R+Ls} \quad (2.10)$$

Dan disubstitusikan ke persamaan (2.8), menjadi

$$Js\omega(s) + b\omega(s) = K\frac{(V(S)-K\omega(s))}{R+Ls} \quad (2.11)$$

Fungsi transfer dari masukan tegangan V(s) berbanding dengan kecepatan motor $\omega(s)$ adalah

$$\frac{\omega(s)}{V(s)} = \frac{K}{(R+Ls)(Js+b)+K.Kb} \quad (2.12)$$

L seringkali diabaikan karena nilainya yang sangat kecil apabila dibandingkan dengan R, maka

$$\frac{\omega(s)}{V(s)} = \frac{K}{RJs+Rb+K.Kb} \quad (2.13)$$

Persamaan (2.5) dan (2.7) dapat dituliskan kedalam bentuk persamaan ruang keadaan (*state-space)* untuk kecepatan motor DC dengan memilih kecepatan rotasi dan arus motor yang mengalir sebagai variabel keadaan, dan tegangan sebagai masukan serta kecepatan rotasi sebagai keluaran:

$$\frac{d}{dt}\begin{bmatrix}\omega \\ i_a\end{bmatrix} = \begin{bmatrix}\frac{-b}{J} & \frac{K}{J} \\ \frac{-K}{L} & \frac{-R}{L}\end{bmatrix}\begin{bmatrix}\omega \\ i_a\end{bmatrix} + \begin{bmatrix}0 \\ \frac{1}{L}\end{bmatrix}V \quad (2.14)$$

$$y = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\omega \\ i_a\end{bmatrix} \quad (2.15)$$

dengan :

b = koefisien gesekan viskos ekivalen dari motor atau beban pada poros motor (N-m/rad/s)
Ψ = Fluks
K_f = konstanta fluks
i_f = arus fluks (ampere)
T = Torsi (N-m)
Kt = Konstanta torsi motor
Kb = konstanta emf balik
R = Tahanan kumparan jangkar, (ohm)
J = Momen inersia eqivalen dari motor,(Kg.m^2)
ω = Kecepatan rotasi, (rpm)
V = Tegangan input servo, (Volt)
ea = Tegangan yang dikenakan pada jangkar (volt)
eb = Gaya gerak listrik balik (volt)
L = Induktansi kumparan jangkar (henry)
i_a = Arus kumparan jangkar (ampere)

2.4 Identifikasi Sistem

Identifikasi sistem adalah suatu metode umum untuk membangun model matematika dan memperkirakan nilai parameter yang optimal secara eksperimen berdasarkan data masukan dan data keluaran (Anindita, 2008). Pada konteks ini memodelkan berarti membangun representasi matematis dari perilaku dinamis suatu sistem atau proses baik dalam waktu atau frekuensi domain, metode ini termasuk dalam teori kendali modern. Langkah-langkah yang diperlukan dalam identifikasi sistem, yaitu :

a. Data masukan dan data keluaran

Langkah awal dalam identifikasi sistem adalah pengambilan data *input-output*. Untuk memperoleh data keluaran diperlukan sinyal uji (data masukan) tertentu yang akan diberikan kepada sistem fisik yang akan diidentifikasi. Agar diperoleh model yang tepat maka dalam pemilihan sinyal uji ini tidak boleh sembarangan. Syarat pemilihannya adalah suatu sinyal uji harus memiliki cakupan frekuensi yang lebar dan standard yang digunakan adalah sinyal *Pseudo Random Binary Sequences* (PRBS). *Pseudo Random Binary Sequence* (PRBS) adalah sinyal kotak yang memiliki pola acak dengan pengulangan dalam jangka waktu yang cukup panjang atau dalam siklus tertentu. (Nusantoro, dkk, 2012). Sehingga dari karakteristik ini membuat PRBS sebagai sebuah sinyal yang kaya akan frekuensi.

b. Pemilihan model

Metode yang sering digunakan untuk mendapatkan model dari identifikasi sistem adalah teknik parametrik. Secara umum metode parametrik terdiri dari beberapa model, seperti *general linear polynomial*, *transfer function*, *zero-pole-gain* dan *state-space*. Pada teknik ini nilai parameter dari suatu sistem dapat diperoleh lebih akurat jika memiliki pengetahuan mengenai sistem yang akan di identifikasi, seperti derajat model, waktu cuplik. Model dengan metode parametrik dapat memberikan perkiraan yang lebih akurat jika pengguna memiliki pengetahuan mengenai dinamika sistem seperti model orders, waktu cuplik, dan sebagainya jika dibandingkan dengan metode non-parametrik, selain itu model parametrik merupakan solusi yang tepat terutama jika terkait dengan pekerjaan *real time* (Roostandy,dkk,2011).

Persamaan fungsi transfer untuk sistem diskrit :

$$y(k) = G(z)u(k) + e(k) \tag{2.16}$$

Fungsi transfer menunjukkan representasi matematis dari hubungan antara satu input dan satu output. Persamaan

berikut menentukan model fungsi transfer diskrit dimana pembilang dan penyebut adalah polinomial

$$G(z) = \frac{b_0+b_1z+\cdots+b_{m-1}z^{m-1}+b_mz^m}{a_0+a_1z+\cdots+a_{m-1}z^{m-1}+a_mz^m} \qquad (2.17)$$

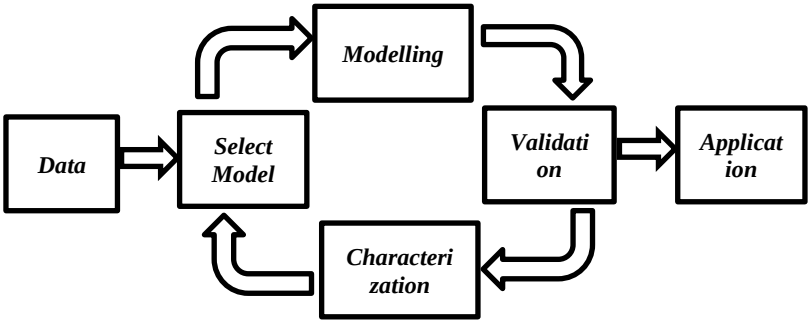
dengan

$y(k)$ = keluaran
$u(k)$ = masukan
$e(k)$ = error
$G(z)$ = fungsi transfer
$a_1, a_2, \ldots, a_n$ = koefisien penyebut
$b_1, b_2, \ldots, b_n$ = koefisien pembilang

c. Validasi model

Setelah model diperoleh, model harus divalidasi untuk mengetahui bagaimana model yang diperoleh merepresentasikan sistem yang sesungguhnya. (Instruments, 2004). Semakin kecil *error* validasi yang diperoleh, semakin baik model yang didapatkan. Validasi dapat dilakukan dengan cara memberikan sinyal masukan kepada sistem dan model dari sistem yang diperoleh.

Langkah-langkah identifikasi sistem akan terus berulang sampai didapatkan hasil pembangunan model yang memuaskan.

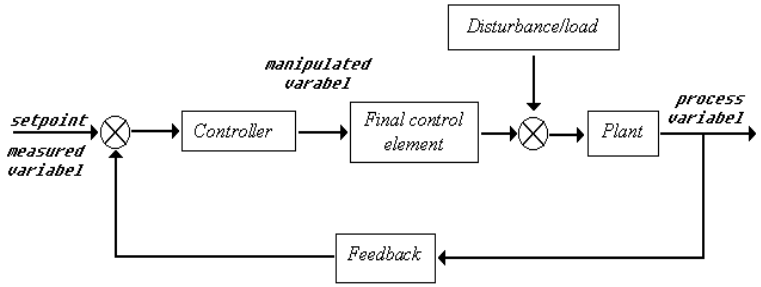


Gambar 2.4 Alur identifikasi sistem (Instruments, 2004)

Berdasarkan pada gambar 2.4, proses identifikasi sistem dimulai dengan memperoleh data dari sistem dan kemudian dianalisis data numerik stimulus dan respon untuk memperkirakan parameter sistem. Kemudian dilakukan pemilihan model dan dilakukan *modelling*, setelah itu melakukan validasi bahwa model yang dihasilkan cocok dengan perilaku sistem yang diamati. Jika hasilnya tidak memuaskan, parameter harus direvisi dan diiterasi.

2.5 Sistem Kendali PI

Sistem pengendalian adalah gabungan dari kerja suatu komponen yang digunakan untuk mempertahankan variabel yang dikendalikan (*process variable*) pada suatu nilai tertentu (*set point*) sehingga sistem dapat dikatakan stabil. Seiring perkembangan ilmu pengetahuan dan teknologi suatu sistem pengendalian sudah memakai unit kendali otomatis seperti pada Gambar 2.5



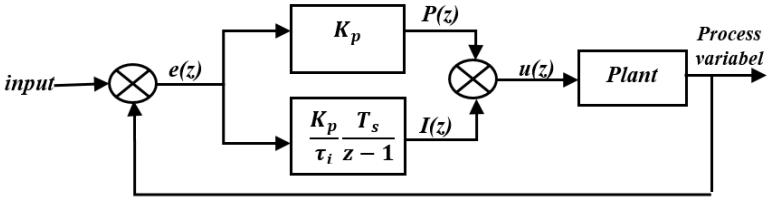
Gambar 2.5 Diagram blok sistem pengendalian tertutup (Basilio, 2002)

Suatu sistem linear (LTI/Linear Time Invariant) dikatakan stabil jika respon natural sistem tersebut mendekati nol pada waktu mendekati tak terhingga. Sistem LTI dikatakan tidak stabil jika respon natural meningkat tanpa batas jika waktu mendekati tak terhingga.

Algoritma pengendali *Proportional-Integral-Derivative* (PID) hingga saat ini dapat dikatakan sebagai algoritma

pengendali terbaik karena keefektifannya, sederhana dalam implementasi dan luas penggunaannya. Karakteristik pengendali PID sangat dipengaruhi oleh kontribusi besaran dari ketiga parameter P, I dan D. Penyetelan konstanta Kp, Ti dan Td akan mengakibatkan penonjolan sifat dari masing – masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat diatur lebih menonjol dibandingkan dengan yang lain, konstanta yang menonjol itulah yang akan memberikan pengaruh pada respon sistem secara keseluruhan

Pengendali PI merupakan penggabungan antara dua macam pengendali, yaitu P (*Propotional*) dan I (*Integral).* Pengendali PI merupakan jenis pengendali untuk menentukan suatu kestabilan atau kepresisian sistem instrumentasi dengan adanya *feedback* atau umpan balik pada sistem tersebut. Diagram blok sistem pengendali PI adalah sebagai berikut :



Gambar 2.6 Diagram blok pengendalian PI

Pengendali P berfungsi untuk mempercepat *rise time* agar respon sistem lebih cepat untuk mencapai *setpoint*, akan tetapi pengendali ini mempunyai kekurangan yaitu meningkatkan *offset*. persamaan pengendali P dalam bentuk transformasi Z adalah sebagai berikut:

$$P(t) = Kp.e(t) \longrightarrow P(k) = Kp.e(k) \longrightarrow P(z) = Kp.e(z) \tag{2.18}$$

dengan

$P(z)$: pengendali P dalam bentuk transformasi z

$e(z)$: sinyal *error* yang merupakan selisih antara *setpoint* dan keluaran proses dalam bentuk transformasiz

Kp : *gain* proportional

Kelemahan dari pengendali P dihilangkan dengan cara menggabungkan dengan pengendali I, selain untuk menghilangkan *offset*, kendali ini mampu mengurangi terjadinya *maksimum overshoot* yang terlalu besar, tetapi kendali jenis I menyebabkan lambatnya respon sistem, yang ditutupi oleh pengendali P.
Persamaan pengendali I dalam bentuk transformasi Z adalah sebagai berikut :

$$I(t) = \frac{Ki}{\tau i}\int_0^t e(t) \longrightarrow \frac{di}{dt} = \frac{Ki}{\tau i}e(t) \longrightarrow \frac{I(k+1)-I(k)}{Ts} = \frac{Ki}{\tau i}e(k) \longrightarrow \frac{z-1}{Ts} = \frac{Ki}{\tau i}e(z) \longrightarrow I(z) = \frac{Ki.Ts}{\tau i(z-1)}e(z) \quad (2.19)$$

Sehingga kendali PI menghasilkan respon yang lebih cepat dari pengendali I tapi mampu menghilangkan *offset* yang ditinggalkan pengendali P.
Persamaan pengendali PI adalah sebagai berikut :

$$u(z) = Kp.e(z) + \frac{Ki.Ts}{\tau i(z-1)}e(z) \quad (2.20)$$

dengan
τi = konstanta waktu integral.
Ts = Waktu cuplik

Jika pemodelan matematis sistem susah untuk dilakukan. maka perancangan pengendali PID secara analitis tidak mungkin dilakukan, sehingga perancangan pengendali PID harus dilakukan secara eksperimental. Perancangan pengendali PID secara eksperimental dengan metode *Ziegler –Nichols* diberikan aturan untuk menentukan nilai Kp, τ_i dan τ_d yang didasarkan pada karakteristik respon transien dari sistem. (Basilio, 2002)

Beberapa parameter dalam algoritma PID yang juga harus diketahui dalam suatu sistem *closed loop* yaitu *maximum overshoot*, *settling time*, dan *error steady state*.

a. *maximum* (*percent*) *overshoot* adalah nilai puncak tertinggi respon sistem pengukuran terhadap keadaan yang diinginkan. Jika nilai *steady state* dari respon berbeda dengan nilai yang diinginkan, umumnya menggunakan *maximum percent overshoot* dengan persamaan :

$$Maximum\ percent\ overshoot = \frac{c(t_p)-c(\infty)}{c(\infty)} \times 100\% \quad (2.21)$$

b. *Settling time* adalah waktu yang dibutuhkan respon sistem untuk mencapai dan berada direntang ±5% atau ±2% dari keadaan yang di inginkan atau nilai akhir. (Ogata, 2010)
c. *error steady state* adalah nilai *error* yang dihasilkan dari keadaan *steady* ketika belum mencapai *set point* yang diinginkan. (Ardhiantama, 2016)

Selain itu hal-hal yang perlu diperhatikan pada nilai dari Kp dan Ki adalah sebagai berikut :

a. Nilai gain Kp yang terlalu besar akan mengakibatkan sistem menjadi semakin sensitif dan cenderung tidak stabil. Jika nilai Kp terlalu kecil maka akan meyebabkan *offset* yang besar.
b. Nilai dari Ti yang kecil akan menghilangkan *offset* tetapi cenderung membuat sistem menjadi lebih sensitif atau mudah berosilasi sedangkan Ti yang besar belum tentu efektif menghilangkan *offset* dan cenderung membuat respon menjadi lambat. (Maulana, 2012)

2.6 Sistem *Active Fault Tolerant Control (AFTC)*

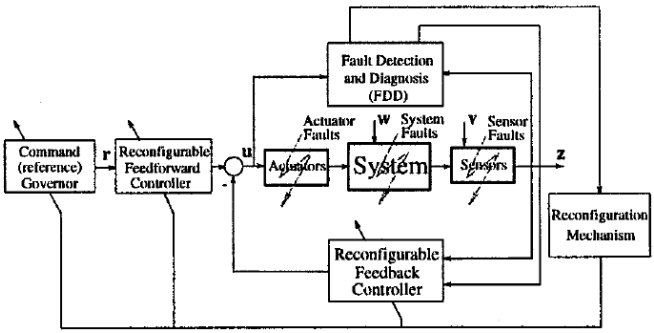
Peningkatan performa dan keamanan sangat dibutuhkan dalam suatu sistem kendali. Pada industri proses, manufaktur, dan lainnya terkadang terjadi beberapa kerusakan pada aktuator, sensor ataupun komponen-komponen yang menyusun sistem tersebut. Untuk mengatasi kerusakan tersebut, dilakukan desain pengendalian yang mampu menoleransi kerusakan yang terjadi dengan tetap menjaga stabilitas sistem yang diinginkan. Sistem *Fault Tolerant Control* (FTC) adalah sebuah pengendali yang mampu mentoleransi kesalahan dalam suatu sistem untuk meningkatkan kinerja yang diinginkan. (Zhang & Jiang, 2003)

FTC dapat diklasifikasikan pada 2 tipe yaitu sistem *Passive Fault Tolerance Control* (PFTC) dan sistem *Active Fault Tolerance Control* (AFTC). Sistem PFCT didesain untuk menjadi sistem pengendali yang *robust* dari kesalahan suatu komponen. Sedangkan sistem AFTC bereaksi terhadap kesalahan yang terjadi pada suatu komponen dengan merekonfigurasi aksi pengendali sehingga kestabilan dan kinerja pada sistem dapat dijaga. Dengan demikian, tujuan utama dari sistem FTC adalah merancang pengendali dengan struktur yang cocok untuk mencapai kestabilan dan kinerja yang diinginkan. Tidak hanya ketika komponen pengendali berfungsi secara normal tetapi ketika terjadi kesalahan pada suatu komponen tersebut (Indriwati, dkk, 2015)

Perbedaan sistem AFTC dan sistem PFTC adalah terletak pada perancangan *Reconfigurable Control* (RC) dan FDD. Sehingga kunci utama pada AFTC adalah merancang pengendali yang dapat di atur ulang sehingga mampu mentoleransi kesalahan yang terjadi (Zhang & Jiang, 2003)

Menurut Zhang dan Jiang (2003), AFTC secara umum terdiri atas 4 sub sistem, yaitu:

a. *Reconfigurable Control* (RC)
b. *Fault Detection Diagnosis* (FDD) *Scheme*
c. *Reconfiguration Mechanishm*
d. *Command reference governor*.



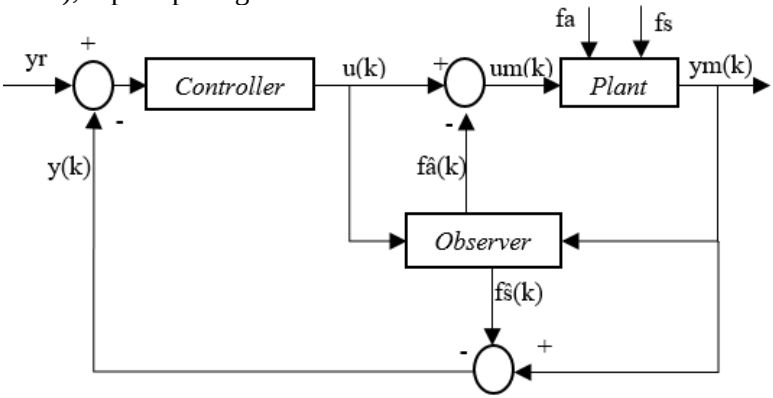
Gambar 2.7 Struktur umum sistem AFTC (Zhang & Jiang, 2008)

Struktur umum dari AFTC dapat dilihat pada gambar 2.7. Berdasarkan Gambar 2.7 diketahui skema FDD diperlukan untuk mengestimasi state yang tidak terukur seperti kesalahan yang terjadi pada sistem, sehingga diharapkan FDD memiliki sensitivas yang tinggi terhadap kesalahan, kemudian terdapat mekanisme pengaturan ulang (*Reconfigurable Controller*) yang sebisa mungkin mampu memulihkan performansi sistem setelah terjadi kesalahan walaupun terdapat ketidakpastian dan *time delay* pada FDI . *Reconfigurable control* harus dapat bekerja secara otomatis untuk menjaga stabilitas dan performansi yang diinginkan. *Reconfiguration mechanishm* digunakan untuk memperbaiki ulang kesalahan awal pada performansi sistem dan untuk menghindari atau mempertimbangkan akan terjadinya penurunan performansi setelah terjadinya kesalahan. Pada intinya, perancangan sistem yang paling penting dalam AFTC terdiri atas 2 sub sistem yaitu *fault detection and identification* (FDI) dan *reconfigurable control*. (Zhang & Jiang, 2003)

Fault Detection and Identification (FDI) disebut juga dengan *Fault Detection and Diagnosis* (FDD) merupakan suatu tahapan untuk mendeteksi *fault* atau kesalahan yang terjadi pada aktuator, sensor, maupun komponen penyusun dalam sistem. Tujuan dari pendeteksian kesalahan ini agar sinyal pengendali yang baru dapat menyampaikan informasi pada pengendali untuk menoleransi kesalahan yang terjadi. Dalam FDI ada 3 hal yang perlu dilakukan yaitu; mendeteksi kesalahan pada sistem serta waktu selang selama terjadinya kesalahan, mengisolasi lokasi kesalahan dan tipe kesalahan yang terjadi dan yang terakhir adalah mengidentifikasi besarnya kesalahan yang terjadi. Untuk itu juga tahapan FDD juga biasa disebut dengan FDI. *Observer* merupakan salah satu jenis FDI yang berfungsi untuk mengestimasi kesalahan berdasarkan model matematik (Indriawati, dkk, 2013)

Untuk merancang sistem kendali yang toleran terhadap kesalahan aktuator dan sensor, maka dilakukan konfigurasi ulang sinyal pengendali atau *reconfigurable control* secara otomatis berdasarkan jenis kesalahan yang terjadi. Konfigurasi ulang

sinyal kendali untuk mengatasi kesalahan pada aktuator dan sensor memerlukan proses diagnosis kesalahan (*fault diagnosis*) untuk mendeteksi dan mengidentifikasi (isolasi) kesalahan yang terjadi sehingga *reconfigurable control* akan bekerja secara otomatis agar dapat memperbaiki kesalahan sehingga sistem tetap beroperasi sesuai keinginan, berdasarkan hal tersebut sejumlah kesalahan yang terjadi diestimasi dan hasil estimasi tersebut digunakan untuk mengatur ulang sinyal kendali (Indriawati, dkk, 2013), seperti pada gambar 2.8



Gambar 2.8 Mekanisme *reconfigurable Control*

2.7 *Observer*

Observer adalah suatu algoritma yang digunakan untuk mengestimasi keadaan (*state*) dari variabel yang tidak terukur berdasarkan model matematis sistem, sehingga observer juga dapat digunakan untuk mengestimasi kesalahan yang terjadi pada suatu sistem jika kesalahan tersebut masuk dalam pemodelan. Suatu sistem umumnya dimodelkan dalam bentuk *state space*, yang dapat didefiniskan secara umum seperti persamaan sistem dalam bentuk diskrit dibawah ini (Ogata, 1987):

$$x(k+1) = Ax(k) + Bu(k) \quad (2.22)$$
$$y(k) = Cx(k) + Du(k) \quad (2.23)$$

dengan:

A = Matriks keadaan (*state*)
B = Matriks kendali
C = Matriks keluaran
D = Matriks gangguan
$u(k)$ = Masukan kendali
$x(k+1)$ = Turunan vektor keadaan
$y(k)$ = Vektor keluaran
$x(k)$ = Vektor keadaan (*state*)

Model matematis dari *observer* sama seperti model matematis pada sistem dengan ditambahkan K_e (*observer gain matrix)* menjadi persamaan sebagai berikut

$$\hat{x}(k+1) = \mathrm{A}\,.\,\hat{x}(k) + \mathrm{B}.u(k) + K_e\,(y(k) - \hat{y}(k)) \quad (2.24)$$
$$\hat{y}(k) = C\hat{x}(k) \quad (2.25)$$

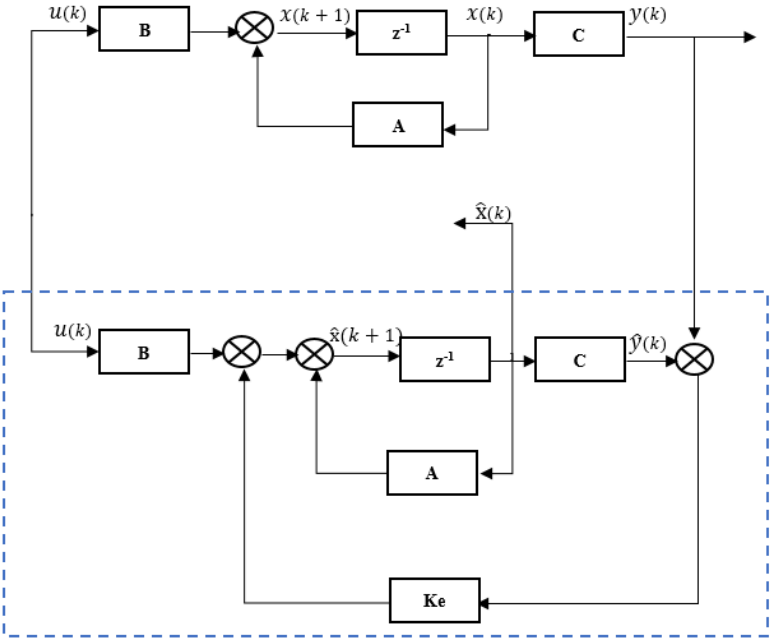
dengan :
$\hat{x}(k+1)$ = Estimasi *state* $x(k+1)$
K_e = *Gain Observer*
$\hat{x}(k)$ = Estimasi *state* $x(k)$
$\hat{y}(k)$ = Estimasi $y(k)$

