TUNING KONTROLER PI MENGGUNAKAN TEKNIK *MODEL REFERENCE ADAPTIVE CONTROL* (MRAC) PADA SISTEM KONTROL KECEPATAN MOTOR DC

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

FAKHRUR ROZI
NIM. 115060300111080

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2015

LEMBAR PENGESAHAN

TUNING KONTROLER PI MENGGUNAKAN TEKNIK *MODEL REFERENCE ADAPTIVE CONTROL* (MRAC) PADA SISTEM KONTROL KECEPATAN MOTOR DC

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

FAKHRUR ROZI
NIM. 115060300111080

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
Pada tanggal 7 Oktober 2015

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Erni Yudaningtyas, M.T.	Ir. Mochammad Rusli, Dipl.-Ing.
NIP. 19650913 199002 2 001	NIP. 19630104 198701 1 001

LEMBAR PENGESAHAN

TUNING KONTROLER PI MENGGUNAKAN TEKNIK *MODEL REFERENCE ADAPTIVE CONTROL* (MRAC) PADA SISTEM KONTROL KECEPATAN MOTOR DC

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

FAKHRUR ROZI
NIM. 115060300111080

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 7 Oktober 2015

MAJELIS PENGUJI

Ir. Purwanto, M.T.	Ir. Retnowati, M.T.
NIP. 19540424 198601 1 001	NIP. 19511224 198203 2 001

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 19741203 200012 1 001

Mengetahui
Ketua Jurusan Teknik Elektro

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 19741203 200012 1 001

PENGANTAR

Assalamu'alaikum Warohmatulloh Wabarokaatuh

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi ini. Tak lepas shalawat serta salam tercurahkan kepada junjungan kita Nabi Muhammad SAW selaku suri tauladan bagi yang mengharapkan rahmat dan hidayah-Nya.

Penyusunan skripsi ini tidak lepas dari bantuan, bimbingan serta dorongan dari berbagai pihak. Pada kesempatan yang baik ini penulis ingin menyampaikan rasa terima kasih kepada:

- Keluarga tercinta, Bapak Abd. Munib dan Ibu Arifa yang selalu memberikan kasih sayang, support dan doa yang tiada akhir. dan selurh keluarga yang juga memberikan support serta doa. Adikku tersayang Shofi, yang selalu menghibur dan memberi semangat.
- Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Hadi Suyono, ST., MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Ir. Purwanto, MT. selaku KKDK Teknik Kontrol, selaku Dosen Pembimbing II atas segala bimbingan, masukan, serta saran yang telah diberikan selama bimbingan.
- Bapak Drs. Ir. Moch. Dhofir, M.T. selaku Dosen Pembimbing Akademik yang telah meluangkan waktu dalam memberi bimbingan akademik.
- Ibu Dr. Ir. Erni Yudaningtyas, M.T. selaku Kepala Laboratorium Sistem Kontrol yang telah meluangkan banyak waktunya untuk kemajuan lab sekaligus Dosen Pembimbing I yang telah memberikan banyak waktu, ilmu, masukan, dan kesabaran dalam membimbing.
- Bapak Ir. Moch. Rusli, Dipl.-Ing. selaku Dosen Pembimbing II yang telah memberikan banyak waktu dan ilmunya ketika bimbingan.
- Bapak Ibu Dosen, karyawan, staf recording dan RBTE atas segala bantuan dan kemudahan.
- Mbak Eka selaku Laboran Lab. Sistem Kontrol sekaligus pendengar keluh kesah selama berada di Lab .

- Keluarga Besar Lab Siskon, Mas Doni, Mas Salmi, Mas Irjan, Mas Along, Mas Gladi, Mbak Yoshiko, Mas Ade, Mas Dito, Mas Sendok, Mas Hakiki, Mas Khoirul, Mbak Hamu, Mbak Ika, Mbak Dina, Mbak Ayu, Mbak Garneta, Azri, Emon, Dimas, Tesu, Zainudin, Mirza, Dennis, Andri, Yudha, Naufal, Afif, Indra, Hilmi, Sura, Zaini, Rifan, Iqbal, Yuda, Diana dan Ana terima kasih telah memberikan banyak bantuan serta pengetahuan dalam belajar dan berkreasi.
- Teman-teman Divisi OTOMASI, Fasihal, Dimas, Naufal, Frengky, Kholid, Andri, Yudha, Dennis, Victor, Oky, Zaini, Rifan, Mukti, Wicak, Fauzan, Diana, Keiko dan Anggota divisi 2014 yang baru terima kasih atas kerjasamanya dan keceriaannya.
- Teman-teman tim robot KRI, KRPAI, dan KRSI yang telah membantu dalam peminjaman alat serta sarana dan prasarana.
- Teman-teman kos Kerto Raharjo 92, Lukman, Jayus, Fajar, Ari, Habibi, Bintang, Jaga, Pani, Halim, Amri, Adi, Yuda, Kholis dan Ariyandi terima kasih telah berbagi kesenangan, pelajaran hidup, serta canda dan tawa.
- Diana Ramadhani atas segala pengertian, kesabaran, semangat yang tak pernah putus, waktu, segala bantuan dan do'a yang telah diberikan.
- Tak lupa keluarga besar INVERTER'11, yang telah memberikan banyak kenangan dan pengalaman.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna, karena keterbatasan ilmu dan kendala yang terjadi selama pengerjaan skripsi. Oleh karena itu, penulis terbuka terhadap kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis juga berharap tulisan ini dapat bermanfaat bagi kita semua.

Malang, Oktober 2015

Penulis

ABSTRAK

Fakhrur Rozi, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Oktober 2015, *Tuning Kontroler PI Menggunakan Model Reference Adaptive Control (MRAC) pada Sistem Kontrol Kecepatan Motor DC*, Dosen Pembimbing: Erni Yudaningtyas dan Mochammad Rusli.

Motor DC dapat menyediakan sebuah torsi awal yang tinggi dan juga memungkinkan untuk mendapatkan berbagai kontrol kecepatan. Motor DC memiliki respon yang cepat, namun masih memiliki *error steady state*. Kontrol kecepatan motor DC dengan menggunakan kontroler PI dapat menghilangkan *error steady state* pada respon motor, namun berdampak pada kecepatan respon yang lambat dalam mencapai nilai *steady state*. Diantara teknik perancangan sistem kontrol adalah dengan menggunakan teknik perancangan kontrol adaptif, yaitu *Model Reference Adaptive Control* (MRAC) yang memiliki ide dasar untuk membuat respon sistem yang dikontrol agar dapat menyerupai perilaku yang sama dengan model referensi. Oleh karena itu, teknik MRAC dapat digunakan sebagai *tuning* kontroler PI pada kontrol kecepatan motor DC untuk meningkatkan kecepatan respon motor DC mencapai keadaan *steady state*. Respon motor DC hasil implementasi dengan setpoint 150 rpm, 250 rpm dan 350 rpm memiliki nilai *error steady state* rata-rata berada dibawah toleransi 2%, masing-masing adalah 1,96%, 1,557%, dan 1,276%. Sedangkan *settling time* masing-masing adalah 14 detik, 10,25 detik dan 3,8 detik. Pada respon sistem dengan perubahan nilai *setpoint* memiliki nilai *error steady state* rata-rata dibawah 2% yaitu 1,6%. Ketika sistem diberi gangguan pada *setpoint* 150 rpm, 250 rpm dan 350 rpm, respon akan mengalami perlambatan dan *recovery time* respon kembali pada keadaan *steady state* masing-masing adalah 4,55 detik, 3,85 detik dan 3,5 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon kembali pada keadaan *steady state* masing-masing adalah 7,4 detik, 9,65 detik dan 10,9 detik.

Kata Kunci: motor DC, kontrol kecepatan, kontroler PI, MRAC.

DAFTAR ISI

PENGANTAR ..... **i**
ABSTRAK ..... **iii**
DAFTAR ISI ..... **iv**
DAFTAR TABEL ..... **vi**
DAFTAR GAMBAR ..... **vii**
BAB I PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Sistematika Penulisan 3
BAB II DASAR TEORI **4**
2.1 Motor DC 4
2.2 *Incremental Rotary Encoder* 5
2.3 Mikrokontroler Arduino Mega 2560 6
2.3.1 Tegangan Sumber 6
2.3.2 Memori 6
2.3.3 *Input* dan *Output* 7
2.3.4 Komunikasi 7
2.4 *Driver* Motor DC (*Dual H-bridge*) 8
2.5 *Pulse Width Modulation* (PWM) 8
2.6 Kontroler *Proporsional Integral* (PI) 9
2.7 *Model Reference Adaptive Control* (MRAC) 10
2.8 Model Referensi 11
2.9 Aturan MIT 11
2.10 Transformasi Z 12
2.11 Sampling Data 13
2.12 Diskritisasi 15
BAB III METODE PENELITIAN **17**
3.1 Perancangan Diagram Blok Sistem 17
3.2 Pembuatan Perangkat Keras 18
3.3 Perancangan Algoritma 18

3.4 Pengujian Sistem 19
3.5 Pengambilan Kesimpulan dan Saran 19
BAB IV PEMBUATAN ALAT DAN PERANCANGAN ALGORITMA 20
4.1. Spesifikasi Alat 20
4.2. Pembuatan Perangkat Keras 21
4.3. Prinsip Kerja Sistem 22
4.4. Desain Kontroler PI menggunakan Teknik MRAC 23
4.5. Penentuan Fungsi Alih Motor DC D-6759 27
4.6. Validasi Fungsi Alih Motor DC D-6759 29
4.7. Penentuan Model Referensi 29
4.8. Penetapan Parameter Kontroler 30
4.9. Desain Persamaan Beda 30
4.10. *Flowchart* Program 34
BAB V PENGUJIAN, SIMULASI DAN ANALISIS SISTEM 36
5.1 Pengujian Sistem 36
5.1.1 Motor DC D-6759 36
5.1.2 Pengujian *Driver* Motor IC L298N (*Dual H-Bridge*) 38
5.1.3 Pengujian *Rotary Encoder* E50S8-360-3-N-24 40
5.2 Simulasi Penggunaan Teknik MRAC pada Kontroler PI 42
5.3 Simulasi 43
5.3.1 Hasil Simulasi Penentuan Nilai Parameter Kontroler 43
5.3.2 Hasil Simulasi Terhadap Perubahan Nilai *Setpoint* 45
5.4 Implementasi 46
5.4.1 Hasil Implementasi dengan Beberapa Nilai *Setpoint* 46
5.4.2 Hasil Implementasi dengan Perubahan Nilai *Setpoint* 49
5.4.3 Hasil Implementasi dengan Diberikan Gangguan 50
BAB VI KESIMPULAN DAN SARAN 53
6.1 Kesimpulan 53
6.2 Saran 53
DAFTAR PUSTAKA 54
Lampiran 1 Foto Alat 55
Lampiran 2 Diagram Blok 58
Lampiran 3 *Listing* Program 60
Lampiran 4 *Datasheet* 66

DAFTAR TABEL

No. Judul Halaman

Tabel 5.1 Data pengujian kecepatan motor DC terhadap tegangan 37
Tabel 5.2 Data pengujian *driver* motor IC L298N 39
Tabel 5.3 Data pengujian *rotary encoder* E50S8-360-3-N-24 41
Tabel 5.4 Nilai parameter kontrol..... 44

DAFTAR GAMBAR

No. Judul Halaman

Gambar 2.1 Elemen-Elemen Dasar Motor DC 4
Gambar 2.2 Rangkaian ekuivalen motor DC magnet permanen 4
Gambar 2.3 Diagram blok lup tertutup 5
Gambar 2.4 Tiga buah sinyal keluaran encoder 5
Gambar 2.5 *Increment rotary encoder* Autonics E50 Series 6
Gambar 2.6 Arduino Mega 2560 6
Gambar 2.7 IC *driver* Motor L298N 8
Gambar 2.8 Sinyal PWM secara Umum 9
Gambar 2.9 Diaram blok kontroler PI 9
Gambar 2.10 Skema *Model Reference Adaptive Control* (MRAC) 10
Gambar 2.11 Sinyal tercuplik dalam bentuk pulsa 13
Gambar 2.12 Sinyal *input* dan *output* dari pencuplik/*data hold* 14
Gambar 2.13 Pencuplik dan *data hold* 14
Gambar 2.14 Representasi dari *pencuplik* dan *data hold* 15
Gambar 3.1 Diagram Blok Sistem dengan *Model Reference Adaptive Control* (MRAC) .. 17
Gambar 3.2 Skema pembuatan perangkat keras 18
Gambar 4.1 *Power Supply Unit* (PSU) 20
Gambar 4.2 Mikrokontroler Arduino Mega 2560 20
Gambar 4.3 *Driver* motor IC L298N 21
Gambar 4.4 Motor DC D-6759 21
Gambar 4.5 *Rotary encoder* Autonics E50S8-360-3-N-24 21
Gambar 4.6 Skema pembuatan perangkat keras 21
Gambar 4.7 Modul motor DC D-6759 dan *rotary encoder* Autonics E50S8-360-3-N-24 . 22
Gambar 4.8 Modul mikrokontroler Arduino Mega 2560 dan *driver* motor IC L298N 22
Gambar 4.9 Diagram blok sistem *loop* tertutup 23
Gambar 4.10 Diagram blok sistem dengan model referensi 24
Gambar 4.11 Diagram blok penggunaan teknik MRAC pada *tuning* kontroler PI 26
Gambar 4.12 Respon sinyal PRBS dan kecepatan motor DC D-6759 27
Gambar 4.13 *System Identification Toolbox* 28
Gambar 4.14 Hasil estimasi model 28

Gambar 4.15 Respon fungsi alih motor DC D-6759 dengan masukan *unit step* 29
Gambar 4.16 Validasi fungsi alih dengan respon motor DC D-6759 29
Gambar 4.17 Respon fungsi alih model referensi dengan masukan *unit step* 30
Gambar 4.18 Diagram blok sistem dengan desain kontroler PI menggunakan teknik MRAC 31
Gambar 5.1 Grafik perubahan kecepatan motor DC terhadap perubahan tegangan 38
Gambar 5.2 Grafik perubahan tegangan *output driver* terhadap *duty cycle* 40
Gambar 5.3 Grafik perubahan respon kecepatan motor DC terhadap *duty cycle* 42
Gambar 5.4 Diagram blok simulink pada Matlab 43
Gambar 5.5 Grafik respon simulasi sistem dengan variasi nilai parameter $\boldsymbol{\gamma p}$ dan $\boldsymbol{\gamma i}$ 44
Gambar 5.6 Grafik respon simulasi *error* sistem 44
Gambar 5.7 Grafik respon simulasi perubahan Kp dan Ki 45
Gambar 5.8 Grafik respon simulasi sinyal kontrol 45
Gambar 5.9 Grafik respon simulasi sistem dengan perubahan nilai *setpoint* 45
Gambar 5.10 Grafik respon simulasi perubahan nilai Kp dan Ki dengna perubahan nilai *setpoint* 46
Gambar 5.11 Grafik perubahan nilai Kp dan Ki *setpoint* 150 rpm 46
Gambar 5.12 Grafik perubahan nilai Kp dan Ki *setpoint* 250 rpm 47
Gambar 5.13 Grafik perubahan nilai Kp dan Ki *setpoint* 350 rpm 47
Gambar 5.14 Grafik respon motor DC dengan *setpoint* 150 rpm 48
Gambar 5.15 Grafik respon motor DC dengan *setpoint* 250 rpm 48
Gambar 5.16 Grafik respon motor DC dengan *setpoint* 350 rpm 48
Gambar 5.17 Grafik respon perubahan nilai Kp dan Ki dengan perubahan nilai *setpoint* .. 49
Gambar 5.18 Grafik respon sistem dengan perubahan nilai *setpoint* 49
Gambar 5.19 Grafik perubahan nilai Kp dan Ki *setpoint* 150 rpm dan diberi gangguan 50
Gambar 5. 20 Grafik perubahan nilai Kp dan Ki *setpoint* 250 rpm dan diberi gangguan ... 50
Gambar 5. 21 Grafik perubahan nilai Kp dan Ki *setpoint* 350 rpm dan diberi gangguan ... 51
Gambar 5. 22 Grafik respon motor DC dengan *setpoint* 150 rpm dan diberi gangguan 51
Gambar 5. 23 Grafik respon motor DC dengan *setpoint* 250 rpm dan diberi gangguan 52
Gambar 5. 24 Grafik respon motor DC dengan *setpoint* 350 rpm dan diberi gangguan 52

BAB I
PENDAHULUAN

1.1 Latar Belakang

Motor *Direct Current* (DC) merupakan aktuator yang banyak digunakan dalm teknologi kontrol. Motor DC memiliki respon yang cepat, namun masih memiliki *error steady state* (*offset*). Motor DC memiliki karakteristik-karakteristik variabel dan telah digunakan secara luas dalam kontrol kecepatan. Motor DC dapat menyediakan sebuah torsi awal yang tinggi dan juga memungkinkan untuk mendapatkan berbagai kontrol kecepatan.

Kontrol kecepatan motor DC dengan menggunakan kontroler PI dapat menghilangkan *error steady state* pada respon motor, namun berdampak pada kecepatan respon yang lambat dalam mencapai nilai *steady state.* Sehingga diharapkan dengan penambahan model referensi, respon yang lambat tersebut dapat dihilangkan.

Di dalam kehidupan sehari-hari, adaptasi diartikan mengubah perilaku untuk menyesuaikan dengan lingkungan baru. Secara intuisi, sebuah kontroler adaptif adalah sebuah kontroler yang mampu memodifikasi perilaku responnya untuk mengubah dinamika proses dan karakter gangguan (*disturbances*) (Astrom, 1995).

Banyak teknik perancangan sistem yang dapat diterapkan untuk mendesain suatu sistem, antara lain teknik perancangan dengan kontrol adaptif. Hal ini terutama untuk melakukan hal-hal sebagai berikut: mengeliminasi gangguan dari luar (*disturbances*), mengeliminasi gangguan dari dalam (perubahan parameter sistem atau pada sistem yang bekerja di luar daerah linier), mengatasi keterbatasan perancangan klasik yang umumnya sukar direalisasikan (Astrom, 1995).

Terdapat banyak teknik perancangan sistem kontrol dengan kontrol adaptif, beberapa diataranya adalah *Model Reference Adaptive Control* (MRAC). MRAC atau juga diketahui sebagai *Model Reference Adaptive System* (MRAS) memiliki ide dasar untuk membuat keluaran sistem yang dikontrol agar dapat menyerupai perilaku yang sama dengan model referensi yang diberikan (Ali dkk, 2012).

Pada skripsi sebelumnya telah dilakukan pengembangan rangkaian dan sistem yang dapat digunakan untuk mengontrol kecepatan motor DC menggunakan kontrol PID dengan respon yang termonitor secara *real time* dalam bentuk grafik sehingga mempermudah dalam mengetahui perubahan yang terjadi. Kelemahan pada skripsi ini terdapat pada penggunaan sensor kecepatan yang memilki resolusi rendah dan fungsi alih

motor DC yang didapat memiliki *best fits sebesar* 82,7 dimana masih dapat dilakukan pencarian fungsi alih motor DC dengan nilai *best fits* yang lebih besar mendekati 100 dengan cara melakukan lebih banyak uji coba (Faishol, 2014).

Pada skripsi ini akan membahas pengembangan alat dan perancangan sistem kontrol dengan menggunakan teknik MRAC pada *tuning* kontroler PI sebagai kontrol kecepatan motor DC. Dalam penggunaan kontrol adaptif pada skripsi ini, diharapkan sistem dapat memiliki tingkah laku yang sama dengan model yang diberikan.

1.2 Rumusan Masalah

Mengacu pada permasalahan yang telah diuraikan pada latar belakang, maka rumusan masalah dapat ditekankan pada poin berikut:

1. Bagaimana perancangan algoritma dan performansi sistem kontrol kecepatan motor DC dengan menggunakan teknik *Model Reference Adaptive Control* (MRAC) pada *tuning* kontroler PI.
2. Bagaiamana respon sistem jika diberikan perubahan nilai *setpoint* dan gangguan.

1.3 Batasan Masalah

Mengacu pada permasalahan pada skripsi ini, maka akan dibatasi pada:

1. Motor DC yang digunakan adalah motor DC D-6759 dengan catu 24 V, arus 0,47 A.
2. Mikrokontroler yang digunakan adalah Arduino Mega 2560.
3. Sensor yang digunakan adalah *rotary encoder* E50S8-360-3-N-24 dengan range pulsa 0 – 360 pulsa per rotasi.
4. Pembahasan ditekankan pada penggunaan teknik *Model Reference Adaptive Control* (MRAC) pada kontroler PI.
5. Gangguan yang diberikan berupa pemberian medan magnet pada piringan besi yang terkopel dengan motor DC.

1.4 Tujuan

Tujuan skripsi ini adalah terwujudnya *tuning* kontroler PI dengan teknik *Model Reference Adaptive Control* (MRAC) pada sistem kontrol kecepatan motor DC.

1.5 Sistematika Penulisan

Adapun sistematika dari penulisan skripsi ini adalah:

BAB I PENDAHULUAN

Menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika penulisan.

BAB II DASAR TEORI

Menjelaskan dasar teori yang mendukung dalam pembuatan alat dan perancangan algoritma yang terdiri atas motor DC, *incremental rotary encoder*, mikrokontroler Arduino Mega 2560, *driver* motor, *Pulse Width Modulation* (PWM), kontroler Proporsional Integral (PI), *Model Reference Adaptive Control* (MRAC), model referensi, aturan MIT, transformasi Z, sampling data pada sistem kontrol dan diskritisasi.

BAB III METODE PENELTITIAN

Merupakan cara ilmiah untuk mendapatkan informasi dengan tujuan dan manfaat tertentu.

BAB IV PEMBUATAN ALAT DAN PERANCANGAN ALGORITMA

Membahas spesifikasi alat, pembuatan perangkat keras, prinsip kerja alat, perancangan algoritma sistem kontrol adaptif dan sistem kontrol digital.

BAB V PENGUJIAN, SIMULASI DAN ANALISIS SISTEM

Membahas hasil pengujian sistem, simulasi sistem dan analisis data secara keseluruhan terhadap alat yang telah direalisasikan.

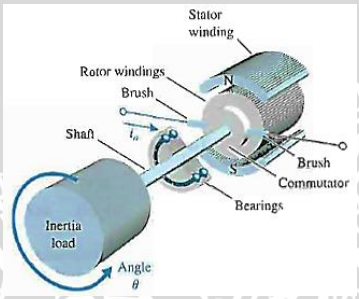
BAB VI KESIMPULAN DAN SARAN

Menjelaskan tentang pengambilan kesimpulan sesuai dengan hasil perancangan algoritma dan pengujian alat serta saran yang diperlukan untuk dilakukan pengembangan selanjutnya.

BAB II
DASAR TEORI

2.1 Motor DC

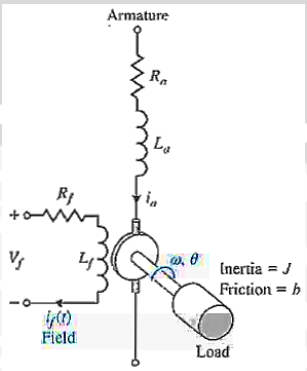
Motor *Direct Current* (DC) memerlukan sumber tegangan yang searah pada kumparan medan untuk diubah menjadi energi mekanik. Bagian utama motor DC pada Gambar 2.1 adalah *stator* (bagian yang tidak berputar) dan kumparan jangkar disebut *rotor* (bagian yang berputar). Tegangan sumber DC dari *power supply* atau baterai menuju ke lilitan melalui sikat yang menyentuh komutator (dua segmen yang terhubung dengan dua ujung lilitan). Kumparan dalam satu lilitan disebut angker dinamo. Angker dinamo adalah sebutan untuk komponen yang berputar diantara medan magnet.



