BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi kontrol saat ini telah memberikan banyak manfaat yang membawa kemajuan diberbagai aspek sosial. Tidak sedikit saat ini penggunaan teknologi kontrol yang dapat membantu menyelesaikan pekerjaan manusia banyak dijumpai dalam kehidupan sehari-hari dan hampir seluruh industri didunia saat ini memanfaatkan perkembangan teknologi kontrol.

Motor Direct Current (DC) merupakan aktuator yang banyak digunakan dalam teknologi kontrol. Motor DC memiliki respon yang cepat, namun masih memiliki *error steady state* (offset) (M. Faishol, 2014). Motor DC memiliki karakteristik-karakteristik variabel dan telah digunakan secara luas dalam pengendalian kecepatan. Motor DC dapat menyediakan sebuah torsi awal yang tinggi dan juga memungkinkan untuk mendapatkan berbagai kontrol kecepatan. Salah satu kontroler yang dapat menghilangkan *error steady state* adalah kontroler PID.

Kontroler PID adalah kontroler yang merupakan gabungan dari kontroler proposional, kontroler integral dan kontroler diferensial. Gabungan dari ketiga kontroler ini diharapkan agar mendapat keluaran sistem yang mempunyai settling time cepat, tidak ada *error*, tidak ada *overshoot* dan dapat digunakan pada *setpoint* yang berbeda-beda.

Dengan adanya modul motor DC dengan tipe D-6759 di Laboratorium Sistem Kontrol Fakultas Teknik Elektro Universitas Brawijaya Malang yang belum dapat digunakan sepenuhnya, maka dalam skripsi ini dilakukan pengembangan rangkaian dan sistem yang dapat digunakan untuk mengontrol kecepatan motor DC D-6759 menggunakan kontrol PID dengan respon yang termonitor secara *real time* dalam bentuk grafik sehingga mempermudah dalam mengetahui perubahan yang terjadi.

Pada penelitian sebelumnya (M. Faishol. 2014) metode yang digunakan untuk mencari parameter Kp, Ki, dan Kd nya menggunakan metode Ziegler Nichols Namun metode tersebut kurang tepat jika dipakai pada Motor DC karena motor DC hanya membutuhkan waktu singkat untuk mencapai *steady state* sehingga nilai parameter yang didapat tidak tepat sehingga diperlukan *tuning* lagi untuk mendapat nilai parameter yang tepat. Oleh karena itu, pada skripsi ini digunakan metode Root

Locus yang dianggap lebih tepat dan tidak memerlukan tuning untuk mendapatkan nilai parameternya.

Pada pengendali digital, pengendali menggunakan suatu rangkaian digital. Dalam banyak kasus, rangkaian digital dimaksud adalah suatu komputer, biasanya berbasiskan mikroprosesor atau mikrokontroler (pengendali-mikro). Komputer akan menjalankan program secara berulang-ulang (setiap perulangan disebut Iterasi atau scan). Program memerintahkan komputer untuk mengambil nilai set-point dan data hasil pengukuran dari sensor dan selanjutnya menggunakan angka-angka ini untuk menghitung keluaran pengendali (yang kemudian dikirim ke aktuator). Program kemudian akan mulai lagi dari awal dan melakukan proses yang sama. Satu siklus kerja untuk proses ini berlangsung dalam waktu kurang 1/1000 detik.

Metode *root locus*/letak kedudukan akar digunakan untuk meneliti perilaku sistem dengan parameter sistem berubah pada lingkup tertentu, misalnya perubahan parameter penguatan K. Di dalam analisis sistem, penguatan K dipilih sedemikian rupa agar sistem stabil serta memberikan respon yang baik. Rancangan dimaksudkan agar letak *pole* dan *zero* dari fungsi alih loop tertutup terletak pada daerah yang ditentukan. Agar sistem stabil, pole dan zero harus terletak pada bidang s sebelah kiri sumbu imajiner. Pada skripsi ini akan dibuat suatu desain pengendali digital yang digunakan sebagai alat pengendali kecepatan motor DC menggunakan Proposional Integral Diferensial (PID) dan dilengkapi dengan metode root locus untuk menentukan parameternya.

1.2 Rumusan Masalah

Berdasarkan uraian dalam latar belakang, didapatkan rumusan masalah sebagai berikut:

1. Bagaimana menemukan parameter PID menggunakan metode *root locus* pada pengendalian kecepatan motor DC D-6759 menggunakan Arduino mega 2560?
2. Bagaimana respon sistem pada saat diberikan gangguan?

1.3 Batasan Masalah

Karena luasnya objek pengkajian, maka perlu dilakukan pembatasan masalah agar pembahasan lebih terfokus pada rumusan masalah. Adapun batasan masalah pada skripsi ini antara lain:

1. Motor DC yang digunakan adalah motor DC D-6759 dengan catu 24 V, arus 0,47 A.
2. Menggunakan Sensor *Rotary Encoder* E50S8-360-3-N-24 dengan *range* pulsa 0 – 360 pulsa per rotasi.
3. Menggunakan Arduino Mega 2560.
4. Kinerja rangkaian driver dan elektronika tidak dibahas secara mendalam.
5. Gangguan yang diberikan berupa pemberian medan magnet pada piringan besi yang terkopel dengan motor DC.

1.4 Tujuan

Merancang parameter PID dengan metode *root locus* pada pengendalian kecepatan motor DC D-6759 menggunakan arduino mega 2560 yang bisa digunakan pada praktikum sistem kontrol modern pada laboratorium sistem kontrol.

1.5 Sistematika Pembahasan

Skripsi ini terdiri dari enam bab, dengan sistematika pembahasan sebagai berikut:

BAB I Pendahuluan

Bab ini menguraikan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika pembahasan.

BAB II Tinjauan Pustaka

Tinjauan pustaka membahas teori-teori yang menunjang dalam perancangan dan pembuatan alat. Pustaka yang diambil adalah pustaka yang relevan dan sesuai serta mendukung penelitian, seperti buku-buku ilmu pengaturan dan lain-lain. Selain dari buku pustaka juga akan diambil dari jurnal, internet, dan sumber pengetahuan yang lain.

BAB III Metode Penelitian

Bab ini membahas tentang metode yang digunakan dalam pengerjaan dan perencanaan alat.

BAB IV Perancangan dan Pembuatan Alat

Bab ini membahas tentang perancangan alat yang meliputi spesifikasi, perencanaan sistem dan diagram balok, prinsip kerja, dan pembuatan alat serta perancangan *hardware* dan *software*. Setelah itu, bagaimana penerapannya dalam sistem secara keseluruhan.

BAB V Pengujian dan Analisis Sistem

Bab ini membahas tentang hasil pengujian sistem yang sudah dibuat, serta analisis hasil yang diperoleh.

BAB VI Kesimpulan dan Saran

Dalam bab ini, semua hal yang sudah dikerjakan pada bab sebelumnya, dianalisis dan diambil kesimpulan. Serta rekomendasi dan saran untuk pengembangan alat selanjutnya

BAB II
TINJAUAN PUSTAKA

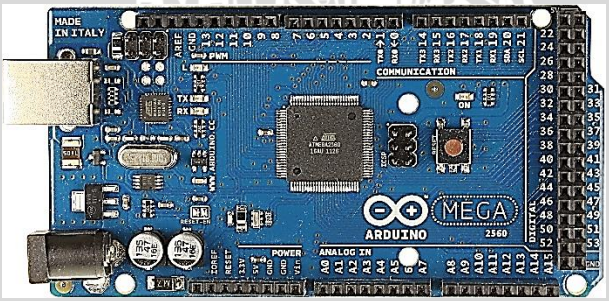
2.1 Pemodelan

Perancangan sistem kontrol loop tertutup memiliki beberapa tujuan yaitu untuk mencapai sistem yang stabil, memastikan bahwa nilai keluaran sistem dekat dengan nilai masukan sistem yang diinginkan, dan menyaring nilai *error* pengukuran dari sinyal kontrol (Johnson A., 1984).