$y(k)$ pada persamaan (2.23) dan $\hat{y}(k)$ pada (2.25) disubstitusikan ke dalam persamaan (2.26), maka diperoleh

$$\hat{x}(k+1) = \mathrm{A}\,.\,\hat{x}(k) + \mathrm{B}.u(k) + K_e(Cx(k) - C\hat{x}(k)) \quad (2.26)$$

persamaan *error observer* atau $(e(k+1))$ didapatkan dari hasil pengurangan persamaan (2.26) dengan persamaan (2.22), menjadi :

$$e(k+1) = x(k+1) - \hat{x}(k+1) \quad (2.27)$$
$$e(k+1) = Ax(k) - \mathrm{A}\hat{x}(k) - K_e\,(Cx(k) - C\hat{x}(k)) \quad (2.28)$$
$$e(k+1) = (A - K_e\,C)(x(k) - \hat{x}(k)) \quad (2.29)$$
$$e(k+1) = (A - K_e\,C)e(k) \quad (2.30)$$



Gambar 2.9 *Observer* (Ogata, 1987)

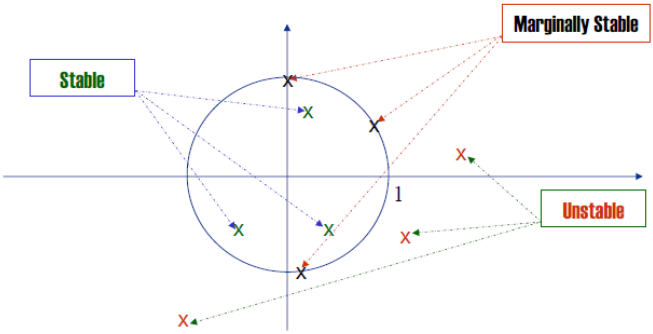
Berdasarkan gambar 2.9, *observer* mengandung nilai *gain observer* atau K_e . *Gain observer* merupakan matriks untuk mengoreksi perbedaan antara keluaran y aktual dan y estimasi sehingga dapat digunakan untuk mengestimasi *state* yang tidak terukur (Ogata, 1970). Untuk menentukan nilai K_e sangat dipengaruhi oleh nilai letak *pole*, seperti pada gambar 2.10 untuk sistem dengan domain diskrit, posisi nilai *pole* untuk menyatakan sistem stabil berada tidak lebih dari 1 atau -1 pada sumbu imajiner maupun sumbu riil. Nilai K_e diperoleh berdasarkan persamaan 2.30 , dengan

$$e(k) = (x(k) - \hat{x}(k)) \tag{2.31}$$

Kemudian dari persamaan (2.30) menjadi persamaan karakteristik observer yang dituliskan

$$|zI - A + K_e C| = 0 \qquad (2.32)$$

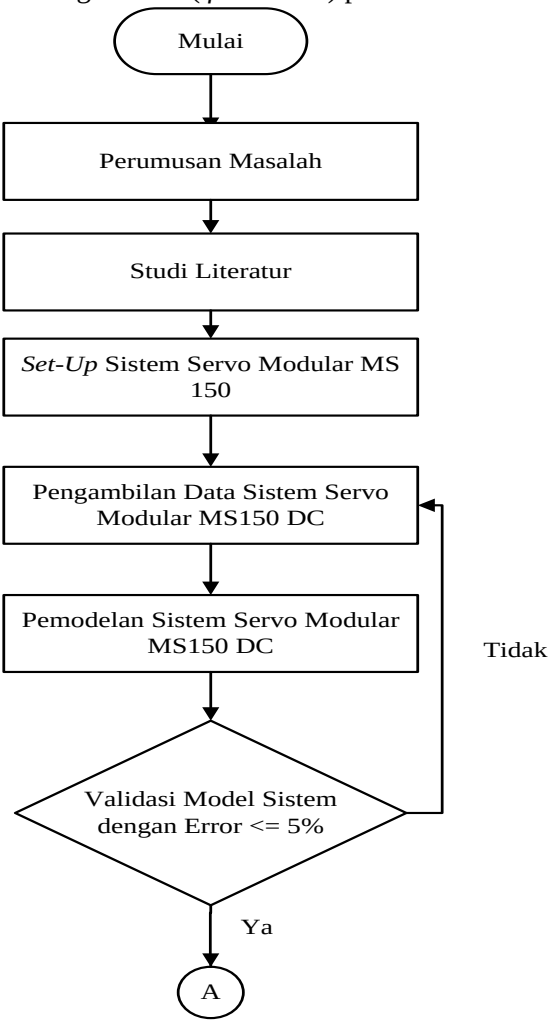
maka dengan menentukan nilai p atau posisi *pole*, diperoleh persamaan karakteristik yang diinginkan, maka nilai K_e diperoleh dari mensetarakan persamaan (2.32) = persamaan karakteristik yang diinginkan. (Ogata, 1970)



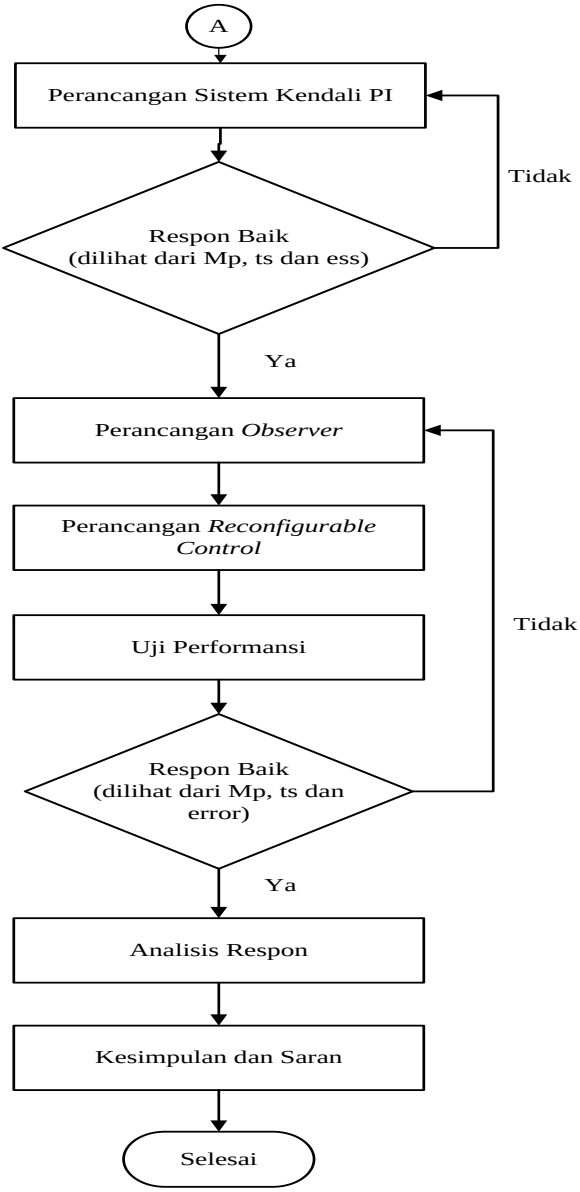
Gambar 2.10 Kestabilan *pole placement* sistem diskrit (M. Chen, 2016)

BAB III
METODOLOGI PENELITIAN

Tahapan yang dilakukan dalam Tugas Akhir ini ditampilkan dengan sebuah diagram alir (*flowchart*) pada Gambar 3.1



Gambar 3.1 Diagram Alir Penelitian Tugas Akhir



Gambar 3.1 (Lanjutan)

3.1 Perumusan Masalah

Kegagalan sistem pada motor DC karena adanya gangguan berupa kesalahan pada aktuator dan sensor menyebabkan kerugian diberbagai bidang. Seperti pada bidang industri, karena menyebabkan terhambatanya proses produksi perusahaan yang dapat merugikan perusahaan, selain itu, pada mobil listrik, kegagalan motor DC untuk bekerja karena adanya gangguan pada kesalahan aktuator dan sensor juga dapat menimbulkan kecelakaan yang dapat membahayakan pengendara maupun masyarakat sekitar. Sehingga dari permasalahan tersebut diperlukan sistem kendali yang mampu mengatasi masalah berupa kesalahan pada komponen aktuator dan sensor secara otomatis berupa sistem *Active Fault Tolerant Control.*

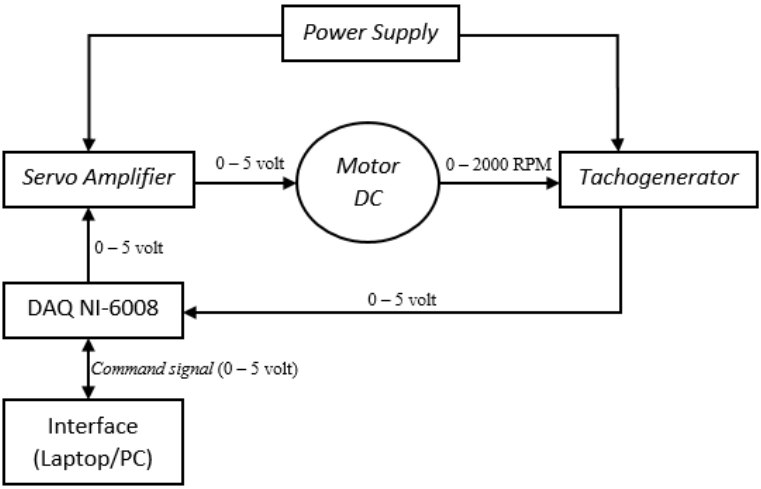
3.2 Studi Literatur

Studi literatur dimaksud untuk membangun pemahaman awal hingga mendalam secara teoritis terhadap materi yang mendukung pada penelitian tugas akhir ini antara lain pemahaman mengenai sistem servo modular MS150 DC, identifikasi sistem, pemahaman sistem kendali PID, sistem *Active Fault Tolerant Control (AFTC),* berupa *Observer* dan *reconfigurable control*.

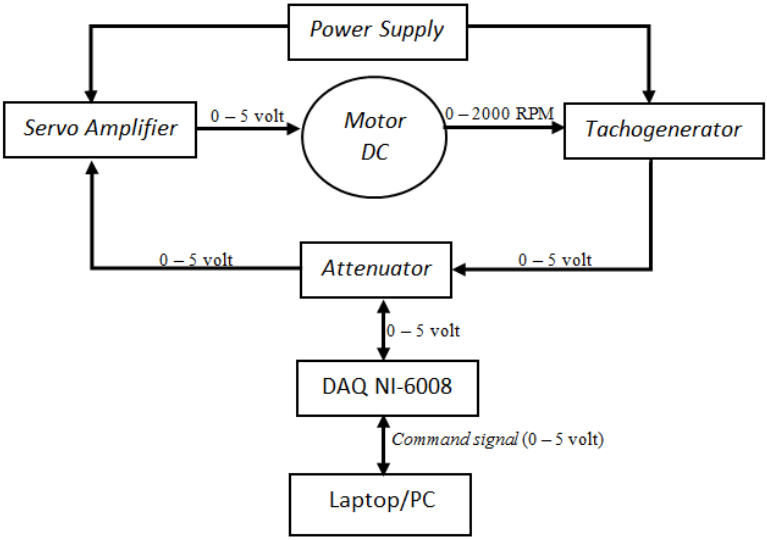
3.3 *Set-Up* Sistem Servo Modular MS150 DC

Pengendalian kecepatan sistem servo modular MS150 DC dapat dilakukan dengan langkah awal *set-up* sistem servo modular MS150 DC, sehingga sistem servo modular MS150 DC dapat bekerja dengan benar. Berdasarkan gambar 3.3, sistem kendali berada pada komputer/laptop atau *user interface*. Sistem kendali berfungsi untuk mengatur sinyal masukan pada motor DC, karena pengendalian kecepatan motor DC dipengaruhi oleh besarnya tegangan masukan yang diberikan. Untuk dapat menjalin komunikasi antara komputer dengan plant maka diperlukan sebuah antarmuka yaitu DAQ NI-6008, selain itu DAQ NI-6008 juga diperlukan sebagai media pengkonversi sinyal *digital-to-analog*. DAQ NI-6008 memiliki karakteristik dapat

menerima dan mengirimkan sinyal dengan rentang 0-5 volt, sehingga dari DAQ NI-6008, sinyal akan dikirim menuju ke *servo amplifier* dengan rentang 0-5 volt. *Servo amplifier* mengirimkan sinyal ke motor DC dengan rentang 0-5 volt sehingga menyebabkan motor DC berputar. Kemudian motor DC berputar dengan rentang kecepatan rotasi sebesar 0-2000 RPM yang selanjutnya akan dikonversi menjadi sinyal tegangan dengan rentang 0–5 volt oleh *tachogenerator*. Keluaran dari *tachogenerator* dihubungkan ke DAQ NI 6008 sebagai data masukan berupa kecepatan rotasi motor DC, selain data tegangan kecepatan motor DC, data arus dalam bentuk tegangan yang diperoleh dari *servo amplifier* juga dapat diukur menggunakan *set-up* ini.



Gambar 3.2 Diagram blok *set-up* sistem servo modular MS150 DC



Gambar 3.3 Diagram blok *set-up* sistem servo modular MS150 DC dengan *attenuator*

Ketika kesalahan pada aktuator dan sensor terjadi secara *realtime*, maka dapat digunakan *attenuator* sebagai bentuk kesalahan yang terjadi pada sistem servo modular MS150 DC, seperti diagram blok pada gambar 3.4, untuk kesalahan pada aktuator, maka sinyal dari DAQ-NI 6008 masuk kedalam *attenuator* yang selanjutnya sinyal masuk kedalam *servoamplifier*. Untuk kesalahan pada sensor, sinyal dari *tachogenerator* masuk kedalam *attenuator* yang selanjutnya masuk kedalam DAQ NI 6008, dan kemudian kembali ke laptop/PC sebagai data masukan atau *process variable*. *Attenuator* berfungsi sebagai hambatan bagi sinyal sehingga, nilai sinyal antara sebelum dan sesudah *attenuator* berbeda. Perbedaan antara kedua sinyal tersebut yang dimaksud sebagai nilai kesalahan pada aktuator dan sensor.

3.4 Identifikasi Sistem Servo Modular MS150 DC

Identifikasi sistem dilakukan dengan langkah – langkah sebagai berikut :

3.4.1 Pengambilan data sistem

Data-data yang dibutuhkan untuk identifikasi sistem berupa data kecepatan dalam bentuk tegangan yang dikonversi oleh *tachogenerator* dan data arus dalam bentuk tegangan dari *servoamplifier*. Untuk mengkonversi data dalam bentuk tegangan ke bentuk kecepatan rotasi dalam satuan rpm, maka menggunakan persamaan konversi yang diperoleh dari pengukuran kecepatan sistem servo modular MS150 DC secara *open loop* sebagai berikut :

$$\omega = V.365 \tag{3.1}$$

dengan

ω = Kecepatan rotasi (rpm)
V = Tegangan (V)

Data kecepatan dan arus diperoleh dengan cara membuat program seperti pada lampiran B dan lampiran C pada perangkat lunak LabVIEW 2013 dengan memberikan sinyal masukan berupa sinyal *Pseudo Random Binary Sequences* (PRBS), kemudian data kecepatan rotasi motor DC dan data arus diambil selama 5 menit dengan waktu cuplik sebesar 1 milidetik, selanjutnya program yang telah dibuat dihubungkan dengan sistem servo modular MS150 DC yang terdapat di ruang laboratorium rekayasa instrumentasi dan kontrol Teknik Fisika ITS dan data disimpan dalam format .LVM. Agar diperoleh model yang tepat maka dalam pemilihan sinyal uji ini tidak boleh sembarangan. Syarat pemilihannya adalah suatu sinyal uji harus memiliki cakupan frekuensi yang lebar dan standard yang digunakan adalah sinyal PRBS (Nusantoro, dkk,2012). Selain itu pemilihan sinyal PRBS ini disebabkan kemampuan PRBS untuk memberikan sinyal dengan rentang amplitudo 0 dan 1 volt secara acak, sehingga baik digunakan untuk mengetahui karakteristik dari sistem.

3.4.2 Pemodelan Sistem Servo Modular MS150 DC

Setelah data masukan dan keluaran dari *plant* diperoleh, selanjutnya dilakukan pemilihan model dengan menggunakan metode parametrik berupa model persamaan fungsi transfer. Model dengan menggunakan metode parametrik dipilih karena pada teknik ini nilai parameter dari suatu sistem dapat diperoleh secara langsung dan merupakan solusi yang tepat jika terkait dengan sistem riil, dan sampai saat ini model dengan metode parametrik lebih banyak digunakan dalam aplikasi riil. (Roostandy, dkk, 2013).

Selain itu metode parametrik mampu menghasilkan model yang lebih akurat jika dibandingkan dengan metode nonparametrik, tetapi dengan syarat karakteristik model dari sistem yang diinginkan sudah diketahui, seperti derajat atau *orde* dari sistem, dan waktu cuplik dari sistem (Instruments, 2004)

Studi literatur mengenai bentuk persamaan ruang keadaan (*state-space*) yang diperlukan untuk kecepatan motor DC seperti pada persamaan (2.14) dan persamaan (2.15), dan bentuk fungsi transfer yang dapat dirubah ke dalam bentuk persamaan ruang keadaan seperti pada persamaan (2.14) dan (2.15), studi literatur mengenai model untuk kecepatan pada motor DC digunakan sebagai dasar memilih orde denumerator dan numerator pada program LabVIEW 2013.

Selanjutnya, dari pemodelan dengan menggunakan perangkat lunak labVIEW 2013 yang dapat dilihat pada lampiran D diperoleh nilai fungsi transfer dalam domain z. Sistem yang dibuat merupakan sistem dengan domain diskrit berupa persamaan dalam bentuk fungsi transfer orde 1 dengan struktur fungsi transfer seperti pada persamaan 2.17.

Persamaan fungsi transfer orde 1 untuk motor DC memiliki nilai sebagai berikut :

a. Kecepatan = $\omega(z) = \frac{0.435322}{z-0.844792} u(z)$ (3.2)

b. Arus = $i(z) = \frac{0.0145632}{z-0.732663} u(z)$ (3.3)

Persamaan (3.2) dan (3.3) dirubah kedalam persamaan dengan domain K yang selanjutnya dirubah ke dalam bentuk persamaan ruang keadaan (*state space)*, yang dituliskan :

$$x(k+1) = Ax(k) + Bu(k) \quad (3.4)$$
$$y(k) = Cx(k) + Du(k) \quad (3.5)$$

d. Kecepatan

$$\omega(z) = \frac{0.435322}{z-0.844792}u(z) \quad (3.6)$$
$$\omega(z)(z - 0.844792) = 0.435322\, u(z) \quad (3.7)$$
$$z\omega(z) - 0.844792\omega(z) = 0.435322\, u(z) \quad (3.8)$$
$$z\omega(z) = 0.844792\omega(z) + 0.435322\, u(z) \quad (3.9)$$

persamaan (3.9) di transformasi balik, menjadi persamaan

$$\omega(k+1) = 0.844792\omega(k) + 0.435322u(k) \quad (3.10)$$

e. Arus

$$i(z) = \frac{0.0145632}{z-0.732663}\, u(z) \quad (3.11)$$
$$i(z)(z - 0.732663) = 0.0145632\, u(z) \quad (3.12)$$
$$zi(z) - 0.732663I(z) = 0.0145632\, u(z) \quad (3.13)$$
$$zi(z) = 0.732663I(z) + 0.0145632\, u(z) \quad (3.14)$$

persamaan (3.14) di transformasi balik, menjadi persamaan

$$\dot{\imath}(k+1) = 0.732663i(k) + 0.0145632u(k) \quad (3.15)$$

Dari persamaan (3.10) dan (3.15) diperoleh bentuk persamaan ruang keadaan untuk sistem servo modular MS150 DC dalam domain K:

$$\begin{bmatrix}\omega(k+1)\\ \dot{i}(k+1)\end{bmatrix} = \begin{bmatrix}0.844792 & 0\\ 0 & 0.732663\end{bmatrix}\begin{bmatrix}\omega(k)\\ i(k)\end{bmatrix} + \begin{bmatrix}0.435322\\ 0.0145632\end{bmatrix}u(k) \quad (3.16)$$

$$y(k) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega(k) \\ i(k) \end{bmatrix} + 0 \quad (3.17)$$

3.4.3 Validasi

Validasi pemodelan dari motor DC, *tachogenerator* dan *servoamplifier* dilakukan untuk memastikan model yang diperoleh sudah merepresentasikan sistem dalam kondisi *riil*.

Validasi dilakukan dengan cara membandingkan respon antara sistem riil dengan model fungsi transfer dari sistem, dengan nilai setpoint yang diberikan sebesar 1 volt. Sehingga, dari nilai sepoint 1 volt, diperoleh sinyal kendali yang masuk kedalam sistem dan model sistem, yang menyebabkan respon antara sistem riil dengan model sama. Program validasi sistem motor DC, *tachogenerator* dan *servo amplifier* pada perangkat lunak labVIEW 2013 dapat dilihat pada lampiran E. Selanjutnya, untuk memperoleh nilai *error* antara model dengan sistem riil menggunakan persamaan sebagai berikut :

$$\text{MAPE} = \left(\frac{100\%}{n}\right) \sum_{t=1}^{n} \frac{|X_t - F_t|}{X_t} \quad (3.18)$$

dengan

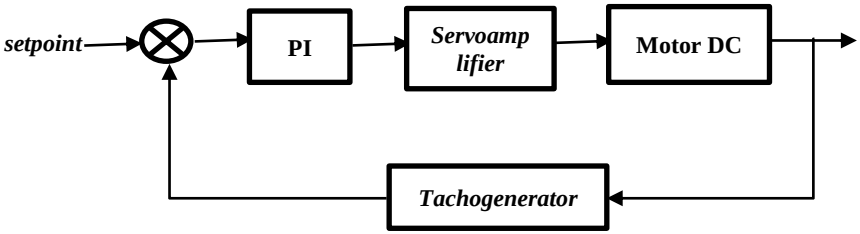
X_t = Data aktual pada periode t
F_t = Data pemodelan pada periode t
n = jumlah data

nilai *error* validasi diperoleh dengan menggunakan persamaan MAPE (*Mean Absolute Percentage Error)* seperti pada persamaan 3.18. MAPE merupakan salah satu metode yang dapat digunakan untuk menghitung validasi dengan menggunakan kesalahan absolut pada tiap periode dibagi dengan nilai data yang nyata untuk periode itu yang selanjutnya hasil pengurangan antara nilai riil dengan model dijumlahkan dan dibagi banyak data, yang kemudian dikalikan 100% sehingga error yang diperoleh dalam bentuk prosentase. MAPE merupakan pengukuran kesalahan yang menghitung ukuran presentase penyimpangan antara data aktual dengan data pemodelan. Batas *error* validasi maksimal yang digunakan sebesar 5%. Pemilihan nilai batas untuk *error*

validasi tidak memiliki dasar yang pasti, namun semakin kecil nilai *error* yang diperoleh maka semakin bagus hasil pemodelan, selain itu hasil model yang diperoleh sudah cukup mereprsentasikan sistem dalam kondisi riil, maka model dapat digunakan.

3.5 Perancangan Sistem Kendali PI

Karakteristik pengendali PI sangat dipengaruhi oleh kontribusi besaran dari ketiga parameter P, dan I. Penyetelan konstanta Kp dan Ti akan mengakibatkan penonjolan sifat dari masing – masing elemen. Diagram blok sistem pengendalian sistem servo modular MS150 DC ditunjukkan oleh gambar 3.5



Gambar 3.4 Diagram blok sistem pengendalian

Dalam pengendalian kecepatan motor DC dilakukan perancangan sistem kendali PI. Nilai Kp, Ki didapatkan secara eksperimen dengan metode *Ziegler–Nichols* untuk perancangan secara simulasi dengan *simulink* pada perangkat lunak matlab R2103a, kemudian setelah sistem kendali dapat bekerja dengan baik, dilakukan perancangan sistem kendali PI untuk diterapkan secara *realtime* pada sistem servo modular MS150 DC dengan menggunakan perangkat lunak LabVIEW 2013. Nilai *gain* Kp dan Ki untuk simulasi terkadang tidak tepat untuk digunakan secara *realtime* jika perangkat lunak yang digunakan berbeda, sehingga diperlukan modifikasi agar diperoleh nilai *gain* yang sesuai untuk diterapkan secara *realtime.*

Perancangan sistem kendali PI bertujuan untuk mengetahui respon dari sistem jika ditambahkan dengan pengendali yang

telah didapatkan nilai *gain*nya serta untuk mengetahui kesesuaian kerja sistem secara keseluruhan. Sistem yang dirancang telah bekerja dengan baik jika respon sistem dapat mencapai nilai *setpoint* yang telah diberikan. Perancangan pengendali PI dengan menggunakan simulink pada perangkat lunak matlab R2013a dapat dilihat pada lampiran F ,dan secara *realtime* pada perangkat lunak labVIEW 2013 dapat dilihat pada lampiran G.