Gambar 2.1 Elemen-Elemen Dasar Motor DC
Sumber : Dorf and Robert, 2008

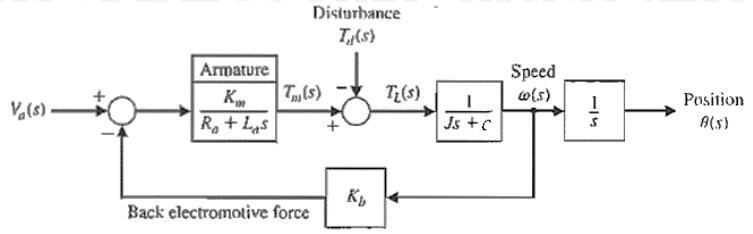
Prinsip kerja motor DC adalah jika sikat arang terhubungkan dengan satu sumber arus serah diluar dengan tegangan V, maka satu arus I masuk ke terminal kumparan rotor dan menghasilkan fluks. Dengan adanya fluks stator dan arus rotor akan menghasilkan satu gaya yang bekerja pada kumparan yang dikenal dengan gaya lorentz.

Rangkaian ekuivalen dari motor DC magnet permanen dengan metode kontrol menggunakan masukan tegangan jangkar ditunjukkan pada Gambar 2.2.



Gambar 2.2 Rangkaian ekuivalen motor DC magnet permanen
Sumber : Dorf and Robert, 2008

Pada pengontrolan arus kumparan jangkar, arus kumparan medan I_f dibuat konstan dan arus kumparan jangkar dikontrol melalui tegangan V_a. Diagram lup tertutup dari pengontrolan motor DC magnet permanen dapat dilhat pada Gambar 2.3.



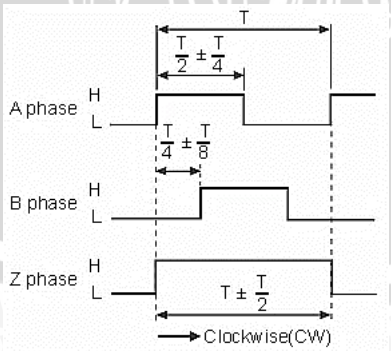
Gambar 2.3 Diagram blok lup tertutup
Sumber: Dorf and Robert, 2008

Berdasarkan diagram blok lup tertutup pada Gambar 2.3, maka akan didapatkan sebuah fungsi alih kecepatan motor DC pada Persamaan 2.1.

$$\frac{\omega(s)}{V_a(s)} = \frac{K_m/L_a J}{(s + R_a/L_a)(s + c/J) + (K_b K_m/L_a J)} \tag{2.1}$$

2.2 *Incremental Rotary Encoder*

Prinsip kerja *Incremental rotary encoder* adalah mengukur nilai sesaat posisi angular dari sebuah *shaft* yang sedang berotasi dan menghasilkan pulsa-pulsa pada chanel-chanelnya. Pulsa-pulsa yang dihasilkan ini berbentuk gelombang *square*. *Incremental rotary encoder* biasanya memiliki tiga buah sinyal keluaran, sinyal A, sinyal B, dan sinyal Z yang ditunjukkan pada Gambar 2.4 dan *Incremental rotary encoder* yang akan digunakan dapat dilihat pada Gambar 2.5.



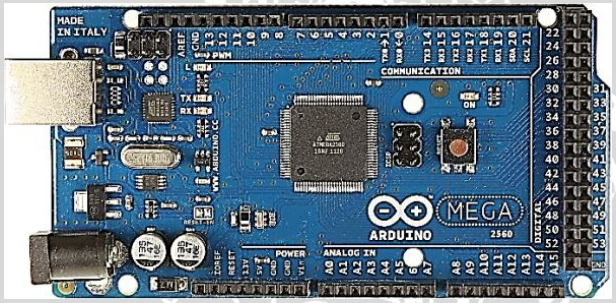
Gambar 2.4 Tiga buah sinyal keluaran encoder
Sumber : Autonics E50 Series datasheet

Untuk kebanyakan peralatan mesin motor atau aplikasi *positioning*, sinyal Z dikenal sebagai indeks sinyal yang memilki peranan penting dalam menentukan *zero position* dengan cara memberikan sebuah pulsa keluaran tunggal per satu revolusi (Morris, S Alan).

Gambar 2.5 *Increment rotary encoder* Autonics E50 Series
Sumber : Autonics E50 Series datasheet

2.3 Mikrokontroler Arduino Mega 2560

Arduino Mega 2560 adalah papan mikrokontroler berdasarkan ATmega328 (lihat Gambar 2.6). *Board* ini memiliki 54 pin digital *input/output* (14 pin dapat digunakan sebagai *output* PWM), 16 *input* analog, 16 MHz osilator kristal, USB koneksi, *jack* listrik, *header* ICSP, dan tombol *reset*.



Gambar 2.6 Arduino Mega 2560
Sumber : *www.electroschematics.com*

2.3.1 Tegangan Sumber

Arduino Mega 2560 dapat diaktifkan melalui koneksi USB atau dengan tegangan sumber eksternal. Eksternal (non-USB) daya dapat berasal baik dari AC ke adaptor DC atau baterai. Adaptor ini dapat dihubungkan dengan menancapkan *plug jack* pusat-positif ukuran 2.1 mm konektor daya. Ujung kepala dari baterai dapat dimasukkan kedalam Gnd dan Vin pin *header* dari konektor *power*. Arduino dapat beroperasi dengan tegangan sumber eksternal 6 V sampai 20 V. Namun jika menggunakan lebih dari 12 V, regulator tegangan dapat panas dan merusak papan. Kisaran yang disarankan adalah 7 V sampai 12 V.

2.3.2 Memori

ATmega 2560 memiliki 256 KB (dengan 8 KB digunakan untuk *bootloader*), 8KB dari SRAM dan 4 KB EEPROM.

2.3.3 *Input* dan *Output*

Masing-masing dari 54 pin digital di Arduino Mega 2560 dapat digunakan sebagai *input* atau *output*, dengan menggunakan fungsi *pinMode*(), *digitalWrite*(), dan *digitalRead* (), beroperasi dengan daya 5 V. Setiap pin dapat memberikan atau menerima maksimum 40 mA dan memiliki internal *pull-up* resistor (secara *default* terputus) dari 20-50 Kohm. Selain itu, beberapa pin memiliki fungsi khusus:

- Serial: 0 (RX) dan 1 (TX); Serial 1: 19 (RX) dan 18 (TX); Serial 2: 17 (RX) dan 16 (TX); Serial 3: 15 (RX) dan 14 (TX). Digunakan untuk menerima (RX) dan mengirimkan (TX) TTL data serial. Pin ini dihubungkan ke pin yang berkaitan dengan *chip* Serial ATmega8U2 USB-to-TTL.
- Eksternal *interrupt*s: 2 (*interrupt* 0), 3 (*interrupt* 1), 18 (*interrupt* 5), 19 (*interrupt* 4), 20 (*interrupt* 3), dan 2 (*interrupt* 2). Pin ini dapat dikonfigurasi untuk memicu *interrupt* pada nilai yang rendah, dengan batasan tepi naik atau turun, atau perubahan nilai.
- PWM : 0 - 13. Menyediakan *output* PWM 8-bit dengan fungsi analogWrite ().
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Pin ini mendukung komunikasi SPI menggunakan SPI *library*.
- LED: 13. Ada built-in LED terhubung ke pin digital 13. Ketika pin bernilai nilai *high* LED menyala dan ketika pin bernilai *low* LED mati.
- I2C: 20 (SDA) dan 21 (SCL). Dukungan I2C (TWI) komunikasi menggunakan *wire*.

2.3.4 Komunikasi

Arduino Mega 2560 memiliki sejumlah fasilitas untuk berkomunikasi dengan komputer, arduino lain, atau mikrokontroler lainnya. Arduino Mega 2560 menyediakan 4 UART TTL (5V) untuk komunikasi serial. Sebuah Arduino Mega 2560 sebagai saluran komunikasi serial melalui USB dan sebagai port virtual com untuk perangkat lunak pada komputer. *Firmware* ’8 U2 menggunakan *driver* USB standar COM, dan tidak ada *driver* eksternal yang diperlukan. Namun pada Windows diperlukan sebuah file inf. Perangkat lunak Arduino terdapat monitor serial yang memungkinkan digunakan memonitor data tekstual sederhana yang akan dikirim ke atau dari papan Arduino. LED RX dan TX di papan tulis akan berkedip ketika data sedang dikirim melalui *chip* USB-to-serial dengan koneksi USB ke komputer (tetapi tidak untuk komunikasi serial pada pin 0 dan 1). Sebuah *Software Serial Library* memungkinkan untuk berkomunikasi secara serial pada salah satu pin digital pada *board* Arduino Mega 2560. Arduino Mega 2560 juga mendukung I2C (TWI) dan

komunikasi SPI. Perangkat lunak Arduino termasuk perpustakaan kawat untuk menyederhanakan penggunaan bus I2C.

2.4 *Driver* Motor DC (*Dual H-bridge*)

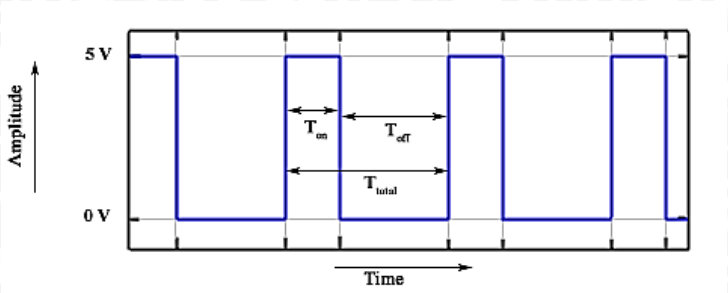
Motor DC tidak dapat dikontrol secara langsung oleh mikrokontroler, karena kebutuhan daya listrik yang cukup besar sedangkan daya keluaran pada mikrokontroler sangat kecil. Untuk dapat melakukan pengontrolan motor DC dibutuhkan suatu *driver* yang mampu memperbesar arus dan tegangan sesuai dengan kebutuhan motor DC. Ada beberapa jenis *driver* motor yang dapat digunakan untuk mengontrol kecepatan motor, yaitu menggunakan rangkaian *H-bridge* transistor, *H-bridge* MOSFET, dan IC *driver* motor. Pada skripsi ini *driver* motor menggunakan IC L298N berbasis *H-bridge,* mampu menangani beban hingga 4 A pada tegangan 6 V – 46 V (lihat Gambar 2.7). Dalam chip terdapat dua rangkaian *H-bridge*. Selain itu *driver* ini mampu mengontrol 2 motor sekaligus dengan arus beban masing-masing 2 A.

Gambar 2.7 IC *driver* Motor L298N
Sumber : store.nerokas.co.ke

2.5 *Pulse Width Modulation* (PWM)

Pengaturan tegangan sumber biasanya menggunakan metode *Pulse Width Modulation* (PWM). Sinyal *Pulse Width Modulation* (PWM) adalah metode yang dapat digunakan untuk mengontrol kecepatan motor DC. Dimana kecepatan motor DC tergantung pada besarnya *duty cycle* yang diberikan pada motor DC tersebut.

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *duty cycle* bervariasi dari 0%-100%. *Duty cycle* adalah besarnya sinyal kontrol yang diberikan pada motor. Persamaan untuk perhitungan *duty cycle* ditunjukkan pada Persamaan 2.2 dengan T_{on} adalah periode logika tinggi, dan T adalah periode keseluruhan. Sinyal PWM secara umum dapat dilihat dalam Gambar 2.8.



Gambar 2.8 Sinyal PWM secara Umum
Sumber : *www.8051projects.net*

$$Duty\ Cycle = \frac{T_{on}}{T_{total}} x100\% \qquad (2.2)$$

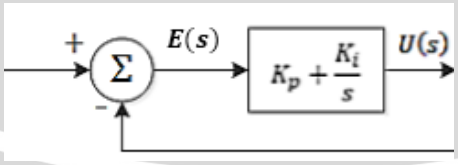
Sedangkan frekuensinya dapat ditentukan dengan Persamaan 2.3:

$$fOCn = \frac{f_{clk}I/O}{N.256} \qquad (2.3)$$

Timer atau counter yang digunakan pada PWM ini yaitu timer atau counter 0 (8 bit) dengan metode fast PWM dan prescaler factor (N) yaitu 256.

2.6 Kontroler *Proporsional Integral* (PI)

Kontroler *Proportional Integral* (PI) (Gambar 2.9) memiliki sifat yang tidak mengeluarkan respon sebelum selang waktu tertentu, pada kontrol P suatu *plant* yang fungsi alihnya tidak mempunyai integrator 1/s, terdapat *error steady state* (*offset*) pada masukan tangga atau *unit step*. *Offset* semacam ini dapat dihilangkan jika ditambahkan aksi integral pada kontroler. kontroler I mengakibatkan respon menjadi lambat walaupun dapat menghilangkan *error steady state* (*offset*). Dalam memperbaiki respon yang lambat tersebut, umumnya kontroler I dipasang paralel dengan kontroler P.



Gambar 2.9 Diaram blok kontroler PI

Bentuk fungsi alih dari kontroler PI seperti pada Persamaan 2.4.

$$U(s) = \left[K_p + \frac{K_i}{s}\right] E(s) \qquad (2.4)$$

Dengan $U(s)$: Sinyal kontrol.

$E(s)$: Nilai *error* sistem.

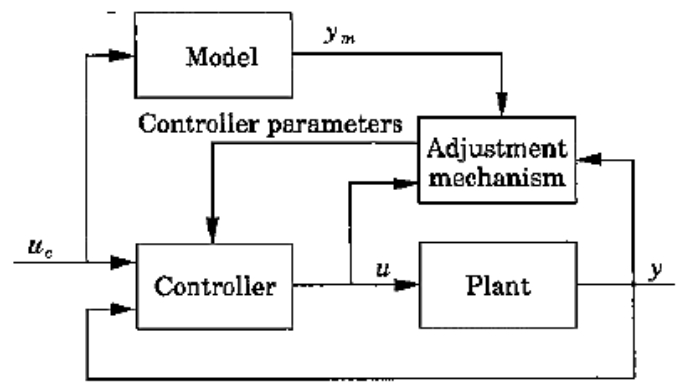
K_p : *Gain* kontroler proporsional.

K_i : *Gain* kontroler integral.

Karena kontroler PI merupakan penggabungan dari dua unit kontroler P dan I, semua kelebuhan serta kekurangan yang ada pada kontroler P dan kontroler I masih tetap ada. Sifat kontroler P yang selalu meninggalkan *error steady state* (*offset*) dapat ditutupi oleh kelebihan kontroler I, sedangkan sifat kontroler I yang lambat dapat ditutupi oleh kontroler P, sehingga kontroler PI menghasilkan respon yang lebih cepat dari kontroler I tapi mampu menghilangkan *error steady state* (*offset*) yang ditinggalkan kontroler P.

2.7 *Model Reference Adaptive Control* (MRAC)

Dasar pemikiran munculnya *Model Reference Adaptive Control* (MRAC) adalah respon sistem yang sesuai dengan keinginan desain dibuat dalam bentuk model $M_{(s)}$. Respon dari model ini dibandingkan dengan respon sistem, selisih respon model dengan respon sistem digunakan untuk mengatur strategi kontrol. Skema dari *Model Reference Adaptive Control* (MRAC) dapat dilihat pada Gambar 2.10.



Gambar 2.10 Skema *Model Reference Adaptive Control* (MRAC)
Sumber : Astrom, 1995

Model referensi digunakan untuk menentukan spesifikasi-spesifikasi sistem yang diinginkan. Kontroler digunakan untuk menghasilkan sinyal kontrol u yang selanjutnya digunakan sebagai *input plant*. *Plant* menghasilkan respon y dan model referensi menghasilkan respon y_m. Respon *plant* dibandingkan dengan respon model referensi, jika terdapat *error* diantara keduanya maka parameter yang ada pada kontroler akan berubah.perubahan parameter kontroler dilakukan melalui hukum adaptasi. Di dalam hukum adaptasi, parameter kontroler ini akan diatur sedemikian rupa sehinga parameter kontroler dapat membentuk sinyal kontrol u yang akan menyebabkan respon *plant* sesuai dengan

model referensi. Apabila *plant* telah mengikuti referensi, maka nilai parameter kontroler sudah tetap seperti yang diharapkan.

Seperti halnya sistem adaptif yang lain, *Model Reference Adaptive Control* mempunyai dua lup yaitu lup dalam dan lup luar. Lup dalam terdiri atas *plant* dan kontroler. Lup luar bertujuan untuk menyesuaikan nilai parameter yang digunakan pada lup dalam. Setelah nilai parameter kontroler sudah tepat, maka lup dalam berjalan seperti sistem kontrol pada umumnya.

Model Reference Adaptive Control bersifat adaptif karena nilai-nilai parameter kontroler dapat ditentukan secara *on-line*. Nilai-nilai parameter ini akan diperbaiki berdasarkan *error* yang terjadi antara respon model dengan respon sistem yang dikontrol.

2.8 Model Referensi

Model referensi ditentukan melalui derajat relatif sistem. Derajat relatif model referensi sistem harus sama dengan derajat relatif *plant* (Butler, 1992). Derajat relatif adalah selisih antara derajat polinominal pole dengan derajat polinominal zero. Orde model referensi disesuaikan dengan orde *plant*.

2.9 Aturan MIT

Aturan MIT adalah pendekatan yang sebenarnya pada MRAC. Nama MIT dipakai karena metode tersebut dikembangkan di Laboratoriun Instrumentasi (sekarang Laboratorium Draper) di MIT.

Untuk mengetengahkan aturan MIT, dimisalkan sebuah sistem lup tertutup di mana kontrolernya mempunyai sebuah parameter θ yang dapat diatur. Respon lup tertutup yang diinginkan ditentukan dengan suatu model yang memiliki respon y_m. Misalnya e sebagai *error* antara respon sistem lup tertutup y dan respon model y_m. Satu kemungkinan untuk mengatur parameter dengan meminimalisasi *loss function* (fungsi kerugian).

$$J(\theta) = \frac{1}{2}e^2 \tag{2.5}$$

Untuk mendapatkan nilai J kecil, merupakan hal yang beralasan untuk mengubah parameter dalam arah negatif gradien dari J, yaitu

$$\frac{d\theta}{dt} = -\gamma \frac{\partial J}{\partial \theta} = -\gamma e \frac{\partial e}{\partial \theta} \quad (2.6)$$

Turunan parsial $\frac{d\theta}{dt}$ yang disebut dengan turunan sensitivitas sistem, menunjukkan bagaimana *error* dipengaruhi oleh parameter yang dapat diatur. Jika diasumsikan bahwa parameter berubah lebih lambat daripada variabel yang lain di dalam sistem, sehingga turunan $\frac{d\theta}{dt}$ dapat dievaluasi dibawah asumsi bahwa θ adalah konstan (Astrom,1995).

Dengan menggunakan aturan MIT, kontroler pada Gambar 2.9 dapat dirumuskan sebagai berikut:

- Respon kontroler sebagai fungsi parameter kontroler yang berubah sesuai dengan perubahan waktu dikalikan dengan *error* atau selisih $y_m(t)$ dan $y(t)$:

  $$u(t) = \theta[y_m(t) - y(t)] \quad (2.7)$$

  Dengan $u(t)$: Respon kontroler.
  θ : Parameter kontroler.
  $y_m(t)$: Respon model referensi.
  $y(t)$: Respon *plant*.

- Perubahan parameter kontroler terhadap waktu dinyatakan dalam model pendekatan gradien, yaitu pada Persamaan 2.8 dan 2.9.

  $$\frac{d\theta}{dt} = -\gamma \frac{\partial}{\partial \theta}[y(t) - y_m(t)]^2 \quad (2.8)$$

  atau

  $$\frac{d\theta}{dt} = -2\gamma[y(t) - y_m(t)] \frac{\partial}{\partial \theta}[y(t) - y_m(t)]^2 \quad (2.9)$$

Dengan γ adalah *gain* adaptasi

2.10 Transformasi Z

Dalam sistem waktu kontinyu *time-invariant*, transformasi laplace dapat dimanfaatkan dalam analisis sistem dan desain. Sedangkan transformasi Z dapat dimanfaatkan dalam menganalisa sistem yang termodelkan dalam waktu diskrit dari persamaan diferensial. Sebuah transformasi ditetapkan sebagai deret angka yang berurutan. Fungsi $E(z)$ ditetapkan sebagai sebuah *power series* dalam z^{-k} dengan koefisien yang sama

untuk nilai dari deret angka $\{e(k)\}$. Transformasi ini disebut sebagai transformasi Z, yang kemudian diungkapkan dalam bentuk Persamaan 2.10.

$$E(z) = \mathcal{Z}[\{e(k)\}] = e(0) + e(1)z^{-1} + e(2)z^{-2} + \cdots \quad (2.10)$$

Dengan,

$$e(k) = \mathcal{Z}^{-1}[E(z)] = \frac{1}{2\pi j}\oint_{\Gamma} E(z)z^{k-1}dz, \qquad j = \sqrt{-1}$$

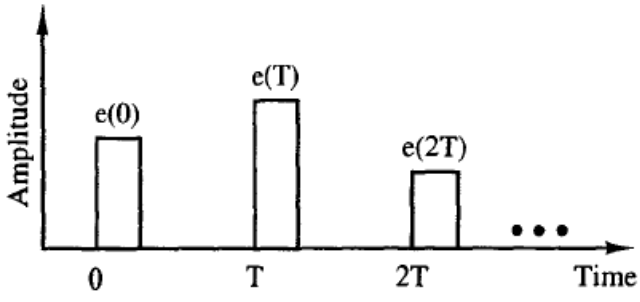
Dimana $\mathcal{Z}(\cdots)$ mengindikasikan operasi transformasi Z dan $\mathcal{Z}^{-1}(\cdots)$ mengindikasikan invers dari transormasi z. $E(z)$ dalam Persamaan 2.10 dapat dituliskan dalam notasi yang disederhanakan seperti pada Persamaan 2.11.

$$E(z) = \mathcal{Z}[\{e(k)\}] = \sum_{k=0}^{\infty} e(k)\, z^{-k} \quad (2.11)$$

Transformasi Z terdefinisikan untuk beberapa deret angka $\{e(k)\}$, dan mungkin digunakan dalam menganalisis dari beberapa jenis sistem yang diuraikan oleh persamaan diferensial *time-invariant* linier.

2.11 Sampling Data

Terdapat sebuah perangkat rekonstruksi data yang dinamakan sebuah penahan data (*data hold*) yang termasuk kedalam sistem secara langsung mengikuti pencuplik (*sampler*). Tujuan dari penahanan data adalah merekonstruksi sinyal yang tercuplik ke dalam sebuah bentuk yang menyerupai sinyal sebelum dicuplik. Perangkat data rekonstruksi sederhana dan sejauh ini yang paling umum adalah *Zero Order Hold* (ZOH). Operasi dari sebuah pencuplik/kombinasi ZOH tergambarkan pada sinyal yang ditunjukkan pada Gambar 2.11. ZOH mencuplik sinyal *output* ke nilai yang sama dengan sinyal *input* pada saat proses pencuplikan (lihat Gambar 2.12).