Pemodelan sistem adalah penggambaran dalam bentuk lain dari sistem untuk mempermudah dalam menganalisisnya. Dengan merepresentasikan setiap blok sistem kedalam bentuk matematis akan dapat diketahui karakteristik suatu sistem dan parameter kontroler yang tepat untuk digunakan tanpa harus menguji berulang kali secara langsung pada alat.

2.2 Mikrokontroller Arduino Mega 2560

Arduino Mega 2560 adalah papan mikrokontroler berdasarkan ATmega328 (lihat Gambar 2.1). *Board* ini memiliki 54 digital *input/output* pin (14 pin dapat digunakan sebagai *output* PWM), 16 *input* analog, 16 MHz osilator kristal, USB koneksi, jack listrik, *header* ICSP, dan tombol reset.



Gambar 2.1 Arduino Mega 2560
Sumber: www.electroschematics.com

Arduino Mega 2560 dapat diaktifkan melalui koneksi USB atau dengan catu daya 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 *power*. Ujung kepala dari baterai dapat dimasukkan kedalam Gnd dan Vin pin *header* dari konektor power. Arduino dapat beroperasi dengan catu

daya eksternal 6 sampai 20 volt. Namun jika menggunakan lebih dari 12V, regulator tegangan bisa panas dan merusak papan. Kisaran yang disarankan adalah 7 sampai 12 volt.

ATmega 2560 memiliki 256 KB (dengan 8 KB digunakan untuk *bootloader*), 8KB dari SRAM dan 4 KB EEPROM. 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 volt. 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 interrupts*: 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.
- I^2C: 20 (SDA) dan 21 (SCL). Dukungan I^2C (TWI) komunikasi menggunakan *wire*.

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

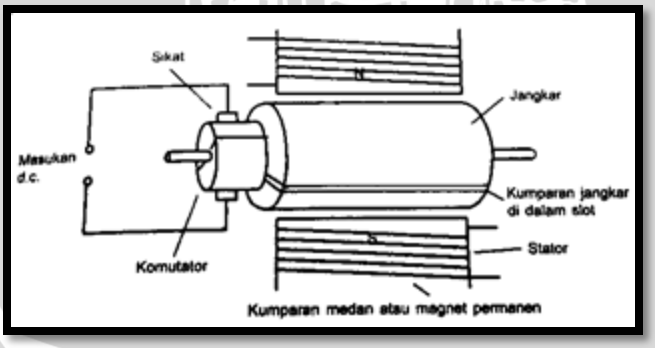
atau dari papan Arduino. LED RX dan TX di papan tulis akan berkedip ketika data sedang dikirim melalui chip USB-to-serial dengan koneksi USB ke komputer (tetapi tidak untuk komunikasi serial pada pin 0 dan 1).

Sebuah *Software Serial Library* memungkinkan untuk berkomunikasi secara serial pada salah satu pin digital pada board Arduino Mega 2560. Arduino Mega 2560 juga mendukung I^2C (TWI) dan komunikasi SPI. Perangkat lunak Arduino termasuk perpustakaan kawat untuk menyederhanakan penggunaan bus I^2C

2.3 Motor DC D-6759

Motor listrik sangat sering digunakan sebagai elemen kontrol akhir dalam sistem kontrol posisi dan kecepatan. Prinsip kerja dasar dari sebuah motor listrik adalah gaya yang bekerja pada konduktor yang berada di dalam suatu medan magnet ketika ada arus yang melewati konduktor tersebut (W. Bolton, 2004).

Motor DC memerlukan suplai tegangan yang searah pada kumparan medan untuk diubah menjadi energi mekanik. Bagian utama motor DC adalah stator (bagian yang tidak berputar) dan kumparan jangkar disebut rotor (bagian yang berputar). Catu tegangan DC dari 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 di antara medan magnet. Elemen-elemen dasar motor DC dijelaskan dalam Gambar 2.2.



Gambar 2.2 Elemen-Elemen Dasar Motor DC D-6759
Sumber: W.Bolton 2004

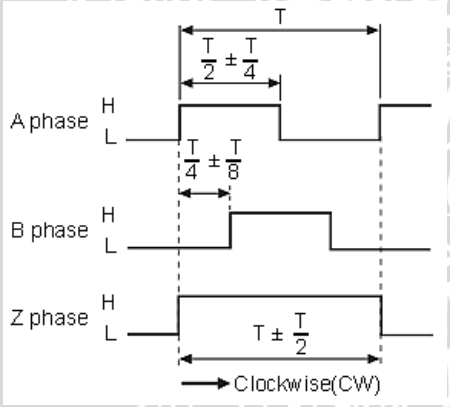
Motor DC memiliki prinsip kerja jika sikat arang terhubungkan dengan satu sumber arus serah di luar dengan tegangan *V*, maka satu arus *I* masuk ke terminal kumparan rotor dan menghasilkan *fluks*. Dengan adanya *fluks* stator dan arus rotor

akan menghasilkan satu gaya yang bekerja pada kumparan yang dikenal dengan gaya lorentz. Motor DC D-6759 ditunjukan dalam Gambar 2.3.

Gambar 2.3 Motor DC D-6759

2.4 *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 dalam Gambar 2.4 berikut.



Gambar 2.4 Sinyal keluaran Encoder
Sumber: Autonics E50 Series datasheet

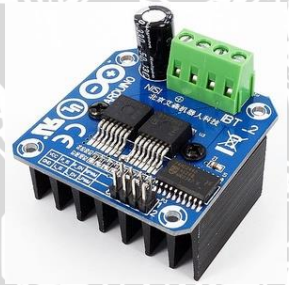
Untuk kebanyakan peralatan mesin motor atau aplikas *positioning*, sinyal Z dikenal sebagai *index signal* yang memilki peranan penting dalam menentukan *zero position* dengan cara memberikan sebuah pulas keluaran tungal per satu revolusi (Morris, S Alan). Sensor *Increment rotary encoder* Autonics E40 Series ditunjukan dalam Gambar 2.5.

Gambar 2.5 *Increment Rotary Encoder E50 series*
Sumber: Autonics E50 series datasheet

2.5 Driver Motor

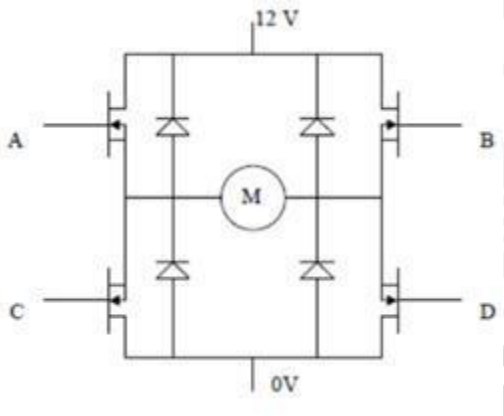
Driver motor berfungsi sebagai mengubah sinyal PWM dari mikrokontroler menjadi tegangan. Dalam aplikasinya *driver motor* biasanya tersusun dari rangkaian transistor-transistor yang tersusun sedemikian tupa sehingga mampu mengendalikan arah putar dan kecepatan motor berdasarkan arah loop dan tegangan kutub motor.

Driver yang akan digunakan adalah driver motor BTS7960 dengan metode H-bridge, gambar driver motor BTS7960 ditunjukkan dalam Gambar 2.6.



Gambar 2.6 Driver Motor H-Bridge BTS7960
Sumber: Driver Motor H-Bridge BTS7960 datasheet

H-bridge adalah sebuah perangkat keras berupa rangkaian yang berfungsi untuk menggerakkan motor. Rangkaian ini diberi nama *H-bridge* karena bentuk rangkaiannya yang menyerupai huruf H seperti pada Gambar 2.7 berikut.