3.6 Perancangan Sistem *Active Fault Tolerant Control (AFTC)* untuk Kasus Kesalahan Simulasi

3.6.1 Perancangan *Observer*

Perancangan *observer* dilakukan untuk mengestimasi kesalahan yang ada, dengan cara mengembangkan persamaan ruang keadaan sistem. Persamaan untuk perancangan *observer* diperoleh dari penelitian yang telah dilakukan (Indriawati, dkk, 2015), tetapi perubahan dilakukan dari betuk domain kontinu ke dalam bentuk domain diskrit. Nilai matriks ruang keadaan untuk perancangan *observer* diperoleh dari pemodelan sistem servo modular MS150 DC pada persamaan (3.16) dan (3.17) :

$$A = \begin{bmatrix} 0.844792 & 0 \\ 0 & 0.732663 \end{bmatrix}$$
$$B = \begin{bmatrix} 0.435322 \\ 0.0145632 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$Fs = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$Fa = \begin{bmatrix} 0.435322 \\ 0.0145632 \end{bmatrix}$$

Persamaan ruang keadaan yang mengandung kesalahan aktuator dan sensor dituliskan sebagai berikut :

$$x(k+1) = Ax(k) + Bu(k) + F_a f_a(k) \quad (3.19)$$
$$y(k) = Cx(k) + Fs.f_s(k) \quad (3.20)$$

Dengan, $x(k) \in R^n$, $u(k) \in R^m$, $y(k) \in R^p$, $f_a(k) \in R^r$, $f_s(k) \in R^n$ berturut-turut adalah keadaan (*state*), masukan, keluaran, kesalahan aktuator dan kesalahan sensor. Sedangkan A, B, C, F_a, F_s adalah matriks keadaan, masukan, keluaran, kesalahan aktuator dan kesalahan sensor.

Kemudian *state* baru $z(k+1)$ ditambahkan untuk memindahkan kesalahan sensor dari persamaan keluaran ke persamaan keadaan

$$z(k+1) = A_z T_s(y(k) - z(k)) + z(k) \tag{3.21}$$

dimana persamaan (3.20) disubstitusi ke dalam persamaan (3.21), sehingga diperoleh persamaan :

$$z(k+1) = -A_z T_s z(k) + A_z T_s\big(Cx(k) + Fs.f_s(k)\big) + z(k) \tag{3.22}$$

persamaan (3.19), (3.22) diubah ke dalam bentuk persamaan yang lebih kompak menjadi

$$\begin{bmatrix} x(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ A_z C & -A_z T_s + I \end{bmatrix} \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} F_a & 0 \\ 0 & AzT_s Fs \end{bmatrix} \begin{bmatrix} f_a(k) \\ f_s(k) \end{bmatrix} \tag{3.23}$$

$$y(k) = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ z(k) \end{bmatrix} \tag{3.24}$$

dengan nilai :
Az = 1000, Ts = 0.001

kemudian bentuk ruang keadaan (3.30) dan (3.31) dirubah kedalam bentuk persamaan keadaan dan persamaan keluaran sistem yang diperluas (mengandung kesalahan) sebagai berikut

$$\bar{x}(k+1) = \bar{\mathrm{A}}\,.\,\bar{x}(k) + \bar{\mathrm{B}}.u(k) + \bar{F}\bar{f}(k) \tag{3.25}$$

$$z(k) = \bar{y}(k) = \bar{C}\bar{x}(k) \tag{3.26}$$

dengan demikian diperoleh nilai matriks

$$\bar{A} = \begin{bmatrix} 0.8448 & 0 & 0 & 0 \\ 0 & 0.7327 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} 0.4353 \\ 0.0146 \\ 0 \\ 0 \end{bmatrix}$$

$$\bar{F} = \begin{bmatrix} 0.4353 & 0 \\ 0.0146 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\bar{C} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Setelah nilai-nilai matriks didapatkan, persamaan keadaan dan persamaan keluaran yang mengandung kesalahan aktuator dan sensor digunakan untuk merancang persamaan *observer* yang dituliskan:

$$\hat{\bar{x}}(k+1) = \bar{\mathrm{A}}\,.\hat{\bar{x}} + \bar{\mathrm{B}}.u(k) + \bar{F}\hat{\bar{f}}(k) + K_e(\bar{y}(k) - \hat{\bar{y}}(k)) \quad (3.27)$$

$$\hat{\bar{y}}(k) = \bar{C}\hat{x}(k) \quad (3.28)$$

Kemudian ditambahkan *state* yang mengandung gain untuk mengestimasi kesalahan aktuator dan sensor (L),

$$\hat{f}(k+1) = T_s L_e(k) + \hat{f}(k) \quad (3.29)$$

Sehingga dari persamaan (3.28) dan (3.30) jika diubah kedalam bentuk yang lebih kompak, menjadi

$$\begin{bmatrix} \hat{\bar{x}}(k+1) \\ \hat{f}(k+1) \end{bmatrix} = \begin{bmatrix} \bar{\mathrm{A}} & \bar{F} \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{\bar{\mathrm{x}}}(k) \\ \hat{f}(k) \end{bmatrix} + \begin{bmatrix} \bar{B} \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} K \\ LT_s \end{bmatrix} e(k) \quad (3.30)$$

$$\hat{\bar{y}}(k) = \begin{bmatrix} \bar{C} & 0 \end{bmatrix} \begin{bmatrix} \hat{\bar{\mathrm{x}}}(k) \\ \hat{f}(k) \end{bmatrix} \quad (3.31)$$

dari bentuk (3.30) dan (3.31) dapat dirubah menjadi persamaan keadaan dan persamaan keluaran *observer*, sebagai berikut :

$$\hat{\tilde{\mathrm{x}}}(k+1) = \tilde{\mathrm{A}}.\hat{\tilde{\mathrm{x}}}(k) + \tilde{\mathrm{B}}.\mathrm{u}(k) + \widetilde{K_e}(\tilde{y}(k) - \hat{\tilde{y}}(k)) \quad (3.32)$$

$$\hat{\tilde{y}}(k) = \tilde{C}.\hat{\tilde{x}}(k) \quad (3.33)$$

dengan memasukan nilai matriks $\bar{A}, \bar{B}, \bar{C}, \bar{F}, I, Ts$ maka, diperoleh nilai matriks sebagai berikut :

$$\tilde{A} = \begin{bmatrix} 0.8448 & 0.0000 & 0.0000 & 0.0000 & 0.4353 & 0.0000 \\ 0.0000 & 0.7327 & 0.0000 & 0.0000 & 0.0146 & 0.0000 \\ 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} 0.3919 \\ 0.0146 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\tilde{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Sehingga dari persamaan observer yang telah diperoleh, untuk menentukan nilai *error observer* sebagai berikut :

$$e(k+1) = \tilde{x}(k+1) - \hat{\tilde{x}}(k+1) \quad (3.34)$$

$$e(k+1) = \left(\tilde{A}\tilde{x}(k) + \tilde{B}u(k)\right) - (\tilde{\mathrm{A}}.\hat{\tilde{\mathrm{x}}}(k) + \tilde{\mathrm{B}}.\mathrm{u}(k) + \widetilde{K_e}\left(\tilde{y}(k) - \hat{\tilde{y}}(k)\right)) \quad (3.35)$$

$$= \tilde{A}\tilde{x}(k) - \tilde{\mathrm{A}}.\hat{\tilde{\mathrm{x}}}(k) - \widetilde{K_e}\left(\tilde{C}.\tilde{x}(k) - \tilde{C}.\hat{\tilde{x}}(k)\right) \quad (3.36)$$

$$= (\tilde{A} - \widetilde{K}_e\tilde{C})(\tilde{x}(k) - \hat{\tilde{x}}(k)) \quad (3.37)$$

dengan,

$$e(k) = \tilde{x}(k) - \hat{\tilde{x}}(k) \tag{3.38}$$

sehingga,

$$e(k+1) = (\tilde{A} - \tilde{K}_e \tilde{C})e(k) \tag{3.39}$$

Tujuan dari perancangan *observer* adalah untuk mengestimasi keadaan, kesalahan aktuator, dan kesalahan sensor. Oleh karena itu persamaan *observer* (sebagai estimator) dapat diterapkan jika memenuhi syarat-syarat berikut (Indriawati,dkk, 2015):

a. Rank$(C) \geq r + q$
b. Rank $(C, Fa) \geq r$
c. $(\tilde{A}, \tilde{C})$ observable

Sehingga untuk memperoleh nilai rank C menggunakan rumus determinan untuk matriks :

$$C = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1 - 0 = 1$$

Karena determinan dari $C \neq 0$, maka C merupakan matriks non-singular, sehingga memiliki nilai rank 2.

Nilai r dan q diperoleh dari ukuran kolom dari matriks $F_a(k) dan\, F_s(k)$ dimana kedua matriks tersebut berdimensi $2x1$ sehingga diperoleh nilai r dan q = 1.

Untuk memperoleh nilai rank (C, F_a) menggunakan rumus determinan :

$$CF_a = \begin{vmatrix} 1 & 0 & 0.3919 \\ 0 & 1 & 0.0146 \end{vmatrix}$$

Berdasarkan (Eriksen, 2010) untuk memperoleh nilai determinan dari matriks $CF_a = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = 1 - 0 = 1$, sehingga memiliki nilai rank 2, dan nilai r = 1

Untuk mencari rank dari $(\tilde{A}, \tilde{C})$ sehingga hasil yang diperoleh *observability*, menggunakan persamaan

$$\begin{bmatrix} C \\ CA \\ \cdot \\ \cdot \\ CA^{n-1} \end{bmatrix} \tag{3.40}$$

Sehingga nilai

$$\left(\tilde{A}, \tilde{C}\right) = \begin{bmatrix} 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0010 & 0.0000 & 0.9990 & 0.0000 & 0.0000 & 0.0010 \\ 0.0000 & 0.0010 & 0.0000 & 0.9990 & 0.0000 & 0.0000 \\ 0.0019 & 0.0000 & 0.9980 & 0.0000 & 0.0004 & 0.0020 \\ 0.0000 & 0.0017 & 0.0000 & 0.9980 & 0.0000 & 0.0000 \\ 0.0026 & 0.0000 & 0.9970 & 0.0000 & 0.0011 & 0.0030 \\ 0.0000 & 0.0023 & 0.0000 & 0.9970 & 0.0000 & 0.0000 \\ 0.0032 & 0.0000 & 0.9960 & 0.0000 & 0.0021 & 0.0040 \\ 0.0000 & 0.0027 & 0.0000 & 0.9960 & 0.0001 & 0.0000 \\ 0.0038 & 0.0000 & 0.9950 & 0.0000 & 0.0034 & 0.0050 \\ 0.0000 & 0.0029 & 0.0000 & 0.9950 & 0.0001 & 0.0000 \end{bmatrix}$$

nilai determinan 6x6

$$\left(\tilde{A}, \tilde{C}\right) = -1.997e-1$$

maka, rank dari $\left(\tilde{A}, \tilde{C}\right) = 6$, maka *observability*.

Sehingga ketiga syarat agar *observer* yang dirancang dapat digunakan sebagai estimator, terpenuhi.

Langkah selanjutnya, Untuk memperoleh nilai gain observer ($\widetilde{K_e}$) Ktild digunakan metode *pole-placement*. Syarat untuk mendapatkan nilai *pole-placement* yang baik terdapat pada subbab 2.6.

Nilai pole yang diinginkan ditentukan dengan metode *trial-and-error*, dengan berdasarkan ilmu kestabilan berdasarkan letak *pole*, diketahui bahwa sebuah sistem diskrit didefiniskan stabil jikalau seluruh akar-akar karakteristik bagian riil atau imajiner bernilai tidak lebih dari 1 atau -1. Sehingga dengan metode *trial-and-error* diperoleh 6 pole yaitu 0,92; 0.91; 0,94; 0,94; 0,9; dan 0,95.

Selanjutnya nilai matriks gain observer didapatkan dengan menggunakan program perangkat lunak matlab R2013a dengan *command* sebagai berikut:

```
p1 = 0.92;
p2 = 0.91;
p3 = 0.94;
p4 = 0.94;
p5 = 0.9;
p6 = 0.95;
Ktild = place(Atild',Ctild',[p1 p2 p3 p4 p5 p6])
K = Ktild(:,1:n+p)
L = Ktild(:,n+p+1:n+p+fa+fs)
```

sehingga diperoleh matriks Ktild atau Ke sebagai berikut :

$$Ke = \begin{bmatrix} 0.0034 & -0.0001 & -0.9146 & 0.0008 & 0.0003 & 0.0023 \\ 0.0525 & 0.0310 & -0.0268 & -1.0679 & 0.0183 & -0.0523 \end{bmatrix} \quad (3.41)$$

Setelah dilakukan perancangan *observer* secara simulasi menggunakan perangkat lunak Matlab R2013a yang dapat dilihat pada lampiran H.3, dan *observer* dapat mengestimasi kesalahan dengan benar, maka selanjutnya dilakukan penerapan *observer* secara *realtime* dengan menggunakan perangkat lunak LabVIEW 2013 yang dapat dilihat pada lampiran I.2. pada lampiran H.3 nilai Aa, Ba, Ca dan K sama seperti dengan $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{K}e$.

3.6.2 Perancangan *Reconfigurable Control*

Pengaturan ulang sistem kendali dengan metode kompensasi digunakan untuk memperbaiki kesalahan sehingga sistem tetap dapat beroperasi sesuai *setpoint* ketika terjadi kesalahan pada aktuator dan sensor.

Untuk kesalahan pada sensor, kompensasi menggunakan persamaan

$$y(k) = y_m(k) - \hat{f}_s(k) \tag{3.42}$$

Untuk kesalahan pada aktuator, kompensasi menggunakan persamaan

$$u_m(k) = u(k) - \hat{f}_a(k) \tag{3.43}$$

dengan :
$y(k)$ = nilai pengukuran sebenarnya
$y_m(k)$ = nilai pengukuran yang mengandung kesalahan
$\hat{f}_s(k)$ = estimasi kesalahan sensor
$u(k)$ = nilai sinyal kendali sebenarnya
$u_m(k)$= nilai sinyal kendali yang mengandung kesalahan
$\hat{f}_a(k)$ = estimasi kesalahan aktuator

Berdasarkan persamaan (3.42) dan (3.43), perancangan skema blok simulasi *reconfigurable control* pada perangkat lunak matlab R2013a dapat dilihat pada lampiran H.4, ditunjukkan dengan gambar lingkaran berwarna hitam, sedangkan untuk gambar lingkaran berwarna merah menunjukkan kesalahan aktuator dan kesalahan sensor yang diberikan secara simulasi dan untuk gambar lingkaran berwarna biru menunjukkan $\hat{f}_a$ dan $\hat{f}_s$. Setelah *reconfigurable control* yang dirancang secara simulasi dapat memperbaiki kesalahan yang terjadi, maka selanjutnya *reconfigurable control* diterapkan secara *realtime* pada sistem servo modular MS150 DC dengan menggunakan perangkat lunak labVIEW 2013 yang dapat dilihat pada lampiran I.3

3.7 Perancangan Sistem *Active Fault Tolerant Control* (AFTC) untuk Kesalahan *Real Time*

Pada subbab 3.6 telah dijelaskan perancangan AFTC untuk kesalahan pada aktuator dan sensor secara simulasi atau kesalahan diberikan masih melewati perangkat lunak labVIEW 2013. Pada subbab 3.7 ini, kesalahan yang diberikan berasal dari komponen-komponen sistem servo modular MS150 DC secara langsung, untuk kesalahan pada aktuator kesalahan terjadi pada

servoamplifier secara langsung dan kesalahan sensor terjadi pada *tachogenerator* secara langsung dengan menggunakan bantuan *attenuator*, yang sudah dijelaskan pada subbab 3.3. perancangan sistem AFTC pada perangkat lunak labVIEW 2013 untuk kesalahan secara *realtime*, dapat dilihat pada lampiran I.4, dari lampiran tersebut terdapat perbedaan perancangan untuk kesalahan *real* dan kesalahan secara simulasi yang dapat dilihat pada lampiran I.1.

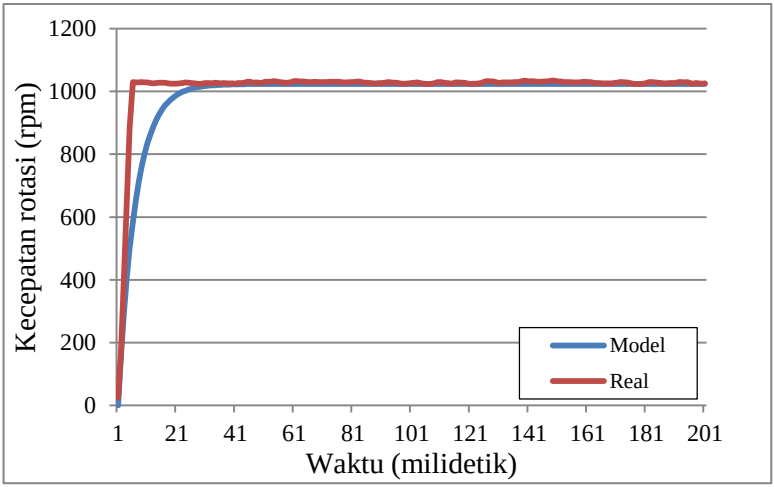
“Halaman ini memang dikosongkan”

BAB IV
ANALISIS DATA DAN PEMBAHASAN

4.1 Validasi Data

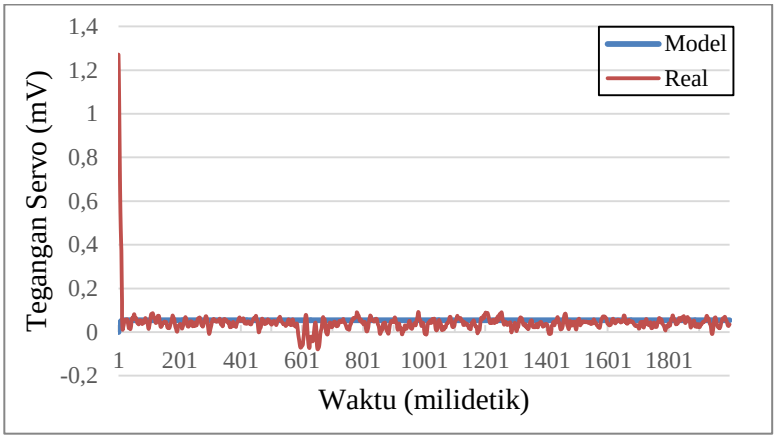
Tahap perancangan yang telah dilakukan berupa pengambilan data kecepatan dan arus motor DC, kemudian dengan menggunakan sistem identifikasi dengan metode parametrik diperoleh persamaan fungsi alih dari kecepatan dan arus sistem servo modular MS150 DC. Kedua fungsi alih tersebut perlu dilakukan validasi untuk mengetahui apakah model yang diperoleh telah sesuai dengan kondisi dari sistem servo modular MS150 DC secara langsung.

Validasi dilakukan dengan membandingkan grafik keluaran model sistem servo modular MS150 DC dan keluaran sistem servo modular MS150 DC secara langsung. Pada gambar 4.1 merupakan grafik validasi kecepatan motor DC, sedangkan pada gambar 4.2 merupakan grafik validasi arus motor DC. Untuk mengkonversi data menjadi bentuk kecepatan rotasi dapat menggunakan persamaan (3.1)



Gambar 4.1 Respon sistem validasi kecepatan

Gambar 4.1 menunjukkan 2 grafik yaitu, yaitu grafik berwarna *orange* yang menunjukkan keluaran kecepatan dari motor DC secara langsung dan grafik berwarna biru yang menunjukkan keluaran kecepatan dari model motor DC yang diperoleh, validasi dilakukan dengan memberikan nilai tegangan sebesar 1 volt atau 365 rpm pada detik ke 0 milidetik, sehingga diperoleh nilai *error* validasi sebesar 0,74%. Pada gambar 4.1 diketahui terdapat perbedaan yang cukup besar terjadi pada awal respon yaitu pada waktu ke 1 hingga 31 milidetik, tetapi pada detik berikutnya grafik respon kecepatan antara model dan sistem secara *realtime* sudah saling mendekati sehingga model dari kecepatan sisten yang diperoleh diperoleh sudah dapat merepresentasikan kecepatan sistem secara *reatime*.



Gambar 4.2 Respon sistem validasi arus dalam bentuk tegangan

Gambar 4.2 menunjukkan 2 grafik yaitu, yaitu grafik berwarna *orange* yang menunjukkan keluaran arus dari motor DC secara langsung dan grafik berwarna biru yang menunjukkan keluaran model arus dari motor DC, validasi dilakukan dengan memberikan nilai tegangan sebesar 1 volt pada detik ke 0 milidetik, dari kedua grafik tersebut terdapat perbedaan yang cukup besar antara hasil arus motor DC dengan model arus motor DC pada detik ke 1 milidetik, dimana pada grafik berwarna

orange terdapat kenaikan tertinggi arus hingga bernilai 1,3 milivolt sementara hasil dari model nilai arus motor DC berada direntang 0,05 milivolt, pada detik berikutnya grafik pengukuran arus sudah berada pada rentang nilai 0,05 milivot (0,092 hingga -0,078 milivot), tetapi hasil pengukuran arus secara *realtime* mengandung *noise* dari lingkungan dengan nilai variansi 0,04 milivolt, sehingga diperoleh nilai *error* validasi pada arus sebesar 4,90%. Walaupun nilai *error* validasi arus cukup besar, namun model arus motor DC sudah cukup merepresentasikan arus motor DC secara langsung karena memiliki nilai yang tidak jauh berbeda dari kondisi arus secara *real.*

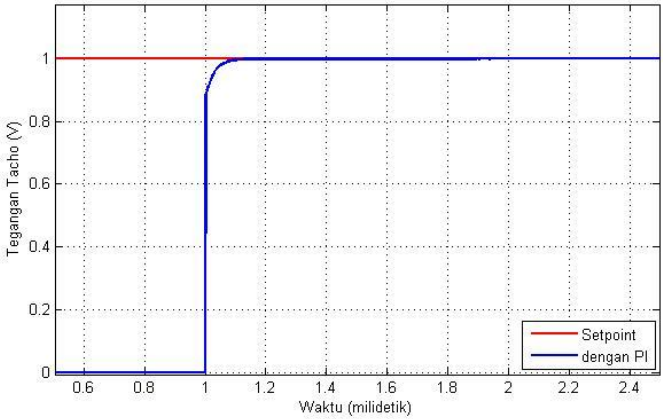
4.2 Sistem Kendali PI

Tahapan perancangan sistem kendali PI dilakukan untuk mendapatkan nilai *gain* pengendali PI yang selanjutnya hasil respon dapat diperoleh sehingga dapat diketahui kinerja dari sistem kendali PI yang dirancang. Dengan menggunakan *simulink* pada perangkat lunak matlab R2013a, diperoleh nilai *gain* PI sebagai berikut :

Kp = 2

Ki = 80

Kemudian dari nilai *gain* yang diperoleh, dilakukan simulasi pada *simulink* perangkat lunak matlab R2013a. Grafik respon sistem kendali PI pada motor DC secara simulasi dengan menggunakan perangkat lunak matlab 2013a ditunjukkan pada gambar 4.3



Gambar 4.3 Respon sistem perancangan sistem kendali PI secara simulasi

Gambar 4.3 menunjukkan 2 grafik, yaitu grafik berwarna biru yang menunjukkan nilai *set point* dan grafik berwarna merah yang menunjukkan hasil PID, nilai *set point* sebesar 1 volt diberikan pada waktu ke 1 milidetik, sehingga dari kedua grafik tersebut dapat diketahui bahwa sistem dengan PID yang dirancang dapat bekerja dengan baik karena mampu mencapai nilai *set point*, dengan karakteristik respon nilai *maximum overshoot* sebesar 0% yang dapat dilihat pada grafik berwarna merah pada waktu ke 1 milidetik, nilai *settling time* sebesar 0,052 milidetik dan *error steady state* sebesar 0%.