Gambar 2.11 Sinyal tercuplik dalam bentuk pulsa
Sumber : Phillips, 1995

Pencuplik dan ZOH dapat direpresentasikan dalam diagram blok yang ditunjukkan pada Gambar 2.12. Sinyal $\bar{e}(t)$ dapat dinyatakan sebagai

$$\bar{e}(t) = e(0)[u(t) - u(t-T)] + e(T)[u(t-T) - u(t-2T)] + e(2T)[u(t-2T) - u(t-3T)] + \cdots \quad (2.12)$$

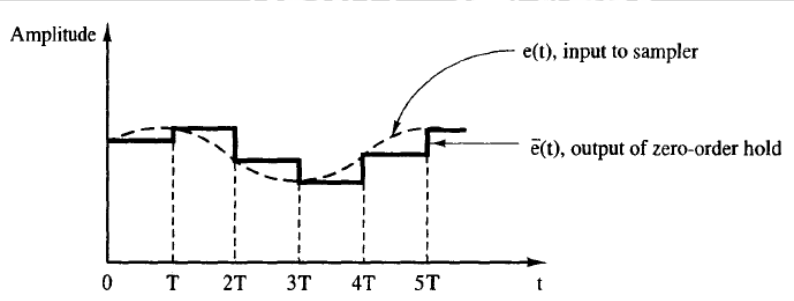
Dimana $u(t)$ adalah fungsi *unit step*. Transformasi laplace dari $\bar{e}(t)$ adalah $\bar{E}(s)$, yang diberikan oleh

$$\bar{E}(s) = e(0)\left[\frac{1}{s} - \frac{e^{-Ts}}{s}\right] + e(T)\left[\frac{e^{-Ts}}{s} - \frac{e^{-2Ts}}{s}\right] + e(2T)\left[\frac{e^{-2Ts}}{s} - \frac{e^{-3Ts}}{s}\right] + \cdots$$

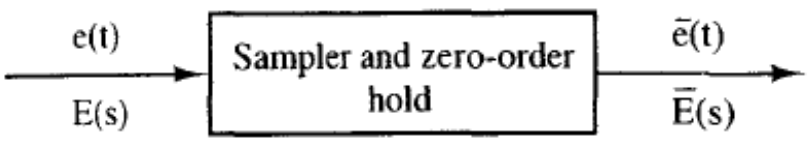
$$= \left[\frac{1 - e^{-Ts}}{s}\right][e(0) + e(T)e^{-Ts} + e(2T)e^{-2Ts} + \cdots$$

$$= \left[\sum_{n=0}^{\infty} e(nT)e^{-nTs}\right]\left[\frac{1 - e^{-Ts}}{s}\right] \quad (2.13)$$

Faktor pertama dalam ekspresi terakhir pada Persamaan 2.13 terlihat sebuah sinyal *input* $e(t)$ dan periode sampling T. Faktor kedua terlihat e(t) yang *independent* dan oleh karena itu dapat dianggap menjadi sebuah fungsi alih.



Gambar 2.12 Sinyal *input* dan *output* dari pencuplik/*data hold*
Sumber : Phillips, 1995

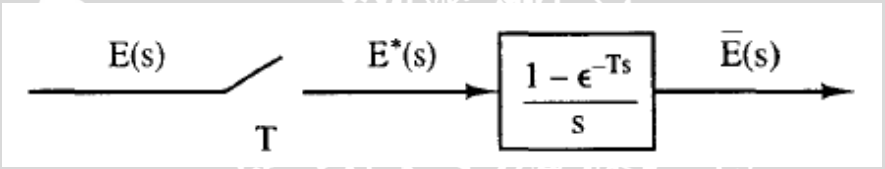


Gambar 2.13 Pencuplik dan *data hold*
Sumber : Phillips, 1995

Sehingga pencuplik dan *data hold* dapat direpresentasikan seperti pada Gambar 2.13. fungsi $E^*(s)$ disebut dengan transformasi terbintang yang didefinisikan sebagai

$$E^*(s) = \sum_{n=0}^{\infty} e(nT)e^{-nTs} \qquad (2.14)$$

Maka Persamaan 2.13 ini terpenuhi dengan representasi pada Gambar 2.13. Operasi yang dilambangkan pada Gambar 2.13 telah terdefinisikan pada Persamaan 2.14 dan disebut dengan pencuplik yang ideal. Operasi yang digambarkan oleh fungsi alih disebut sebagai *data hold*. Hal ini harus ditekankan bahwa $E^*(s)$ tidak tampak dalam bentuk fisik sistem tetapi muncul pada sebuah hasil dari mengfaktorkan Persamaan 2.13. Pencuplik (*switch*) dalam Gambar 2.14 bukanlah sebuah model fisik pencuplik dan blok juga bukan sebuah model fisik dari *data hold*. Bagaimanapun juga, kombinasi tersebut bukan merpakan keakuratan model karakteristik *input-output* dari pencuplik perangkat *data hold* seperti yang ditunjukkan sebelumnya.



Gambar 2.14 Representasi dari *pencuplik* dan *data hold*
Sumber: Phillips, 1995

2.12 Diskritisasi

Banyak cara yang dapat digunakan untuk proses diskritisasi (mengubah bentuk analog menjadi diskrit), tiga diantaranya yang banyak digunakan dalam bidang kontrol adalah *backward difference*, *forward difference* (metode Euler) dan *bilinear transformation*. Semua metode yang digunakan hanya merupakan pendekatan (*approximations*), sehingga hasilnya tidak akan persis sama dengan bentuk analog. Hal ini dikarenakan bentuk diskrit selalu ada sebagian informasi yang hilang, yaitu informasi yang terjadi antara satu cuplikan dengan cuplikan lainnya walaupun frekuensi cuplik yang digunakan tinggi dan karena formula matematika yang digunakan dalam proses diskritisasi juga diturunkan berdasarkan pendekatan agar lebih mudah digunakan.

Menurut teori sampling Nyquist dan Shannon mengatakan bahwa jika suatu fungsi waktu $e(t)$ tidak mengandung komponen frekuensi yang lebih tinggi dari f_0 hertz, dalam hal ini nilai dai nilai $e(t)$ dapat ditentukan dengan memberikan titik sampling berjarak $1/2f_0$ detik terpisah.

Berikut tahapan diskritisasi yang dapat digunakan:

1. Tulis algoritma analog dalam bentuk trasformasi laplace.
2. Lakukan diskritisasi menjadi bentuk transformasi Z dengan mengganti operator s dengan menggunakan salah satu dari tiga metode diskritisasi, yaitu:
   - *Backward difference* :

$$s = \frac{1 - z^{-1}}{T_s} \tag{2.15}$$

   - *Forward difference* :

$$s = \frac{1 - z^{-1}}{T_s z^{-1}} \tag{2.16}$$

   - *Bilinear transform* :

$$s = \frac{2(1 - z^{-1})}{T_s(1 + z^{-1})} \tag{2.17}$$

Dimana T_s adalah waktu cuplik (*time sampling*)

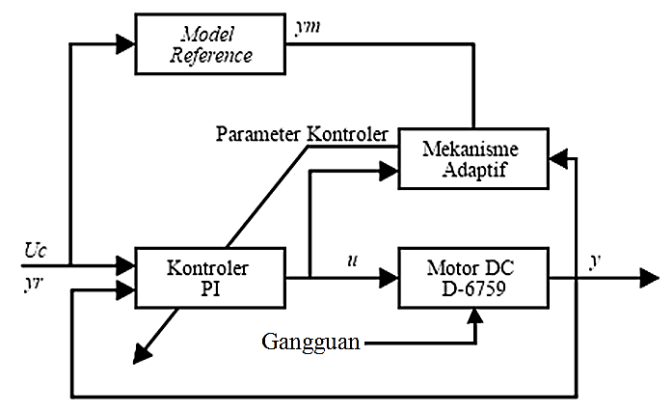
Hingga tahap 2, algoritma sudah didapat dalam bentuk diskrit yang dinyatakan dalam transformasi Z. Pada implementasinya, bentuk transformasi Z tersebut perlu diubah menjadi time domain (persamaan beda), yaitu dengan mengubah operator z^nmenjadi n kali waktu delay (z^{-1} berarti 1 kali waktu delay, z^{-2} berarti 2 kali waktu delay dan seterusnya).

BAB III
METODE PENELITIAN

Metode penelitian pada dasarnya merupakan cara ilmiah untuk mendapatkan informasi dengan tujuan dan manfaat tertentu. Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan skripsi yang terdapat di bab pendahuluan maka diperlukan langkah-langkah untuk menyelesaikan masalah tersebut. Metode yang digunakan diuraikan sebagai berikut.

3.1 Perancangan Diagram Blok Sistem

Pada perencanaan alat diperlukan perancangan diagram blok sistem yang dapat menjelaskan sistem secara garis besar dan diharapkan alat dapat bekerja sesuai dengan rencana dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Blok Sistem dengan *Model Reference Adaptive Control* (MRAC)

Keterangan:

Uc=yr	: *Setpoint* (nilai masukan sistem yang diinginkan pada nilai keluaran sistem).
ym	: *Respon* model referensi.
y	: *Respon* motor DC D-6759.
u	: *Respon* kontroler PI.
Model reference	: Persamaan matematis yang akan menjadi model referensi
Mekanisme Adaptif	: Algoritma kontrol adaptif yang akan menghasilkan *gain* kontroler yang baru untuk kontroler PI.
Kontroler PI	: Kontroler yang akan menghasilkan sinyal kontrol untuk Motor DC D-6759.

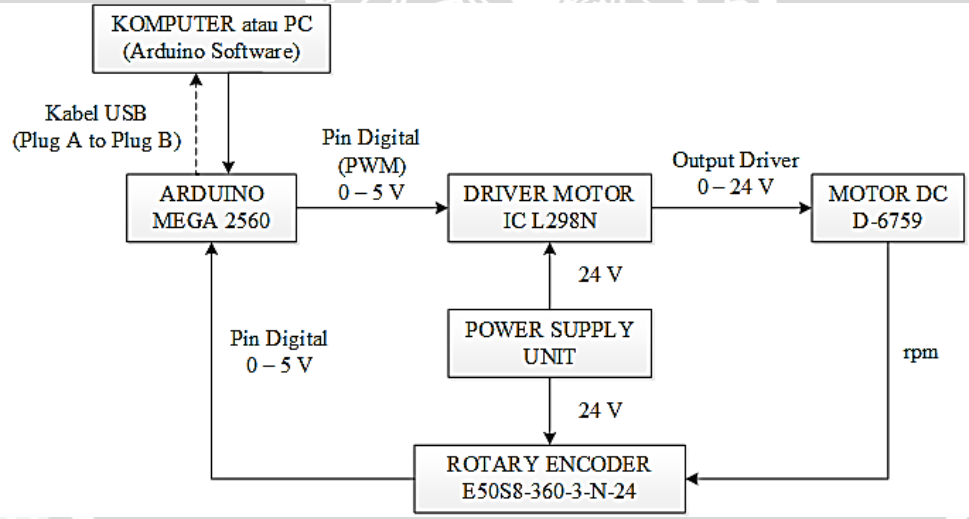
Motor DC D-6759 : *Plant* (objek fisik yang akan dikontrol)
Gangguan : Gangguan yang diberikan ke sistem

3.2 Pembuatan Perangkat Keras

Pembuatan perangkat keras dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem beserta pemrogramannya, hal ini dimaksudkan agar sistem kontrol kecepatan motor DC D-6759 dengan penggunaan teknik *Model Reference Adaptive Control* (MRAC) pada *tuning* kontroler PI dapat berjalan sesuai deskripsi awal yang telah direncanakan.

Pembuatan perangkat keras yang dilakukan meliputi:

1. Skema pembuatan perangkat keras (Gambar 3.2).
2. Penentuan modul elektronik yang digunakan meliputi:
   - Power Supply Unit (PSU).
   - Mikrokontroler Arduino Mega 2560.
   - *Driver* motor DC D-6759 (Dual H- Bridge).
   - *Rotary encoder* Autonics E40S8-1024-3-N-24.



Gambar 3.2 Skema pembuatan perangkat keras

3.3 Perancangan Algoritma

Perancangan algoritma pada skripsi ini meliputi:

- Perancangan algoritma teknik MRAC pada *tuning* pengendali PI dalam persamaan fungsi s.
- Konversi persamaan fungsi s ke dalam bentuk diskrit melalui transformasi z yang kemudian dikonversi ke dalam bentuk persamaan beda.
- Pembuatan *flow chart* untuk pembuatan program di perangkat lunak.

3.4 Pengujian Sistem

Setelah semua komponen pada alat sudah terhubung sesuai dengan skema pembuatan perangkat keras dan perangkat lunak untuk mendukung sistem yang telah dibuat, maka diadakan pengujian sistem. Metode pengujian sitem adalah sebagai berikut:

- Menguji sistem pada tiap-tiap blok.
- Menggabungkan sistem dari beberapa blok menjadi keseluruhan sistem.
- Mengevaluasi hasil pengujian keseluruhan sistem.

3.5 Pengambilan Kesimpulan dan Saran

Kesimpulan dapat berdasarkan dari hasil perealisasian dan pengujian alat sesuai dengan tujuan dan rumusan masalah. Saran diberikan setelah melihat adanya kekurangan dalam sistem yang telah dibuat dengan harapan agar alat ini dapat dikembangkan dengan baik.

BAB IV
PEMBUATAN ALAT DAN PERANCANGAN ALGORITMA

Pembuatan alat dan perancangan algoritma dalam skripsi ini bertujuan untuk memberikan gambaran pemuatan alat secara keseluruhan dan gambaran algoritma proses pengontrolan. Pembuatan alat bertujuan untuk menghasilkan semua perangkat keras maupun perangkat lunak yang sudah ditentukan. Perancangan algoritma meliputi algoritma kontrol adaptif, kontrol digital dan *flowchart* program.

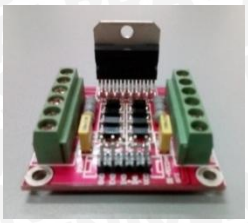
4.1. Spesifikasi Alat

Spesifikasi alat yang meliputi komponen-komponen pendukung pada skripsi ini adalah sebagai berikut:

1. Komputer atau PC yang sudah terinstall *software* arduino.
2. *Power Supply Unit* (PSU) yang digunakan memiliki range 0-24 Volt 2560 (lihat Gambar 4.1)
3. Perangkat kontroler yang digunakan adalah Arduino Mega 2560 (lihat Gambar 4.2).
4. *Driver* motor yang digunakan adalah IC L298N (*Dual H-Bridge*) (lihat Gambar 4.3).
5. Motor DC D-6759 dengan catu daya maksinmal 24 Volt (lihat Gambar 4.4).
6. *Rotary encoder* yang digunakan adalah Autonics E50S8-360-3-N-24 (lihat Gambar 4.5).

Gambar 4.1 *Power Supply Unit* (PSU)

Gambar 4.2 Mikrokontroler Arduino Mega 2560



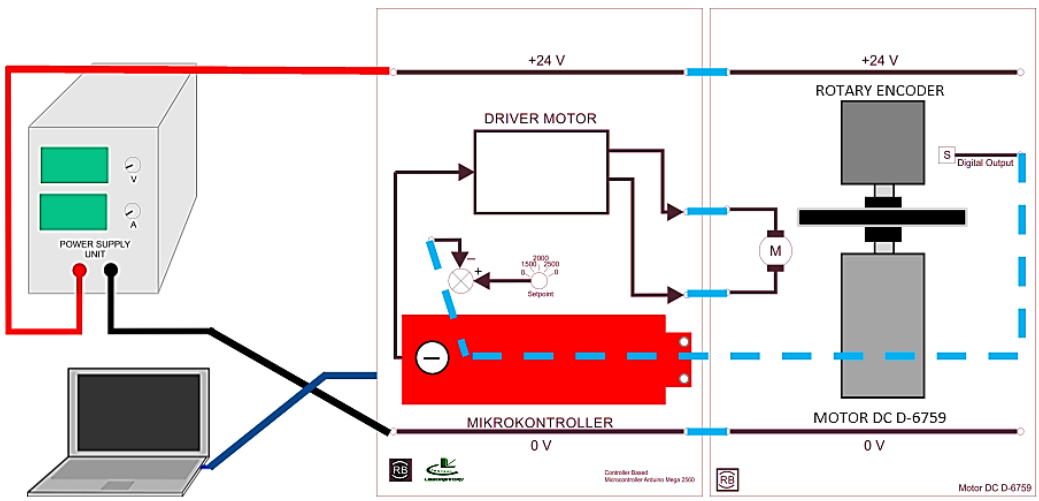
Gambar 4.3 *Driver* motor IC L298N

Gambar 4.4 Motor DC D-6759

Gambar 4.5 *Rotary encoder* Autonics E50S8-360-3-N-24

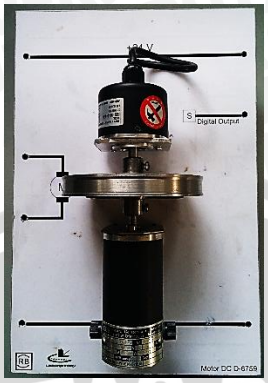
4.2. Pembuatan Perangkat Keras

Skema pembuatan alat dan pembuatan modul untuk pengontrolan kecepatan motor DC D-6759 dengan kontrol adaptif ditunjukkan dalam Gambar 4.6.



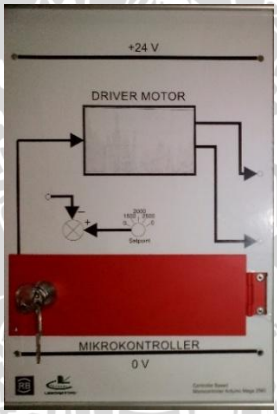
Gambar 4.6 Skema pembuatan perangkat keras

Pembuatan modul motor DC D-6759 yang telah dikopel dengan *rotary encoder* Autonics E50S8-360-3-N-24 dapat dilihat pada Gambar 4.7.



Gambar 4.7 Modul motor DC D-6759 dan *rotary encoder* Autonics E50S8-360-3-N-24

Pembuatan modul mikrokontroler Arduino Mega 2560 yang terhubung dengan *driver* motor IC L298N dapat dilihat pada Gambar 4.8.



Gambar 4.8 Modul mikrokontroler Arduino Mega 2560 dan *driver* motor IC L298N

4.3. Prinsip Kerja Sistem

Prinsip kerja sistem adalah sebagai berikut:

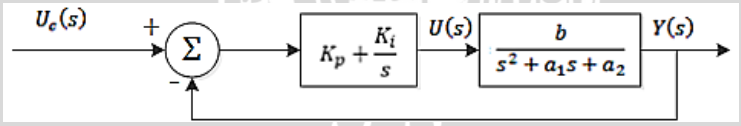
1. *Power Supply Unit* (PSU) diberi catu 110 VAC.
2. *Power Supply Unit* (PSU) mengkonversi tegangan 110 VAC menjadi 24 VDC sebagai *supply* rangkaian *driver* motor dan *rotary encoder*.
3. Catu daya Arduino Mega 2560 berupa tegangan 5 VDC yang didapat dari kabel USB (*plug* A *to plug* B) yang dihubungkan pada komputer.
4. Keluaran *rotary encoder* berupa pulsa dengan tegangan 0 V sampai 5 V sebagai masukan digital pada pin *external interrupt* Arduino Mega 2560 yang kemudian di konversi menjadi nilai pembacaan kecepatan motor.

5. Sinyal kontrol dari Arduino Mega 2560 masuk ke *driver* IC L298N (*Dual H-Bridge*). *Driver* berfungsi menguatkan sinyal yang dihasilkan Arduino dari 0-5 V menjadi 0-24 V.
6. Motor DC D-6759 yang beputar telah dikopel dengan *rotary encoder* Autonics E50S8-360-3-N-24 sebagai pembaca putaran motor DC D-6759.
7. Mencari respon kecepatan motor DC D-6759 terhadap perubahan sinyal *Pulse Width Modulation (*PWM).
8. Mencari fungsi alih motor DC D-6759 dengan memberikan sinyal *Pseudo Random Binary Sequence* (PRBS).
9. Mensimulasikan algoritma adaptif dengan meggunakan *software* Matlab.
10. Membandingkan respon sistem *simulink* dengan respon sistem motor DC D-6759 yang didapat.
11. Mengimplementasikan hasil desain perancanan pada sistem.

4.4. Desain Kontroler PI menggunakan Teknik MRAC

Fungsi alih dari motor DC (lihat Persamaan 2.1) merupakan orde dua, apabila $b = \frac{K_m}{L_a J}$, $a_1 = \frac{R_a}{L_a} + \frac{C}{J}$ dan $a_2 = \frac{R_a C}{L_a J} + K_b \frac{K_m}{L_a}$, maka akan didapatkan bentuk fungsi alih motor DC pada Persamaan 4.1.

$$\frac{Y(s)}{U(s)} = \frac{b}{s^2 + a_1 s + a_2} \tag{4.1}$$



Gambar 4.9 Diagram blok sistem *loop* tertutup

Fungsi alih dari diagram blok sistem *loop* tertutup pada Gambar 4.9 adalah

$$\frac{Y(s)}{U_c(s)} = \frac{bK_p s + bK_i}{s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i} \tag{4.2}$$

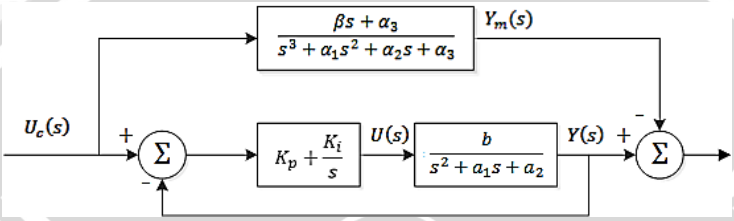
Model referensi ditentukan melalui derajat relatif sistem. Derajat relatif model referensi sistem harus sama dengan derajat relatif *plant*. Derajat relatif adalah selisih antara derajat *polinominal pole* dengan derajat *polinominal zero*. Orde model referensi

disesuaikan dengan orde *plant*, sehingga model referesi dapat diasumsikan dalam bentuk fungsi alih orde 3 pada Persamaan 4.3.

$$\frac{Y_m(s)}{U_c(s)} = \frac{\beta s + \alpha_3}{s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3} \tag{4.3}$$

Dimana: $\beta = bK_p$ $\alpha_1 = a_1$ $\alpha_2 = a_2 + bK_p$ $\alpha_3 = bK_i$

Dalam penggunaan teknik MRAC pada *tuning* kontroler PI pada *plant*, model dan kontroler dapat didibuat dalam bentuk diagram blok pada Gambar 4.10.