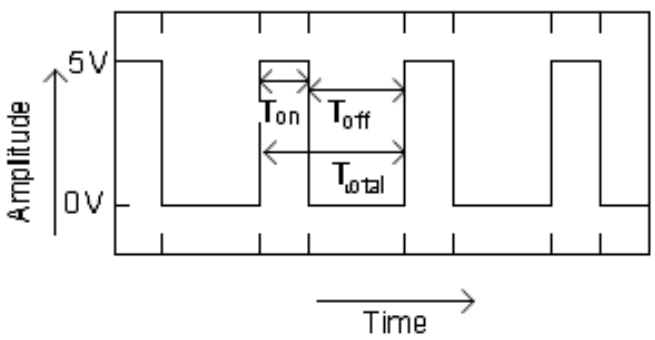
Gambar 2.7 Konfigurasi Driver Motor H-Bridge BTS7960
Sumber: Driver Motor H-Bridge BTS7960 datasheet

Rangkaian ini terdiri dari dua buah MOSFET kanal P dan dua buah MOSFET kanal N. Prinsip kerja rangkaian ini adalah dengan mengatur mati-hidupnya ke empat MOSFET tersebut. Huruf M pada gambar adalah motor DC yang akan dikendalikan. Bagian atas rangkaian akan dihubungkan dengan sumber daya kutub positif, sedangkan bagian bawah rangkaian akan dihubungkan dengan sumber daya kutub negatif. Pada saat MOSFET A dan MOSFET D *on* sedangkan MOSFET B dan MOSFET C *off*, maka sisi kiri dari gambar motor akan terhubung dengan kutub positif dari catu daya, sedangkan sisi sebelah kanan motor akan terhubung dengan kutub negatif dari catu daya sehingga motor akan bergerak searah jarum jam, jika kita melakukan proses sebaliknya maka arah putar akan berlawan arah jarum jam.

2.6 *Pulse Width Modulation* (PWM)

Pulse Width Modulation (PWM) adalah metode yang dapat digunakan untuk mengatur kecepatan dari 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%-10%. *Duty cycle* adalah besarnya sinyal kontrol yang diberikan pada motor. Persamaan untuk perhitungan *duty cycle* ditunjukkan pada persamaan 2.2 dengan Ton 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*

$$Duty\ Cycle = \frac{T_{on}}{T} x\ 100\% \quad (2.1)$$

dimana :

Ton = lebar pulsa saat logika tinggi

T = periode pulsa

Sedangkan frekuensinya dapat ditentukan dengan rumus sebagai berikut:

$$fOCn = \frac{fclk\ \ I/O}{N.256} \quad (2.2)$$

dimana:

$fOCn$ = frekuensi PWM

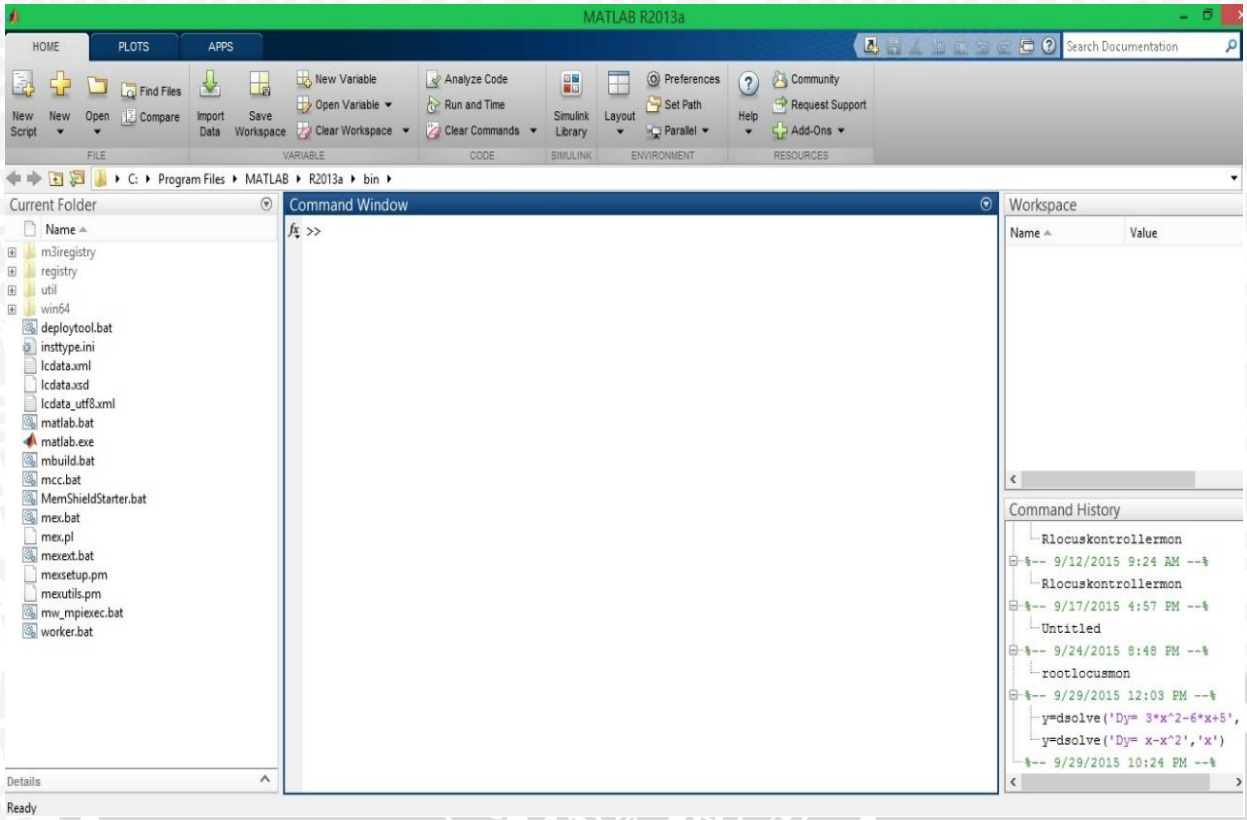
$fclk\ \ I/O$ = frekuensi I/O

$N.256$ = prescaler factor

2.7 Matlab R2013a

Matlab (matrix laboratory) adalah sebuah lingkungan komputasi numerical dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The Math Work, matlab memungkinkan manipulasi matriks, pemplotan fungsi dan data, implementasi algoritma, pembuatan antarmuka, dan peng-antarmuka-an dengan program dalam bahasa lain.

Dalam skrispi ini aplikasi matlab dipergunakan untuk mencari fungsi alih motor DC melalui data respon kecepatan motor yang diolah menggunakan beberapa fungsi yang terdapat di dalalam matlab. Serta dengan aplikasi matlab kita dapat melakukan simulink untuk mencari nilai kp, ki, kd yang tepat untuk digunakan..



Gambar 2.9 Tampilan Software Matlab R2013a

2.8 *Pseudo Random Binary Square*

Pseudo Random Binary Sequence (PRBS) adalah sinyal kotak yang termodulasi pada lebarnya dan berlangsung secara sekuensial. Sinyal ini biasanya dibangkitkan menggunakan *Linear Feedback Shift Register* (LFSR). Pada LFSR memiliki 2 parameter dasar yang menentukan sifat sekuensial yang dihasilkan, yaitu: panjang dari *shift register* dan susunan umpan balik. PRBS memiliki variasi panjang sekuensialnya tergantung dari panjangnya *shift register* seperti ditunjukkan dalam Tabel 2.1 berikut.

Tabel 2.1 Tabel Variasi Panjang Sekuensial PRBS
Sumber : Landau 2006

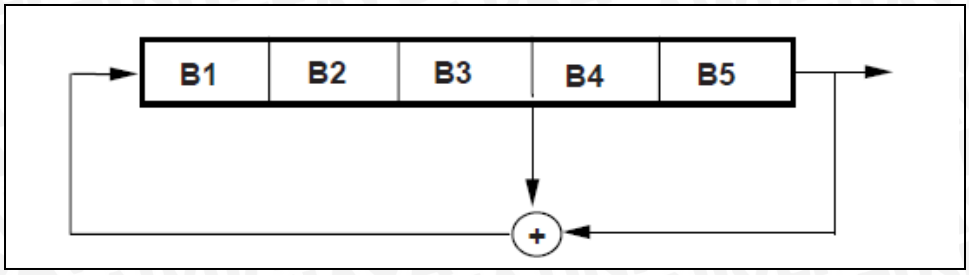
Panjang Register (N)	Panjang Sekuensial $L=2^N-1$	Posisi Tap Umpan Balik
2	3	1 dan 2
3	7	1 dan 3
4	15	3 dan 4
5	31	3 dan 5
6	63	5 dan 6
7	127	4 dan 7
8	255	2, 3, 4, dan 8
9	511	5 dan 9
10	1023	7 dan 10

Panjang dari shift register menentukan periode maksimum yang dapat dihasilkan dari sekuensial *Pseudo Random Binary Sequence* (PRBS) yang tidak berulang dan dapat dinyatakan dengan persamaan:

$$LPRBS=2n-1 \quad (2.3)$$

Dimana n adalah panjang dari register *linear feedback shift register* (LFSR). Panjang maksimum dari PRBS disebut M-sequence.