Nilai *gain* yang telah diperoleh dengan menggunakan perangkat lunak matlab R2103a kemudian dirubah dengan menggunakan persamaan (3.19) sehingga diperoleh nilai Kp dan τ_i sebesar :

Kp = 2

τ_i = 0,025

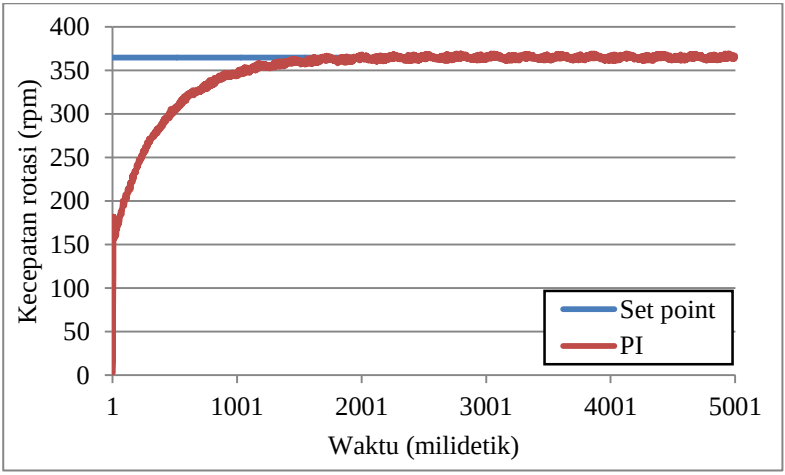
namun nilai *gain* pengendali PI yang diperoleh dari simulasi perangkat lunak MATLAB R2013 terkadang tidak dapat atau tidak tepat untuk digunakan pada perangkat lunak labVIEW 2013, karena perbedaan perangkat lunak yang digunakan,

sehingga perlu dilakukan modifikasi untuk mendapatkan nilai *gain* pada perangkat lunak labVIEW 2013 dengan menggunakan metode eksperimen, sehingga diperoleh nilai *gain* pada labVIEW 2013 sebesar :

Kp = 0.2103

τ_i = 0.0026418

Grafik respon sistem pengendali PI pada pengendalian kecepatan sistem servo modular MS150 DC dengan menggunakan perangkat lunak labVIEW 2013 ditunjukkan pada gambar 4.4



Gambar 4.4 Respon sistem penerapan sistem kendali PI secara *realtime*

Gambar 4.4 menunjukkan 2 grafik yaitu grafik berwarna biru yang menunjukkan nilai *set point* dan grafik berwarna *orange* yang menunjukkan sistem dengan PI, nilai *set point* yang diberikan sebesar 1 volt atau sebesar 365 rpm pada detik ke 0 milidetik, dari kedua grafik tersebut dapat diketahui bahwa sistem kendali PID dapat berkerja dengan baik karena mampu mencapai nilai *set point* dengan karakteristik respon *maximum overshoot*

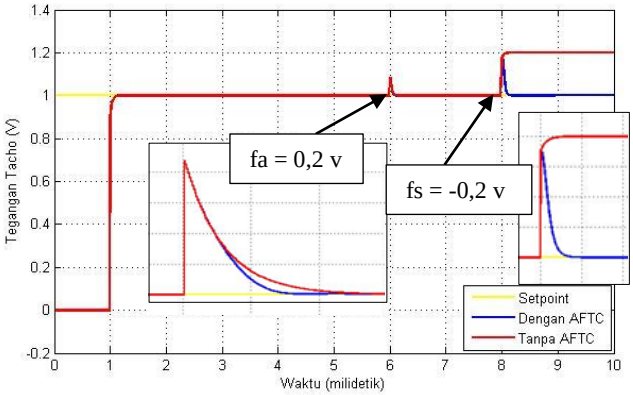
sebesar 0,36%, *settling time* sebesar 1.027 milidetik, dan *error steady-state* sebesar 0,36%.

4.3Uji Performansi

Uji performansi dilakukan untuk menguji algortima *active fault tolerant control* yang telah dibuat dapat memperbaiki kesalahan yang terjadi pada aktuator dan sensor. Uji secara simulasi dilakukan dengan memberikan kesalahan pada aktuator dan sensor berupa kesalahan minor yaitu kesalahan bias. kesalahan bias secara simulasi pada aktuator diberikan sebesar 0,2 volt; 0,3 volt; 0,4 volt dan 0,6 volt dari sinyal kendali, sedangkan kesalahan bias pada sensor secara simulasi diberikan sebesar -0,2 volt; -0,3 volt; -0,4 volt dan -0,6 volt dari sinyal pengukuran. Kesalahan bias merupakan penyimpangan nilai dari nilai tetapan yang telah ditentukan. Bias dalam pengukuran juga bisa disebut dengan nilai *error* sistematik. e*rror* sistematik mendeskripsikan *error* pada pembacaan *output* sistem pengukuran yang secara konsisten *(error steady state)* pada satu sisi pembacaan yang benar, yaitu seluruh *error* adalah positif atau seluruh *error* adalah negatif (Ardhiantama, 2016). Kesalahan bias pada sensor kecepatan dapat berupa adanya zero bias, yaitu perubahan nilai zero kecepatan pada sensor *tachogenerator,* sedangkan kesalahan bias pada aktuator dapat berupa gangguan yang menghambat tegangan atau arus dari servoamplifier untuk masuk kedalam motor DC. Uji performasi dilakukan secara simulasi dengan menggunakan bantuan perangkat lunak matlab R2013a untuk mengetahui apakah algoritma AFTC yang dirancang sudah mampu bekerja dengan baik, dan kemudian uji performansi sistem AFTC dilakukan secara *realtime* pada sistem servo modular MS150 DC dengan kesalahan yang terjadi pada aktuator dan sensor menggunakan bantuan komponen *attenuator* yang telah dijelaskan pada subbab 3.3 menggunakan bantuan perangkat lunak labVIEW 2013.

4.3.1 Uji Kesalahan Bias Pada Aktuator dan Sensor Secara Simulasi

- Hasil respon uji kesalahan bias pada aktuator sebesar 0,2 dan pada sensor sebesar -0,2 volt secara simulasi.



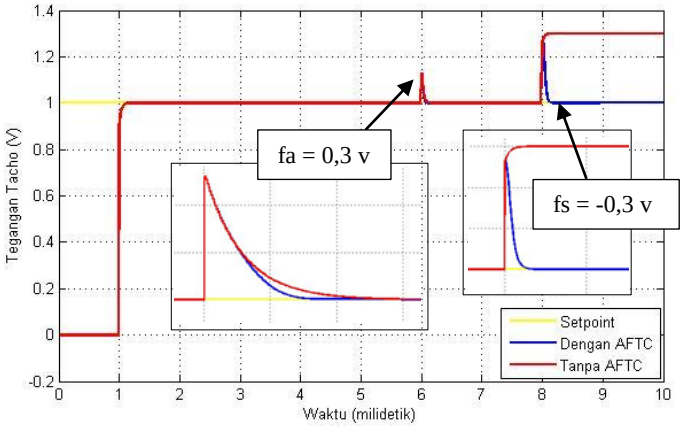
Gambar 4.5 Respon sistem AFTC dengan kesalahan bias 0,2 volt pada aktuator dan -0,2 volt pada sensor secara simulasi.

Tabel 4.1 Performansi sistem dengan kesalahan bias pada aktuator 0,2 volt dan sensor -0,2 volt secara simulasi

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	8,7%	8,7%	17,8%	20 %
Error Steady State	0%	0%	0%	20 %
Settling Time	0,039 milidetik	0,043 milidetik	0,076 milidetik	- milidetik

Pada gambar 4.5 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna biru dan sistem tanpa AFTC yang ditunjukkan grafik berwarna merah dapat mencapai nilai *setpoint,* yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 1 volt pada waktu ke 1 milidetik. Ketika kesalahan bias pada aktuator terjadi di 6 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun selanjutnya, sistem dapat kembali mencapai nilai *setpoint* sebesar 1 volt, ketika kesalahan bias pada sensor terjadi di 8 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun dapat kembali ke nilai *setpoint* sebesar 1 volt, tetapi pada sistem tanpa AFTC, sistem mengalami *maximum overshoot* dan tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru sebesar 1,2 volt.

- Hasil respon uji kesalahan bias pada aktuator sebesar 0,3 dan pada sensor sebesar -0,3 volt secara simulasi.



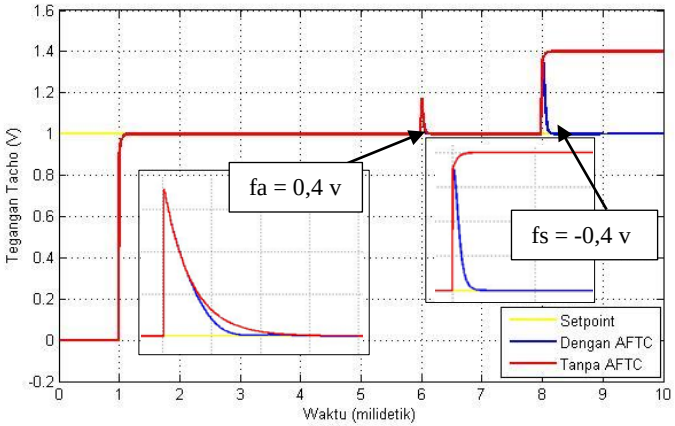
Gambar 4.6 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,3 volt dan sensor -0,3 volt secara simulasi

Tabel 4.2 Performansi sistem dengan kesalahan bias pada aktuator 0,3 volt dan sensor -0,3 volt secara simulasi

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	13%	13%	25%	30 %
Error Steady State	0%	0%	0%	30 %
Settling Time	0,045 milidetik	0,054 milidetik	0,088 milidetik	- milidetik

Pada gambar 4.6 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna biru dan sistem tanpa AFTC yang ditunjukkan grafik berwarna merah dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 1 volt pada waktu ke 1 milidetik. Ketika kesalahan bias pada aktuator terjadi di 6 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun selanjutnya, sistem dapat kembali mencapai nilai *setpoint* sebesar 1 volt, ketika kesalahan bias pada sensor terjadi di 8 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun dapat kembali ke nilai *setpoint* sebesar 1 volt, tetapi pada sistem tanpa AFTC, sistem mengalami *maximum overshoot* dan tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru sebesar 1,3 volt.

- Hasil respon uji kesalahan bias pada aktuator sebesar 0,4 dan pada sensor sebesar -0,4 volt secara simulasi



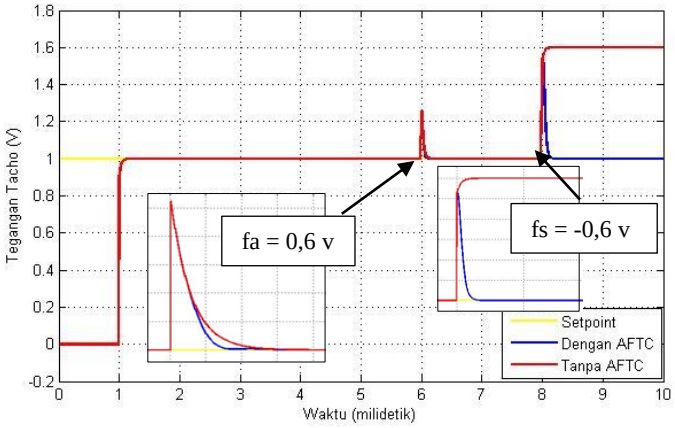
Gambar 4.7 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,4 volt dan sensor -0,4 volt secara simulasi.

Tabel 4.3 Performansi sistem dengan kesalahan bias pada aktuator 0,4 volt dan sensor -0,4 volt secara simulasi

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	17,5%	17,5%	35%	40 %
Error Steady State	0%	0%	0%	40 %
Settling Time	0,05 milidetik	0,061 milidetik	0,09 milidetik	- milidetik

Pada gambar 4.7 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna biru dan sistem tanpa AFTC yang ditunjukkan grafik berwarna merah dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 1 volt pada waktu ke 1 milidetik. Ketika kesalahan bias pada aktuator terjadi di 6 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun selanjutnya, sistem dapat kembali mencapai nilai *setpoint* sebesar 1 volt, ketika kesalahan bias pada sensor terjadi di 8 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun dapat kembali ke nilai *setpoint* sebesar 1 volt, tetapi pada sistem tanpa AFTC, sistem mengalami *maximum overshoot* dan tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru sebesar 1,4 volt.

- Hasil respon uji kesalahan bias pada aktuator sebesar 0,6 dan pada sensor sebesar -0,6 volt secara simulasi.



Gambar 4.8 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,6 volt dan sensor -0,6 volt secara simulasi

Tabel 4.4 Performansi sistem dengan kesalahan bias pada aktuator 0,6 volt dan sensor -0,6 volt secara simulasi

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	26%	26 %	53%	60 %
Error steady state	0%	0%	0%	60 %
Settling Time	0,058 milidetik	0,073 milidetik	0,102 milidetik	- milidetik

Pada gambar 4.8 respon sistem dengan AFTC (yang ditunjukkan oleh grafik berwarna biru) dan sistem tanpa AFTC (yang ditunjukkan grafik berwarna merah) dapat mencapai nilai *setpoint,* (yang ditunjukkan grafik berwarna kuning), dengan nilai *setpoint* sebesar 1 volt pada waktu ke 1 milidetik. Ketika kesalahan bias pada aktuator terjadi di 6 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun selanjutnya, sistem dapat kembali mencapai nilai *setpoint* sebesar 1 volt, ketika kesalahan bias pada sensor terjadi di 8 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun dapat kembali ke nilai *setpoint* sebesar 1 volt, tetapi pada sistem tanpa AFTC, sistem mengalami *maximum overshoot* dan tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru sebesar 1,6 volt.

Secara keseluruhan hasil respon sistem dengan AFTC, ketika kesalahan bias pada aktuator sebesar 0,2 volt;0,3 volt;0,4 volt dan 0,6 volt terjadi di waktu 6 milidetik, sistem dengan AFTC maupun tanpa AFTC memiliki nilai *error steady state* sebesar 0%, Penggunaan algoritma AFTC secara simulasi tidak mempengaruhi nilai *maximum overshoot,* karena sistem dengan AFTC dan sistem tanpa AFTC memiliki nilai *maximum overshoot* yang sama. Pada nilai *settling time,* penggunaan algortima AFTC mempengaruhi nilai *settling time,* yaitu sistem dengan AFTC

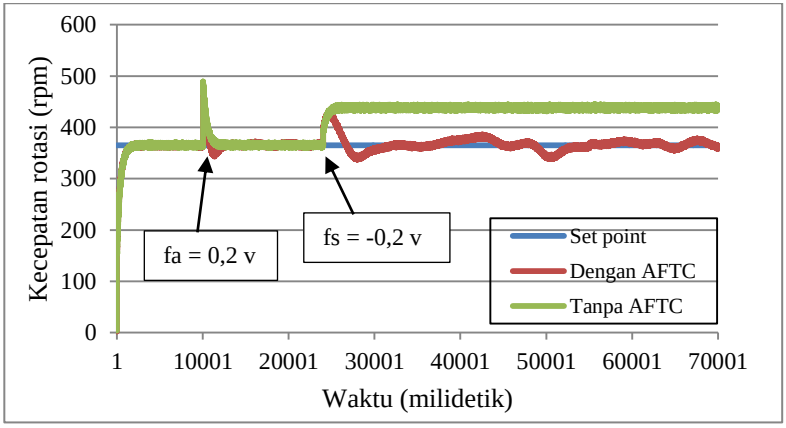
memiliki nilai *settling time* yang lebih kecil dibandingkan sistem tanpa AFTC.

Ketika kesalahan bias pada sensor terjadi sebesar -0,2 volt;-0,3 volt;-0,4 volt dan -0,6 volt di waktu 8 milidetik, sistem dengan AFTC memiliki nilai *maximum overshoot* yang lebih kecil dibandingkan sistem tanpa AFTC, walaupun selisih nilai yang diperoleh tidak terlalu banyak. Nilai *settling time* untuk sistem dengan AFTC yang diperoleh lebih kecil dibandingkan dengan sistem tanpa AFTC, dikarenakan sistem tanpa AFTC tidak dapat kembali mencapai setpoint dengan nilai *error steady stat*e sebesar 20% untuk kesalahan bias -0,2 volt; 30% untuk kesalahan bias -0,3 volt; 40% untuk kesalahan bias -0,4 volt dan 60% untuk kesalahan bias -0,6 volt sehingga memiliki nilai *settling time* tidak terhingga. Sedangkan untuk sistem dengan AFTC memiliki nilai *error staedy state* sebesar 0% untuk semua kesalahan bias yang terjadi.

Nilai *error* yang diperoleh untuk sistem dengan AFTC ketika terjadi kesalahan pada aktuator dan sensor, menunjukkan bahwa sistem AFTC yang dirancang dapat bekerja dengan baik karena mampu mengkompensasi kesalahan sehingga rancangan algoritma AFTC dapat diterapkan secara *realtime*. Sistem dengan AFTC memiliki nilai *maximum overshoot, settling time,* dan *error steady state* yang lebih kecil karena terdapat *observer* yang berfungsi untuk mengestimasi kesalahan yang terjadi pada aktuator dan sensor, yang selanjutnya oleh *reconfigurable control* estimasi kesalahan tersebut digunakan untuk mengkompensasi atau memperbaiki kesalahan sehingga menyebabkan respon dari sistem dapat kembali ke nilai *setpoint*

4.3.2 Uji Kesalahan Bias pada Aktuator dan Sensor Secara Simulasi pada Sistem Servo Modular MS150 DC Secara *Realtime.*

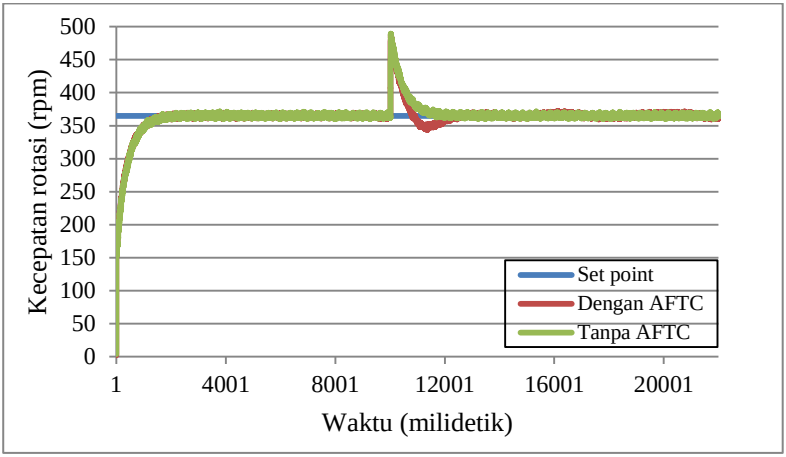
- Hasil respon uji kesalahan bias pada aktuator sebesar 0,2 dan pada sensor sebesar -0,2 volt secara *realtime* pada sistem servo modular MS150 DC



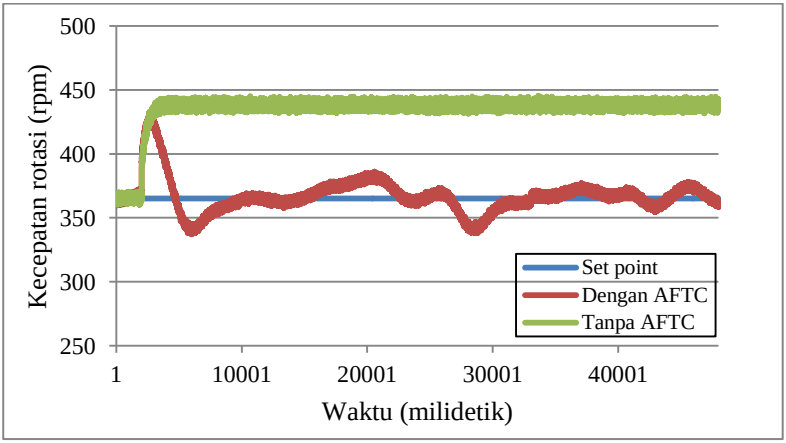
Gambar 4.9 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,2 volt dan sensor -0,2 volt penerapan secara *realtime*

Pada gambar 4.9 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna *orange* dan sistem tanpa AFTC yang ditunjukkan grafik berwarna abu-abu dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada aktuator terjadi di 10.000 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada sensor terjadi di 24.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint*, tetapi pada sistem tanpa

AFTC, sistem tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru.



Gambar 4.10 Perbesaran respon sistem AFTC dengan kesalahan bias pada aktuator 0,2 volt penerapan secara *realtime*

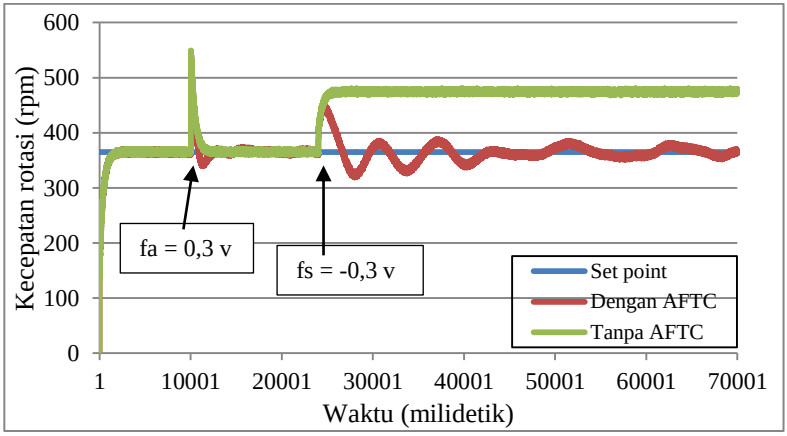


Gambar 4.11 Perbesaran respon sistem AFTC dengan kesalahan bias pada sensor -0,2 volt penerapan secara *realtime*

Tabel 4.5 Performansi sistem dengan kesalahan bias pada aktuator 0,2 volt dan sensor -0,2 volt penerapan secara *realtime*

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	33,83%	34,02%	18,33%	20,76%
Error Avearge	0,71%	0,50%	1,67%	20%
Settling time	1.252 milidetik	1.050 milidetik	5.001 milidetik	- milidetik

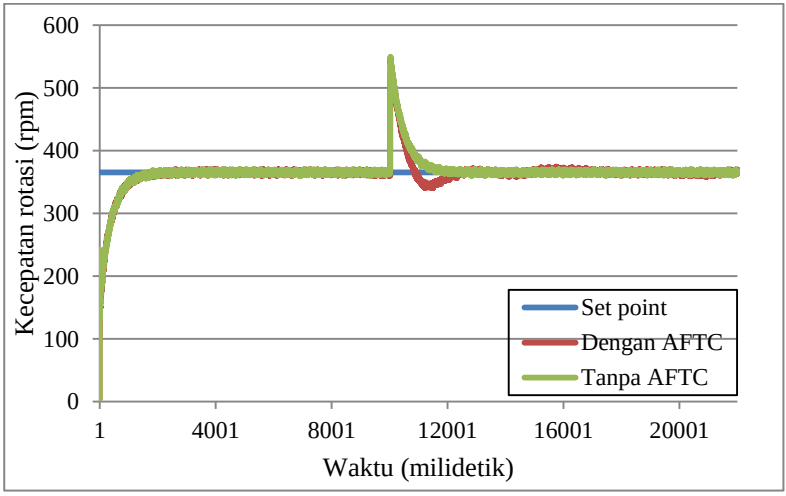
- Hasil respon uji kesalahan bias pada aktuator sebesar 0,3 dan pada sensor sebesar -0,3 volt secara *realtime* pada sistem servo modular MS150 DC



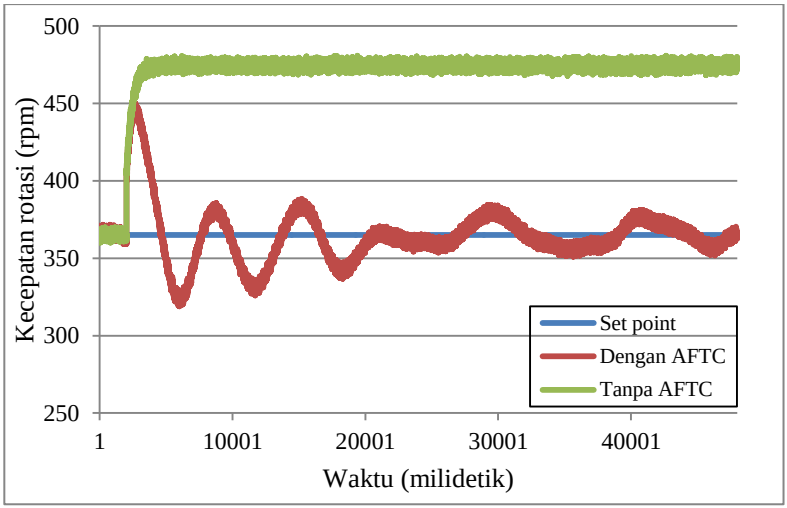
Gambar 4.12 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,3 volt dan sensor -0,3 volt penerapan secara *realtime.*

Pada gambar 4.12 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna *orange* dan sistem tanpa AFTC

yang ditunjukkan grafik berwarna abu-abu dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada aktuator terjadi di 10.000 milidetik, sistem dengan AFTC dan tanpa AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada sensor terjadi di 24.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint*, tetapi pada sistem tanpa AFTC, sistem tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru.