Gambar 4.10 Diagram blok sistem dengan model referensi

Perubahan parameter kontroler terhadap waktu dinyatakan dalam model pendekatan gradien dimana *error* didefinisikan sebagai selisih antara respon *plant* y dan respon model referensi $y_m (e = y - y_m)$. Sehingga memungkinkan untuk didapatkan parameter kontroler K_p dan K_i menerapkan aturan MIT $\frac{d\theta}{dt} = -\gamma e \frac{\partial e}{\partial \theta}$ dengan θ didapat dari penurunan berikut:

$$\frac{\partial e}{\partial \theta} = \frac{\partial (y - y_m)}{\partial \theta} \tag{4.4}$$

$$\frac{d\theta}{dt} = s\theta = -\gamma e \frac{\partial e}{\partial \theta} \tag{4.5}$$

Dari Persamaan 4.4 dan 4.5 didapatkan bahwa

$$\theta = \frac{-\gamma}{s} e \frac{\partial e}{\partial \theta} \tag{4.6}$$

Dengan mengganti θ dengan K_p dan K_i akan diperoleh Persamaan 4.7 dan 4.8.

$$\frac{dK_p}{dt} = -\gamma_p \left(\frac{\partial J}{\partial K_p}\right) = -\gamma_p \left(\frac{\partial J}{\partial e}\right)\left(\frac{\partial e}{\partial y}\right)\left(\frac{\partial y}{\partial K_p}\right) \tag{4.7}$$

$$\frac{dK_i}{dt} = -\gamma_i \left(\frac{\partial J}{\partial K_i}\right) = -\gamma_i \left(\frac{\partial J}{\partial e}\right)\left(\frac{\partial e}{\partial y}\right)\left(\frac{\partial y}{\partial K_i}\right) \tag{4.8}$$

Dimana $e = y - y_m \Rightarrow \frac{\partial e}{\partial y} = 1$; $\frac{\partial J}{\partial y} = e$

Dengan menggunakan hubungan Persamaan 4.7 dan 4.8 didapatkan

$$\frac{dK_p}{dt} = -\gamma_p \left(\frac{\partial J}{\partial K_p}\right) = -\gamma_p e \left(\frac{\partial y}{\partial K_p}\right) \tag{4.9}$$

$$\frac{dK_i}{dt} = -\gamma_p \left(\frac{\partial J}{\partial K_i}\right) = -\gamma_p e \left(\frac{\partial y}{\partial K_i}\right) \tag{4.10}$$

Cara agar mendapatkan $\frac{\partial y}{\partial K_p}$ dan $\frac{\partial y}{\partial K_i}$ dengan cara mendiferensialkan Persamaan 4.2, sehingga

$$\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right) = u_c\left(bK_p s + bK_i\right)$$

$$\left(y \frac{\partial}{\partial K_p}\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right)\right)$$

$$+ \left(\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right)\frac{\partial y}{\partial K_p}\right) = \frac{\partial}{\partial K_p} u_c\left(bK_p s + bK_i\right)$$

$$\frac{\partial y}{\partial K_p} = b \frac{s}{s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i}(u_c - y) \tag{4.11}$$

Dengan cara yang sama untuk $\frac{\partial y}{\partial K_i}$

$$y\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right) = u_c\left(bK_p s + bK_i\right)$$

$$\left(y \frac{\partial}{\partial K_i}\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right)\right)$$

$$+ \left(\left(s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i\right)\frac{\partial y}{\partial K_i}\right) = \frac{\partial}{\partial K_i} u_c\left(bK_p s + bK_i\right)$$

$$\frac{\partial y}{\partial K_i} = b \frac{1}{s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i}(u_c - y) \tag{4.12}$$

Dengan mensubstitusi Persamaan 4.11 dan 4.12 ke dalam Persamaan 4.9 dan 4.10 didapatkan Persamaan 4.13 dan 4.14.

$$\frac{dK_p}{dt} = -\gamma_p e \frac{bs}{s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i}(u_c - y) \tag{4.13}$$

$$\frac{dK_i}{dt} = -\gamma_i e \frac{b}{s^3 + a_1 s^2 + (a_2 + bK_p)s + bK_i}(u_c - y) \tag{4.14}$$

Dari Persamaan 4.13 dan 4.14 *gain parameters* tidak dapat ditemukan ketika nilai a_1, a_2 dan b tidak diketahui. Jadi rumus yang diturunkan dari aturan MIT ini belum dapat digunakan. Sebagai gantinya, beberapa pendekatan diperlukan. Jika berfikir sistem merupakan model yang sempurna, bandingkan hubungan antara *input-output* dari sistem dengan model referensi, pendekatan yang akan digunakan seperti pada persamaan 4.15.

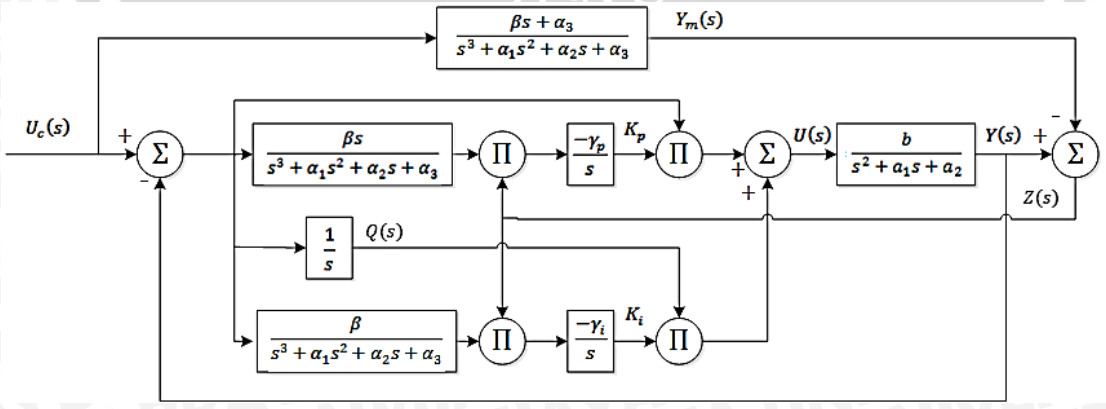
$$\begin{cases} \beta = b \\ \alpha_1 = a_1 \\ \alpha_2 = bK_p + a_2 \\ \alpha_3 = bK_i \end{cases} \tag{4.15}$$

Sehingga persamaan yang baru adalah

$$K_p = \frac{-\gamma_p}{s} e \frac{\beta s}{s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3}(u_c - y) \tag{4.16}$$

$$K_i = \frac{-\gamma_i}{s} e \frac{\beta}{s^3 + \alpha_1 s^2 + \alpha_2 s + \alpha_3}(u_c - y) \tag{4.17}$$

Sebuah representasi diagram dari skema adaptif berdasarkan penurunan persamaan yang telah diperoleh dapat dilihat pada Gambar 4.11.



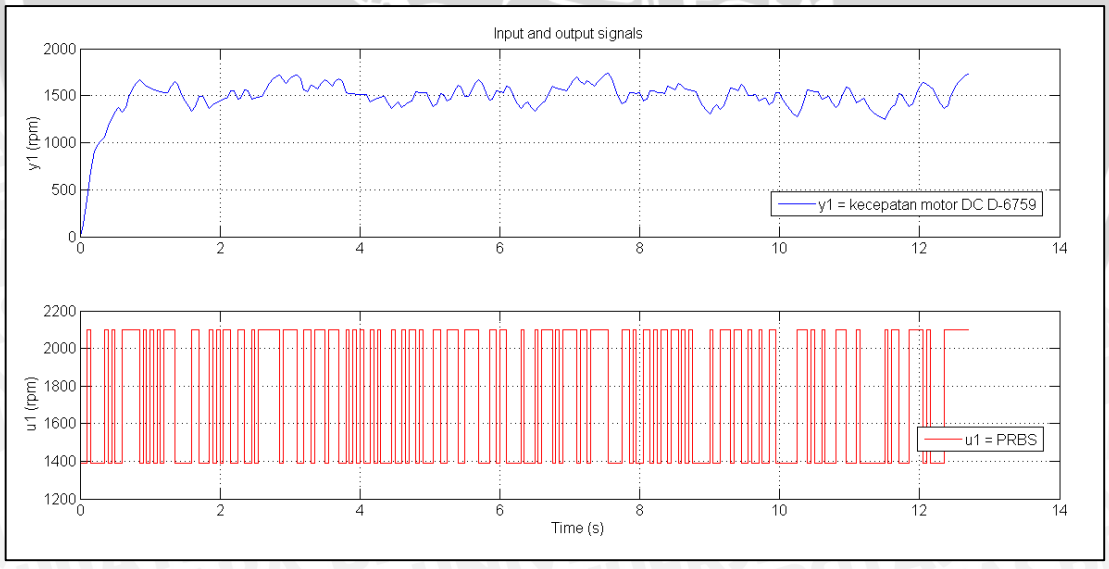
Gambar 4.11 Diagram blok penggunaan teknik MRAC pada *tuning* kontroler PI

4.5. Penentuan Fungsi Alih Motor DC D-6759

Pengontrolan kecepatan motor DC D-6759 menggunakan Arduino Mega 2560 sebagai pengolah dan memberikan data berupa *Pulse Width Modulation* (PWM) agar motor bergerak. Motor DC yang digunakan pada perancangan ini tidak diketahui karakteristiknya, sehingga yang perlu dilakukan adalah melakukan pengujian dengan menggunakan *rotary encoder*. Karakteristik motor DC D-6759 pada perancangan ini didapatkan dengan cara memberikan masukan *unit step*.

Fungsi alih dari motor didapatkan dari pemodelan dengan cara membangkitkan sinyal *Pseudo Random Binary Sequence* (PRBS). Langkah yang dilakukan untuk membangkitkan sinyal PRBS adalah sebagai berikut:

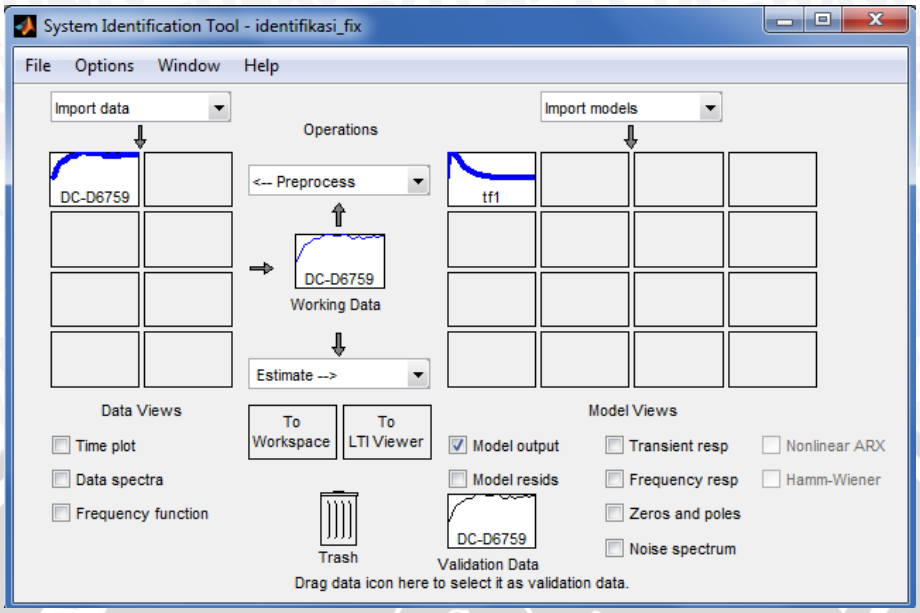
1. Mencari nilai yang linear dari hasil kecepatan motor terhadap *duty cycle* PWM.
2. Memasukkan nilai batas atas dan bawah berdasarkan nilai yang linier untuk membangkitkan sinyal PRBS.
3. Sinyal PRBS yang telah dibangkitkan kemudian digunakan sebagai masukan motor DC.
4. Setelah didapatkan data sinyal PRBS dan data kecepatan motor DC D-6759 (lihat Gambar 4.12), selanjutnya adalah melakukan identifikasi dengan menggunakan *software* Matlab.



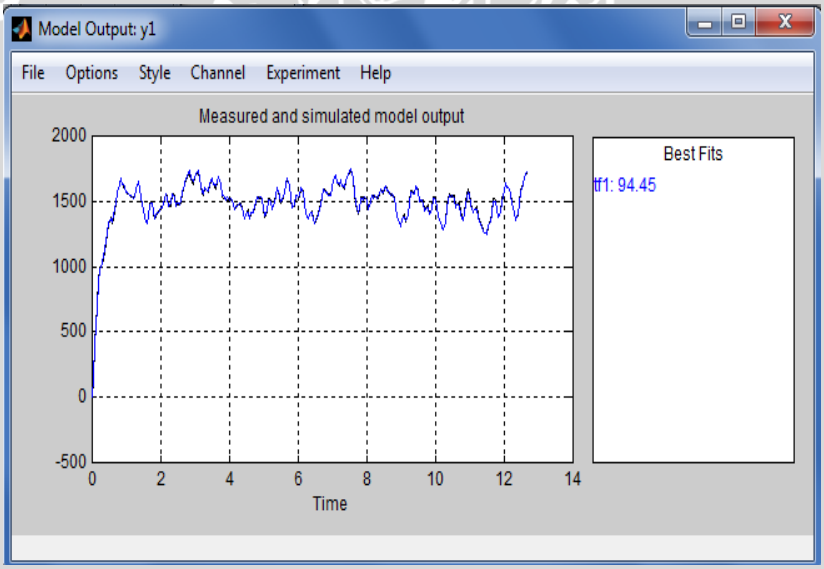
Gambar 4.12 Respon sinyal PRBS dan kecepatan motor DC D-6759

5. Dengan menggunakan sintaks ident pada *command window* pada Matlab, data sinyal PRBS dan data kecepatan motor yang telah disimpan kemudian di *import* pada blok *System Identification Toolbox* (lihat Gambar 4.13). Setelah melakukan beberapa

estimasi model berdasarkan data yang telah di *import* didapatkan fungsi alih dari motor dengan *best fit* sebesar 94.45 (lihat Gambar 4.14).



Gambar 4.13 *System Identification Toolbox*

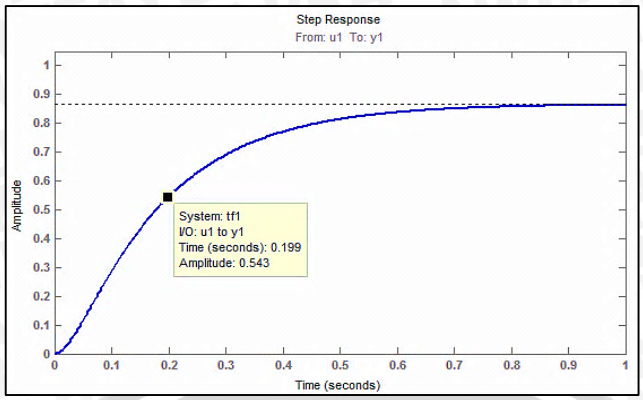


Gambar 4.14 Hasil estimasi model

6. Dari hasil identifikasi, fungsi alih motor yang didapat adalah

$$\frac{Y(s)}{U(s)} = \frac{163.3}{s^2 + 37{,}24s + 188.4} \tag{4.18}$$

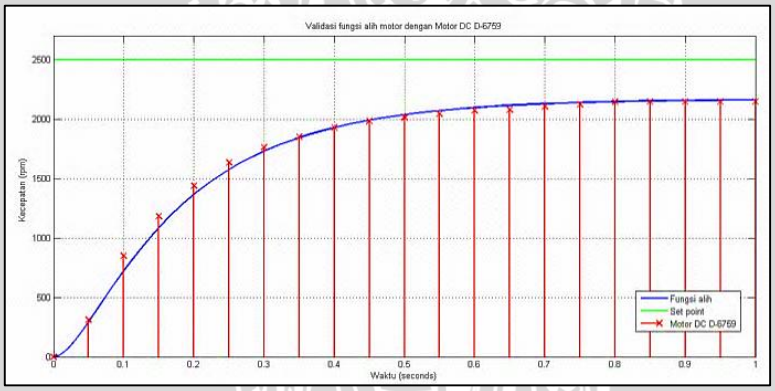
Dengan memberikan masukan *unit step* pada program Matlab didapatkan respon pada Gambar 4.15, nilai *time constant* fungsi alih merupakan waktu yang dibutuhkan respon untuk mencapai 63,2% dari nilai *steady state*, yaitu 0.199 seconds.



Gambar 4.15 Respon fungsi alih motor DC D-6759 dengan masukan *unit step*

4.6. Validasi Fungsi Alih Motor DC D-6759

Dalam melakukan validasi fungsi alih motor dilakukan dengan cara membandingkan respon fungsi alih dan respon kecepatan motor DC D-6759 yang dapat dari pembacaan *rotary encoder* dengan memberikan masukan pulsa *unit step*. Berikut perbandingan kedua respon yang didapat dengan menggunakan Matlab (lihat Gambar 4.16).



Gambar 4.16 Validasi fungsi alih dengan respon motor DC D-6759

Dari grafik pada Gambar 4.16 dapat dilihat bahwa respon keluaran fungsi alih yang telah didapat dari proses identifikasi hampir menyerupai respon kecepatan motor DC D-6759. Jadi fungsi alih yang telah didapatkan dianggap dapat mewakili pemodelan *plant* motor DC D-6759.

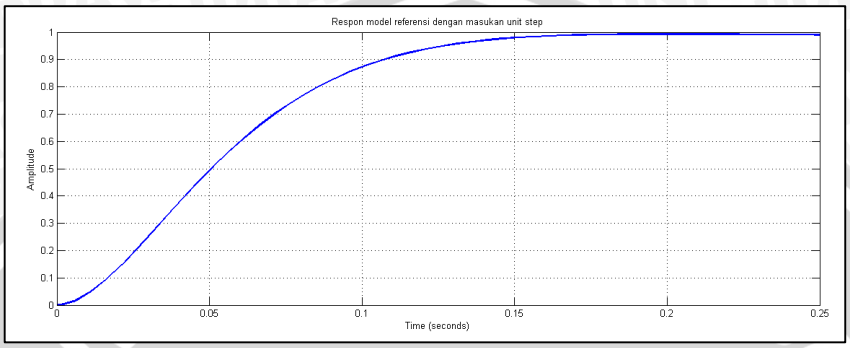
4.7. Penentuan Model Referensi

Penentuan model referensi dilakukan dengan cara *try and error* sehingga didapatkan respon yang stabil yang mengikuti karakteristik dari *plant*. Model referensi yang akan ditentukan tidak memilki *error steady state*, tidak terdapat osilasi dan memilki *settling time* yang cepat.

Fungsi alih dari model referensi yang digunakan adalah:

$$\frac{Y_m(s)}{U_c(s)} = \frac{952{,}3s + 279{,}8}{s^3 + 53{,}79s^2 + 979{,}8s + 279{,}8} \tag{4.19}$$

Respon transien dari model referensi terhadap masukan *unit step* dapat dilihat pada Gambar 4.17.



Gambar 4.17 Respon fungsi alih model referensi dengan masukan *unit step*

4.8. Penetapan Parameter Kontroler

Penetapan parameter kontroler terhadap waktu dinyatakan dalam pendekatan gradien dimana *error* didefinisikan sebagai selisih antara keluaran *plant* y dan keluaran model referensi $y_m (e = y - y_m)$. Sehingga memungkinkan untuk didapatkan parameter kontroler K_p dan K_i dengan menerapkan aturan MIT. Berdasarkan desain kontroler PI dengan teknik MRAC yang telah dijelaskan sebelumnya, maka akan didapatkan bentuk persamaan 4.20 dan 4.21:

$$K_p = \frac{-\gamma_p}{s} e \frac{952{,}3s}{s^3 + 53{,}79s^2 + 979{,}8s + 279{,}8} (u_c - y) \tag{4.20}$$

$$K_i = \frac{-\gamma_i}{s} e \frac{952{,}3}{s^3 + 53{,}79s^2 + 979{,}8s + 279{,}8} (u_c - y) \tag{4.21}$$

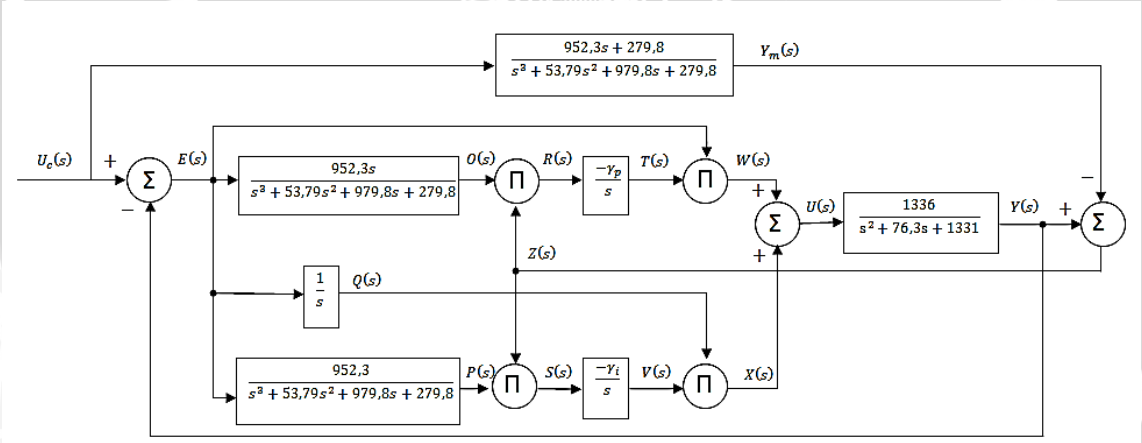
Pada persamaan 4.20 dan 4.21 terdapat dua parameter yang harus ditentukan nilainya terlebih dahulu dalam mendesain sistem yang akan dikontrol, yaitu parameter γ_p dan γ_i. Dimana cara menetukan nilai parameter tersebut dilakukan dengan cara *try and error*.

4.9. Desain Persamaan Beda

Terdapat banyak cara yang dapat digunakan untuk proses diskitisasi, tiga diantaranya yang banyak digunakan dalam bidang control adalah *backward difference*,

forward difference (Metode Euler), dan *bilinear transformation.* Ketiga metode tersebut hanya merupakan pendekatan, sehingga hasilnya tidak akan persis sama dengan bentuk analog karena dalam bentuk diskrit selalu ada sebagian informasi yang hilang, yaitu informasi yang terjadi antara satu cuplikan dengan cuplikan lainnya walaupun frekuensi cuplik yang digunakan tinggi. Kedua, karena formula matematika yang digunakan dalm proses diskritisasi juga diturunkan berdasarkan pendekatan agar lebih mudah digunakan. Pada perancangan ini akan digunakan metode *backward difference* karena seluruh daerah stabil dan sebagian daerah tidak stabil di bidang s di-*mapping* ke daerah stabil z, hal inilah yang menyebabkan metode *backward difference* jauh lebih *robust* dibandingkan dengan kedua metode lainnya. (Asro, 2009)

Bedasarkan hasil dari penentuan fungsi alih motor DC, model referensi, dan parameter kontroler jika disusun menjadi sebuah sistem yang direpresentasikan dalam diagram blok seperti yang telah didapatkan pada desain kontroler PI yang menggunakan teknik MRAC, maka akan didapatkan diagram pada Gambar 4.18.