Panjang maksimum yang dapat dihasilkan PRBS untuk panjang tertentu dari *shift* register dapat dicapai dengan mempersiapkan konfigurasi dari feedback. Pada dasarnya ada 2 kemungkinan untuk realisasi *linear feedback shift register* (LFSR), yaitu: Fibonacci (*many to one*) dan Galois (*one to many*). Keduanya dapat didasarkan pada gerbang XOR atau XNOR menggunakan bermacam-macam angka dan kombinasi dari *feedback taps* keluaran dari *shift* register khusus. Dengan mengubah konfigurasi umpanbalik (jumlah tap dan posisinya) memungkinkan untuk menemukan M-sequence yang berbeda untuk panjang tertentu dari *shift* register (lihat Gambar 2.10).



Gambar 2.10 Register Geser 5 Bit dengan umpan balik
Sumber : Landau 2006

Prinsip pengujian proses dengan sinyal PRBS adalah membuat perubahan input kecil secara acak untuk membangkitkan gangguan (*perturbation*) yang kontinyu pada variabel *output*. Salah satu keuntungan penggunaan dari pendekatan ini adalah amplitudo perubahan input yang dibutuhkan dapat lebih kecil jika dibandingkan dengan perubahan *step* pada *step testing*. Selain itu, proses pengujian dapat dilakukan tanpa harus menunggu proses dalam keadaan tunak (*steadystate*). Jika pengujian dengan sinyal PRBS dilakukan, sinyal input secara teoritis disebut *white* (tidak berkorelasi) dan akan menghasilkan parameter model estimasi yang lebih baik. Frekuensi dari PRBS dapat dipilih untuk putaran (*flips*) cepat (*fast*) atau lambat (*slow*). Pemilihan frekuensi ini dapat menentukan jenis informasi terbaik yang akan didapat, misalnya untuk *fast* akan memberikan informasi yang akurat mengenai *dead time*, *slow* akan memberikan informasi *steady state gain* yang tepat sedangkan *medium* memberikan informasi *time constant* lebih baik.

2.9 Kontroller

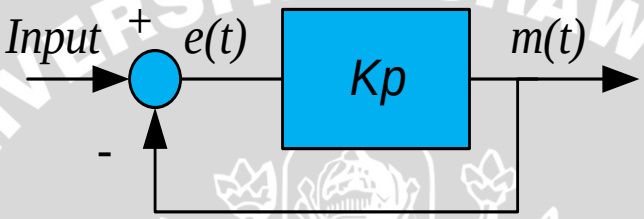
Kontroler dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadap perilaku sistem. Hal itu disebabkan oleh tidak dapat dirubahnya komponen penyusun sistem tersebut. Artinya, karakteristik plant harus diterima sebagaimana adanya, perubahan perilaku sistem hanya dapat dilakukan melalui penambahan kontroler.

Salah satu fungsi kontroler adalah mengurangi sinyal *error*, sinyal *error* adalah perbedaan nilai *setpoint* dengan nilai *output plant*. *Setpoint* adalah nilai referensi atau nilai yang diinginkan, sedangkan *output plant* adalah nilai aktual yang terukur pada *output plant*. Semakin kecil nilai sinyal *error* maka kinerja sistem kontrol dinilai semakin baik.

Prinsip kerja kontroler adalah membandingkan nilai *output plant* dengan nilai *setpoint*, menentukan nilai kesalahan dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan *error* (Ogata K, 1995).

2.9.1 Kontroller Proporsional (P)

Kontroler *proportional* memiliki *output* yang besarnya sebanding dengan besarnya sinyal *error*. *Output* kontroler merupakan perkalian antara penguatan proporsional dengan sinyal *error*. Gambar 2.11 menunjukan diagram blok kontroler *proporsional* dan Persamaan 2.4 menunjukan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.11 Diagram Blok Kontroller Proporsional
Sumber: K. Ogata, 1997

$$m(t) = Kp\, e(t) \tag{2.4}$$

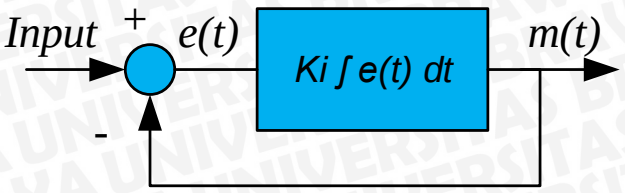
dimana:

Kp = adalah penguatan proporsional
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Penambahan Kp akan mempercepat kecepatan respon *transient* dan mengurangi kesalahan keadaan mantap.

2.9.2 Kontroller Integral (I)

Kontroler *integral* memiliki karakteristik seperti sebuah operasi *integral*, *output* kontroler dipengaruhi oleh perubahan yang sebanding dengan perubahan nilai sinyal *error*. *Output* kontroler merupakan penjumlahan terus menerus dari perubahan sinyal *error*. Gambar 2.12 menunjukan diagram blok kontroler *Integral* dan Persamaan 2.5 dan Persamaan 2.6 menunjukan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.12 Diagram Blok Kontroller Integral
Sumber: K. Ogata, 1997

$$\frac{dm(t)}{dt} = Ki\ e(t) \quad (2.5)$$

$$m(t) = Ki \int e(t)\ dt \quad (2.6)$$

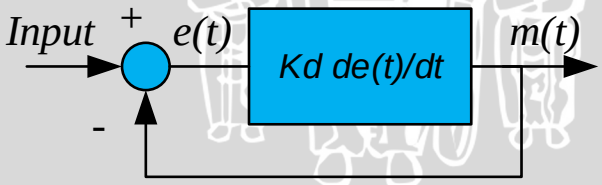
dimana:

Ki = adalah penguatan integral
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Aksi kontrol *integral* digunakan untuk menghilangkan sinyal *error* dalam keadaan mantap.

2.9.3 Kontroller Differensial (D)

Kontroler differensial memiliki sifat seperti suatu operasi differensial. Perubahan yang mendadak pada masukkan kontroler mengakibatkan perubahan yang sangat besar dan cepat. Kontroler ini tidak akan menghasilkan output saat sinyal *error* konstan sehingga tidak akan mempengaruhi keadaan mantap. Gambar 2.13 menunjukan diagram blok kontroler differensial dan Persamaan 2.7 menunjukan hubungan antara *output* kontroler dengan sinyal *error*.



Gambar 2.13 Diagram Blok Kontroller Differensial
Sumber: K. Ogata, 1997

$$m(t) = Kd\ \frac{de(t)}{dt} \quad (2.7)$$

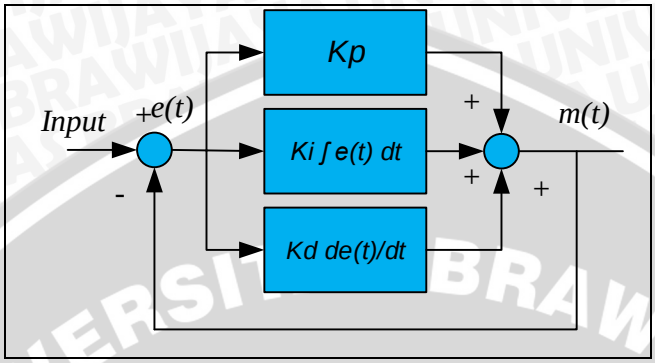
dimana:

Kd = adalah penguatan *derivative*
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

Kontroler ini digunakan untuk memperbaiki atau mempercepat respon *transient*.