Gambar 4.13 Perbesaran respon sistem AFTC dengan kesalahan bias pada aktuator 0,3 volt penerapan secara *realtime*

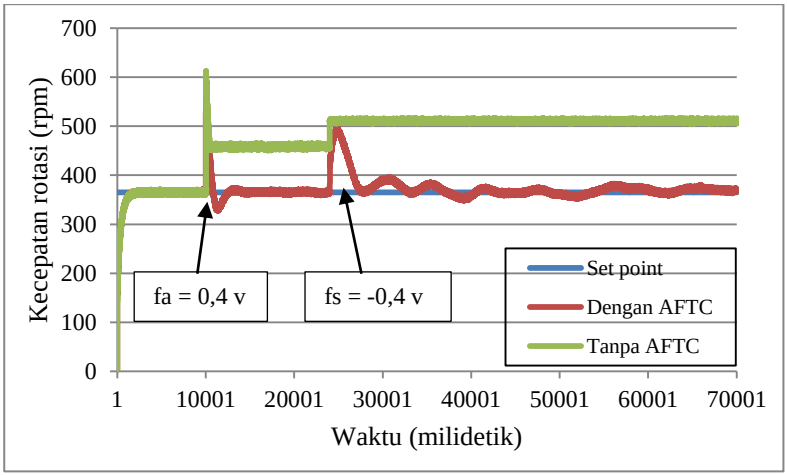


Gambar 4.14 Penerapan respon sistem AFTC dengan kesalahan bias pada sensor -0,3 volt penerapan secara *realtime*

Tabel 4.6 Karakteristik respon sistem AFTC dengan kesalahan bias pada aktuator 0,3 volt dan sensor -0,3 volt penerapan secara *realtime*

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	50,09%	50,49%	23,31%	30,76%
Error	0,87%	0,56%	1,71%	29,99 %
Settling Time	1.049 milidetik	1.042 milidetik	17.006 milidetik	- milidetik

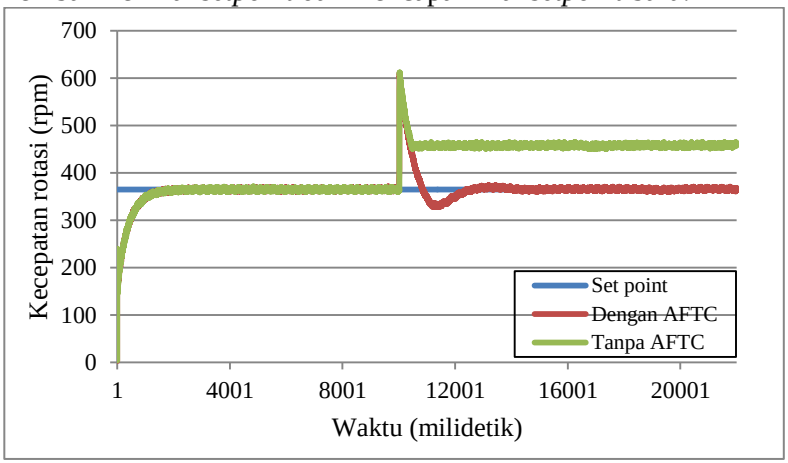
- Hasil respon uji kesalahan bias pada aktuator sebesar 0,4 dan pada sensor sebesar -0,4 volt secara *realtime* pada sistem servo modular MS150 DC



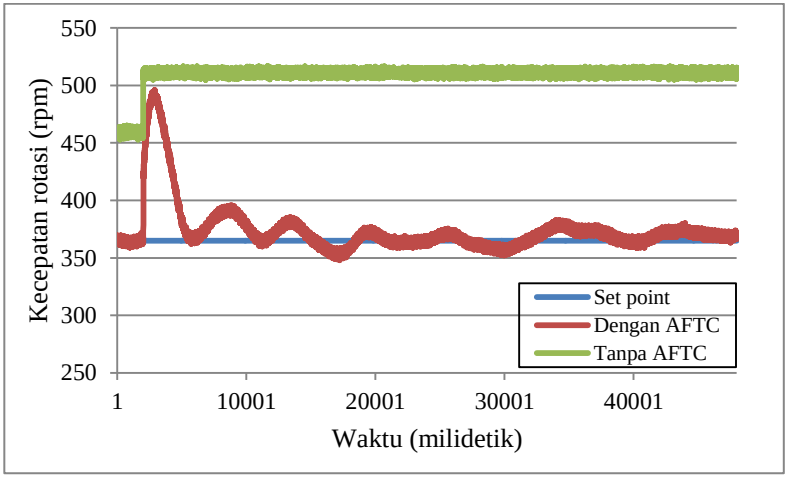
Gambar 4.15 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,4 volt dan sensor -0,4 volt penerapan secara *realtime*

Pada gambar 4.15 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna *orange* dan sistem tanpa AFTC yang ditunjukkan grafik berwarna abu-abu dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada aktuator terjadi di 10.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint* sebesar 365 rpm, tetapi pada sistem tanpa AFTC, sistem tidak dapat memperbaiki kesalahan, sehingga tidak dapat mencapai nilai *setpoint* 365 rpm dan mencapai nilai *setpoint* baru. Ketika kesalahan bias pada sensor terjadi di 24.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai

nilai *setpoint*, tetapi pada sistem tanpa AFTC, sistem tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru.



Gambar 4.16 Perbesaran respon sistem AFTC dengan kesalahan bias pada aktuator 0,4 volt penerapan secara *realtime*

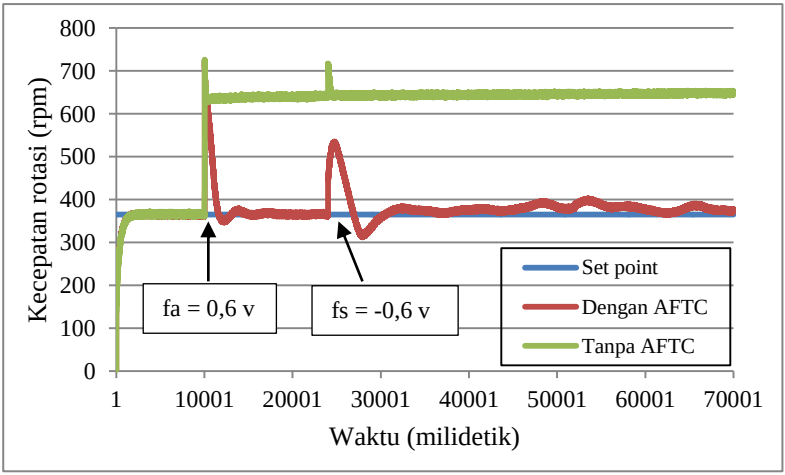


Gambar 4.17 Perbesaran respon sistem AFTC dengan kesalahan bias pada sensor -0,4 volt penerapan secara *realtime*

Tabel 4.7 Performansi sistem dengan kesalahan bias pada aktuator 0,4 volt dan sensor -0,4 volt penerapan secara *realtime*

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	67,38%	67,73%	35,93%	41,15%
Error	0,57%	25,56%	1,56%	40%
Settling Time	2.001 milidetik	- milidetik	8.001 milidetik	- milidetik

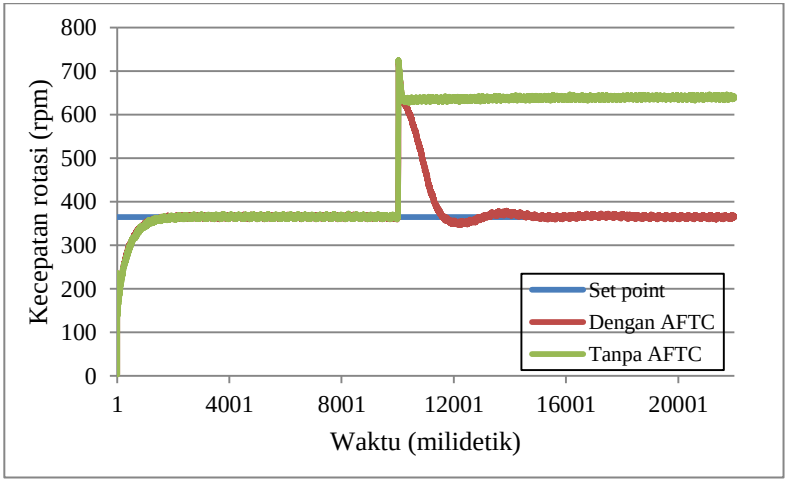
- Hasil respon uji kesalahan bias pada aktuator sebesar 0,6 dan pada sensor sebesar -0,6 volt secara *realtime* pada sistem servo modular MS150 DC



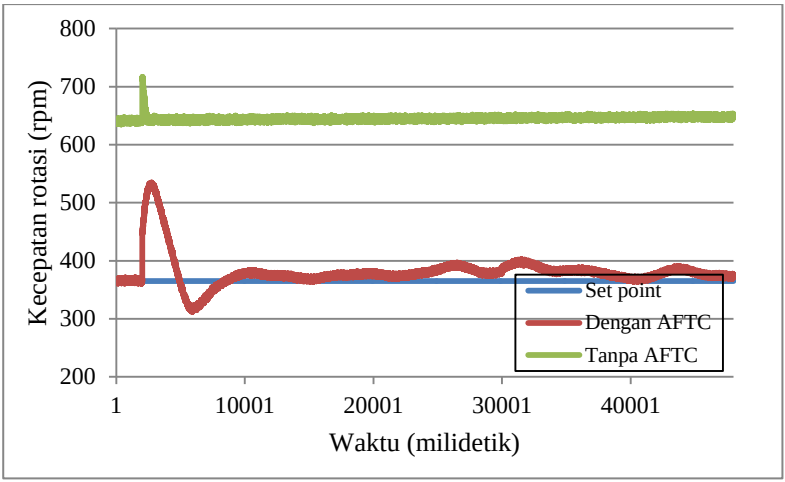
Gambar 4.18 Respon sistem AFTC dengan kesalahan bias pada aktuator 0,6 volt dan sensor -0,6 volt penerapan secara *realtime*

Pada gambar 4.18 respon sistem dengan AFTC yang ditunjukkan oleh grafik berwarna *orange* dan sistem tanpa AFTC

yang ditunjukkan grafik berwarna abu-abu dapat mencapai nilai *setpoint*, yang ditunjukkan grafik berwarna kuning, dengan nilai *setpoint* sebesar 365 rpm. Ketika kesalahan bias pada aktuator terjadi di 10.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint* sebesar 365 rpm, tetapi pada sistem tanpa AFTC, sistem tidak dapat memperbaiki kesalahan, sehingga tidak dapat mencapai nilai *setpoint* 365 rpm dan mencapai nilai *setpoint* baru. Ketika kesalahan bias pada sensor terjadi di 24.000 milidetik, sistem dengan AFTC mengalami *maximum overshoot* namun, sistem AFTC dapat memperbaiki kesalahan sehingga dapat mencapai nilai *setpoint*, tetapi pada sistem tanpa AFTC, sistem tidak dapat kembali ke nilai *setpoint* dan mencapai nilai *setpoint* baru.



Gambar 4.19 Perbesaran respon sistem AFTC dengan kesalahan bias pada aktuator 0,6 volt penerapan secara *realtime*



Gambar 4.20 Perbesaran respon sistem AFTC dengan kesalahan bias pada sensor -0,6 volt penerapan secara *realtime*

Tabel 4.8 Performansi sistem dengan kesalahan bias pada aktuator 0,6 volt dan sensor -0,6 volt penerapan secara *realtime*

Parameter	**Kesalahan Aktuator**		**Kesalahan Sensor**	
	Dengan AFTC	Tanpa AFTC	Dengan AFTC	Tanpa AFTC
Maximum overshoot	98,25 %	98,71 %	46,52%	96,43%
Error	0,85%	75,05%	3,67%	76,75%
Settling Time	1.934 milidetik	-milidetik	6.024 milidetik	- milidetik

Setelah uji algoritma AFTC secara simulasi menunjukkan bahwa algoritma AFTC yang dirancang dapat bekerja dengan baik, selanjutnya algoritma AFTC tersebut diterapkan secara *realtime* pada motor DC dengan menggunakan bantuan perangkat lunak labVIEW 2013.

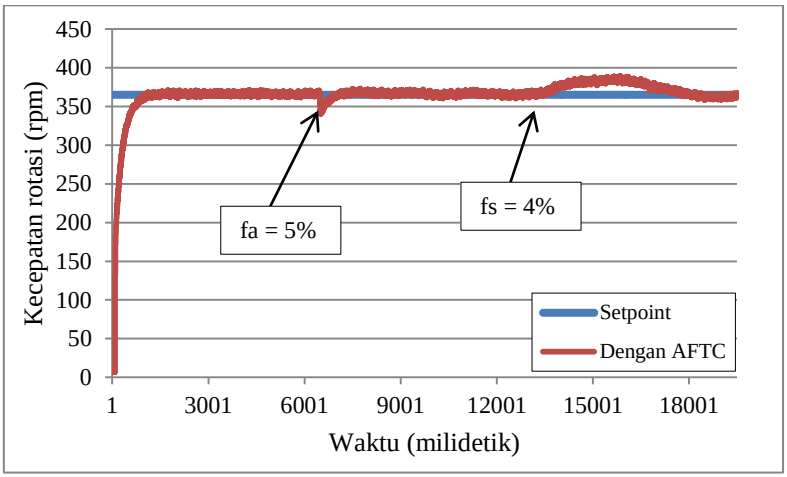
Respon sistem dengan AFTC dan tanpa AFTC ketika diberikan kesalahan bias sebesar 0,2 volt dan 0,3 volt pada aktuator dapat kembali ke nilai *set point* dengan nilai *error* rata-rata 0,71% (+2,6 rpm) dan 0,87% (+3,2 rpm) untuk sistem dengan AFTC serta 0,50% (+1,86 rpm) dan 0,56% (+2,07 rpm) untuk sistem tanpa AFTC. Nilai *error* rata-rata yang dihasilkan pada sistem dengan AFTC lebih besar dibandingkan sistem tanpa AFTC dikarenakan adanya pengaruh dari pengukuran arus, dimana pengukuran arus yang diperoleh mengandung *noise* dengan nilai variansi sebesar 0,04 *milivolt*. Adanya noise pada pengukuran arus dapat menyebabkan munculnya *error* antara model riil dengan *observer*, sehingga berpengaruh pada nilai kesalahan aktuator yang diestimasi oleh *observer* dan berpengaruh pada kerja dari *reconfigurable control* yang berfungsi untuk mengkompensasi kesalahan. Sedangkan untuk sistem tanpa AFTC, kesalahan aktuator yang masih dapat dikompensasi sebesar 3 volt, untuk sistem tanpa AFTC, untuk memperbaiki kesalahan bias pada aktuator tidak diperlukan nilai estimasi kesalahan aktuator hasil *observer* yang dipengaruhi oleh pengukuran arus untuk aktuator, maupun *reconfigurable control*. Hal ini juga berdampak pada nilai *settling time* yang diperoleh untuk sistem dengan AFTC lebih besar dibandingkan sistem tanpa AFTC. Ketika kesalahan bias yang diberikan pada aktuator sebesar 0,4 volt dan 0,6 volt sistem tanpa AFTC tidak dapat kembali ke nilai *setpoint* dengan *error* rata-rata sebesar 25,56 % (+93,4 rpm) dan 75,05% (+274 rpm) sedangkan untuk sistem dengan AFTC sistem dapat kembali ke nilai *setpoint* dengan nilai *error* rata-rata sebesar 0,57% (+2,1 rpm) dan 0,85% (+3,12 rpm), dan nilai *settling time* yang diperoleh lebih kecil dibandingkan dengan sistem tanpa AFTC. Pada penerapan secara *real*, nilai *maximum overshoot* yang dihasilkan oleh sistem dengan AFTC lebih kecil dibandingkan dengan sistem tanpa AFTC.

Pada kesalahan sensor, respon sistem dengan AFTC mampu kembali ke nilai *set point* dengan nilai *error* rata-rata sebesar 1,67% (+6,1 rpm) untuk kesalahan bias -0,2 volt; 1,71% (+6,24 rpm) untuk kesalahan bias -0,3 volt; 1,56% (+5,7 rpm)

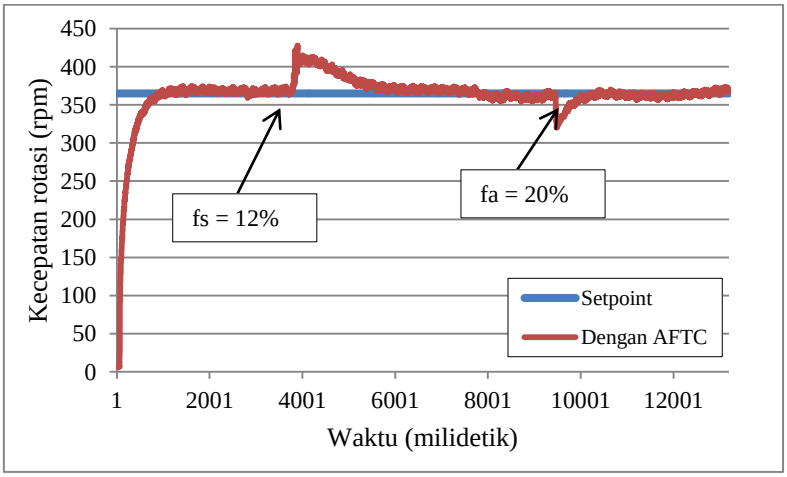
untuk kesalahan bias -0,4 volt dan 3,67% (+13,4 rpm) untuk kesalahan bias -0,6 volt , berbeda dengan sistem tanpa AFTC yang tidak mampu kembali ke nilai *set point* dan memiliki *error* rata-rata sebesar 20% (+73 rpm) untuk kesalahan bias -0,2 volt; 29,99% (+109,4 rpm) untuk kesalahan -0,3 volt; 40% (+146 rpm) untuk kesalahan -0,4 volt dan 76,75% (+280 rpm) untuk kesalahan -0,6 volt. Penggunaan sistem dengan AFTC mempengaruhi nilai *maximum overshoot* yang nilainya lebih kecil dibandingkan sistem tanpa AFTC, selain itu untuk sistem dengan AFTC nilai *settling time* yang diperoleh lebih kecil dibandingkan dengan sistem tanpa AFTC, karena untuk sistem tanpa AFTC, respon dari sistem tidak dapat kembali ke nilai *setpoint*, sehingga nilai *settling time* untuk sistem tanpa AFTC tidak diketahui atau tidak terhingga.

Respon yang diperoleh ketika menggunakan algortima AFTC menunjukkan bahwa algoritma AFTC yang dirancang dapat bekerja dengan baik karena mampu memperbaiki kesalahan bias pada aktuator dan sensor sehingga respon sistem dapat kembali ke nilai *setpoint* yang membuat performansi dari sistem lebih terjaga dibandingkan tanpa AFTC, namun respon yang dihasilkan secara *realtime* tidak sebagus hasil yang diperoleh secara simulasi, karena ketika algortima AFTC diuji secara *realtime* terdapat faktor pengukuran secara langsung, serta adanya faktor *disturbance* atau *noise* pada pengukuran yang dapat mempengaruhi hasil keluaran estimasi *observer* karena terdapat perbedaan antara model yang digunakan *observer* dan sistem secara riil, namun pada simulasi, sistem secara riil dan model menggunakan persamaan yang sama sehingga menyebabkan hasil respon yang diperoleh lebih bagus.

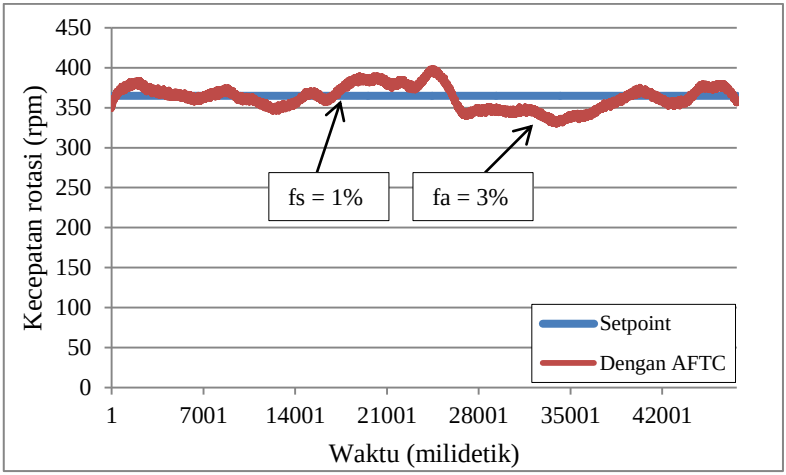
4.3.3 Hasil Respon Uji Kesalahan Bias Aktuator dan Sensor Secara *Real* Pada Sistem Servo Modular MS150 DC



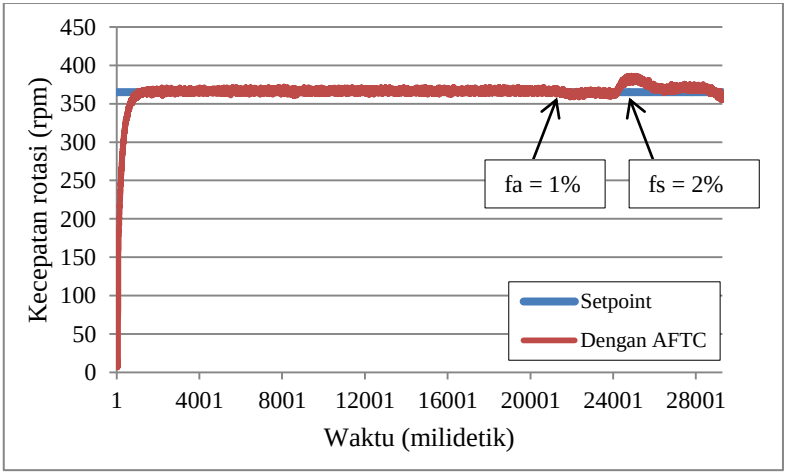
Gambar 4.21 Respon sistem AFTC dengan kesalahan pada aktuator dan sensor secara *real*.



Gambar 4.22 Respon sistem AFTC dengan kesalahan pada sensor dan aktuator secara *real*.



Gambar 4.23 Respon sistem AFTC dengan kesalahan pada sensor dan aktuator secara *real* menggunakan kompensasi awal.



Gambar 4.24 Respon sistem AFTC dengan kesalahan pada aktuator dan sensor secara *real* menggunakan kompensasi awal.

Sistem AFTC yang dirancang dapat bekerja dengan baik, ketika kesalahan aktuator dan sensor diberikan secara simulasi maupun secara *real*. Pada gambar 4.21 dan gambar 4.22, merupakan hasil penerapan sistem AFTC dengan kesalahan aktuator dan sensor terjadi secara *real.* Dari kedua gambar tersebut, *reconfigurable control* pada AFTC diterapkan secara bergiliran, yaitu pada gambar 4.21 kompensasi untuk aktuator dijalankan terlebih dahulu, kemudian kesalahan pada aktuator terjadi di waktu 6166 milidetik, selanjutnya kompensasi untuk sensor dijalankan, kemudian kesalahan sensor terjadi di waktu 13179 milidetik. Berbeda dengan gambar 4.21, pada gambar 4.22 kompensasi untuk sensor dijalankan terlebih dahulu, yang selanjutnya kesalahan sensor terjadi di waktu 3543 milidetik, kemudian kompensasi untuk aktuator dijalankan, selanjutnya kesalahan aktuator terjadi di waktu 9129 milidetik. Dari gambar 4.21 dan 4.22, sistem dengan AFTC dapat memperbaiki kesalahan pada aktuator dan sensor yang terjadi, sehingga sistem dapat kembali ke nilai *setpoint* sebesar 365 rpm dengan *error steady state* kesalahan pada aktuator sebesar 0,56% dan pada sensor sebesar 0,72% pada gambar 4.21, sedangkan pada gambar 4.22 *error steady state* kesalahan pada aktuator sebesar 0,75% dan pada sensor sebesar 1,2%.