Gambar 4.18 Diagram blok sistem dengan desain kontroler PI menggunakan teknik MRAC

Model referensi dari sistem dinyatakan dalam bentuk laplace (lihat Persamaan 4.19). Dengan menggunakan metode *backward difference*, ganti operator s dalam persamaan analog bentuk s dengan persamaan 4.22.

$$s = \frac{1 - z^{-1}}{T_s} \tag{4.22}$$

Dimana T_s = waktu cuplik (*time sampling*), maka akan didapatkan penurunan persamaan 4.19.

$$Y_m(z) = \left. \frac{952{,}3\left(\frac{1-z^{-1}}{T_s}\right) + 279{,}8}{\left(\frac{1-z^{-1}}{T_s}\right)^3 + 53{,}79\left(\frac{1-z^{-1}}{T_s}\right)^2 + 979{,}8\left(\frac{1-z^{-1}}{T_s}\right) + 279{,}8} U_c(z) \right| \times \frac{T_s}{T_s}$$

$$Y_m(z) = \frac{952{,}3{T_s}^2(1-z^{-1}) + 279{,}8{T_s}^3}{(1-z^{-1})^3 + 53{,}79T_s(1-z^{-1})^2 + 979{,}8{T_s}^2(1-z^{-1}) + 279{,}8{T_s}^3} U_c(z)$$

$$Y_m(z) = \frac{952{,}3{T_s}^2(1-z^{-1}) + 279{,}8{T_s}^3}{(1-3z^{-1}+3z^{-2}-z^{-3}) + 53{,}79T_s(1-2z^{-1}+z^{-2}) + 979{,}8{T_s}^2(1-z^{-1}) + 279{,}8{T_s}^3} U_c(z)$$

$$Y_m(z) = \frac{952{,}3{T_s}^2 - 952{,}3{T_s}^2 z^{-1} + 279{,}8{T_s}^3}{1-3z^{-1}+3z^{-2}-z^{-3} + 53{,}79T_s - 107{,}58T_s z^{-1} + 53{,}79T_s z^{-2} + 979{,}8{T_s}^2 - 979{,}8{T_s}^2 z^{-1} + 279{,}8{T_s}^3} U_c(z)$$

$$Y_m(z) = \frac{(279{,}8{T_s}^3 + 952{,}3{T_s}^2) - 952{,}3{T_s}^2 z^{-1}}{(279{,}8{T_s}^3 + 979{,}8{T_s}^2 + 53{,}79T_s + 1) - (979{,}8{T_s}^2 + 107{,}58T_s + 3)z^{-1} + (53{,}79T_s + 3)z^{-2} - z^{-3}} U_c(z)$$

Kalikan setiap fraksi/komponen dengan $\left((279{,}8{T_s}^3 + 979{,}8{T_s}^2 + 53{,}79T_s + 1) - (979{,}8{T_s}^2 + 107{,}58T_s + 3)z^{-1} + (53{,}79T_s + 3)z^{-2} - z^{-3}\right)$ sehingga diperoleh persaamaan 4.23.

$$\begin{aligned} Y_m(z)\Big((279{,}8{T_s}^3 + 979{,}8{T_s}^2 + 53{,}79T_s + 1) - (979{,}8{T_s}^2 + 107{,}58T_s + 3)z^{-1} \\ + (53{,}79T_s + 3)z^{-2} - z^{-3}\Big) \\ = \left((279{,}8{T_s}^3 + 952{,}3{T_s}^2) - 952{,}3{T_s}^2 z^{-1}\right) U_c(z) \end{aligned} \quad (4.23)$$

Susun kembali persamaan 4.23 berdasarkan jenis variable (Y_m atau U_c) dan berdasarkan jenis operator (z^{-n}), sehingga menjadi Persamaan 4.24.

$$\begin{aligned} Y_m(z)(279{,}8{T_s}^3 + 979{,}8{T_s}^2 + 53{,}79T_s + 1) \\ = (979{,}8{T_s}^2 + 107{,}58T_s + 3)Y_m(z)z^{-1} - (53{,}79T_s + 3)Y_m(z)z^{-2} \\ + Y_m(z)z^{-3} + (279{,}8{T_s}^3 + 952{,}3{T_s}^2)U_c(z) \\ - 952{,}3{T_s}^2 z^{-1} U_c(z) \end{aligned} \quad (4.24)$$

Jika Persamaan 4.24 disederhanakan lagi akan didapatkan persamaan 4.25.

$$Y_m(z) = \frac{A}{G}Y_m(z)z^{-1} - \frac{B}{G}Y_m(z)z^{-2} + \frac{C}{G}Y_m(z)z^{-3} + \frac{D}{G}U_c(z) - \frac{E}{G}z^{-1}U_c(z) \quad (4.25)$$

Dalam bentuk time domain, dengan mengganti operator z^0 menjadi waktu saat ini (k), z^{-1} menjadi waktu sesaat sebelumnya atau satu kali waktu tunda $(k-1)$, dan seterusnya maka akan didapatakan Persamaan 4.26

$$y_m(k) = \frac{A}{G}y_m(k-1) - \frac{B}{G}y_m(k-2) + \frac{C}{G}y_m(k-3) + \frac{D}{G}u_c(k) - \frac{E}{G}u_c(k-1) \quad (4.26)$$

Dengan menggunakan cara yang sama, maka akan didapatkan Persamaan 4.27 sampai 4.31.

- Model referensi untuk Kp:

$$o(k) = \frac{A}{G}o(k-1) - \frac{B}{G}o(k-2) + \frac{C}{G}o(k-3) + \frac{E}{G}e(k) - \frac{E}{G}e(k-1) \quad (4.27)$$

- Model referensi untuk Ki:

$$p(k) = \frac{A}{G}p(k-1) - \frac{B}{G}p(k-2) + \frac{C}{G}p(k-3) + \frac{F}{G}e(k) \quad (4.28)$$

Dimana:

$A = 979{,}8{T_s}^2 + 107{,}58T_s + 3$

$B = 53{,}79T_s + 3$

$C = 1$

$D = 279{,}8{T_s}^3 + 952{,}3{T_s}^2$

$E = 952{,}3{T_s}^2$

$F = 952{,}3{T_s}^3$

$G = 279{,}8{T_s}^3 + 979{,}8{T_s}^2 + 53{,}79T_s$

- Integrator:

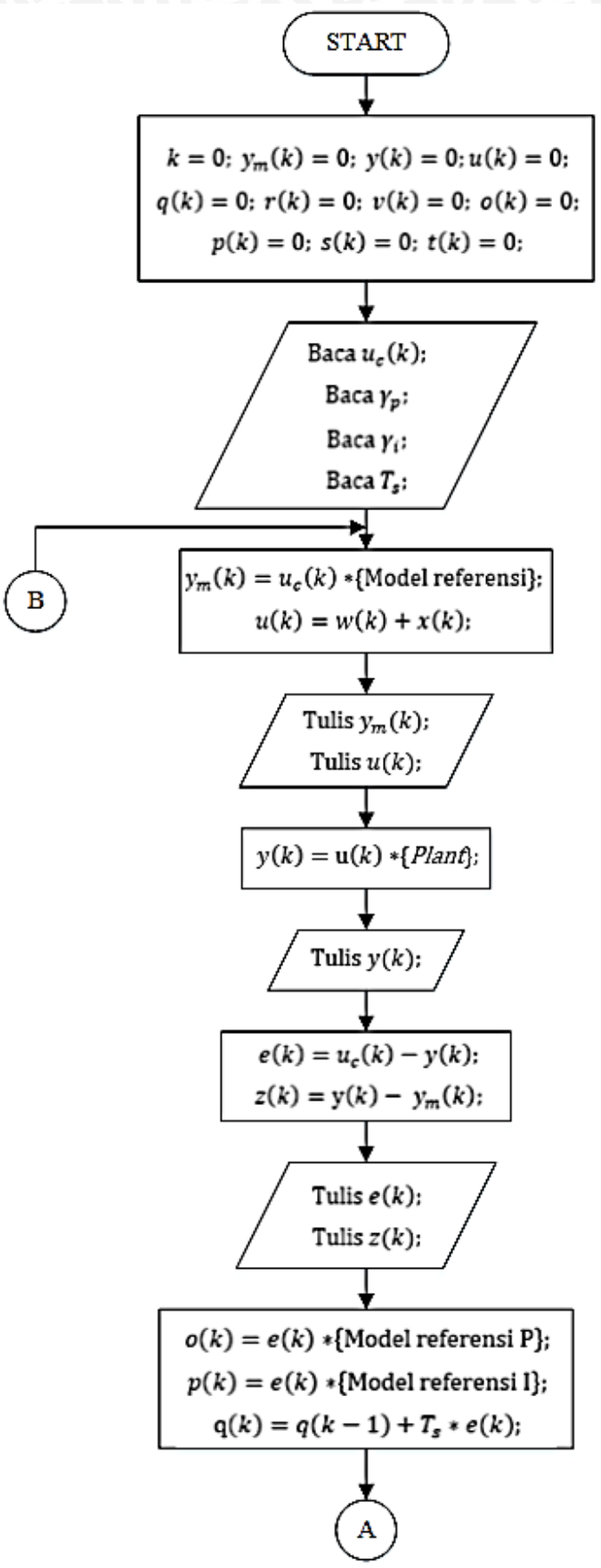
$$q(k) = q(k-1) + T_s e(k) \quad (4.29)$$

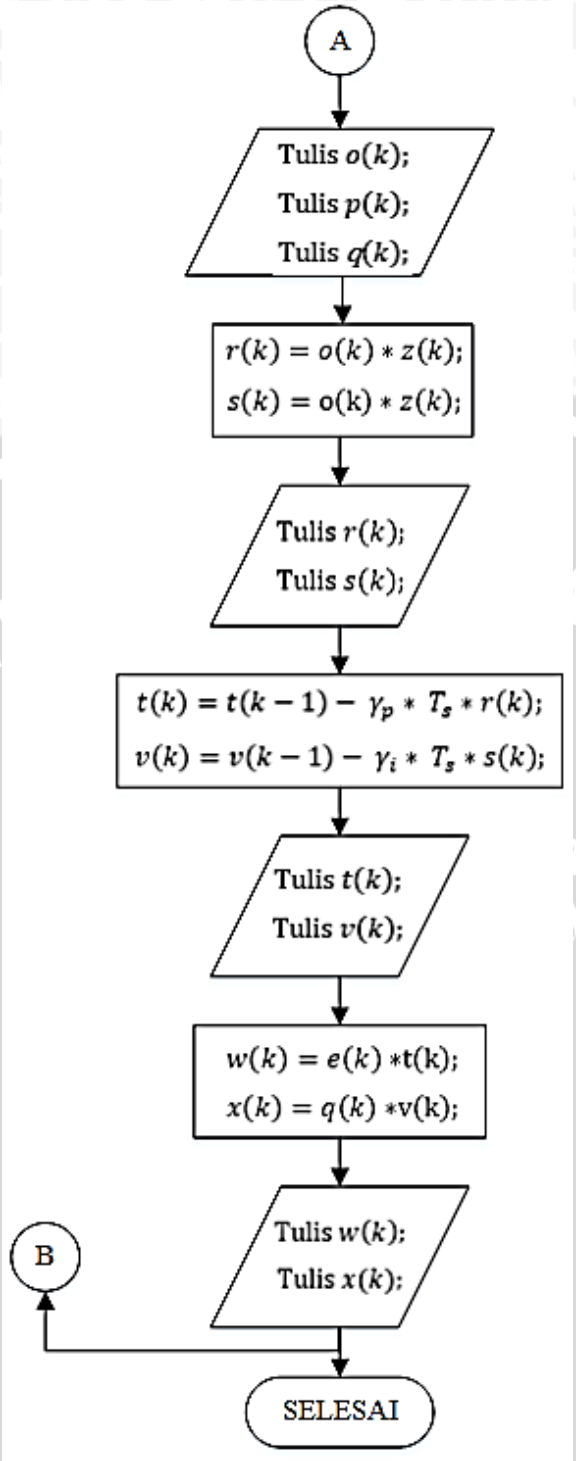
- Parameter kontroler:

$$t(k) = t(k-1) - \gamma_p T_s r(k); \quad r(k) = o(k) * z(k); \quad z(k) = y_m(k) - y(k) \quad (4.30)$$

$$v(k) = v(k-1) - \gamma_i T_s s(k); \quad s(k) = p(k) * z(k) \quad (4.31)$$

4.10. *Flowchart* Program




A
Tulis $o(k)$;
Tulis $p(k)$;
Tulis $q(k)$;
$r(k) = o(k) * z(k)$;
$s(k) = o(k) * z(k)$;
Tulis $r(k)$;
Tulis $s(k)$;
$t(k) = t(k-1) - \gamma_p * T_s * r(k)$;
$v(k) = v(k-1) - \gamma_i * T_s * s(k)$;
Tulis $t(k)$;
Tulis $v(k)$;
$w(k) = e(k) * t(k)$;
$x(k) = q(k) * v(k)$;
Tulis $w(k)$;
Tulis $x(k)$;
B
SELESAI

BAB V
PENGUJIAN, SIMULASI DAN ANALISIS SISTEM

Pengujian dan analisis sistem dilakukan untuk mengetahui apakah alat yang telah dibuat berfungsi dengan baik dan sesuai dengan skema pembuatan. Pengujian pada sistem ini meliputi pengujian setiap blok maupun pengujian secara keseluruhan. Pengujian setiap blok dilakukan untuk mendapatkan letak kesalahan dan mempermudah analisis pada sistem.

Pengujian dibagi menjadi beberapa bagian, yaitu:

1. Pengujian motor DC D-6759.
2. Pengujian *driver* motor IC L298N (*Dual H-Bridge*).
3. Pengujian *rotary encoder* E50S8-360-3-N-24.

Analisis Sistem dibagi menjadi 2 bagian, yaitu:

1. Analisis sistem berdasarkan simulasi.
2. Analisis sistem berdasarkan implementasi.

5.1 Pengujian Sistem

5.1.1 Motor DC D-6759

a. Tujuan

Mengetahui karakteristik motor DC D-6759 pada setiap kenaikan *input* tegangan

b. Peralatan yang digunakan

1. *Power Supply Unit* (PSU).
2. Motor DC D-6759.
3. Tachometer digital.
4. Perangkat komputer.
5. Kabel penghubung

c. Langkah pengujian

1. Hubungkan *output* tegangan *Power Supply Unit* (PSU) dengan motor DC D-6759.
2. Atur *output* tegangan *Power Supply Unit* (PSU) dari 0 V sampai 24 V sebagai tegangan sumber motor DC D-6759.
3. Gunakan tachometer digital untuk mendapatkan nilai putaran motor.

4. Amati dan catat hasil pengukuran putaran motor di setiap kenaikan 1V.

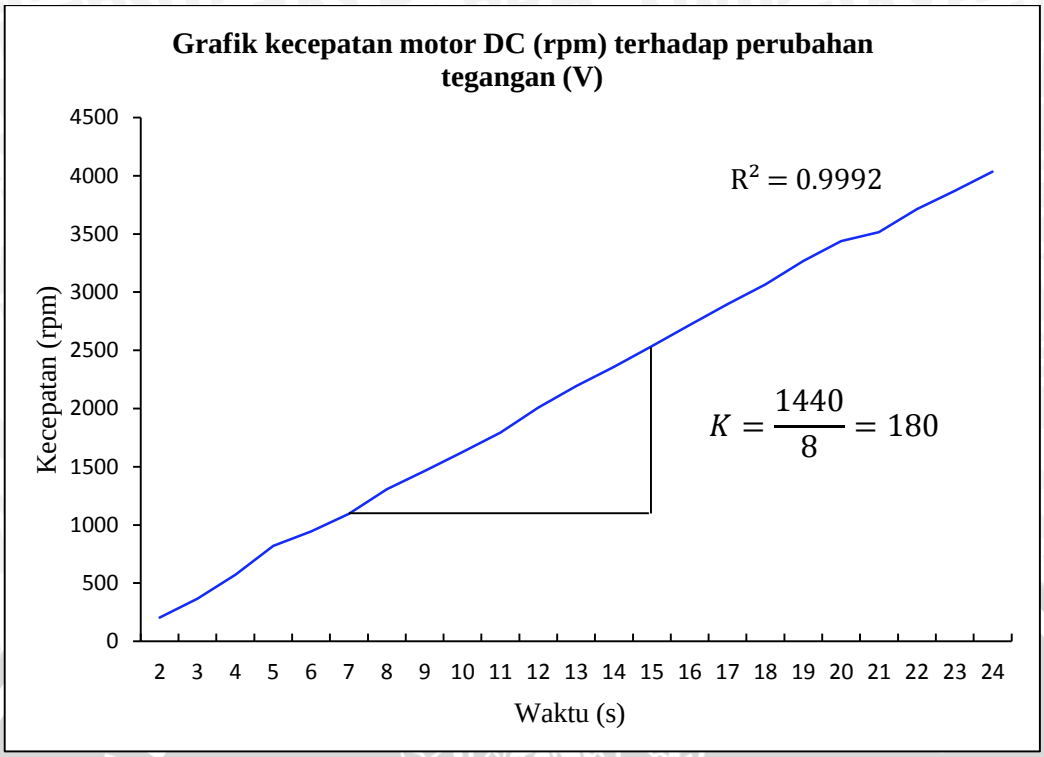
d. Hasil pengujian

Data hasil pengujian kecepatan dengan perubahan *input* tegangan pada motor DC D-6759 ditunjukkan dalam Tabel 5.1.

Tabel 5.1 Data pengujian kecepatan motor DC terhadap tegangan

Input Tegangan (V)	Kecepatan Motor (rpm)
0	0
1	0
2	205.1
3	367.5
4	573.4
5	819.2
6	943.8
7	1096
8	1307
9	1463
10	1627
11	1795
12	2008
13	2192
14	2358
15	2536
16	2718
17	2898
18	3067
19	3269
20	3438
21	3516
22	3713
23	3871
24	4036

Pada *input* tegangan 0 V dan 1 V, terlihat bahwa motor DC D-6759 tidak berputar, pada daerah ini dapat disebut dengan daerah *dead time*. Berdasarkan Tabel 5.1 dengan mengambil data tegangan dimulai dari 2 V, maka akan didapatakan kurva kecepatan motor (rpm) terhadap *input* tegangan (V) seperti ditunjukkan pada Gambar 5.1.



Gambar 5.1 Grafik perubahan kecepatan motor DC terhadap perubahan tegangan

5.1.2 Pengujian *Driver* Motor IC L298N (*Dual H-Bridge*)

a. Tujuan

Mengetahui kinerja dan respon rangkaian *driver* motor IC L298N (*Dual H-Bridge*) dengan membandingkan *output* tegangan efektif *driver* dengan masukan *duty cycle* sinyal PWM yang diberikan oleh Arduino Mega 2560.

b. Peralatan yang digunakan

1. Komputer atau PC.
2. *Power Supply Unit* (PSU).
3. *Driver* motor IC L298N (*Dual H-Bridge*).
4. Multimeter.
5. *Digital osciloscope* Vellen PCSU1000 (PC-Lab).
6. Arduino Mega 2560.
7. Kabel penghubung.

c. Langkah Pengujian

1. Hubungkan *output* tegangan *Power Supply Unit* (PSU) pada *input* tegangan referensi *driver* motor IC L298N (*Dual H-Bridge*).
2. Hubungkan *input* tegangan *driver* motor IC L298N (*Dual H-Bridge*) dengan pin *output* PWM di Arduino Mega 2560.

3. Hubungkan *output* tegangan *driver* motor IC L298N (*Dual H-Bridge*) dengan multimeter dan PC-Lab.
4. Atur *duty cycle* sinyal PWM pada Arduino Mega 2560 dengan nilai 0%-100%.
5. Amati dan catat hasil pembacaan multimemter dan PC-Lab disetiap kenaikan 5%.

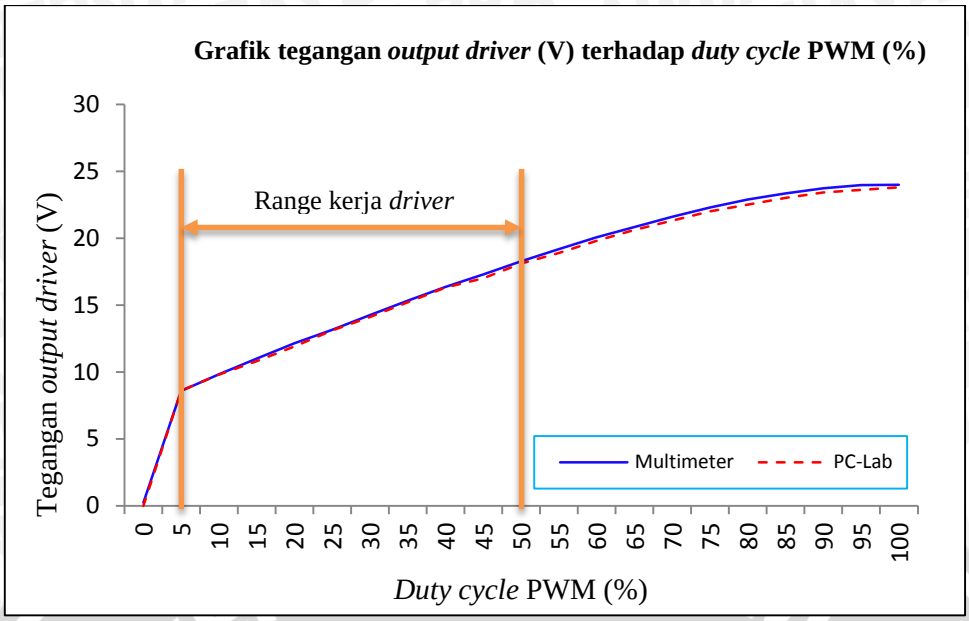
d. Hasil pengujian

Data pengujian *driver* motor IC L298N (*Dual H-Bridge*) ditunjukkan dalam Tabel 5.2.

Tabel 5.2 Data pengujian *driver* motor IC L298N

Duty cycle (%)	*Output driver* (V) dengan multimeter	*Output driver* (V) dengan PC-Lab
0	0.238	0
5	8.59	8.6
10	9.82	9.8
15	11	10.8
20	12.14	11.9
25	13.16	13.1
30	14.26	14.1
35	15.32	15.2
40	16.36	16.3
45	17.28	17
50	18.26	18.1
55	19.18	18.9
60	20.06	19.8
65	20.83	20.6
70	21.6	21.3
75	22.28	22
80	22.88	22.5
85	23.34	23
90	23.72	23.4
95	23.95	23.6
100	23.98	23.8

Berdasarkan Tabel 5.2 akan didapatkan kurva *output* tegangan *driver* (V) terhadap masukan *duty cycle* sinyal PWM (%) seperti ditunjukkan pada Gambar 5.2.



Gambar 5.2 Grafik perubahan tegangan *output driver* terhadap *duty cycle*

Dari Gambar 5.2 dapat dilihat bahwa kurva tidak linier, sehingga pemilihan range kerja sistem yang akan digunakan dalam proses pengontrolan adalah dari *dutycycle* PWM 5%-50%.

5.1.3 Pengujian *Rotary Encoder* E50S8-360-3-N-24

a. Tujuan

Mengetahui tingkat kelinieran dari *rotary encoder* dalam pembacaan putaran motor.

b. Peralatan yang digunakan:

1. Komputer atau PC.
2. *Power Supply Unit* (PSU).
3. *Rotary encoder* E50S8-360-3-N-24.
4. *Driver* motor IC L298N (*Dual H-Bridge*).
5. Motor DC D-6759.
6. Arduino Mega 2560
7. Kabel penghubung.
8.

c. Langkah pengujian

1. Hubungkan *output* tegangan *Power Supply Unit* (PSU) pada *input* tegangan sumber *rotary encoder* E50S8-360-3-N-24 dan *input* tegangan referensi *driver* motor IC L298N (*Dual H-Bridge*).

2. Hubungkan pin *output rotary encoder* E50S8-360-3-N-24 pada pin *interrupt* eksternal Arduino Mega 2560.
3. Atur *duty cycle* sinyal PWM pada Arduino Mega 2560 dengan nilai 0%-100%.
4. Catat hasil pembacaan *rotary enoder* E50S8-360-3-N-24 di setiap kenaikan 5%.

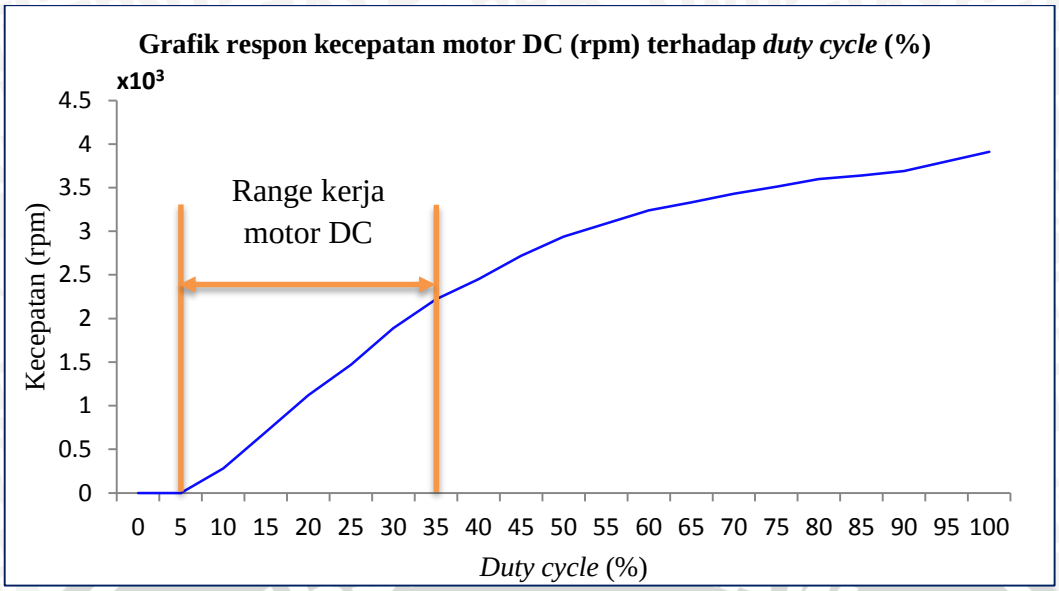
d. Hasil Pengujian

Data hasil pengujian *rotary encoder* E50S8-360-3-N-24 ditunjukkan dalam Tabel 5.3.