2.9.4 Kontroller Proporsional Integral Differensial (PID)

Gabungan aksi kontrol *proportional*, *integral*, dan *derivative* yang terlihat dalam Gambar 2.14 mempunyai keunggulan dapat saling menutupi kekurangan dan kelebihan dari masing-masing kontroler. Persamaan kontroler PID ini dapat dinyatakan sebagai berikut (Persamaan 2.8):



Gambar 2.14 Diagram Blok Kontroller Proporsional Integral Differensial
Sumber: K. Ogata, 1997

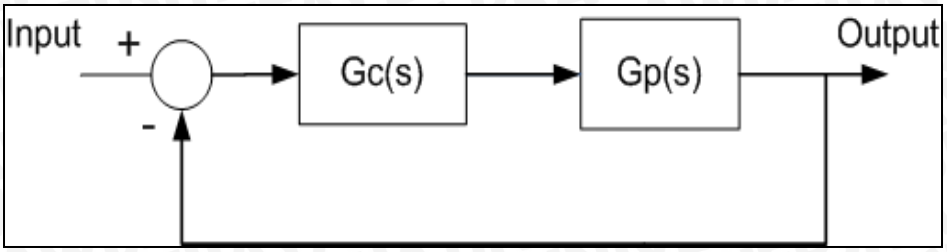
$$m(t) = Kp\,.e(t) + Ki.\int e(t)dt + Kd\,\frac{de(t)}{dt} \qquad (2.8)$$

dimana:

Kp = adalah penguatan *proportional*
Ki = adalah penguatan *integral*
Kd = adalah penguatan *derivative*
$e(t)$ = sinyal *error*
$m(t)$ = *output* kontroler

2.10 Metode Perancangan Kontroller Proporsional Integral Differensial (PID) Menggunakan Metode Root Locus

Rancangan sistem kendali loop tertutup menggunakan *root locus* memungkinkan untuk mengatur sekurang-kurangnya beberapa letak *pole* sistem *loop* tertutup sehingga dapat mengatur tanggapan transien pada tingkat tertentu dan pengaruhnya terhadap tanggapan keadaan mantap (Philips,1996:209). Prosedur analitis perancangan kontroler PID menggunakan metode *root locus* dapat dilihat dalam Gambar 2.15 berikut:



Gambar 2.15 Sistem Kendali Loop Tertutup
Sumber: K.Ogata, 1997

Untuk sistem tersebut, persamaan karakteristik dapat dilihat dalam persamaan 2.9:

$$1 + Gc(s)Gp(s) = 0 \tag{2.9}$$

Misalkan diinginkan lokus akar melalui $s = s_1$, maka hasil dari persamaan ditunjukkan pada persamaan 2.10:

$$\begin{aligned} &Gc(s_1)Gp(s_1) = -1 \\ &Gc(s_1)|Gp(s_1)|e^{j\psi} = 1e^{j\Pi} \end{aligned} \tag{2.10}$$

Fungsi alih kontroler PID setelah ditransformasi laplace dinyatakan oleh persamaan 2.11:

$$Gc(s) = Kp + \frac{Ki}{s} + Kd\,s \tag{2.11}$$

Perhitungan dari persamaan 2.9 ditunjukkan pada persamaan 2.12:

$$Gc(s_1) = \frac{1}{|Gp(s_1)|}e^{j(\Pi-\psi)} \tag{2.12}$$

Substitusi persamaan 2.12 pada persamaan 2.8 didapatkan persamaan 2.13:

$$Kd\,s_1^2 + Kp\,s_1 + Ki = \frac{e^{j(\Pi-\psi)}}{|Gp(s_1)|} \tag{2.13}$$

Dengan

$$s_1 = |s_1|e^{j\beta} \tag{2.14}$$

Hasil dari substitusi persamaan 2.13 ke persamaan 2.14, didapatkan pada persamaan 2.15:

$$\begin{aligned} &Kd|s_1|^2(\cos 2\beta + j\sin 2\beta) + Kp|s_1|(\cos\beta + j\sin\beta) + Ki \\ &= \frac{|s_1|}{|Gp(s_1)|}[\cos(\beta + \Pi - \psi) + j\sin(\beta + \Pi - \psi)] \end{aligned} \tag{2.15}$$

Menyamakan real dengan real dan imajiner dengan imajiner, didapatkan hasil pada persamaan 2.16:

$$\begin{bmatrix} |s_1|^2 & |s_1|\cos\beta \\ |s_1|^2 & |s_1|\sin\beta \end{bmatrix} \begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} \frac{|s_1|}{Gp(s_1)}\cos(\beta + \Pi + \psi) - Ki \\ \frac{|s_1|}{Gp(s_1)}\sin(\beta + \Pi + \psi) \end{bmatrix} \quad (2.16)$$

Atau dapat ditunjukkan pada persamaan 2.17:

$$\begin{bmatrix} |s_1|^2 & |s_1|\cos\beta \\ |s_1|^2 & |s_1|\sin\beta \end{bmatrix} \begin{bmatrix} Kd \\ Kp \end{bmatrix} = \begin{bmatrix} -\frac{|s_1|}{Gp(s_1)}\cos(\psi - \beta) - Ki \\ \frac{|s_1|}{Gp(s_1)}\sin(\psi - \beta) \end{bmatrix} \quad (2.17)$$

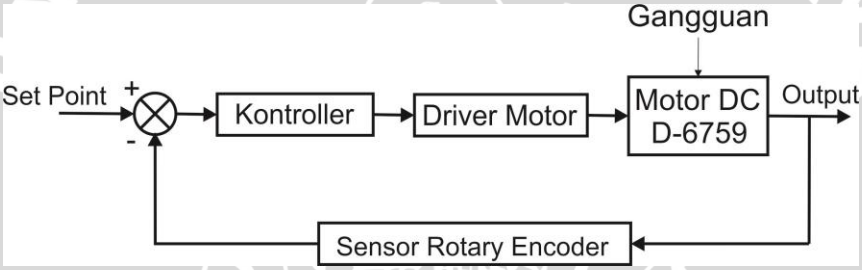
Dari persamaan dapat dilihat bahwa untuk perancangan kontroler PID, satu dari tiga penguatan Kp. Ki, Kd, harus ditentukan dahulu. Sedangkan untuk perancangan PI atau PD, penguatan yang sesuai pada persamaan dibuat sama dengan nol.

BAB III
METODE PENELITIAN

Metode penelitian pada dasarnya merupakan cara ilmiah untuk mendapatkan informasi dengan tujuan dan manfaat tertentu. Untuk menyelesaikan rumusan masalah dan merealisasi kan tujuan skripsi yang terdapat pada 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 perancanaan blok diagram sistem yang dapat menjelaskan sistem secara garis besar dan diharapkan alat dapat bekerja sesuai dengan rencana diagram blok sistem dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Blok Sistem loop tertutup

Keterangan :

1. *Setpoint* sistem berupa putaran motor yang memilliki empat nilai yaitu 0 rpm, 150 rpm, 250 rpm, dan 350 rpm.
2. Kontroler yang digunakan adalah kontroler *Proporsional Integral Differensial* (PID) menggunakan perangkat keras Arduino Mega 2560.
3. *Driver* motor menggunakan *H-Bridge* BTS7960.
4. Aktuator yang digunakan adalah motor DC D-6759 24 volt.
5. Sensor *Rotary Encoder* sebagai pembaca putaran motor digunakan untuk *feedback* sistem.