Pada gambar 4.23 dan 4.24 sistem AFTC yang dirancang dapat bekerja dengan baik, sehingga sistem dapat kembali ke nilai *setpoint.* Dari kedua gambar tersebut, *reconfigurable control* pada AFTC diterapkan secara bersamaan, yaitu kompensasi untuk aktuator dan sensor dijalankan terlebih dahulu, yang selanjutnya kesalahan pada aktuator dan sensor terjadi, sehingga dapat diketahui algoritma AFTC yang dirancang dapat memperbaiki kesalahan pada aktuator terlebih dahulu, kemudian kesalahan sensor ataupun kesalahan sensor terlebih dahulu, kemudian kesalahan aktuator. Namun, ketika kesalahan pada sensor terjadi terlebih dahulu seperti pada gambar 4.23, respon dari sistem dengan AFTC menjadi lebih berfluktuatif dibandingkan ketika kesalahan pada aktuator terjadi telebih dahulu seperti pada

gambar 4.24, hal ini disebabkan karakteristik dari sistem servo modular MS150 DC yang nonlinier, sehingga ketika kesalahan pada sensor terjadi menyebabkan error antara model dengan sistem secara riil menjadi lebih besar, sehingga berdampak pada estimasi kesalahan sensor, yang juga berdampak pada estimasi kesalahan pada aktuator. Sedangkan pada gambar 4.24 ketika kesalahan aktuator terjadi terlebih dahulu, respon sistem lebih *steady* dikarenakan ketika terjadi kesalahan aktuator, nilai *error* antara model dengan sistem riil tidak terlalu besar, selain itu sistem kendali PI juga memiliki peranan penting dalam mengatasi kesalahan pada aktuator sehingga sistem dapat kembali ke nilai *setpoint* lebih baik dibandingkan ketika kesalahan sensor terjadi. Nilai *error steady state* untuk untuk kesalahan pada sensor sebesar 5% dan pada aktuator sebesar 2,15% pada gambar 4.23, dan nilai *error steaady state* untuk kesalahan pada aktuator sebesar 0,47% dan pada sensor sebesar 1,2% pada gambar 4.24.

Dari hasil perancangan algoritma AFTC juga diperoleh nilai kesalahan maksimum yang dapat diatasi untuk aktuator sebesar 100% atau ketika motor DC dari sistem servo modular MS150 DC sudah tidak berputar dikarenakan kesalahan aktuator, maka algoritma AFTC tidak dapat memperbaiki, dan untu kesalahan pada sensor, nilai maksimum yang dapat diatasi sebesar 12%, ketika kesalahan pada sensor sebesa 13%, respon sistem memiliki *error* lebih dari 5%.

"Halaman ini memang dikosongkan"

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis data yang telah dilakukan, didapatkan kesimpulan dari tugas akhir mengenai perancangan sistem *Active Fault Tolerant Control* pada pengendalian kecepatan sistem servo modular MS150 DC dengan kesalahan pada aktuator dan sensor adalah sebagai berikut :

a. Telah dilakukan perancangan algoritma AFTC berupa perancangan *observer* dan *reconfigurable control* dan kesalahan aktuator dan sensor secara simulasi dan secara *real* pada pengendalian kecepatan sistem servo modular MS150 DC sehingga sistem mampu memperbaiki kesalahan bias pada aktuator dan sensor yang menyebabkan sistem dapat kembali mencapai nilai *setpoint* dibandingkan dengan tanpa AFTC.
b. Sistem dengan AFTC lebih mampu menjaga performansi ketika terjadi kesalahan pada aktuator dan sensor dibandingkan dengan sistem tanpa AFTC, dengan nilai kesalahan bias terbesar yang terjadi pada aktuator sebesar 0,6 volt dan pada sensor sebesar -0,6 volt.

   Dalam hasil uji performansi untuk sistem secara *realtime* diperoleh, ketika terjadi kesalahan pada aktuator dan sensor sistem dengan AFTC memiliki nilai *maximum overshoot* lebih kecil dibandingkan dengan sistem tanpa AFTC, selain itu penggunaan sistem dengan AFTC menyebabkan nilai *settling time* yang dihasilkan lebih kecil dibandingkan dengan sistem tanpa AFTC untuk kesalahan bias pada sensor dan untuk kesalahan bias pada aktuator dengan nilai bias sebesar 4–6 volt, kemudian sistem dengan AFTC mempengaruhi nilai *error* rata-rata, dimana ketika terjadi kesalahan pada sensor, nilai *error* rata-rata untuk sistem dengan AFTC lebih kecil dibandingkan dengan sistem tanpa AFTC, begitu juga ketika terjadi kesalahan bias pada aktuator, sistem dengan AFTC memiliki nilai *error* rata-rata

lebih kecil dibandingkan dengan sistem tanpa AFTC dengan nilai kesalahan bias yang terjadi bernilai 4-6 volt.

Sistem AFTC dengan kesalahan secara *real,* dapat memperbaiki kesalahan yang terjadi dengan nilai kesalahan terbesar yang terjadi pada aktuator sebesar 100% dan pada sensor sebesar 13%, selain itu sistem AFTC yang dirancang dapat memperbaiki kesalahan aktuator - sensor ataupun sensor - aktuator.

5.2 Saran

Dalam pengerjaan tugas akhir ini terdapat beberapa saran yang dapat digunakan untuk penelitian selanjutnya, yaitu adanya tambahan sensor arus yang memiliki filter *noise* didalamnya sehingga hasil pengukuran arus dapat lebih baik sehingga hasil respon yang diperoleh jauh lebih baik. Selain itu, sistem AFTC yang dirancang sebaiknya digunakan untuk sistem linier, sedangkan untuk sistem nonlinier, model sistem yang diperoleh harus dapat beroperasi di berbagai macam *setpoint* sehingga nilai *error* antra model dan sistem riil tidak terlalu besar yang menyebabkan sistem dengan AFTC dapat memperbaiki kesalahan dengan baik.

DAFTAR PUSTAKA

Anindita, P. (2008). Pemodelan Keterkaitan Suku Bunga dan Kurs dengan Sistem Kontrol. Bandung: ITB.

Ardhiantama, A. (2016). Perancangan Active Fault Tolerant Control Pada Sistem Pengendalian Temperatur Fuel Gas Superheat Burner PT PETROKIMIA GRESIK Dengan Kesalahan Pada Sensor Temperatur. Surabaya: Institut Teknologi Sepuluh Nopember.

Basilio, J. C. (2002). *Design of PI and PID Controlers With Transient Performance Specification*. *IEEE Transactions On Education, Vol. 45, No.4*.

Eriksen, E. (2010, september 3).
Diambil kembali dari dr-eriksen: http://www.dr-eriksen.no/teaching/GRA6035/2010/lecture2-hand.pdf

Hudaya, C. (2013). *Motor DC.* Jakarta: Universitas Indonesia.

Indriawati, K., Agustinah, T., & Jazidie, A. (2013). *Reconfigurable Fault-Tolerant Control of Linear System with Actuator and Sensor Faults*. *IEEE*, 22-27.

Indriawati, K., Agustinah, T., & Jazidie, A. (2015). *Robust Fuzzy Observer-Based Fault Tolerant Tracking Control for Nonlinear Systems with Stimultaneous Actuator and Sensor Faults: Application to a DC Series Motor Speed Drive*. *Praise Worthy Prize, VIII*(6), 375-385.

Instruments, N. (2004). *labVIEW System Identification Toolkit User Manual.* USA: National Instruments.

M. Chen, B. (2016). *Digital Control System.* Singapore: The National University of Singapore.
Diambil kembali dari NUS Web Site: http://uav.ece.nus.edu.sg/~bmchen/courses/ee3304.pdf

Maulana, N. (2012). Penerapan Robust PID Pada Pengendalian Kecepatan MS 150 DC Motor Servo System. Surabaya: Institut Teknologi Sepuluh Nopember.

Nusantoro, G., Muslim, M., & Budi, T. (2012). Identifikasi Sistem Plant Suhu dengan Metode *Recursive Least Square. EECCIS*, 67-75.

Ogata, K. (1970). *Modern Control Engineering.* New Jersey: Prentice-Hall.

Ogata, K. (1987). *Discrete-Time Control Systems.* USA: Prentice-Hall, Inc.

Ogata, K. (2004). *System Dynamics.* New Jersey: Prentice-Hall.

Ogata, K. (2010). *Modern Control Engineering.* United States of America: Pearson Prentice Hall.

Pengaturan, A. L. (2015). Sistem Pengaturan Kecepatan Motor DC. Surabaya: Teknik Elektro-Institut Teknologi Sepuluh Nopember.

Roostandy, J., sumardi, & Andromeda, T. (2011). Pengidentifikasian Parameter Fungsi Alih Sistem. Semarang: Universitas Diponegoro.

Setyaningrum, D. (2012). Desain dan Implementasi *Model Reference Adaptive Control* untuk Pengaturan Tracking Optimal Posisi Motor DC. Surabaya: Institut Teknologi Sepuluh Nopember.

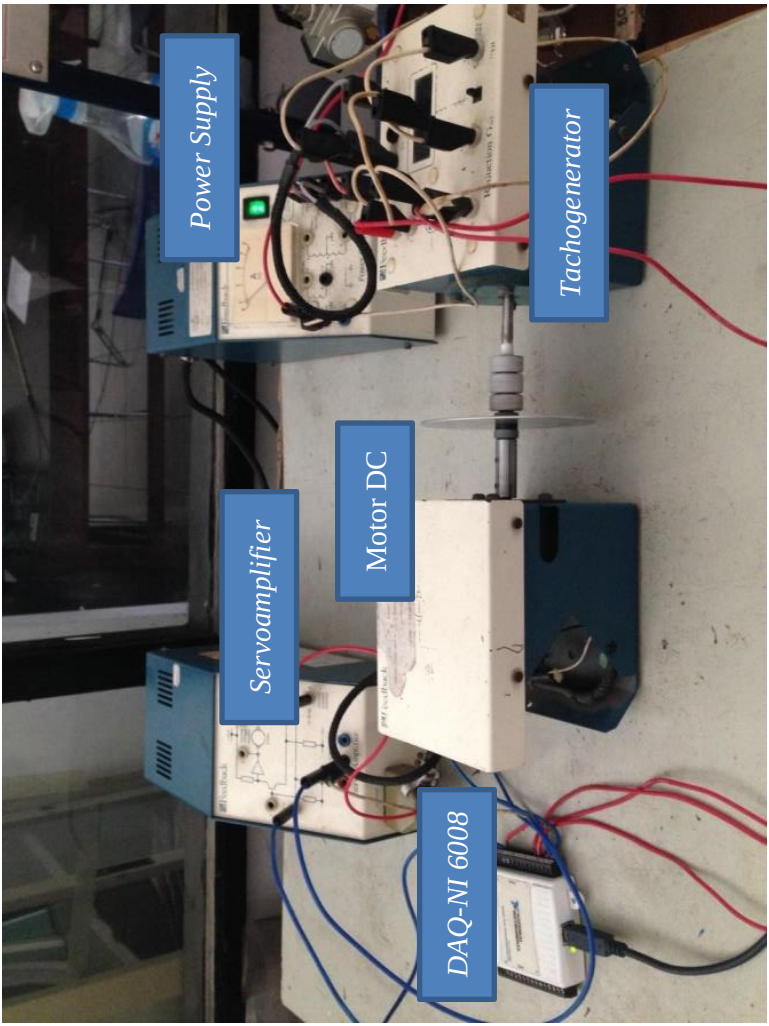
Sugiono, D. (2015). Model Matematik Motor DC. Malang: PPPP TK VEDC.

Zhang, Y., & Jiang, J. (2003). *Fault Tolerant Control System Design with Explicit Consideration of Performance Degradation. IEEE*, 838-848.

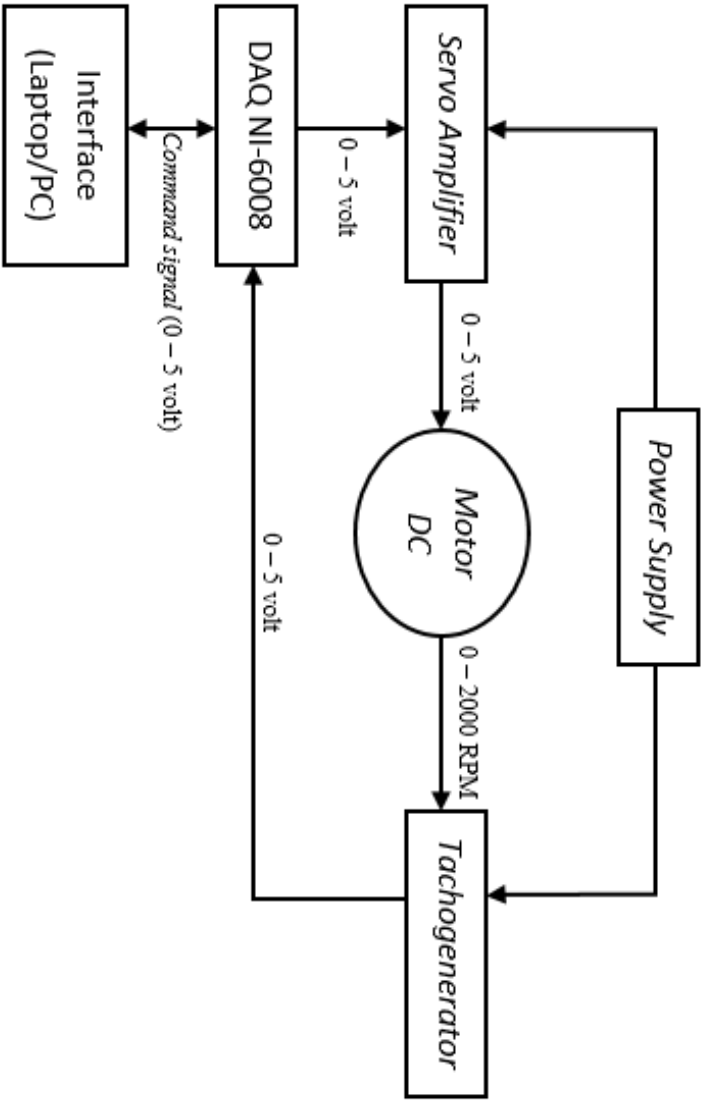
Zhang, Y., & Jiang, J. (2008). *Bibliographical review on reconfigurable fault-tolerant control systems*. *Science Direct*, 229-252.

LAMPIRAN A

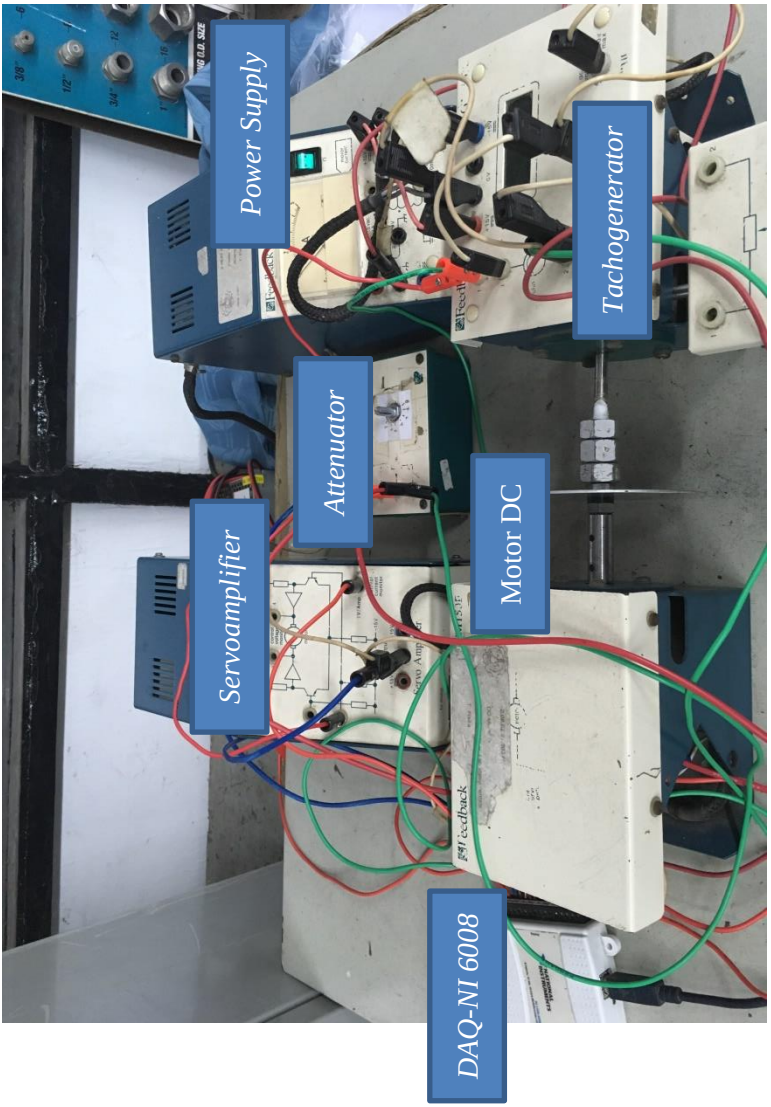
A.1 *Set-up* pengendalian kecepatan sistem servo modular MS150 DC



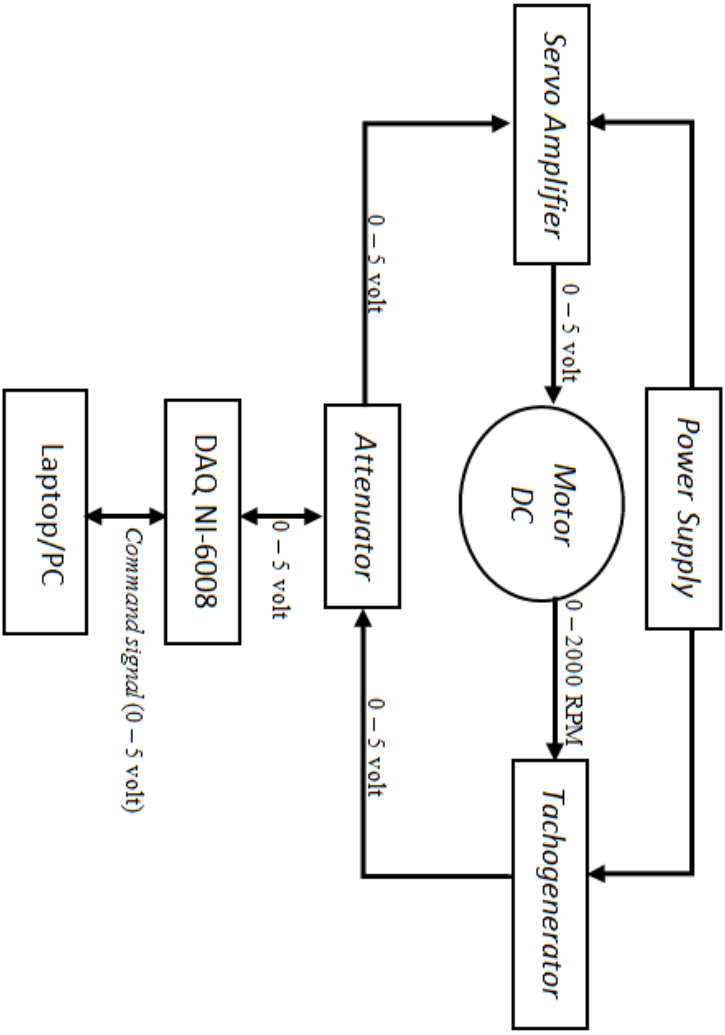
A.2 Diagram blok pengendalian kecepatan sistem servo modular MS150 DC



A.3 *Set-up* pengendalian kecepatan sistem servo modular MS150 DC dengan *attenuator*

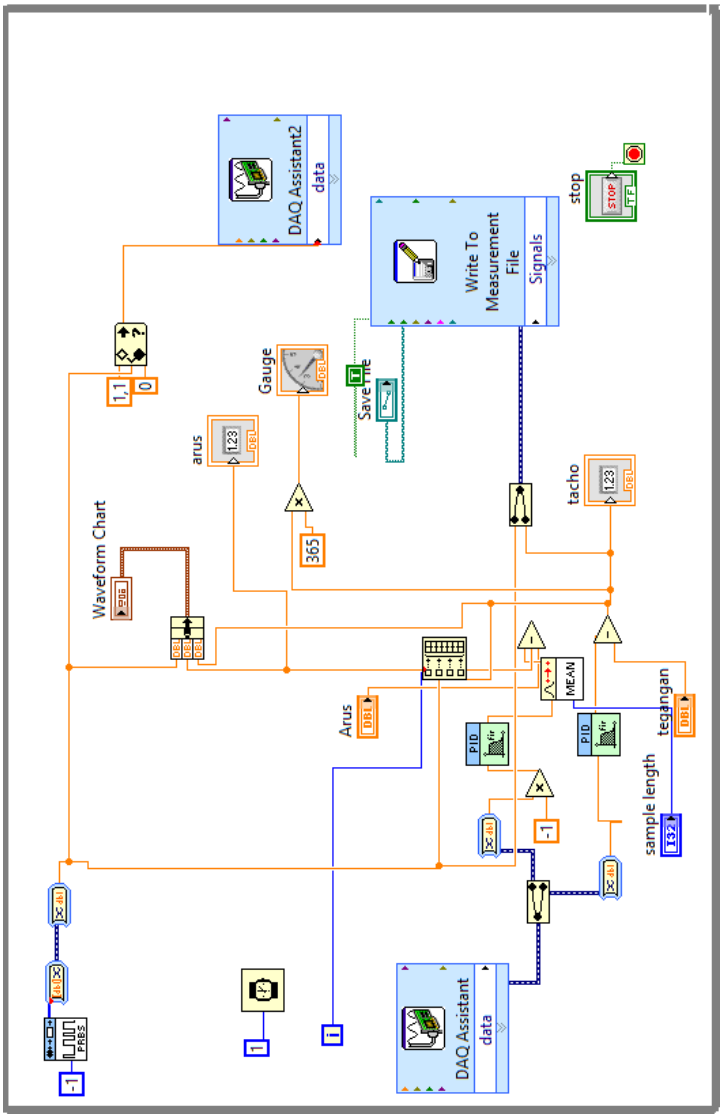


A.4 Diagram blok *set- up* pengendalian kecepatan sistem servo modular MS150 DC dengan *attenuator*



LAMPIRAN B

B.1 Program untuk pengambilan data kecepatan pada sistem servo modular MS150 DC



B.2 Data kecepatan sistem servo modular MS150 DC

Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
1	1	0,020957
2	1	0,429598
3	1	1,093034
4	1	1,769945
5	1	2,432045
6	1	2,837402
7	0	2,835063
8	0	2,790801
9	0	2,611463
10	1	2,284556
11	1	1,889273
12	1	1,577183
13	0	1,466497
14	0	1,420024
15	0	1,329824
16	1	1,144166
17	0	0,860806
18	1	0,660633
19	0	0,652847
20	0	0,7439
21	1	0,823362
22	1	0,83494
23	1	0,88512
24	1	1,042517
25	1	1,315441
26	0	1,703172

Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
27	0	2,023093
28	1	2,100837
29	0	2,050664
30	0	1,940748
31	0	1,737991
32	0	1,512076
33	0	1,269922
34	0	0,927635
35	1	0,601506
36	1	0,426784
37	1	0,475561
38	1	0,72482
39	1	1,103999
40	1	1,531862
41	1	1,89278
42	1	2,138251
43	1	2,311978
44	1	2,44118
45	0	2,538958
46	1	2,566782
47	1	2,483796
48	0	2,376999
49	0	2,256476
50	1	2,046537
51	1	1,837992
52	0	1,716254
53	0	1,607371

Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
54	1	1,497261
55	1	1,429183
56	0	1,402321
57	0	1,369027
58	0	1,31939
59	1	1,248993
60	0	1,113367
61	0	0,948754
62	0	0,805737
63	1	0,703886
64	1	0,648965
65	1	0,746054
66	0	0,99252
67	0	1,330796
68	0	1,599103
69	0	1,620313
70	0	1,350128
71	0	0,982033
72	0	0,635955
73	1	0,375519
74	0	0,232735
75	0	0,241149
76	0	0,298703
77	1	0,347414
78	0	0,396219
79	1	0,43326
80	0	0,468736

Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
81	1	0,60179
82	0	0,766748
83	1	0,899518
84	0	0,986621
85	1	0,998917
86	1	0,953491
87	1	1,038145
88	0	1,232009
89	0	1,477179
90	1	1,665294
91	1	1,745775
92	1	1,730913
93	1	1,774469
94	1	1,939615
95	1	2,189664
96	1	2,391883
97	0	2,528127
98	0	2,560859
99	0	2,438332
100	0	2,149909
101	1	1,740049
102	0	1,324319
103	0	1,055927
104	1	0,918021
105	1	0,876654
106	0	0,965212
107	1	1,078977