Tabel 5.3 Data pengujian *rotary encoder* E50S8-360-3-N-24

Duty cycle (%)	Kecepatan motor DC (rpm) dengan r*otary encoder*
0	0
5	0
10	280
15	700
20	1120
25	1470
30	1890
35	2220
40	2450
45	2720
50	2940
55	3090
60	3240
65	3330
70	3430
75	3510
80	3600
85	3640
90	3690
95	3800
100	3910

Berdasarkan Tabel 5.3 akan didapatkan kurva kecepatan motor dengan menggunakan *rotary encoder* E50S8-360-3-N-24 terhadap masukan *duty cycle* sinyal PWM (%) seperti ditunjukkan pada Gambar 5.3.



Gambar 5.3 Grafik perubahan respon kecepatan motor DC terhadap *duty cycle*

Pada Gambar 5.3, kurva respon tidak linear, sehingga pemilihan daerah kerja sistem yang akan digunakan dalam proses pengontrolan tidak dapat meggunakan semua range kecepatan yang ada. Dalam skripsi ini pemilihan range kerja sistem berada pada range kecepatan 0-2400 rpm, yaitu 150, 250 dan 350 rpm.

5.2 Simulasi Penggunaan Teknik MRAC pada Kontroler PI

Tahapan-tahapan simulasi yang dilakukan adalah

1. Simulasi sistem dengan beberapa nilai parameter kontroler γ_p dan γ_i.
2. Penentuan nilai parameter kontroler γ_p dan γ_i dilakukan untuk mendapatkan respon keluaran *plant* yang dapat menjejaki respon keluaran dari model referensi. Penentuan parameter kontroler γ_p dan γ_i dilakukan dengan cara memberikan nilai awal parameter pada nilai tertentu sebelum simulasi dilakukan.
3. Simulasi sistem dengan adanya pengaruh gangguan berupa perubahan nilai *setpoint*, simulasi ini dilakukan agar unjuk kerja dari sistem terhadap perubahan nilai *setpoint* dapat teramati.

Langkah-langkah yang dilakukan:

1. Tentukan nilai parameter kontroler γ_p dan γ_i dimulai dari nilai yang terkecil dan amati keluaran sistem yang meliputi respon sinyal masukan, model referenai, respon model, sinyal kontrol dan parameter kontroler. Dari simulasi ini dapat diketahui bagaimana pengaruh perubahan parameter kontroler γ_p dan γ_i pada sistem.

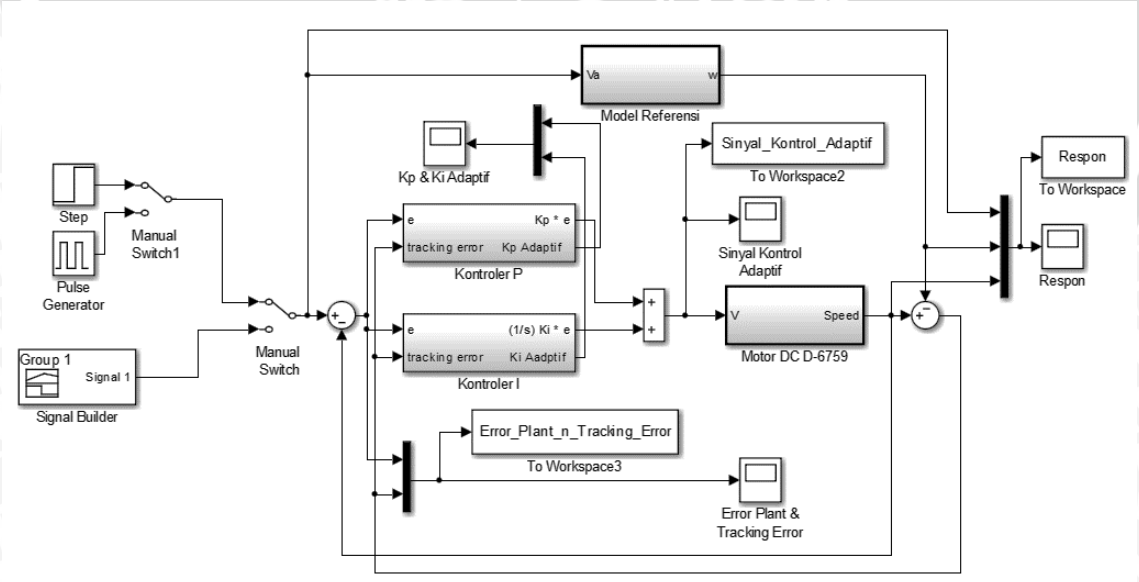
2. Melakukan simulasi dengan pengruh gangguan berupa perubahan nilai *setpoint*. Sistem yang digunakan adalah sistem yang tebaik yang didapatkan dari simulasi yang telah dilakukan sebelumnya. Dari simulasi ini akan dapat diamati bagaimana sistem beradaptasi untuk menjejaki model referensi.

5.3 Simulasi

Setelah desain matematik dapat diselesaikan, insinyur kontrol mensimulasikan model pada komputer untuk menguji perilaku sistem yang diperoleh dalam bentuk respon terhadap berbagai sinyal dan gangguan. Biasanya, konfigurasi sistem hasil desain awal belum memenuhi spesifikasi yang diinginkan. Oleh karena itu sistem ini harus didesain ulang berdasarkan informasi hasil analisis yang telah dilakukan. Proses desain dan analisis ini diulang sampai diperoleh sistem yang memuaskan. Selanjutnya, dari hasil simulasi pada komputer dapat dibuat sistem fisik prototipe (Ogata,1995).

5.3.1 Hasil Simulasi Penentuan Nilai Parameter Kontroler

Simulasi ini dilakukan pada sistem dengan sinyal masukan berupa unit step dengan menggunakan diagram blok simulink pada Matlab (lihat Gambar 5.4). Dari hasil simulasi ini diharapkan dapat memperoleh nilai parameter kontroler γ_p dan γ_i yang sesuai.

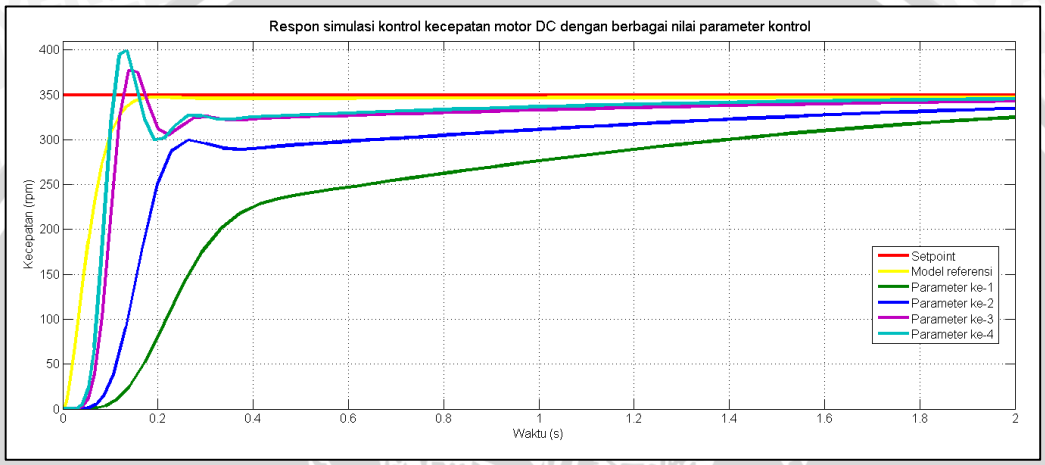


Gambar 5.4 Diagram blok simulink pada Matlab

Beberapa nilai parameter kontroler yang disimulasikan terdapat pada Tabel 5.4 dan respon sinyal masukan, model referensi dan keluaran sistem dapat dilihat pada Gambar 5.5.

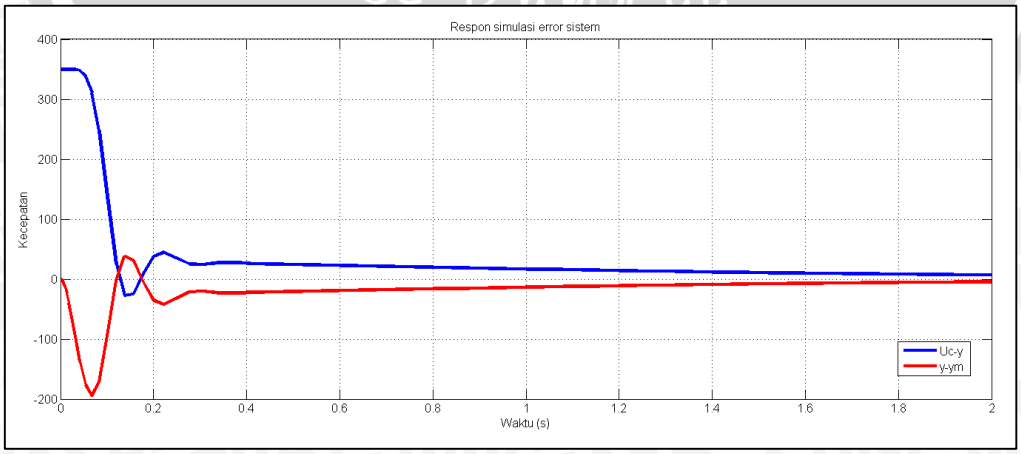
Tabel 5.4 Nilai parameter kontrol

Parameter ke-	γ_p	γ_i
1	0,0001	0,0001
2	0,0005	0.0005
3	0,005	0,005
4	0,01	0,01

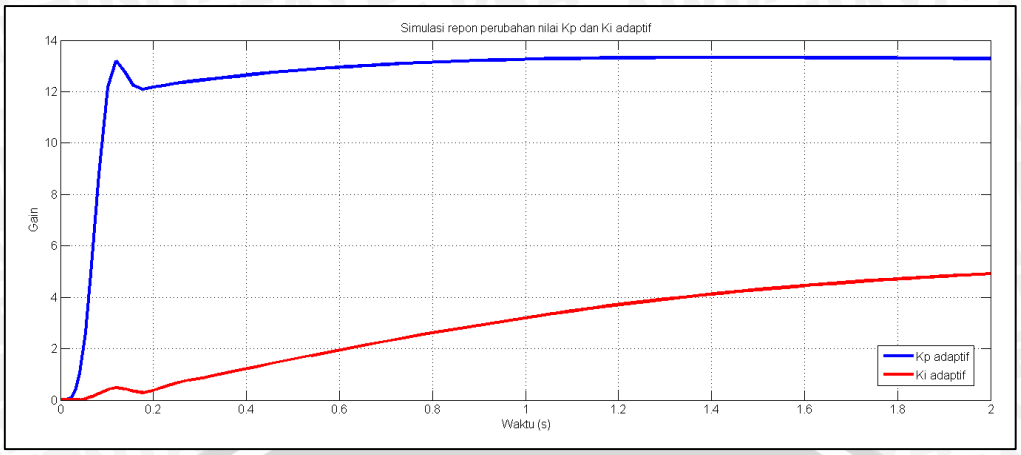


Gambar 5.5 Grafik respon simulasi sistem dengan variasi nilai parameter γ_p dan γ_i

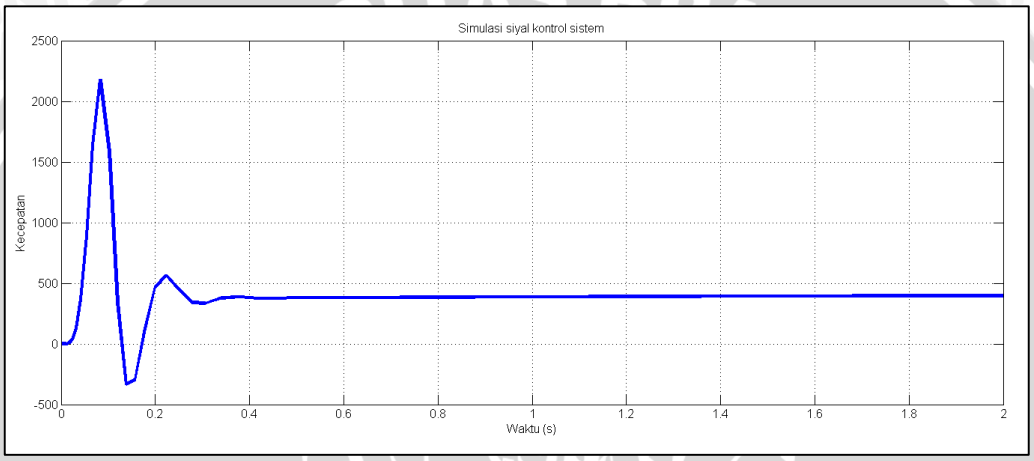
Berdasarkan hasil simulasi, jika nilai parameter γ_p dan γ_i ditentukan dengan melihat respon yang memiliki *settling time* dan *maximum overshoot* yang paling kecil, yaitu parameter ke-3. Hasil simulasi respon dari *error* perubahan parameter kontroler dan sinyal kontrol dengan parameter ke-3 dapat dilihat pada Gambar 5.6, 5.7 dan 5.8.



Gambar 5.6 Grafik respon simulasi *error* sistem



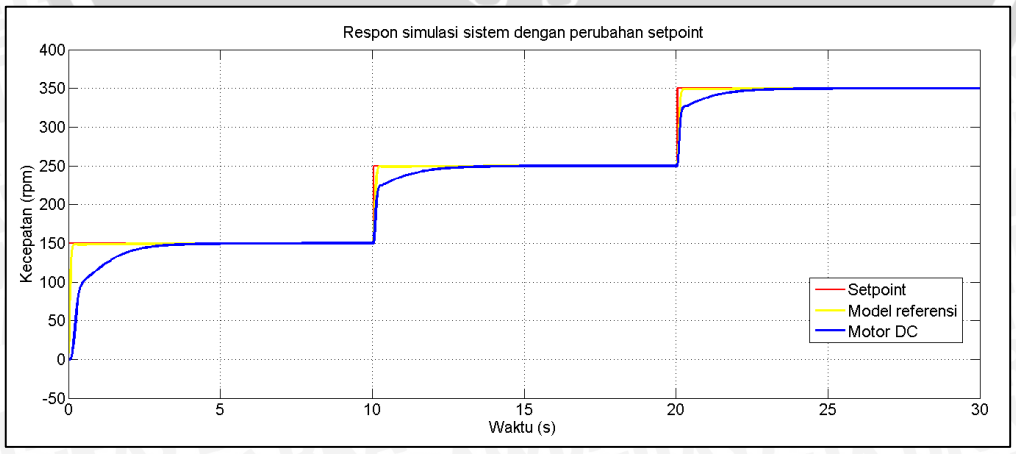
Gambar 5.7 Grafik respon simulasi perubahan Kp dan Ki



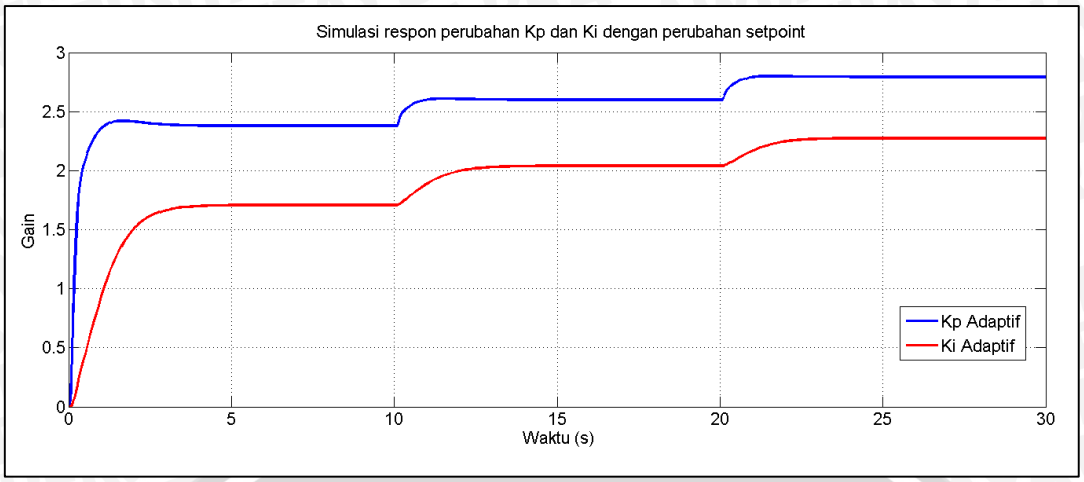
Gambar 5.8 Grafik respon simulasi sinyal kontrol

5.3.2 Hasil Simulasi Terhadap Perubahan Nilai *Setpoint*

Simulasi ini dilakukan untuk mendapatkan respon sistem dengan perubahan nilai *setpoint*. Hasil respon sistem dan perubahan nilai Kp dan Ki dapat dilihat pada Gambar 5.9 dan 5.10.



Gambar 5.9 Grafik respon simulasi sistem dengan perubahan nilai *setpoint*

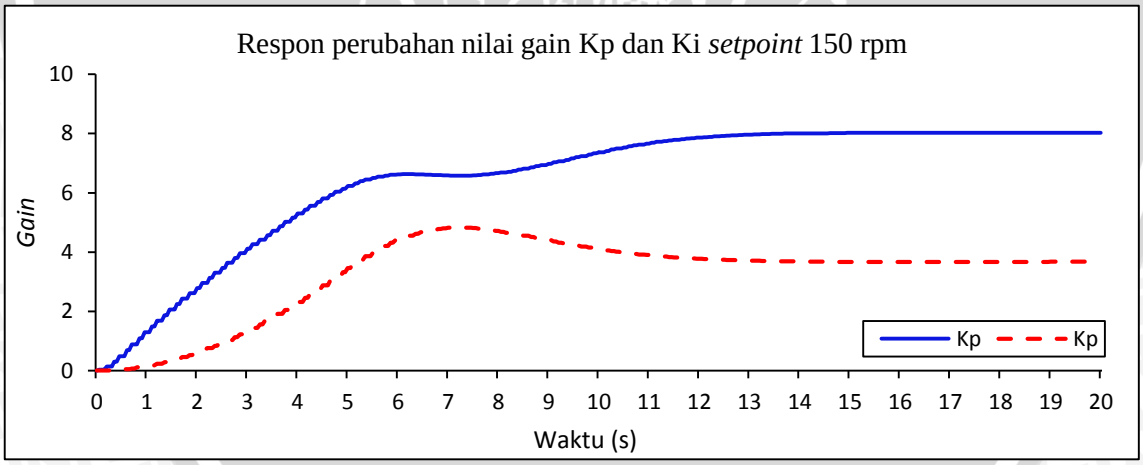


Gambar 5.10 Grafik respon simulasi perubahan nilai Kp dan Ki dengna perubahan nilai *setpoint*

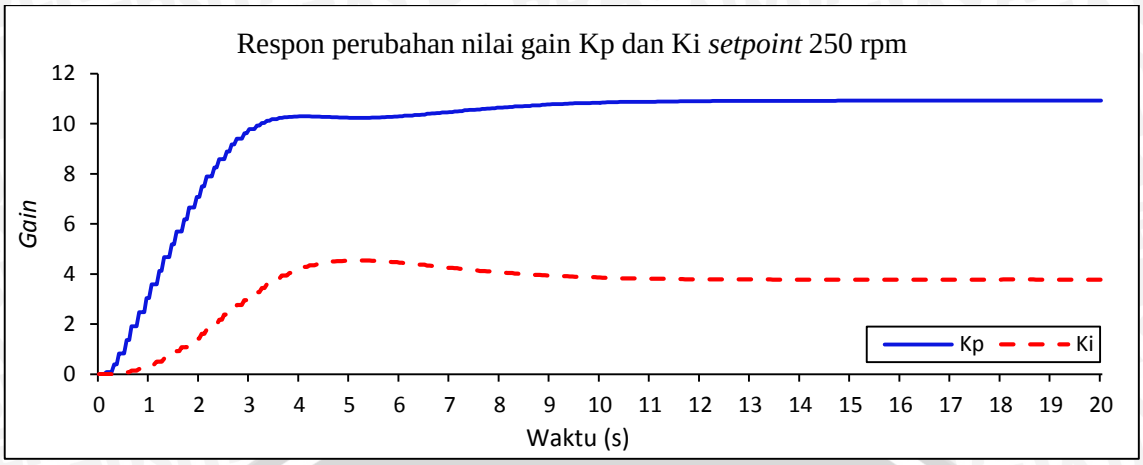
5.4 Implementasi

5.4.1 Hasil Implementasi dengan Beberapa Nilai *Setpoint*

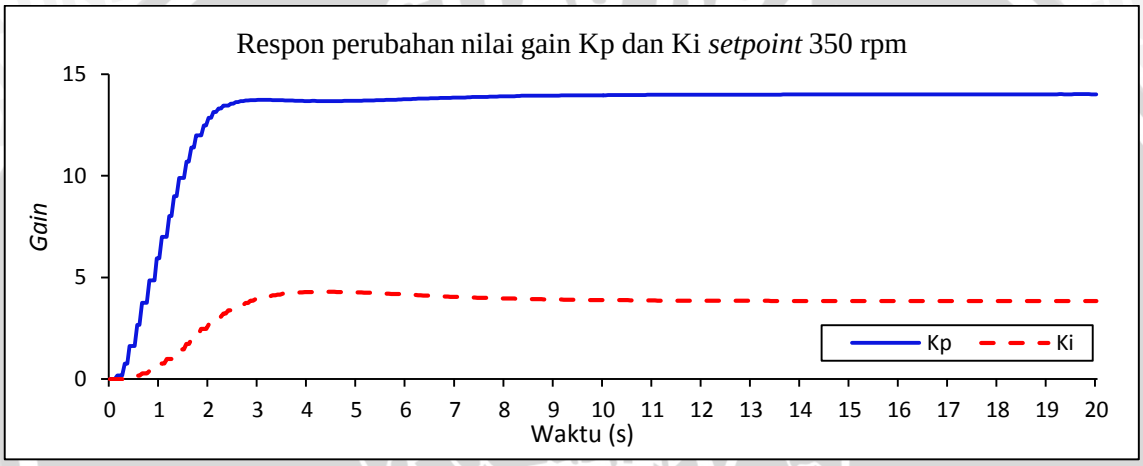
Implementasi dari sistem menggunakan *sampling* setiap 0,05 detik dan 3 nilai *setpoint*, yaitu 150, 250 dan 350 rpm. Data yang diperoleh adalah perubahan nilai Kp dan Ki pada setiap nilai *setpoint* dan data nilai respon kecepatan dari motor DC D-6759 pada setiap *setpoint* dengan hasil perubahan Kp dan Ki. Berikut data yang diperoleh dalam bentuk grafik respon pada Gambar 5.11, 5.12 dan 5.13.