   Gangguan pada motor berupa magnet U yang ditempelkan pada piringan besi pada Motor DC D-6759

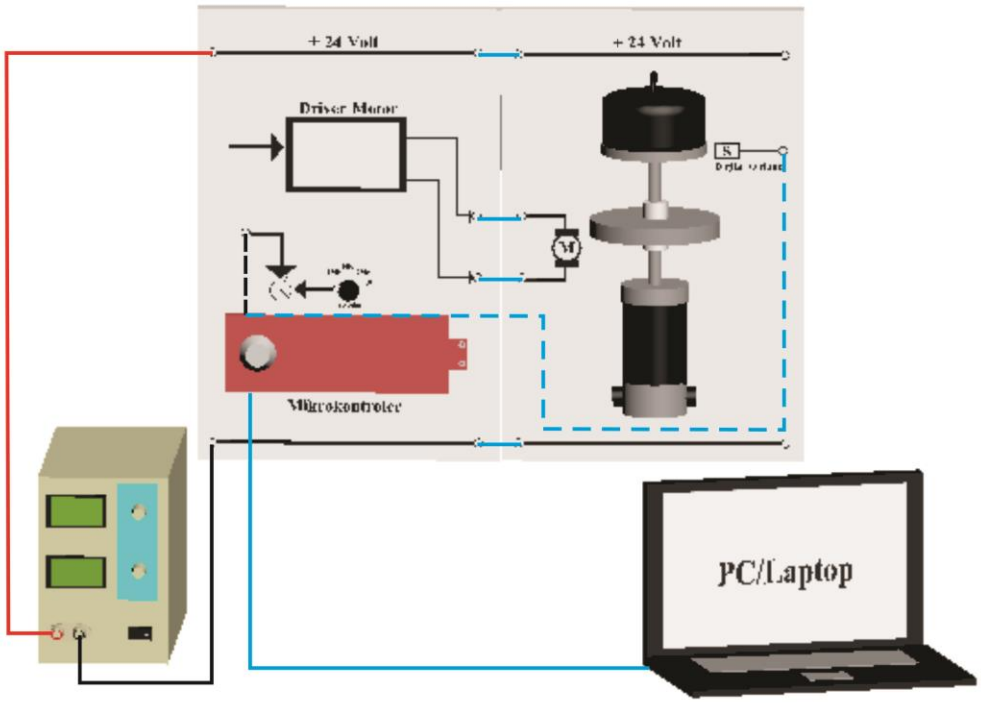
3.2 Pembuatan Perangkat Keras

Pembuatan perangkat keras dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem beserta pemrogramannya, hal ini dimaksudkan agar

perancangan parameter PID pada pengendalian kecepatan motor DC D-6579 dengan menggunakan metode *root locus* 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) tipe AD-8723.
    - Mikrokontroler Arduino Mega 2560.
    - *Driver* motor DC D-6759 (*Dual H- Bridge*).
    - *Rotary encoder* Autonics E40S8-1024-3-N-24.
3. Perancangan Sistem meliputi:
    - Modelling Motor DC D-6579.
    - Modelling Sensor *Rotary encoder* Autonics E40S8-1024-3-N-24.
    - Algoritma pengendali *root locus*.
    - Pembuatan *flow chart*.
    - Verifikasi Sistem meliputi respon, *error steady state*, *time settling*, dan *rise time*.



Gambar 3.2 Skema pembuatan perangkat keras

3.3 Pengujian Alat

Setelah semua komponen pada alat sudah terhubung sesuai dengan diagram blok sistem yang telah dirancang dan perangkat lunak (*software*) untuk mendukung sistem yang telah dibuat, maka diadakan pengujian dan analisis alat.

Pengujian dan analisis yang dilakukan adalah sebagai berikut:

1. Pengujian sistem pada tiap-tiap blok
2. Menggabungkan sistem dari beberapa blok menjadi keseluruhan sistem
3. Mengevaluasi hasil pengujian Sistem Keseluruhan

3.4 Pengambilan Kesimpulan dan Saran

Kesimpulan diambil berdasarkan data yang didapat dari hasil pengujian sistem secara keseluruhan. Apabila hasil yang didapatkan sesuai dengan apa yang diharapkan sebelumnya, maka sistem kendali tersebut sudah berhasil memenuhi

tujuan awal dan selanjutnya dapat dimasukkan ke dalam saran guna dikembangkan untuk disempurnakan di penelitian selanjutnya.

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, kontrol digital dan *flowchart* program.

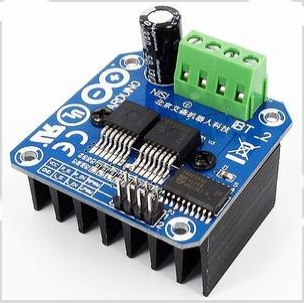
4.1 Spesifikasi Alat

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

1. Komputer atau PC yang sudah terdapat *software* arduino.
2. *Power Supply Unit* (PSU) yang digunakan memiliki range 0-24 Volt (lihat Gambar 4.1).
3. Perangkat Kontroler yang digunakan adalah Arduino Mega 2560 (lihat Gambar 4.2).
4. *Driver* motor yang digunakan adalah *H-Bridge* BTS7960 (lihat Gambar 4.3).
5. Motor DC D-6759 dengan catu daya maksimal 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 Arduino Mega 2560



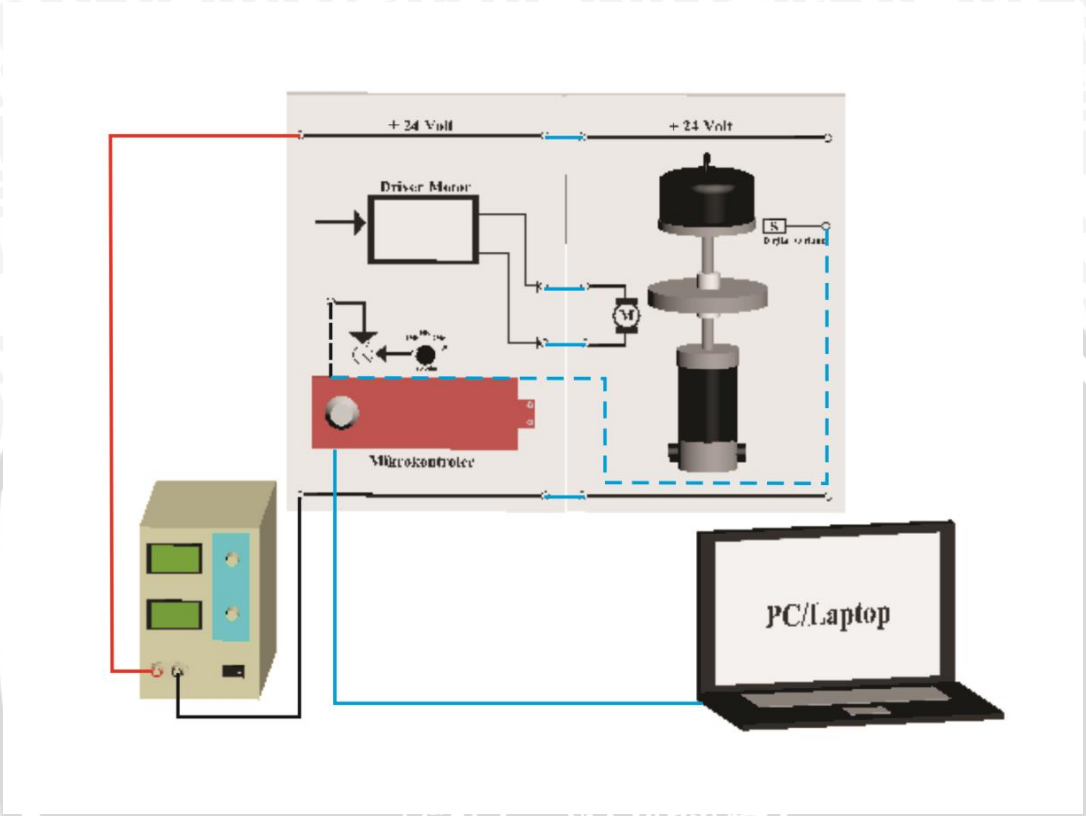
Gambar 4.3 Driver Motor H-Bridge BTS7960

Gambar 4.4 Motor DC D-6759

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

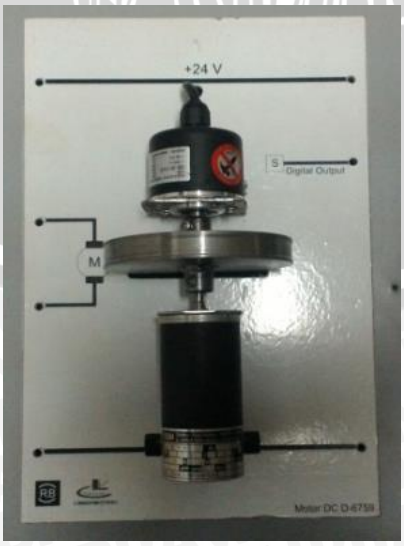
4.2 Pembuatan Perangkat Keras

Skema pembuatan alat dan pembuatan modul untuk pengontrolan kecepatan motor DC D-6759 dengan menggunakan metode root locus ditunjukkan dalam Gambar 4.6.