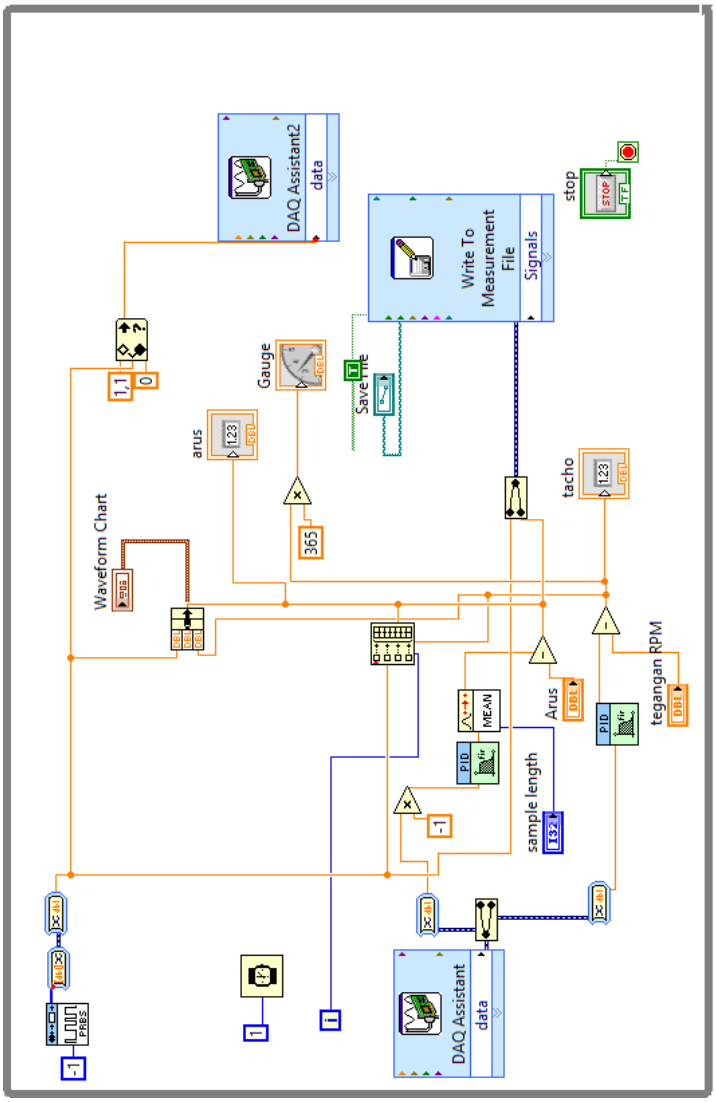
Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
108	1	1,17319
109	1	1,347578
110	1	1,53306
111	1	1,700072
112	1	1,94223
113	1	2,215315
114	0	2,417382
115	1	2,521278
116	1	2,480158
117	1	2,374692
118	0	2,291658
119	1	2,222864
120	1	2,159696
121	0	2,13642
122	0	2,079482
123	0	1,919045
124	0	1,682658
125	1	1,383553
126	1	1,042353
127	1	0,868125
128	0	0,945841
129	1	1,159861
130	0	1,379489
131	0	1,52325
132	0	1,475724
133	1	1,272132
134	0	1,066344

Time (ms)	Sinyal PRBS (V)	Sinyal Kecepatan Sistem Servo Modular MS 150DC (V)
135	0	0,925909
136	0	0,797095
137	1	0,711154
138	0	0,674655
139	0	0,656591
140	0	0,613849
141	0	0,580971
142	1	0,509812
143	0	0,42454
144	0	0,408693
145	1	0,443059
146	1	0,521805
147	0	0,687737
148	0	0,84082
149	0	0,929967
150	1	0,962701
151	1	0,906083
152	0	0,852433
153	1	0,895674
154	1	1,028137
155	1	1,245273
156	1	1,477754
157	0	1,675975
158	1	1,863371

“Halaman ini memang dikosongkan”

LAMPIRAN C

C.1 Program untuk pengambilan data arus pada sistem servo modular MS150 DC



C.2 Data arus pada sistem servo modular MS150 DC

Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
1	1	-0,120052
2	0	-0,105327
3	0	-0,143423
4	0	-0,21207
5	1	-0,280417
6	0	-0,309878
7	0	-0,319179
8	1	-0,311043
9	1	-0,278207
10	1	-0,23326
11	0	-0,194467
12	1	-0,151334
13	0	-0,092786
14	0	-0,038374
15	0	-0,003428
16	1	0,009777
17	0	0,0346
18	0	0,047325
19	0	0,043826
20	1	0,025188
21	0	0,0069
22	1	-0,026133
23	1	-0,041594
24	0	-0,005132
25	1	0,054668
26	1	0,118609

Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
27	0	0,175566
28	0	0,203932
29	0	0,200688
30	0	0,183257
31	1	0,130406
32	1	0,101091
33	1	0,09545
34	0	0,094837
35	0	0,107104
36	0	0,103959
37	1	0,048933
38	1	0,007066
39	1	0,002655
40	0	0,037267
41	1	0,099977
42	0	0,149408
43	1	0,133642
44	1	0,076368
45	0	0,03503
46	0	0,016094
47	1	0,014879
48	1	0,038037
49	0	0,044819
50	1	0,015775
51	0	0,007449
52	0	-0,004726
53	0	-0,025248

Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
54	0	-0,047925
55	0	-0,093141
56	1	-0,148538
57	1	-0,150496
58	0	-0,122869
59	1	-0,082805
60	1	-0,021111
61	1	0,024878
62	1	0,057992
63	1	0,121598
64	1	0,201804
65	1	0,268691
66	1	0,334458
67	1	0,359589
68	1	0,343452
69	0	0,315976
70	1	0,255952
71	1	0,180498
72	0	0,124409
73	0	0,055911
74	1	-0,027651
75	0	-0,077925
76	1	-0,130661
77	0	-0,169189
78	0	-0,189231
79	1	-0,217529
80	0	-0,221469

Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
81	1	-0,211954
82	0	-0,204421
83	0	-0,1719
84	1	-0,142409
85	1	-0,124121
86	0	-0,068459
87	0	-0,027508
88	0	-0,003793
89	1	0,012806
90	0	0,013425
91	0	0,00813
92	0	0,005802
93	0	-0,004828
94	0	-0,01246
95	0	-0,040834
96	0	-0,084091
97	1	-0,114776
98	0	-0,121673
99	0	-0,101305
100	0	-0,074389
101	1	-0,050455
102	0	-0,028263
103	0	-0,021818
104	1	-0,006231
105	1	0,041211
106	0	0,096077
107	0	0,133899

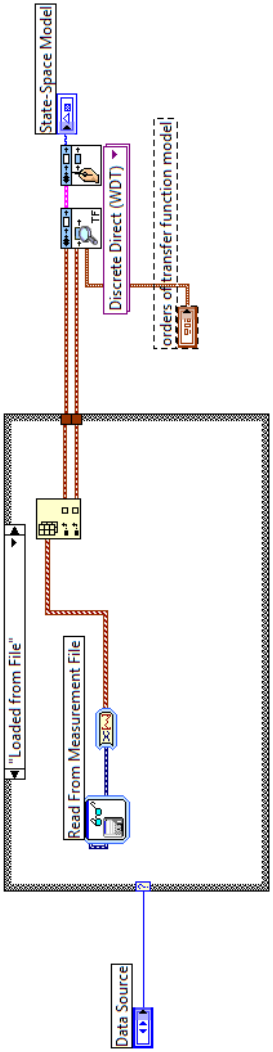
Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
108	0	0,14508
109	1	0,131373
110	1	0,107835
111	0	0,100267
112	0	0,10032
113	1	0,107588
114	0	0,121342
115	1	0,098245
116	1	0,060892
117	1	0,070009
118	0	0,094213
119	0	0,115048
120	1	0,133212
121	0	0,12183
122	0	0,077914
123	0	0,026823
124	0	-0,022647
125	1	-0,055395
126	0	-0,075789
127	0	-0,108947
128	1	-0,151036
129	1	-0,155762
130	0	-0,128814
131	0	-0,100969
132	0	-0,057504
133	1	-0,017971
134	0	0,004276

Time (ms)	Sinyal PRBS (V)	Sinyal Arus Sistem Servo Modular MS 150 DC (V)
135	1	0,026602
136	1	0,061229
137	1	0,104973
138	1	0,152844
139	0	0,178336
140	1	0,177075
141	0	0,172472
142	0	0,153764
143	1	0,127586
144	0	0,11853
145	1	0,11776
146	1	0,095599
147	1	0,074063
148	0	0,060407
149	0	0,036751
150	0	0,007536
151	1	-0,031879
152	0	-0,054607
153	1	-0,053988
154	0	-0,031153
155	0	-0,000428
156	0	0,012188
157	1	-0,023512
158	0	-0,060879

“Halaman ini memang dikosongkan”

LAMPIRAN D

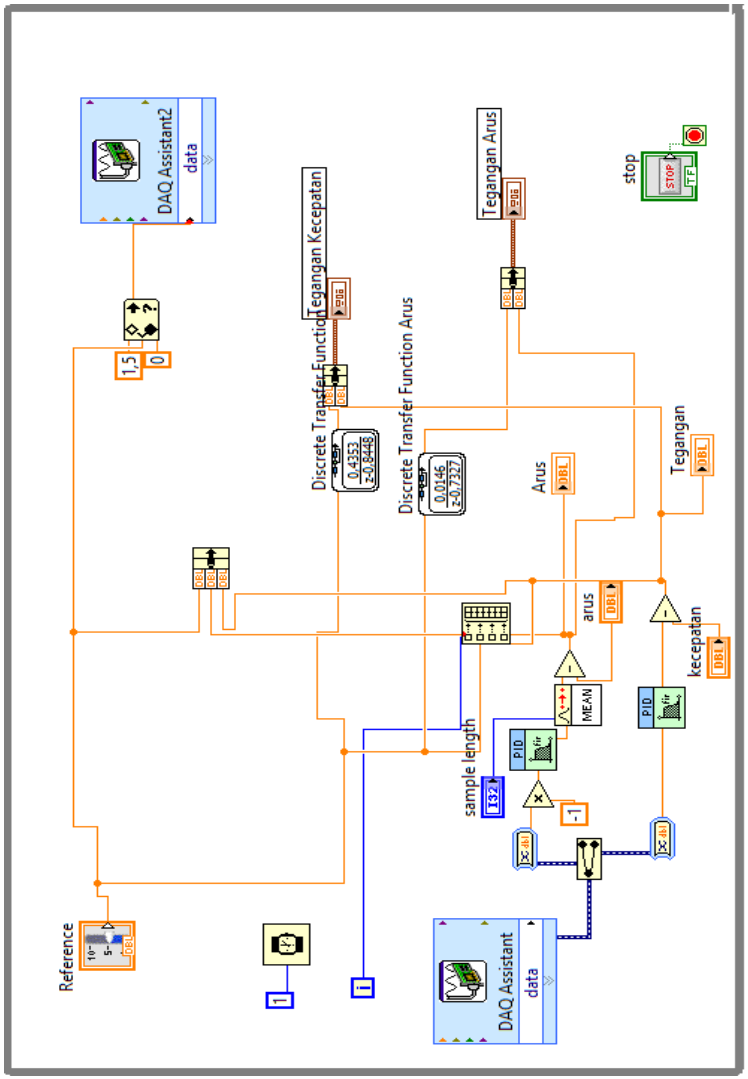
Program pemodelan sistem servo modular MS150 DC dengan metode identifikasi sistem pada perangkat lunak labVIEW 2013



“Halaman ini memang dikosongkan”

LAMPIRAN E

E.1 Program validasi model dengan sistem servo modular MS150 DC pada perangkat lunak labVIEW 2013



E.2 Data validasi kecepatan sistem servo modular MS150 DC

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
1	0	0,067427	1
2	0,435322	0,469442	0,072682035
3	0,803079	1,118385	0,281929747
4	1,113756	1,778652	0,373820174
5	1,376214	2,427372	0,433043637
6	1,597937	2,821095	0,433575615
7	1,785246	2,817039	0,366268625
8	1,943484	2,818589	0,31047627
9	2,077162	2,821496	0,263808278
10	2,190091	2,819532	0,223243077
11	2,285494	2,816974	0,188670538
12	2,366089	2,814032	0,159181914
13	2,434175	2,811101	0,13408483
14	2,491694	2,813235	0,11429582
15	2,540285	2,816367	0,098027707
16	2,581334	2,815902	0,083301194
17	2,616013	2,81643	0,071159944
18	2,645308	2,810481	0,058770367
19	2,670057	2,80798	0,049118227
20	2,690965	2,807229	0,04141593
21	2,708628	2,80638	0,034832061
22	2,723549	2,808777	0,030343456
23	2,736155	2,811937	0,026950106
24	2,746803	2,817311	0,025026701
25	2,7558	2,816411	0,021520652
26	2,763399	2,811611	0,017147465

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
27	2,76982	2,808859	0,013898526
28	2,775244	2,805818	0,010896644
29	2,779826	2,805127	0,009019556
30	2,783696	2,80863	0,008877638
31	2,786966	2,813481	0,009424268
32	2,789729	2,813965	0,008612758
33	2,792063	2,811004	0,006738162
34	2,794034	2,814607	0,007309369
35	2,7957	2,81347	0,006316044
36	2,797107	2,81108	0,004970687
37	2,798295	2,812975	0,005218674
38	2,7993	2,811013	0,004166825
39	2,800148	2,808788	0,00307606
40	2,800865	2,809734	0,003156527
41	2,80147	2,807573	0,002173764
42	2,801981	2,811728	0,003466552
43	2,802413	2,813106	0,003801137
44	2,802778	2,815654	0,004573005
45	2,803087	2,823053	0,007072485
46	2,803347	2,822548	0,006802719
47	2,803567	2,816903	0,004734277
48	2,803753	2,819122	0,005451697
49	2,80391	2,814936	0,003916963
50	2,804043	2,813618	0,003403092
51	2,804155	2,822644	0,006550242
52	2,80425	2,823881	0,00695178
53	2,80433	2,824193	0,00703316
54	2,804397	2,830127	0,009091465

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
55	2,804454	2,826695	0,007868199
56	2,804503	2,821379	0,005981472
57	2,804543	2,819698	0,005374689
58	2,804578	2,816837	0,004352045
59	2,804607	2,815817	0,003981083
60	2,804632	2,821181	0,005865983
61	2,804652	2,828704	0,008502834
62	2,80467	2,829799	0,008880136
63	2,804685	2,827943	0,008224352
64	2,804697	2,827953	0,008223616
65	2,804708	2,824105	0,006868371
66	2,804717	2,82093	0,005747395
67	2,804724	2,82184	0,006065546
68	2,804731	2,823558	0,006667828
69	2,804736	2,820915	0,005735373
70	2,80474	2,81987	0,005365496
71	2,804744	2,821219	0,005839674
72	2,804748	2,820987	0,005756496
73	2,80475	2,824461	0,006978677
74	2,804753	2,824912	0,007136151
75	2,804755	2,823767	0,00673285
76	2,804756	2,82509	0,007197647
77	2,804758	2,820748	0,005668709
78	2,804759	2,817866	0,004651392
79	2,80476	2,819423	0,00520071
80	2,804761	2,819819	0,005340059
81	2,804761	2,822019	0,00611548
82	2,804762	2,823203	0,006531943

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
83	2,804762	2,827388	0,008002439
84	2,804763	2,821991	0,00610491
85	2,804763	2,815054	0,003655703
86	2,804764	2,814293	0,00338593
87	2,804764	2,811403	0,002361454
88	2,804764	2,808882	0,001466064
89	2,804764	2,811293	0,002322419
90	2,804764	2,812025	0,002582125
91	2,804765	2,812202	0,002644547
92	2,804765	2,814371	0,003413196
93	2,804765	2,821046	0,005771264
94	2,804765	2,818516	0,004878809
95	2,804765	2,815237	0,003719758
96	2,804765	2,815283	0,003736036
97	2,804765	2,810298	0,00196883
98	2,804765	2,807194	0,000865277
99	2,804765	2,808501	0,001330247
100	2,804765	2,81096	0,002203873
101	2,804765	2,813625	0,003148963
102	2,804765	2,814608	0,003497112
103	2,804765	2,817879	0,004653855
104	2,804765	2,813685	0,00317022
105	2,804765	2,807093	0,000829328
106	2,804765	2,805238	0,000168613
107	2,804765	2,805207	0,000157564
108	2,804765	2,806811	0,000728941
109	2,804765	2,81342	0,003076327
110	2,804765	2,821164	0,005812849

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
111	2,804765	2,821541	0,005945687
112	2,804765	2,815306	0,003744176
113	2,804765	2,813677	0,003167386
114	2,804765	2,808757	0,001421269
115	2,804765	2,806821	0,000732501
116	2,804765	2,817085	0,004373315
117	2,804765	2,818706	0,004945887
118	2,804765	2,81464	0,003508442
119	2,804765	2,815673	0,003874029
120	2,804765	2,810467	0,002028844
121	2,804765	2,804748	6,06115E-06
122	2,804765	2,806103	0,000476818
123	2,804765	2,808103	0,001188703
124	2,804765	2,809728	0,001766363
125	2,804765	2,816223	0,00406857
126	2,804765	2,826745	0,007775728
127	2,804765	2,828208	0,008288994
128	2,804765	2,825426	0,007312526
129	2,804765	2,82732	0,007977519
130	2,804765	2,819087	0,005080368
131	2,804765	2,814111	0,00332112
132	2,804765	2,817621	0,004562714
133	2,804765	2,816955	0,004327368
134	2,804765	2,817022	0,004351049
135	2,804765	2,817586	0,004550349
136	2,804765	2,820322	0,005516037
137	2,804765	2,821365	0,005883677
138	2,804765	2,823187	0,00652525

Time (ms)	Data Model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
139	2,804765	2,831415	0,009412255
140	2,804765	2,832172	0,009677025
141	2,804765	2,827797	0,008144856
142	2,804765	2,829923	0,008889995
143	2,804765	2,826605	0,007726584
144	2,804765	2,824139	0,006860144
145	2,804765	2,824454	0,006970905
146	2,804765	2,827515	0,008045934
147	2,804765	2,827414	0,0080105
148	2,804765	2,829254	0,008655639
149	2,804765	2,834933	0,010641521
150	2,804765	2,831311	0,009375869
151	2,804765	2,825655	0,007392976
152	2,804765	2,827087	0,00789576
153	2,804765	2,821793	0,006034461
154	2,804765	2,821211	0,005829412
155	2,804765	2,821873	0,00606264
156	2,804765	2,819526	0,005235277
157	2,804765	2,819048	0,005066604
158	2,804765	2,818529	0,004883398
159	2,804765	2,821554	0,005950267
160	2,804765	2,822999	0,006459088
161	2,804765	2,819956	0,005386963
162	2,804765	2,821647	0,00598303
163	2,804765	2,816856	0,004292374

E.3 Data validasi arus pada sistem servo modular MS150 DC

Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
1	0	1,271585	1
2	0,014563	1,190601	0,987768362
3	0,025233	1,066312	0,976336194
4	0,033051	0,924004	0,964230674
5	0,038778	0,779145	0,95023006
6	0,042975	0,64804	0,933684649
7	0,046049	0,548428	0,916034557
8	0,048302	0,475435	0,898404619
9	0,049952	0,417831	0,880449273
10	0,051161	0,371985	0,862464884
11	0,052047	0,245415	0,787922499
12	0,052696	0,138699	0,620069359
13	0,053172	0,064933	0,18112516
14	0,05352	0,024265	1,205645992
15	0,053775	0,010078	4,335880135
16	0,053962	0,016193	2,332427592
17	0,0541	0,025234	1,143932789
18	0,0542	0,031022	0,747147186
19	0,054273	0,039308	0,380711306
20	0,054327	0,048989	0,108963237
21	0,054367	0,055264	0,016231181
22	0,054396	0,057759	0,058224692
23	0,054417	0,057463	0,053008023
24	0,054433	0,055741	0,023465672
25	0,054444	0,055006	0,010217067
26	0,054452	0,056062	0,028718205

Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
27	0,054458	0,058495	0,069014446
28	0,054463	0,059154	0,079301484
29	0,054466	0,058419	0,067666341
30	0,054468	0,053903	0,010481791
31	0,05447	0,044885	0,213545728
32	0,054472	0,036661	0,485829628
33	0,054472	0,027068	1,012413182
34	0,054473	0,018485	1,946875845
35	0,054474	0,017814	2,057931964
36	0,054474	0,019528	1,789532978
37	0,054474	0,02	1,7237
38	0,054475	0,020676	1,634697234
39	0,054475	0,016614	2,278861201
40	0,054475	0,013714	2,972218171
41	0,054475	0,017249	2,158154096
42	0,054475	0,025019	1,177345218
43	0,054475	0,03809	0,430165398
44	0,054475	0,053704	0,014356473
45	0,054475	0,062651	0,13050071
46	0,054475	0,064062	0,1496519
47	0,054475	0,062057	0,122177998
48	0,054475	0,061176	0,10953642
49	0,054475	0,064504	0,15547873
50	0,054475	0,071173	0,23461144
51	0,054475	0,078466	0,305750261
52	0,054475	0,081548	0,331988522
53	0,054475	0,077051	0,29300074
54	0,054475	0,067639	0,194621446

Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
55	0,054475	0,06021	0,095249958
56	0,054475	0,058539	0,069423803
57	0,054475	0,061836	0,119040688
58	0,054475	0,064859	0,160101142
59	0,054475	0,064068	0,149731535
60	0,054475	0,059338	0,081954228
61	0,054475	0,051232	0,063300281
62	0,054475	0,043483	0,252788446
63	0,054475	0,040436	0,347190622
64	0,054475	0,040717	0,337893263
65	0,054475	0,03962	0,374936901
66	0,054475	0,036627	0,487290796
67	0,054475	0,035501	0,534463818
68	0,054475	0,037828	0,440070847
69	0,054475	0,043593	0,249627234
70	0,054475	0,049594	0,098419164
71	0,054475	0,053631	0,015737167
72	0,054475	0,053659	0,015207141
73	0,054475	0,051188	0,064214269
74	0,054475	0,050829	0,071730705
75	0,054475	0,052458	0,038449807
76	0,054475	0,051923	0,049149702
77	0,054475	0,047068	0,157368063
78	0,054475	0,040386	0,348858515
79	0,054475	0,036585	0,488998223
80	0,054475	0,039719	0,371509857
81	0,054475	0,044729	0,21788996
82	0,054475	0,048758	0,117252553

Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
83	0,054475	0,049281	0,105395589
84	0,054475	0,045173	0,205919465
85	0,054475	0,042384	0,285272744
86	0,054475	0,045129	0,207095216
87	0,054475	0,052099	0,045605482
88	0,054475	0,06053	0,100033041
89	0,054475	0,064899	0,160618808
90	0,054475	0,060424	0,098454257
91	0,054475	0,053213	0,023716009
92	0,054475	0,047187	0,154449319
93	0,054475	0,045188	0,205519164
94	0,054475	0,046015	0,183853091
95	0,054475	0,044862	0,214279346
96	0,054475	0,038571	0,412330507
97	0,054475	0,028962	0,88091292
98	0,054475	0,019208	1,836057893
99	0,054475	0,014555	2,742700103
100	0,054475	0,015078	2,612879692
101	0,054475	0,01889	1,883800953
102	0,054475	0,02634	1,068147304
103	0,054475	0,034903	0,56075409
104	0,054475	0,043842	0,242529994
105	0,054475	0,053947	0,009787384
106	0,054475	0,065339	0,166271293
107	0,054475	0,075684	0,28023096
108	0,054475	0,082	0,335670732
109	0,054475	0,083787	0,349839474
110	0,054475	0,084003	0,351511256

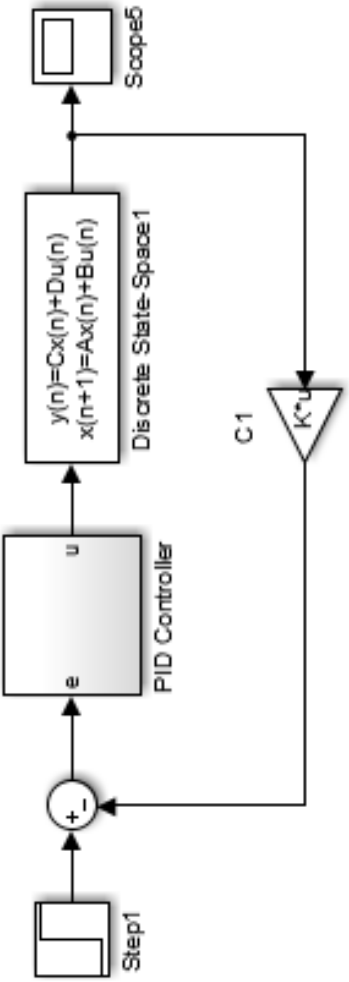
Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
111	0,054475	0,084364	0,354286188
112	0,054475	0,085876	0,365655131
113	0,054475	0,085764	0,364826734
114	0,054475	0,081934	0,335135597
115	0,054475	0,076029	0,283497087
116	0,054475	0,068107	0,200155637
117	0,054475	0,062602	0,129820134
118	0,054475	0,063657	0,144241796
119	0,054475	0,062627	0,1301675
120	0,054475	0,057942	0,059835698
121	0,054475	0,053543	0,01740657
122	0,054475	0,047607	0,144264499
123	0,054475	0,041756	0,304602931
124	0,054475	0,040643	0,340329208
125	0,054475	0,042368	0,285758119
126	0,054475	0,047529	0,146142355
127	0,054475	0,054373	0,001875931
128	0,054475	0,060413	0,098290103
129	0,054475	0,068018	0,199109059
130	0,054475	0,074287	0,266695384
131	0,054475	0,074937	0,273056034
132	0,054475	0,074818	0,27189981
133	0,054475	0,074298	0,266803952
134	0,054475	0,06806	0,199603291
135	0,054475	0,058941	0,075770686
136	0,054475	0,047442	0,148244172
137	0,054475	0,034641	0,572558529
138	0,054475	0,025795	1,111843381