Gambar 5.11 Grafik perubahan nilai Kp dan Ki *setpoint* 150 rpm



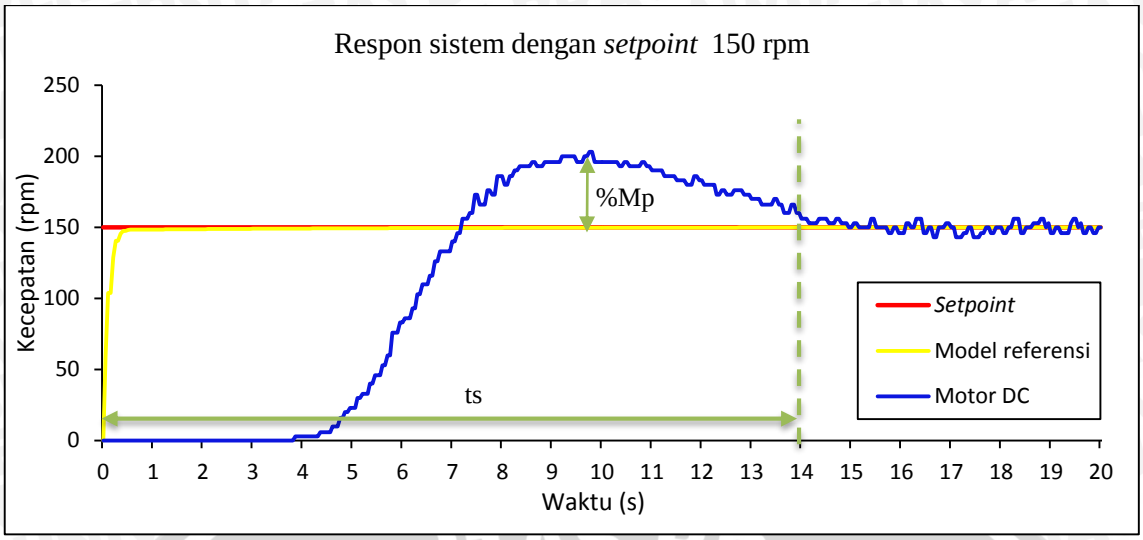
Gambar 5.12 Grafik perubahan nilai Kp dan Ki *setpoint* 250 rpm



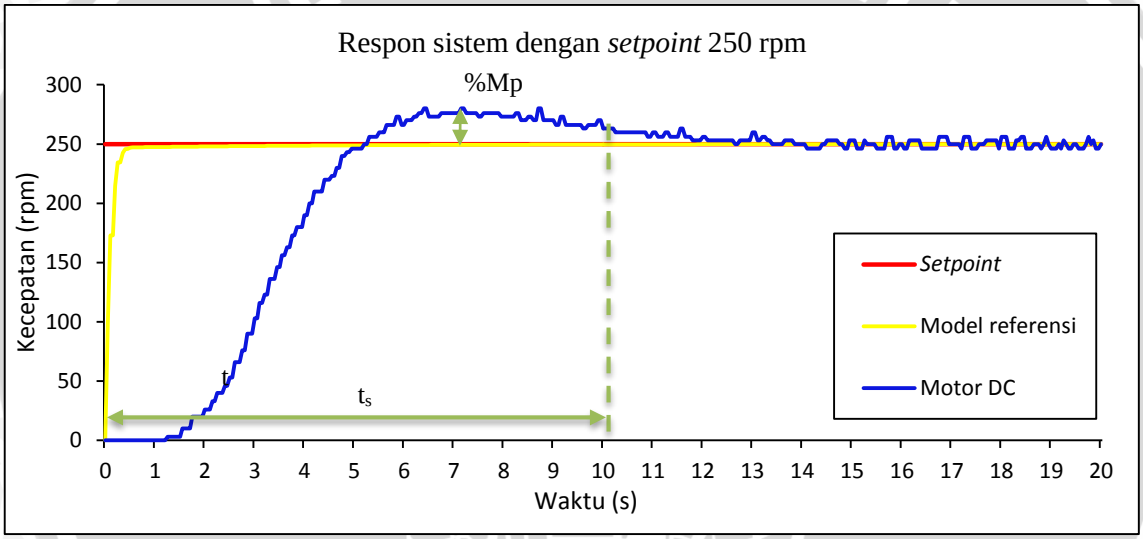
Gambar 5.13 Grafik perubahan nilai Kp dan Ki *setpoint* 350 rpm

Dari ketiga grafik (Gambar 5.11, 5.12 dan 5.13), nilai perubahan Kp dan Ki saat motor mulai dinyalakan mengalami kenaikan nilai yang tinggi. Sedangkan ketika motor sudah mencapai keadaan *steady state*, kenaikan nilai Kp dan Ki secara perlahan.

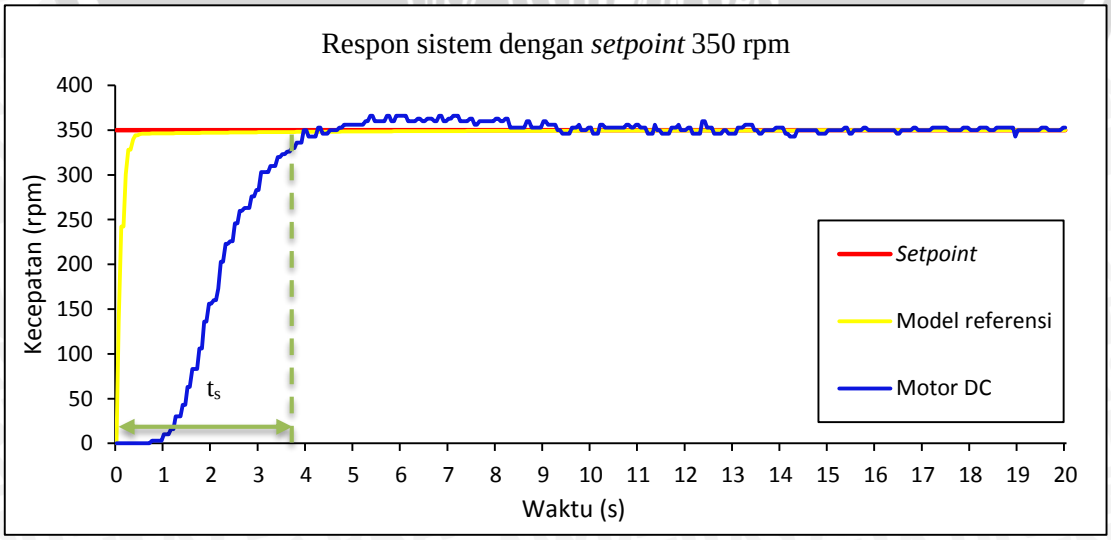
Pada grafik respon motor DC (Gambar 5.14, 5.15 dan 5.16) dengan nilai *setpoint* 150 rpm memilki nilai *maximum overshoot* sebesar 35,3%, nilai *setling time* adalah 14 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,96%. Respon motor DC dengan nilai *setpoint* 250 rpm memilki nilai *maximum overshoot* sebesar 12%, nilai *setling time* adalah 10,25 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,557%. Respon motor DC dengan nilai *setpoint* 350 rpm memilki tidak terjadi *overshoot*, nilai *setling time* adalah 3,8 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,276%.



Gambar 5.14 Grafik respon motor DC dengan *setpoint* 150 rpm



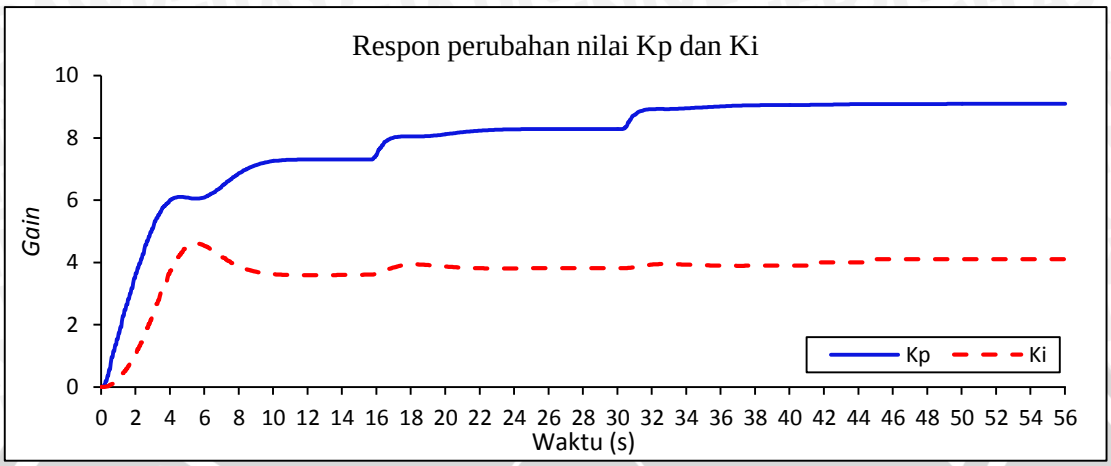
Gambar 5.15 Grafik respon motor DC dengan *setpoint* 250 rpm



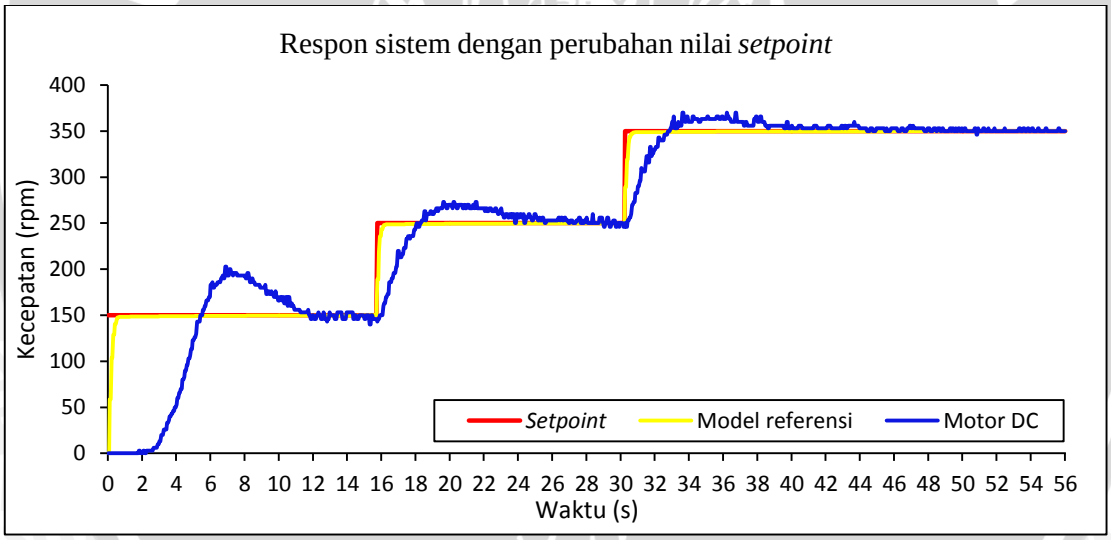
Gambar 5.16 Grafik respon motor DC dengan *setpoint* 350 rpm

5.4.2 Hasil Implementasi dengan Perubahan Nilai *Setpoint*

Pada implementasi sistem dengan memberikan perubahan nilai *setpoint* pada saat sistem sedang berada pada keadaan *steady state*, maka akan didapatkan grafik respon perubahan nilai Kp dan Ki dan motor DC seperti pada Gambar 5.17 dan 5.18.



Gambar 5.17 Grafik respon perubahan nilai Kp dan Ki dengan perubahan nilai *setpoint*



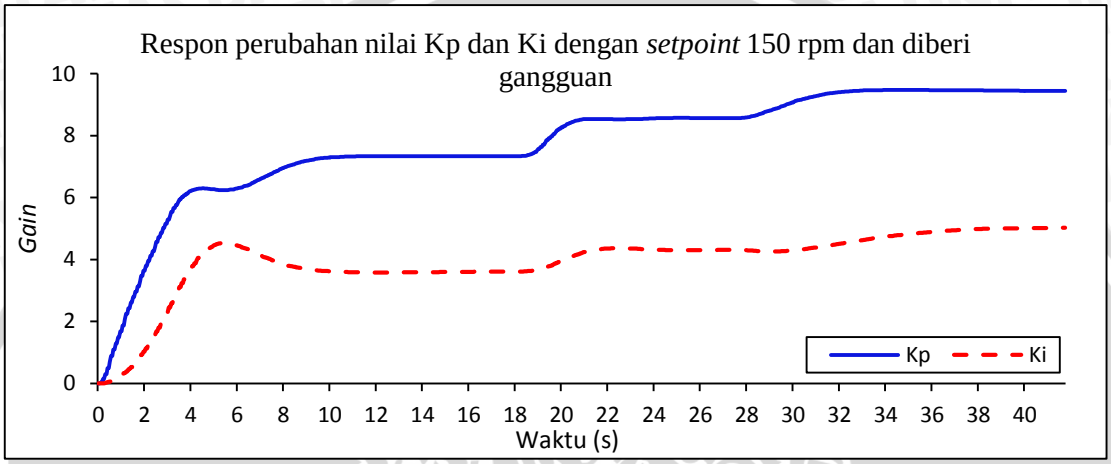
Gambar 5.18 Grafik respon sistem dengan perubahan nilai *setpoint*

Pada respon sistem dengan perubahan nilai *setpoint* memiliki nilai *error steady state* dibawah 2% yaitu 1,6%. Pada *setpoint* 150 rpm, respon sistem terjadi *overshoot* sebesar 33,33%, nilai *setling time* adalah 10,9 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,931%. Pada saat terjadi perubahan *setpoint* dari 150 rpm menjadi 250 rpm, respon sistem mengalami *overshoot* sebesar 9,2%, nilai *setling time* adalah 7,3 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,564%. Pada saat terjadi perubahan dari *setpoint* 250 rpm menjadi 350 rpm, respon sistem mengalami *overshoot* sebesar 5,71%, nilai *setling time* adalah

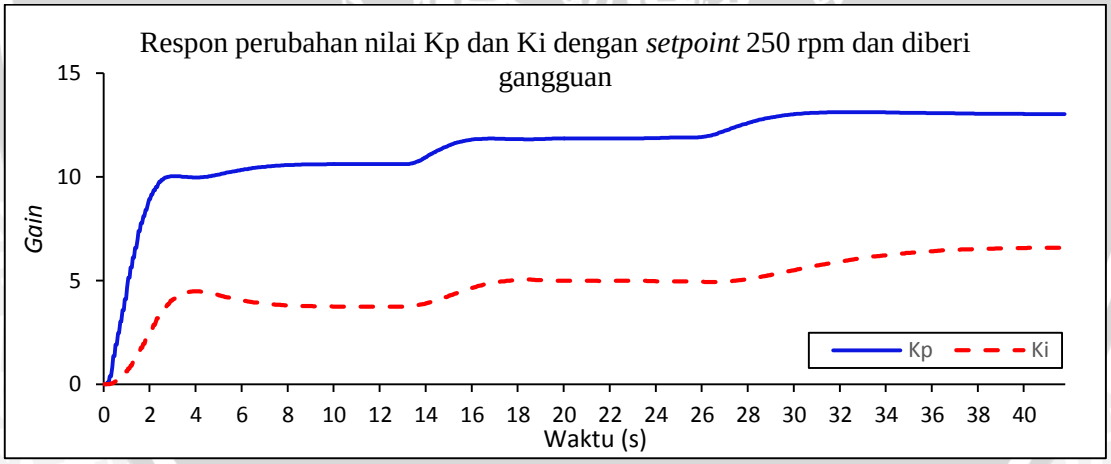
6,55 detik dengan nilai *error steady state* rata-rata berada dibawah toleransi 2% yaitu 1,31%.

5.4.3 Hasil Implementasi dengan Diberikan Gangguan

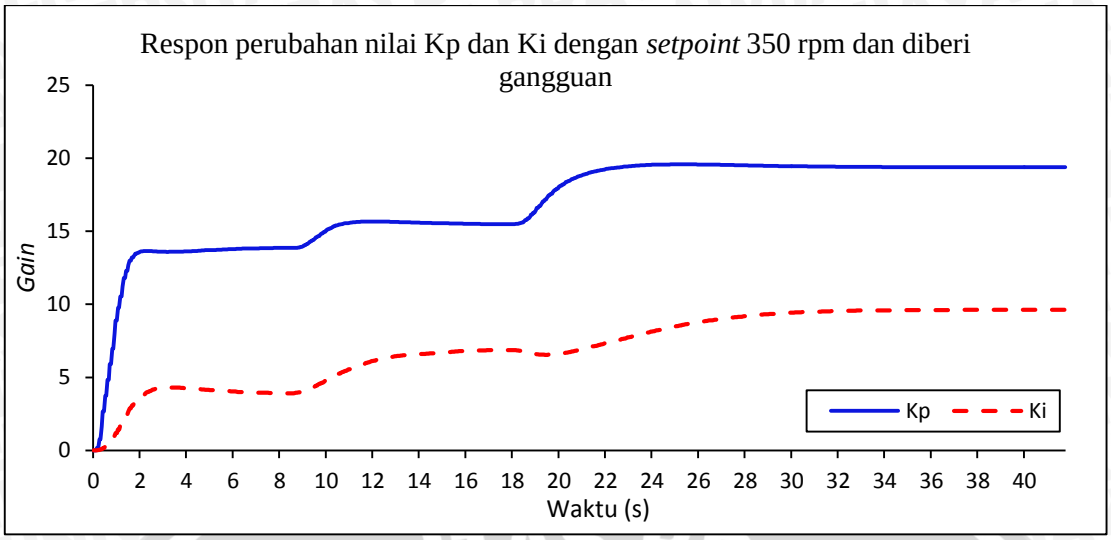
Pada implementasi sistem dengan memberikan gangguan pada saat sistem sedang berada pada keadaan *steady state*, maka akan didapatkan grafik respon Kp dan Ki dan respon motor DC seperti pada Gambar 5.19, 5.20 dan 5.21.



Gambar 5.19 Grafik perubahan nilai Kp dan Ki *setpoint* 150 rpm dan diberi gangguan

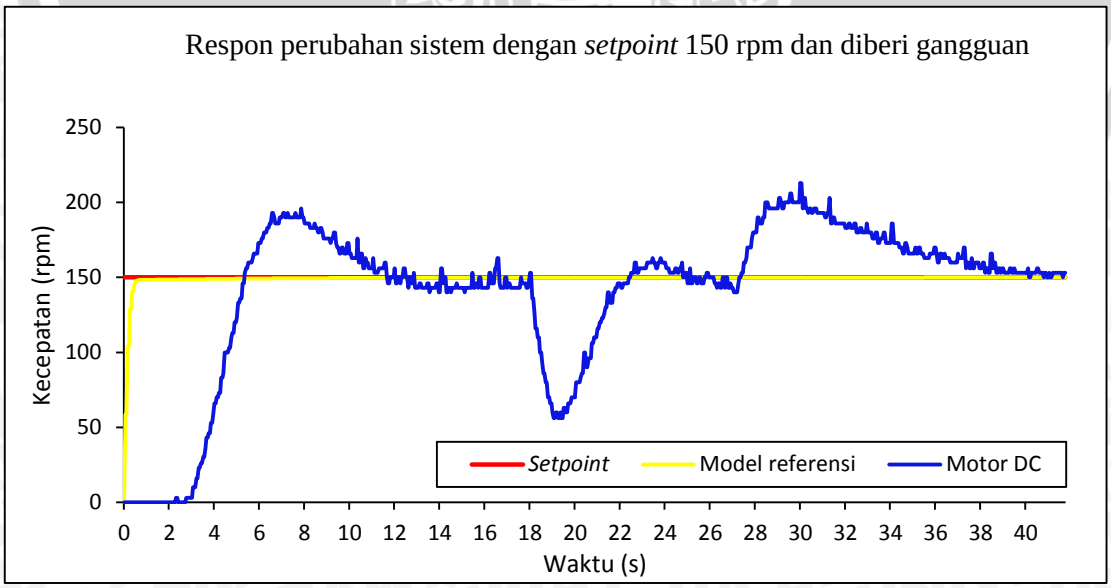


Gambar 5. 20 Grafik perubahan nilai Kp dan Ki *setpoint* 250 rpm dan diberi gangguan

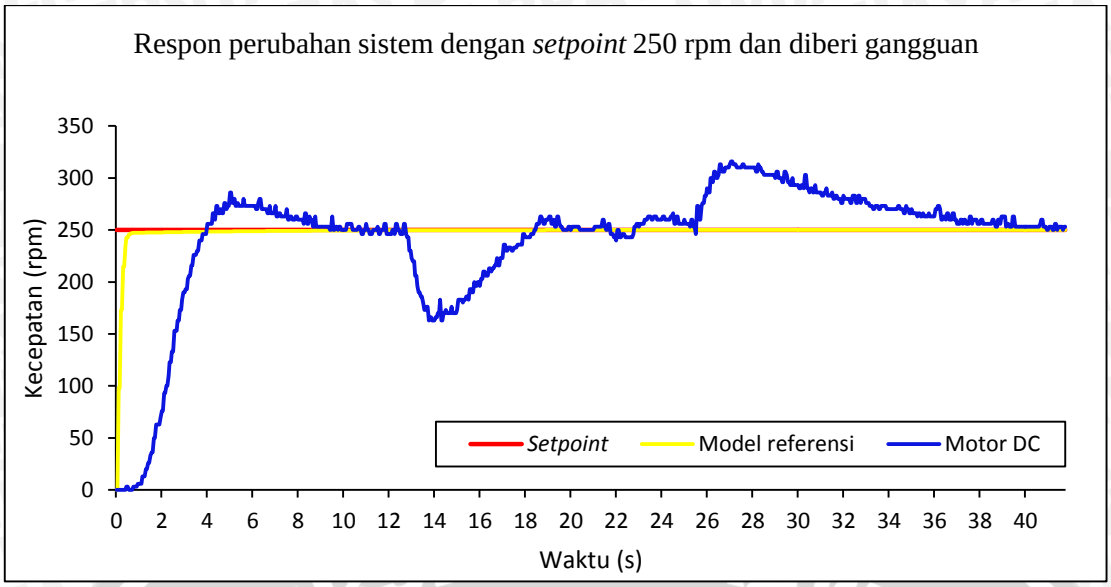


Gambar 5. 21 Grafik perubahan nilai Kp dan Ki *setpoint* 350 rpm dan diberi gangguan

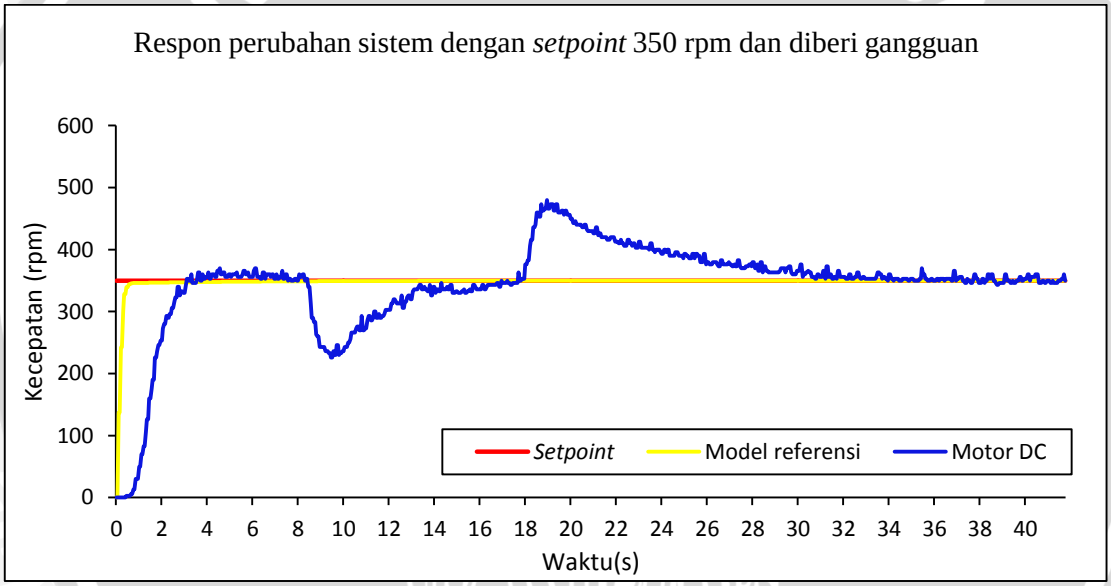
Pada respon sistem (Gambar 5.22, 5.23 dan 5.24) dengan diberikan gangguan pada *setpoint* 150 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 4,55 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 7,4 detik. Pada *setpoint* 250 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 3,85 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 9,65 detik. Pada *setpoint* 350 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 3,5 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 10,9 detik.