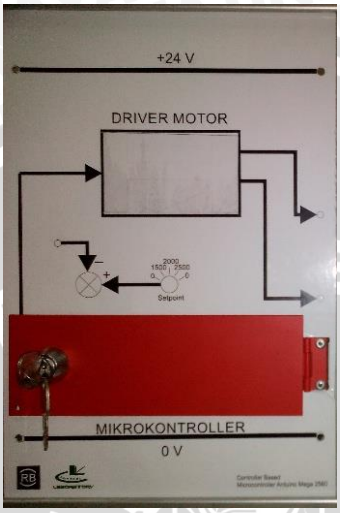
Gambar 4.6 Skema pembuatan perangkat keras

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 Pembuatan modul Motor DC D-6759 dan *Rotary Encoder* E50 Series

Pembuatan modul Mikrokontroler Arduino Mega 2560 yang terhubung dengan *driver motor H-Bridge* BTS7960 dapat dilihat pada Gambar 4.8.



Gambar 4.8 Pembuatan modul Arduino Mega 2560 dan *Driver* motor.

4.3 Prinsip Kerja Sistem

Prinsip kerja sistem berikut 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* motor H-Bridge BTS7960. *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).

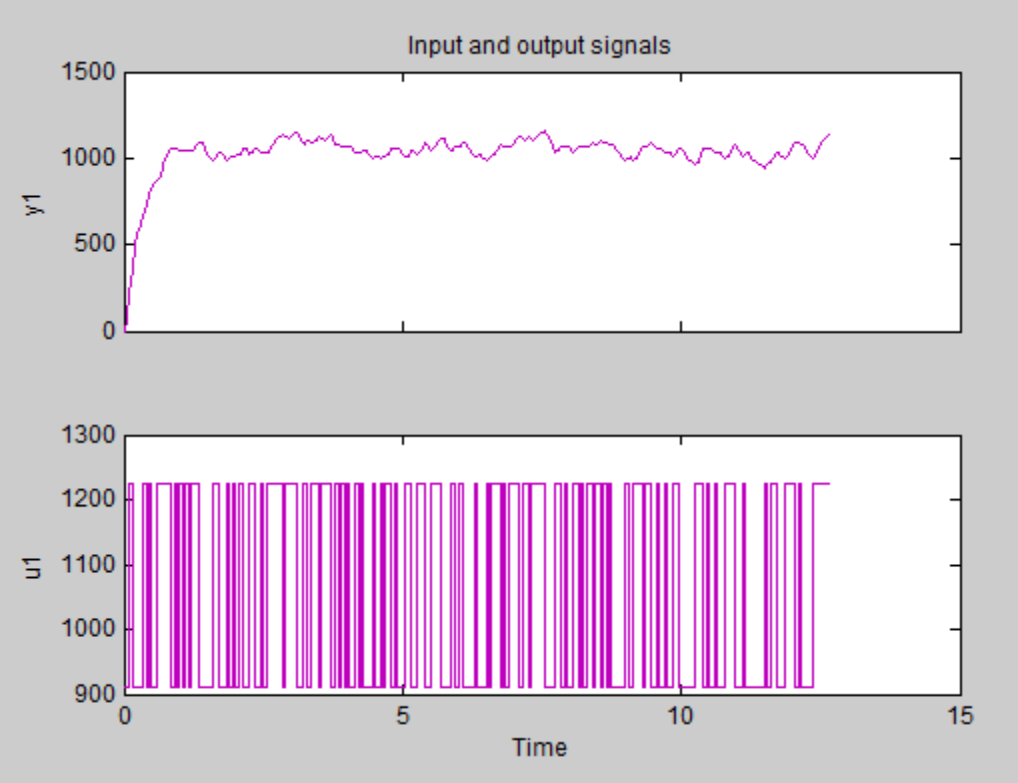
8. Mencari fungsi alih motor DC D-6759 dengan memberikan sinyal *Pseudo Random Binary Sequence* (PRBS).
9. Mencari parameter kontroler PID menggunakan metode *root locus*.
10. Menguji parameter kontroler PID yang didapat dalam simulasi *simulink* Matlab.
11. Membandingkan respon sistem *simulink* dengan respon sistem motor DC D-6759 yang didapat.
12. Mengimplementasikan hasil desain perancanan pada sistem.

4.4 Penentuan Fungsi Alih Motor DC D-6759

Untuk mengendalikan motor DC D-6759 digunakan 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*. Untuk mendapatkan karakteristik motor DC D-6759 pada perancangan ini diberikan masukan unit step.

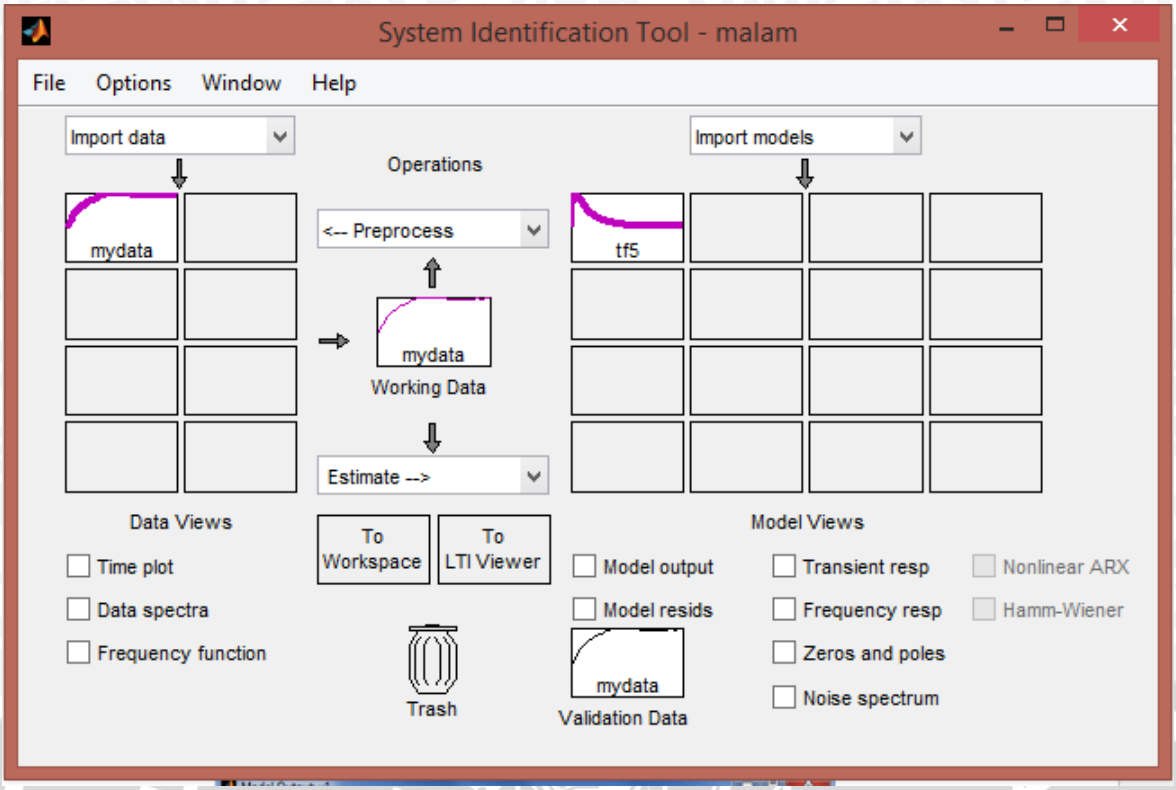
Untuk mendapatkan fungsi alih dari motor dilakukan pemodelan dengan cara membangkitkan sinyal *Pseudo Random Binary Sequence* (PRBS). Langkah yang dilakukan untuk membangkitkan sinyal PRBS adalah sebagai berikut :

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 4.9), selanjutnya adalah melakukan identifikasi dengan menggunakan *software* Matlab.