Time (ms)	Data model (Ft)	Data Real (Xt)	(Xt - Ft)/Xt
139	0,054475	0,022969	1,371674866
140	0,054475	0,023458	1,322235485
141	0,054475	0,025954	1,098905756
142	0,054475	0,027748	0,963204555
143	0,054475	0,029831	0,826120479
144	0,054475	0,033276	0,637065753
145	0,054475	0,036525	0,491444216
146	0,054475	0,039973	0,362794887
147	0,054475	0,042026	0,296221387
148	0,054475	0,041185	0,3226903
149	0,054475	0,041612	0,309117562
150	0,054475	0,04493	0,212441576
151	0,054475	0,047875	0,137859008
152	0,054475	0,048514	0,122871748
153	0,054475	0,047758	0,140646593
154	0,054475	0,047468	0,147615236
155	0,054475	0,047051	0,157786232
156	0,054475	0,046582	0,169443133
157	0,054475	0,045161	0,206239897
158	0,054475	0,039873	0,366212725
159	0,054475	0,031999	0,70239695
160	0,054475	0,02499	1,179871949
161	0,054475	0,020225	1,693448702
162	0,054475	0,018742	1,906573471
163	0,054475	0,019832	1,746823316
164	0,054475	0,019864	1,742398308

“Halaman ini memang dikosongkan”

LAMPIRAN F

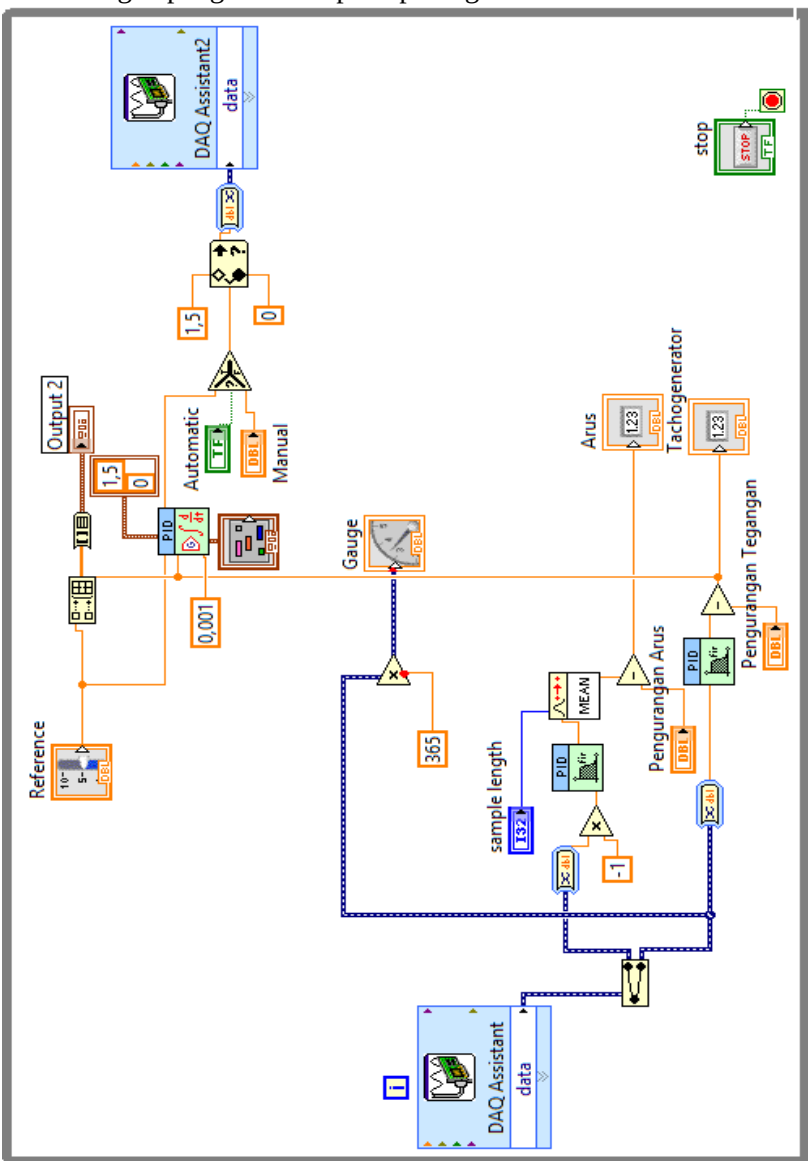
Perancangan pengendali PI pada perangkat lunak MATLAB R2013a



“Halaman ini memang dikosongkan”

LAMPIRAN G

Perancangan pengendali PI pada perangkat lunak labVIEW 2013



“Halaman ini memang dikosongkan”

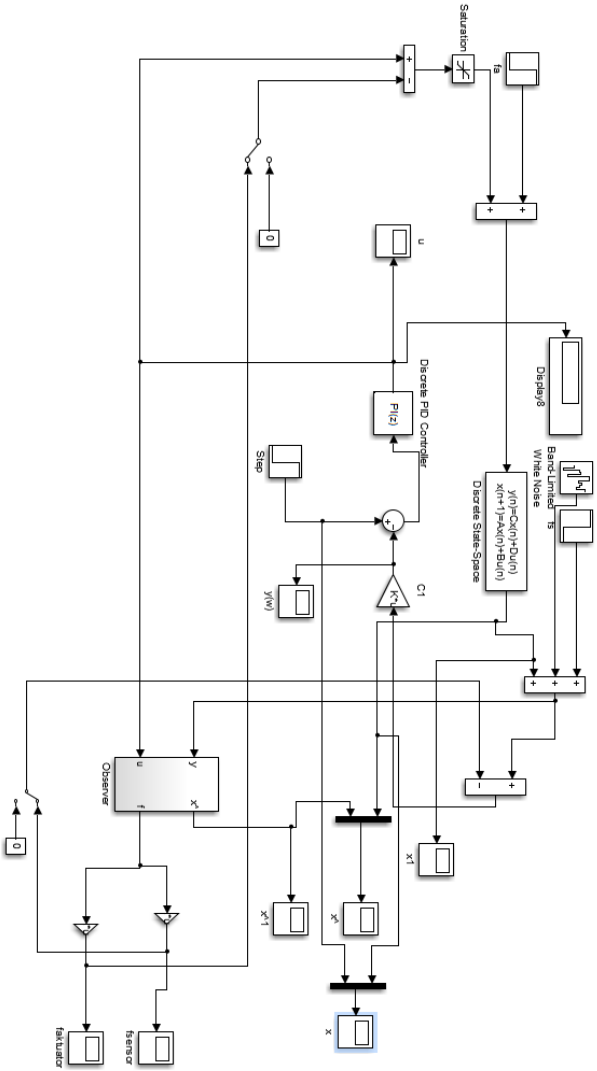
LAMPIRAN H

Perancangan Sistem *Active Fault Tolerant Control* (AFTC) Pada Perangkat Lunak MATLAB R2013a

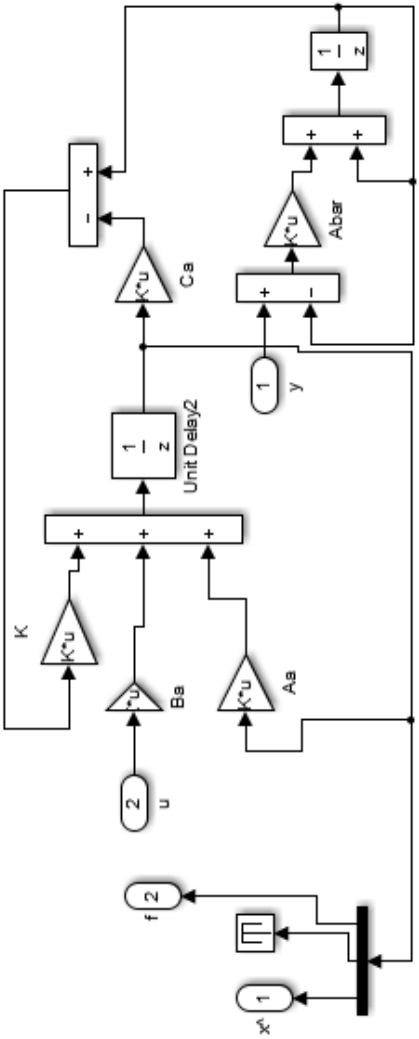
H.1 M-file perancangan sistem AFTC pada perangkat lunak MATLAB R2013a

```
clear all
clc
A= [0.844792 0; 0 0.732663]
B= [0.435322 ; 0.0145632]
C=[1 0;0 1]
D=[0;0];
Ts= 0.001;
n = length(A)
p = size(C,1)
m = size(B,2)
Fa= B;
Fs=[1;0];
fs = size(Fs,2);%y=Cx+Fsfs -> (1;0)*fs
fa = size(Fa,2); %Jumlah Baris Fa menunjukan
state
Abar = 1000;
Aa = [A zeros(n,p); Ts*Abar*C (1-
Ts*Abar)*eye(p)]
Ba = [B; zeros(p,m)]
Ea = [B zeros(n,fs); zeros(p,fa) Ts*Abar*Fs]
Ca = [zeros(p,n) eye(p)];
Atild = [Aa Ea; zeros(fa+fs,n+p) eye(fa+fs)]
Btild = [Ba; zeros(fa+fs,m)]
Ctild = [Ca zeros(p,fa+fs)]
p1 = 0.92;
p2 = 0.91;
p3 = 0.94;
p4 = 0.94;
p5 = 0.9;
p6 = 0.95;
Ktild = place(Atild',Ctild',[p1 p2 p3 p4 p5 p6])
K = Ktild(:,1:n+p)
L = Ktild(:,n+p+1:n+p+fa+fs)
```

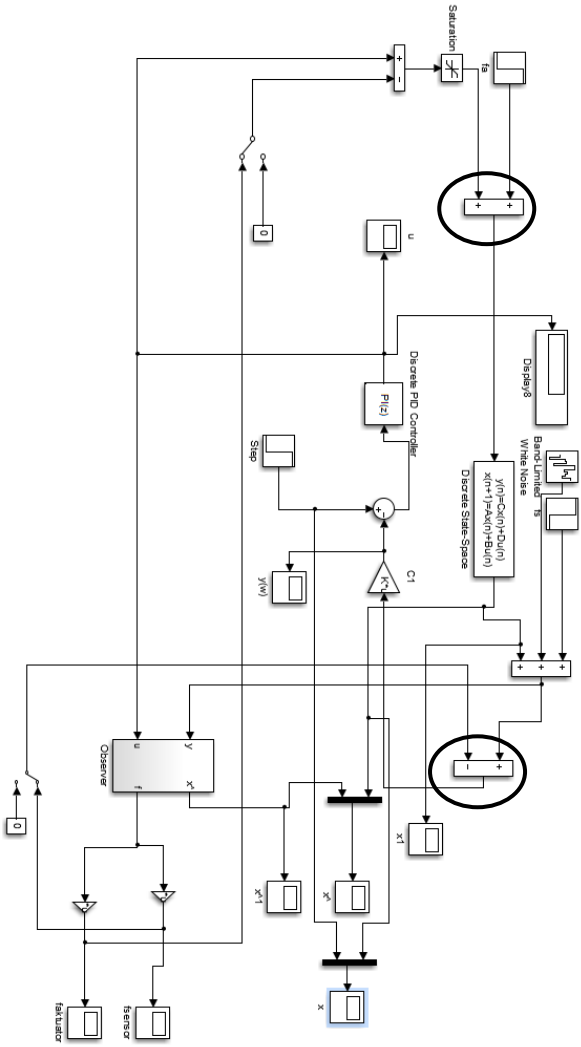
H.2 Simulink perancangan sistem AFTC pada perangkat lunak MATLAB R2013a



H.3 Simulink perancangan *observer* pada perangkat lunak MATLAB R2013a



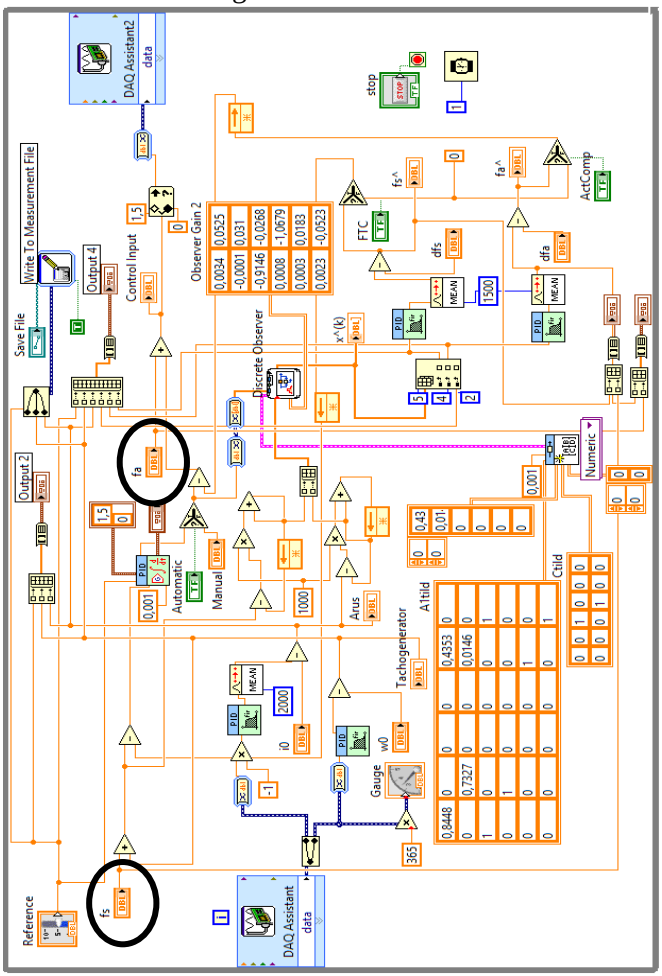
H.4 Perancangan *reconfigurable control* pada perangkat lunak MATLAB R2013a



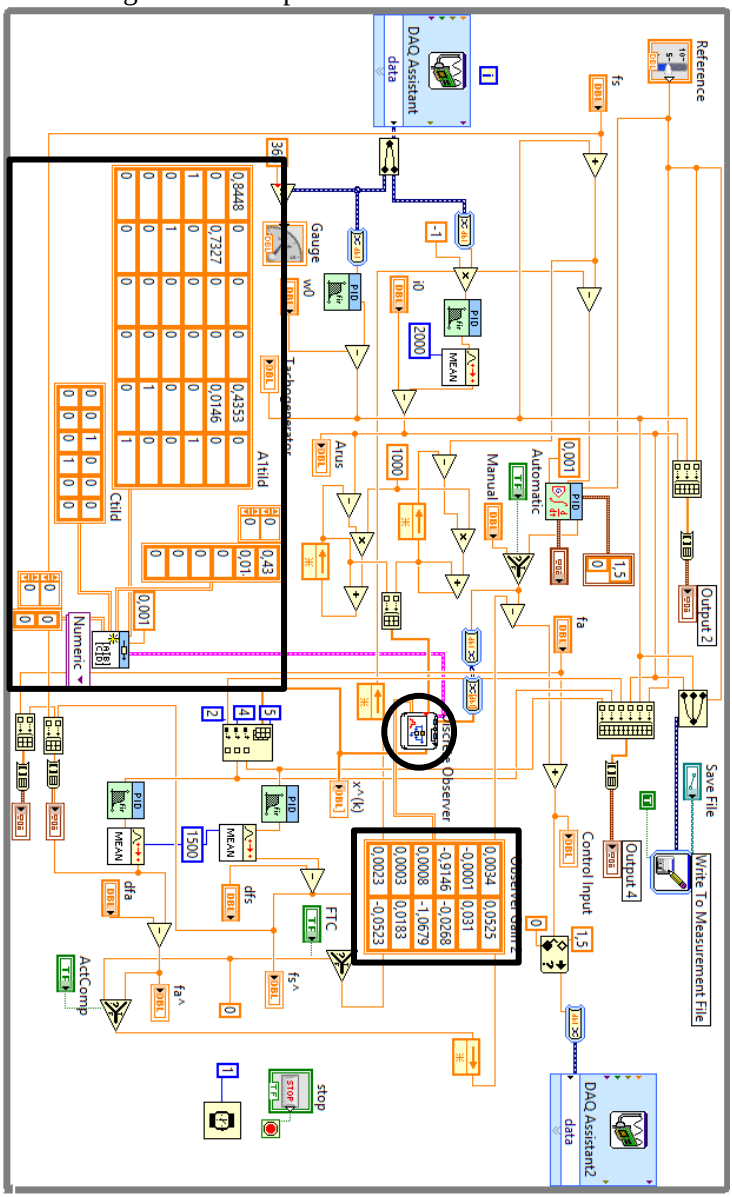
LAMPIRAN I

Perancangan Sistem *Active Fault Tolerant Control* (AFTC) Pada Perangkat Lunak labVIEW 2013

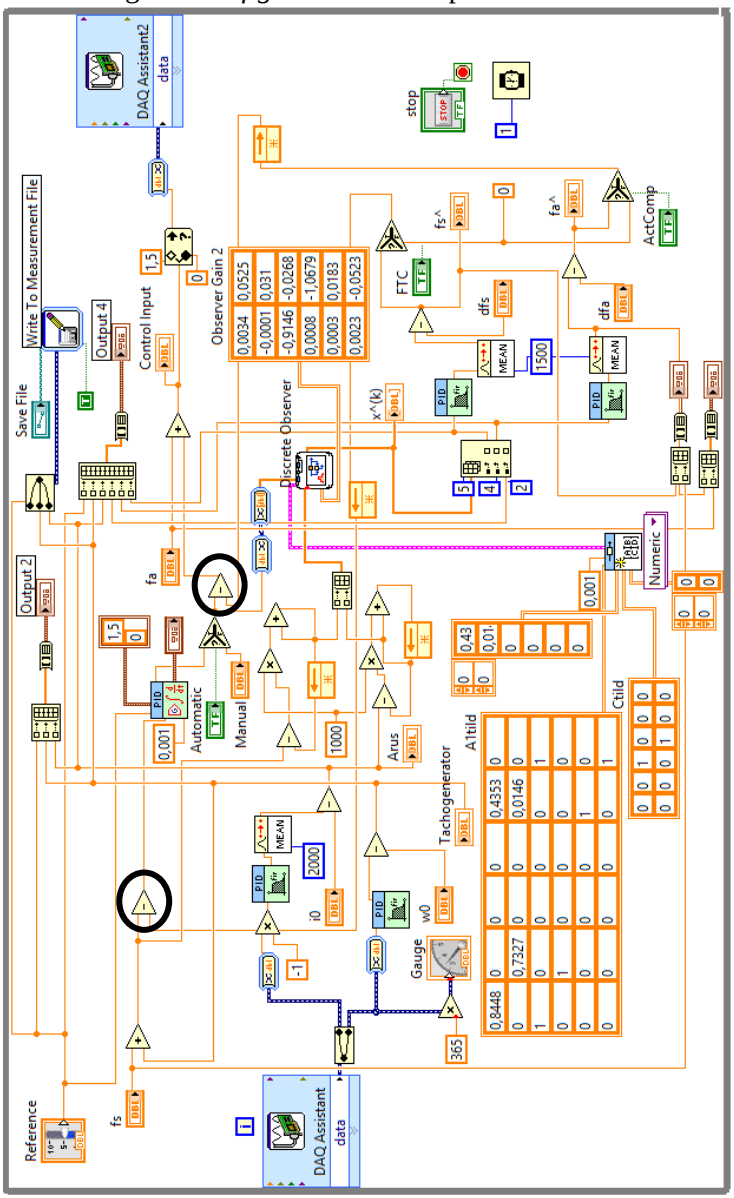
I.1 Perancangan sistem AFTC untuk diterapkan pada sistem servo modular MS150 DC dengan kesalahan simulasi



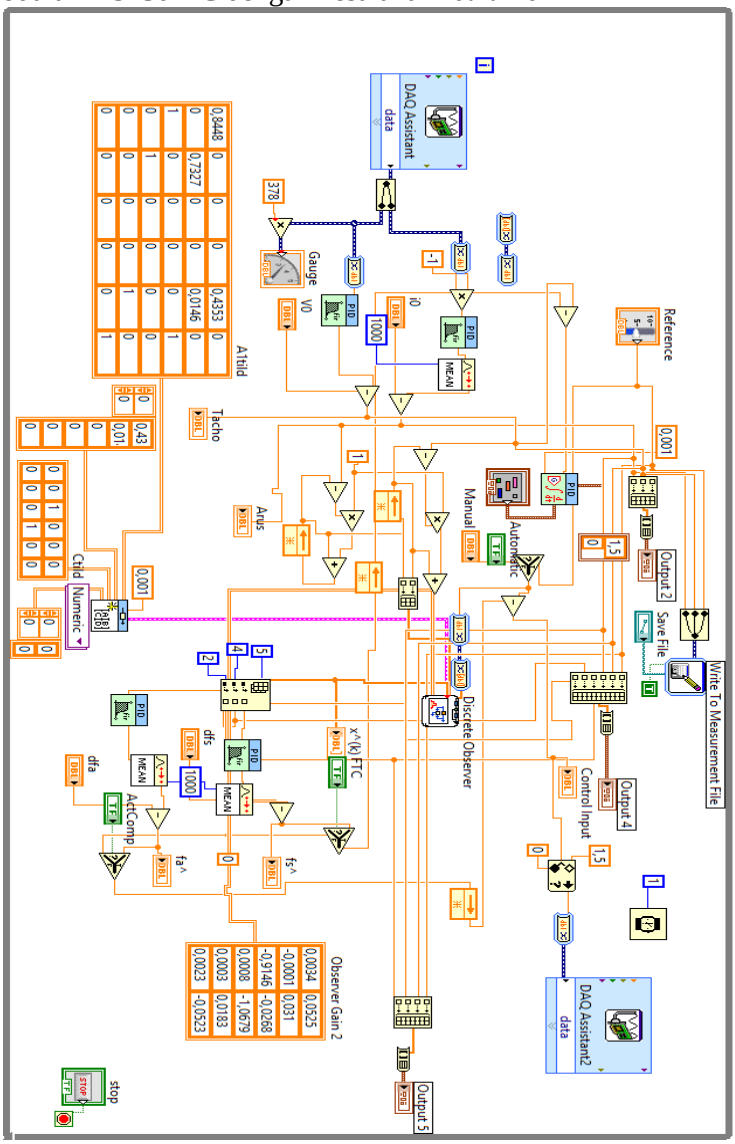
I.2 Perancangan *observer* pada sistem AFTC



I.3 Perancangan *reconfigurable control* pada sistem AFTC



I.4 Perancangan sistem AFTC untuk diterapkan pada sistem servo modular MS150 DC dengan kesalahan *realtime*



BIODATA PENULIS

Nama penulis Viqi Bhagaskara Kresna dilahirkan di Malang, tanggal 04 Maret 1995. Saat ini penulis tinggal di Jalan Sidomakmur no 22, Ngadilangkung-Kepanjen, Malang, Provinsi Jawa Timur. Penulis telah menyelesaikan pendidikan di SDN 7 Kepanjen pada tahun 2007, pendidikan di SMPN 4 Kepanjen pada tahun 2010, pendidikan di SMAN 1 Kepanjen pada tahun 2013 dan sedang menempuh pendidikan S1 Teknik Fisika FTI di Institut Teknologi Sepuluh Nopember Surabaya hingga sekarang.

Pada bulan Juli 2017 penulis telah menyelesaikan Tugas Akhir dengan judul **Perancangan Sistem *Active Fault Tolerant Control* Pada Pengendalian Kecepatan Sistem Servo Modular MS150 DC Dengan Kesalahan Pada Aktuator dan Sensor.** Bagi pembaca yang memiliki kritik, saran atau ingin berdiskusi lebih lanjut mengenai tugas akhir ini, maka dapat menghubungi penulis melalui *email*: viqibagas@gmail.com