Gambar 5. 22 Grafik respon motor DC dengan *setpoint* 150 rpm dan diberi gangguan



Gambar 5. 23 Grafik respon motor DC dengan *setpoint* 250 rpm dan diberi gangguan



Gambar 5. 24 Grafik respon motor DC dengan *setpoint* 350 rpm dan diberi gangguan

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil identifikasi *plant* menggunakan sinyal uji PRBS dan sintaks ident pada *software* Matlab, fungsi alih motor DC memilki *best fit* sebesar 94.45 dan telah divalidasi. Berdasarkan hasil simulasi didapatkan nilai parameter kontrol yang memenuhi kestabilan respon sistem adalah $\gamma_p = 0{,}005$ dan $\gamma_i = 0{,}005$. Berdasarkan hasil implementasi, respon motor DC dengan nilai *setpoint* 150 rpm, 250 rpm dan 350 rpm memiliki nilai *error steady state* rata-rata berada dibawah toleransi 2% dengan nilai masing-masing adalah 1,96%, 1,557%, dan 1,276%. Sedangkan *settling time* adalah 14 detik, 10,25 detik dan 3,8 detik. Respon sistem dengan perubahan nilai *setpoint* memiliki nilai *error steady state* rata-rata dibawah 2% yaitu 1,6%. Saat sistem diberi gangguan, pada *setpoint* 150 rpm, 250 rpm dan 350 rpm, respon akan mengalami perlambatan dan *recovery time* respon kembali keadaan *steady state* masing-masing adalah 4,55 detik, 3,85 detik dan 3,5 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon masing-masing adalah 7,4 detik, 9,65 detik dan 10,9 detik.

6.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah dengan mengimplementasikan motor DC sebagai aktuator seperti pada lift barang, konveyor, dan sebagainya.

DAFTAR PUSTAKA

Ali, A.T., Eisa B. dan Omar B. 2012. *Adaptive PID Controller for DC Motor Speed Control.* International Journal of Engineering Inventions 1(5), 26-30.

Arif, M. Faishol. 2015. *Sistem Kontrol Kecepatan Motor DC D-6759 Berbasis Arduino Mega 2560*. Malang: Skripsi Teknik Elektro Universitas Universitas Brawijaya Malang.

Astrom, K.J. dan B. Wittenmark. 1995. *Adaptive Control*. Addison-Wesley Publishing Company, Inc. USA.

Butler, H. 1992. Model Reference Adaptive Systems, From Theory to Practice. UK: Prentice-Hall, Inc.

Dorf, R.C. dan Robert H.B. 2008. *Modern Control Systems, 11thEd.* NJ: Prentice-Hall, Inc.

Gunterus, F. 1994. *Falsafah Dasar: Sistem Pengendali Proses*. Jakarta: PT Elex Media Komputindo.

Morris, S Alan. 2011. *Measurement and Instrumentation Principles,* Oxford: Butterworth-Heinemann.

Ogata, K. 1995. *Teknik Control Automatik (Sistem Pengaturan)*. Jilid 1. Diterjemahkan oleh: Leksono, Edi. Jakarta: Erlangga.

Phillips, Charles L., H. Troy Nagle. 1995. *Digital Control System Analysis and Design.*Engelwood Cliffs, NJ: Prentice-Hall, Inc.

Pirabakaran, K. dan V. M. Becerra. 2001. *Automatic Tuning of PID Controllers Using Model Reference Adaptive Control Techniques.* IEEE: *Industrial Electronics Society.*

Sar, S.K. dan lillie D. 2014. *MRAC Based PI Controller for Speed Control of DC Motor Using Lab View.* WSEAS Transactions on Systems and Control.Wain, Y. Suban. https://asro.wordpress.com/2009/01/16/diskritisasi/(diakses pada 23 Juli 2015).

Wain, Y. Suban. https://asro.wordpress.com/2009/01/16/diskritisasi.(diakses pada 23 Juli 2015).

Xiog, Ai dan Yogkun Fan. 2007. *Application of a PID Controller using MRAC Techniques for Control of the DC Electromotor Drive*. IEEE: *International Conference on Mechatronis and Automation.*

Lampiran 1

Foto Alat

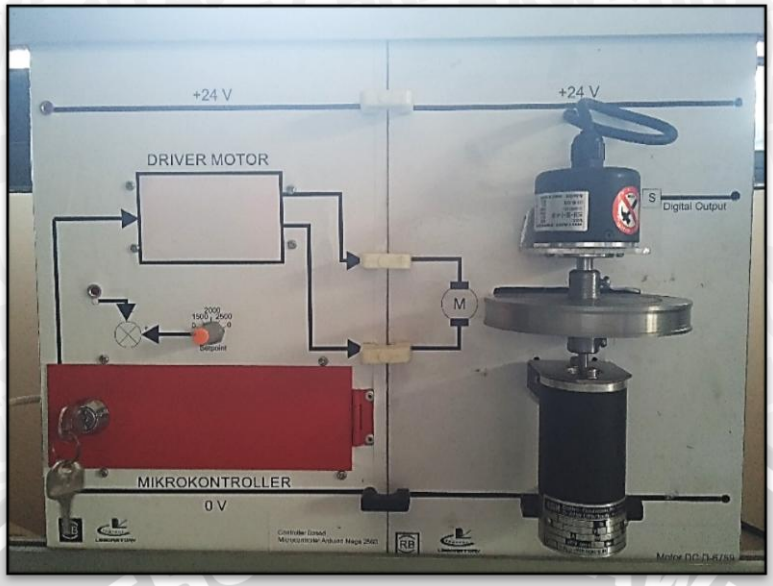


Foto modul mikrokontroler, *driver*, motor DC dan *rotary encoder*

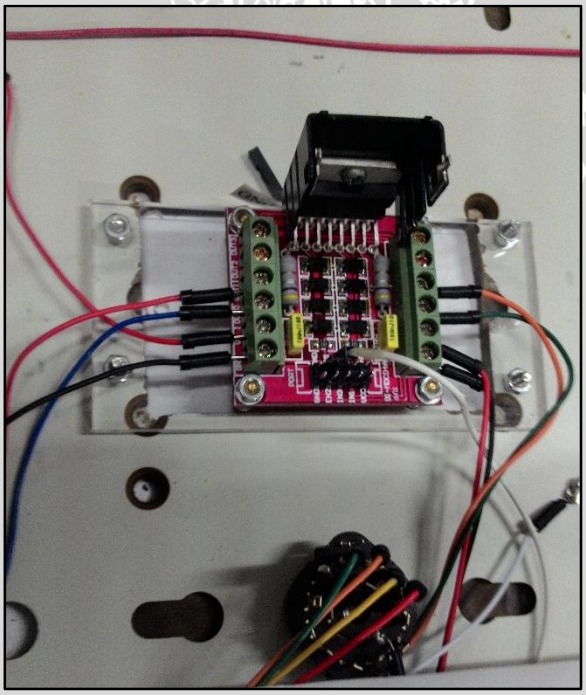


Foto *driver* motor IC L298N (*Dual H-Bridge*) di dalam modul

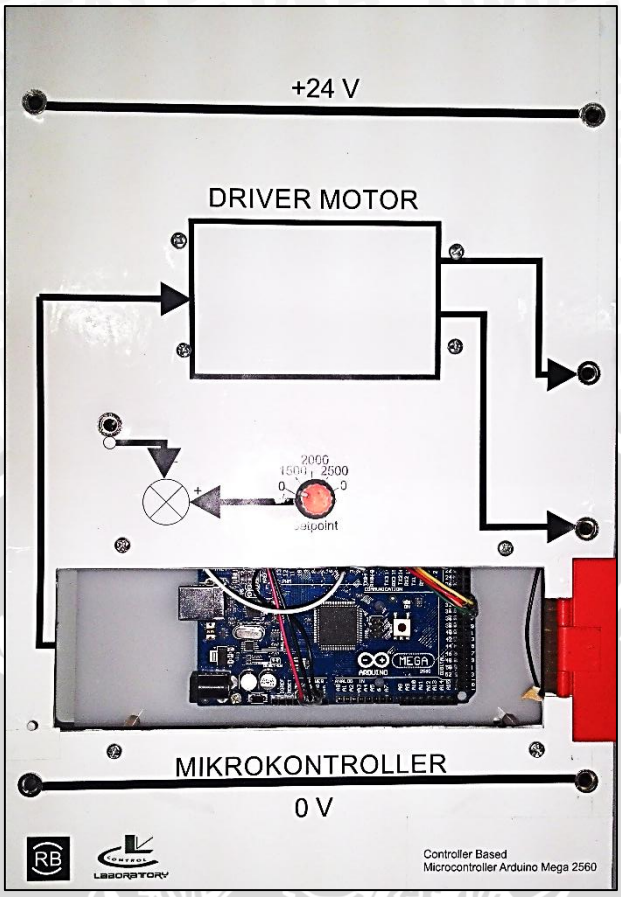


Foto mikrokontroler Arduino Mega 2560 di dalam modul

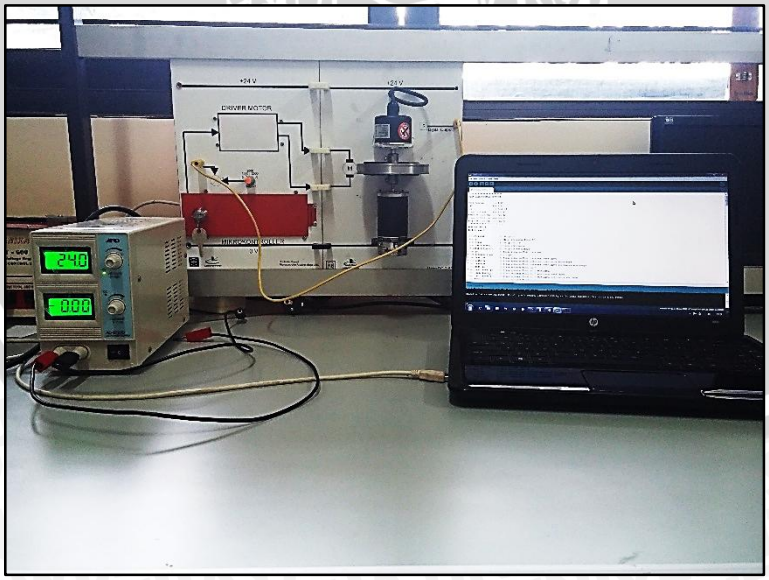


Foto pengujian keseluruhan

Lampiran 2

Diagram Blok

Diagram Blok SIMULINK Subsistem Kontroler P

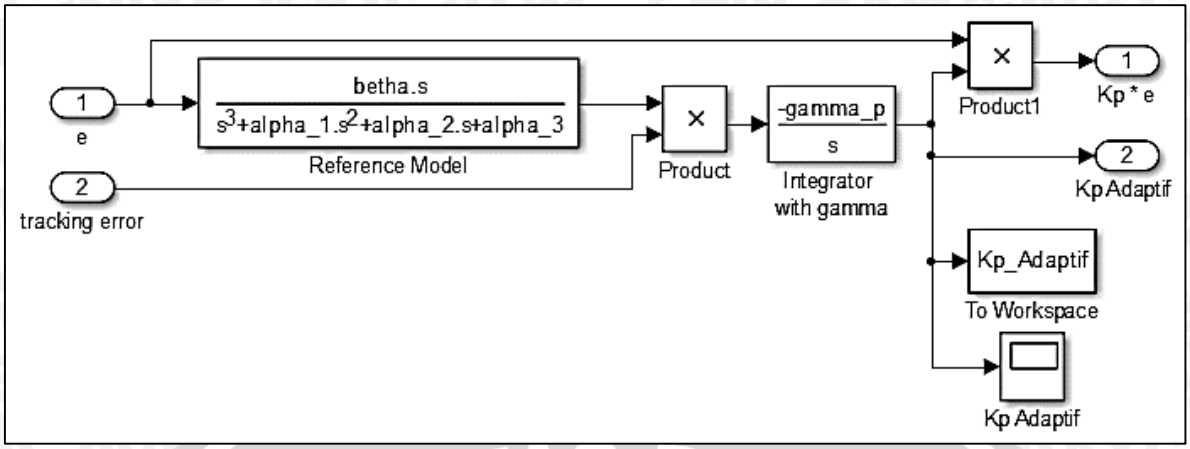


Diagram Blok SIMULINK Subsistem Kontroler I

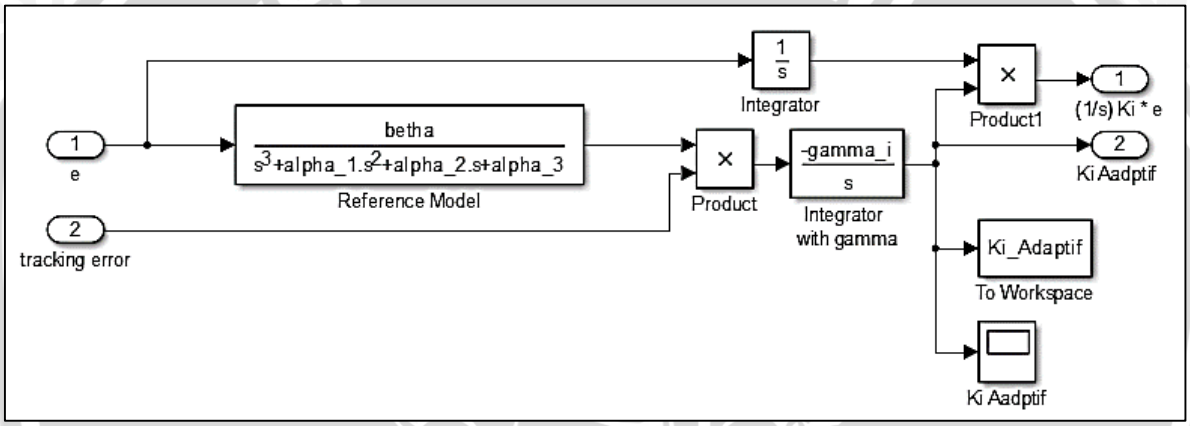
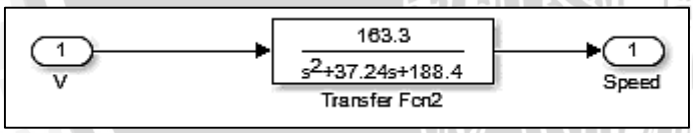


Diagram Blok SIMULINK Subsistem Motor DC D-6759



Lampiran 3

Listing Program

Listing program M-File simulasi

```
clear all
clc

%%%%%PENENTUAN MODEL REFERENSI%%%%%

betha       = 952.3;
alpha_1     = 53.79;
alpha_2     = 979.8;
alpha_3     = 279.8;


%%%%%PENENTUAN PARAMETER GAMMA%%%%%
gamma_p = 0.005; gamma_i = 0.005;
```

Listing program Sitem kontrol adaptif dengan MRAC

```
/*************************
Baca Putaran              ==> PIN2
PWM                       ==> PIN5
LED                       ==> Pout 13
setpoint 0 rpm            ==> Pin 30
setpoint 150 rpm          ==> Pin 31
setpoint 250 rpm          ==> Pin 32
setpoint 350 rpm          ==> Pin 33
-**************************/
#define led 13
#define pwm 5

int set_point;            // Setpoint
float y;                  // Nilai kecepatan Motor DC
float error;              // set_point - y
float last1_error;        // error sebelumnya
int last1_set_point;      // Setpoint sebelumnya
float ym;                 // Nilai keluaran Model referensi
float tracking_error;     // y - ym
float last1_ym;           // Nilai keluaran Model referensi sebelumnya
float last2_ym;           // Nilai keluaran Model referensi sebelumnya sebelumnya
float last3_ym;           // Nilai keluaran Model referensi sebelumnya sebelumnya sebelumnya
float model_p;            // Nilai keluaran Model di P
float last1_model_p;      // Nilai keluaran Model di P sebelumnya
float last2_model_p;      // Nilai keluaran Model di P sebelumnya sebelumnya
float last3_model_p;      // Nilai keluaran Model di P sebelumnya sebelumnya sbelumnya
float model_i;            // Nilai keluaran Model di I
float last1_model_i;      // Nilai keluaran Model di I sebelumnya
float last2_model_i;      // Nilai keluaran Model di I sebelumnya sebelumnya
float last3_model_i;      // Nilai keluaran Model di I sebelumnya sebelumnya sbelumnya
long int pulsa;           // pulsa rotary encoder
float gamma_p;            // parameter kendali P
float gamma_i;            // parameter kendali I
float Ts;                 // Time Sampling Model Referensi
float A, B, C, D, E, F, G; // Konstanta Model
float pwmMotor;           // PWM Motor
float Prop;               // Var sementara P
float Intg;               // Var sementara I
float last1_I;            // Var sementara I sebelumnya
double MV;                // Nilai keuaran PI (sinyal kontrol)
```

```
double last1_MV;        // Nilai keuaran PI (sinyal kontrol) sebelumnya
float Kp;               // Nilai konstanta Kp
float last1_Kp;         // Nilai konstanta Kp sebelumnya
float Ki;               // Nilai konstanta Ki
float last1_Ki;         // Nilai konstanta Ki sebelumnya
float integrator;       // Nilai integrator Kontroler I
float last1_integrator; // Nilai integrator Kontroler I sebelumnya
float R, S;             // Hasil perkalian model dengan tracking error
int as,ap;

void setup()
{
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(pwm,OUTPUT);
 pinMode(led,OUTPUT);
 pinMode(30, INPUT_PULLUP);
 pinMode(31, INPUT_PULLUP);
 pinMode(32, INPUT_PULLUP);
 pinMode(33, INPUT_PULLUP);

 Ts = 0.05;
 gamma_p = 0.0002;
 gamma_i = 0.0001;

 //Konstanta Model
 A = 3+(107.58*Ts)+(979.8*Ts*Ts);
 B = 3+(53.79*Ts);
 C = 1;
 D = (952.3*Ts*Ts)+(279.8*Ts*Ts*Ts);
 E = (952.3*Ts*Ts);
 F = (952.3*Ts*Ts*Ts);
 G = 1+(53.79*Ts)+(979.8*Ts*Ts)+(279.8*Ts*Ts*Ts);

 last1_ym = 0;    last2_ym = 0;     last3_ym = 0;     last1_set_point = 0;
 last1_error = 0; last1_integrator = 0;
 last1_model_p = 0;        last2_model_p = 0;        last3_model_p = 0;
 last1_model_i = 0;        last2_model_i = 0;        last3_model_i = 0;
 last1_Kp = 0;    last1_Ki = 0;

 noInterrupts();
 TCCR1A = 0;
 TCCR1B = 0;
 TCNT1  = 0;

 OCR1A   = 3124;                 // compare match register 16MHz/256/50Hz/50ms
 TCCR1B |= (1 << WGM12);         // CTC mode
 TCCR1B |= (1 << CS12);          // 256 prescaler
 TIMSK1 |= (1 << OCIE1A);        // enable timer compare interrupt

 TCCR2A = 0b00000010;
 TCCR2B = 0b00000111;
 TIMSK2 |= (1 << OCIE2A);        // enable timer compare interrupt
 OCR2A   = 156;                  // compare match register 16MHz/64/10000=10ms

 TCCR0A = 0b00000010;
 TCCR0B = 0b00000101;
 TIMSK0 |= (1 << OCIE0A);        // enable timer compare interrupt
 OCR0A   = 156;                  // compare match register 16MHz/64/10000=10ms
```

```
  attachInterrupt(0, hitung_pulsa, FALLING);
  interrupts();                         // enable all interrupts
  Serial.begin(9600);
}

// Timer rotary encoder
ISR(TIMER2_COMPA_vect)          // timer compare interrupt service routine
{
  Rot_switch();
  if(as==5)
  {
    as=0;
    y = (pulsa*1200)/360;
    pulsa = 0;
   }
  as++;
}

// Timer Model referensi
ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  MRAC();
}

//Timer tuning
ISR(TIMER0_COMPA_vect)          // timer compare interrupt service routine
{
  if(ap==5)
  {
  ap=0;
  error =  set_point - y;
  tracking_error = y - ym;
  Tuning_Kp();
  Integrator();
  Tuning_Ki();
  Kontroler();
  }
 ap++;
}

void MRAC()
{
  ym = ((A/G)*last1_ym) - ((B/G)*last2_ym) + ((C/G)*last3_ym) + ((D/G)*set_point)-
((E/G)*last1_set_point);
  last1_set_point = set_point;
  last3_ym = last2_ym;
  last2_ym = last1_ym;
  last1_ym = ym;
}

void Tuning_Kp()
{
  model_p = ((A/G)*last1_model_p) - ((B/G)*last2_model_p) + ((C/G)*last3_model_p) + ((E/G)*error) -
((E/G)*last1_error);
  last3_model_p = last2_model_p;
  last2_model_p = last1_model_p;
  last1_model_p = model_p;
  last1_error = error;
  R = model_p*tracking_error;
  Kp = last1_Kp + ((-gamma_p)*Ts*R);
```

```
  last1_Kp = Kp;
}

void Tuning_Ki()
{
  model_i = ((A/G)*last1_model_i) - ((B/G)*last2_model_i) + ((C/G)*last3_model_i) + ((F/G)*error);
  last3_model_i = last2_model_i;
  last2_model_i = last1_model_i;
  last1_model_i = model_i;
  S = model_i*tracking_error;
  Ki = last1_Ki + ((-gamma_i)*Ts*S);
  last1_Ki = Ki;
}

void Integrator()
{
  integrator = last1_integrator + (Ts * error);
  last1_integrator = integrator;
}

void Kontroler()
{
  Prop = Kp * error;
  Intg = Ki * integrator;
  MV = Prop + Intg;

  if (MV<0)MV = 0;
  pwmMotor = MV*0.046;

}

void Rot_switch()
{
  if(digitalRead(30)==LOW)          // baca input setpoint
  {
    set_point=0;
  }
  else if(digitalRead(31)==LOW)
  {
    set_point=150;
  }
  else if(digitalRead(32)==LOW)
  {
    set_point=250;
  }
  else if(digitalRead(33)==LOW)
  {
    set_point=350;
  }
  else;
}

void loop()
{
  digitalWrite(7,LOW);
  digitalWrite(8,HIGH);
  analogWrite(pwm,pwmMotor);
  Serial.print(Kp);
  Serial.print("\t\t");
```

```
  Serial.print(Ki);
  Serial.print("\t\t");
  Serial.print(set_point);
  Serial.print("\t\t");
  Serial.print(ym);
  Serial.print("\t\t");
  Serial.print(y);
  Serial.print("\t\t");
  Serial.print("\n");
}

void hitung_pulsa()
{
  pulsa++;
}
```

Lampiran 4

Datasheet