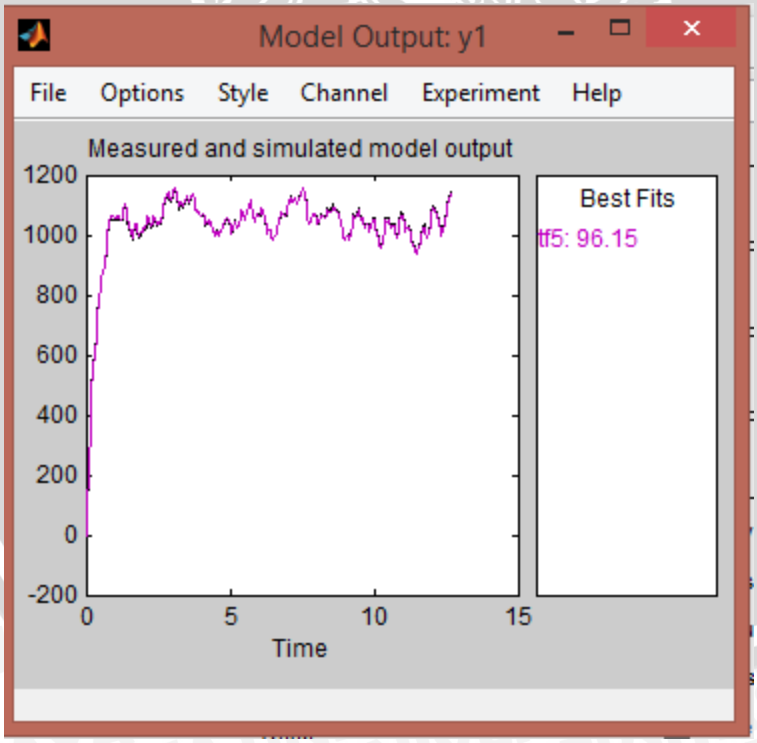


Gambar 4.9 Respon sinyal PRBS dan kecepatan motor DC D-6759

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 4.10). Setelah melakukan beberapa estimasi model berdasarkan data yang telah di *import* didapatkan fungsi alih dari motor dengan *best fit* sebesar 94.45 (lihat Gambar 4.11).



Gambar 4.10 *System Identification Toolbox*



Gambar 4.11 Hasil estimasi model

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

$$\frac{Y(s)}{U(s)} = \frac{140.5}{s^2 + 41.79s + 142.4} \quad (4.1)$$

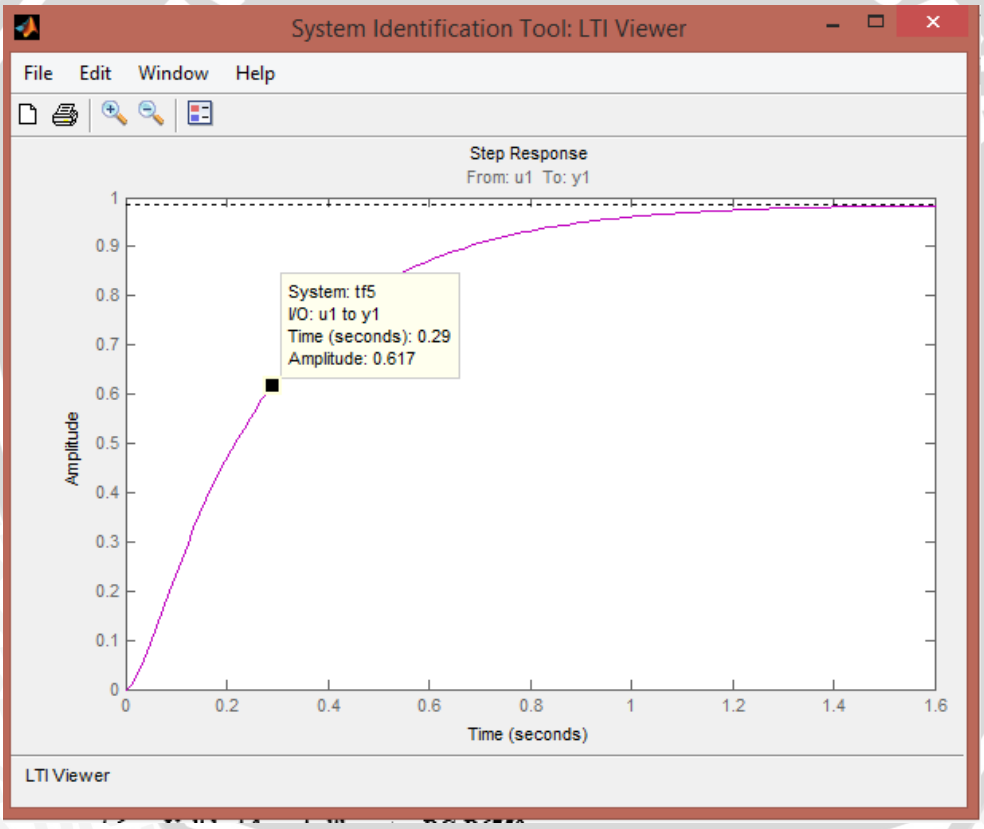
$$F(s) = \frac{140.5}{S^2 + 41.79s + 142.4} = \frac{{\omega_n}^2}{s^2 + 2\xi\omega_n s + {\omega_n}^2} \quad (4.2)$$

$$2\xi\omega_n = 41.79 \quad (4.3)$$

$$\omega_n = \sqrt{142.4} = 11.93 \quad (4.4)$$

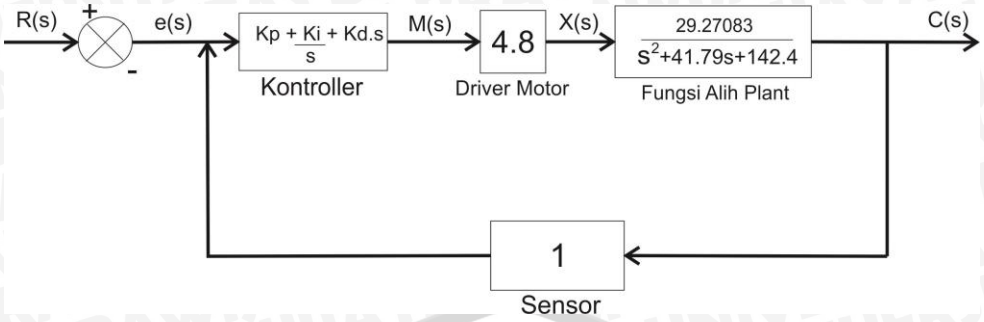
$$\xi = \frac{41.79}{2 \text{ x } 11.93} = 1.7514 \quad (4.5)$$

Dengan memberikan masukan *unit step* pada program Matlab didapatkan respon dalam Gambar 4.12, nilai *time constant* fungsi alih merupakan waktu yang dibutuhkan respon untuk mencapai 63,2% dari nilai *steady state* yaitu 0.29 seconds.



Gambar 4.12 Respon fungsi alih motor DC D-6759 dengan masukan *unit step*.

Sedangkan untuk fungsi alih sistem secara keseluruhan adalah sebagai berikut :



Gambar 4.13 Diagram Blok Sistem keseluruhan

$$\text{Fungsi Alih Plant} = \frac{29.27083}{s^2+41.79s+142.4}$$

$$\text{Fungsi Alih Kontroler} = \frac{2.2386s+8+0.0345s^2}{s}$$

$$\text{Fungsi Alih Driver } K = \frac{span\ out}{span\ in} = \frac{0-24v}{0-5v} = 4.8$$

$$\frac{C_{(s)}}{e_{(s)}} = \frac{4.84725s^2 + 327.1683s + 1.124}{s^3 + 41.79s^2 + 142.4s}$$

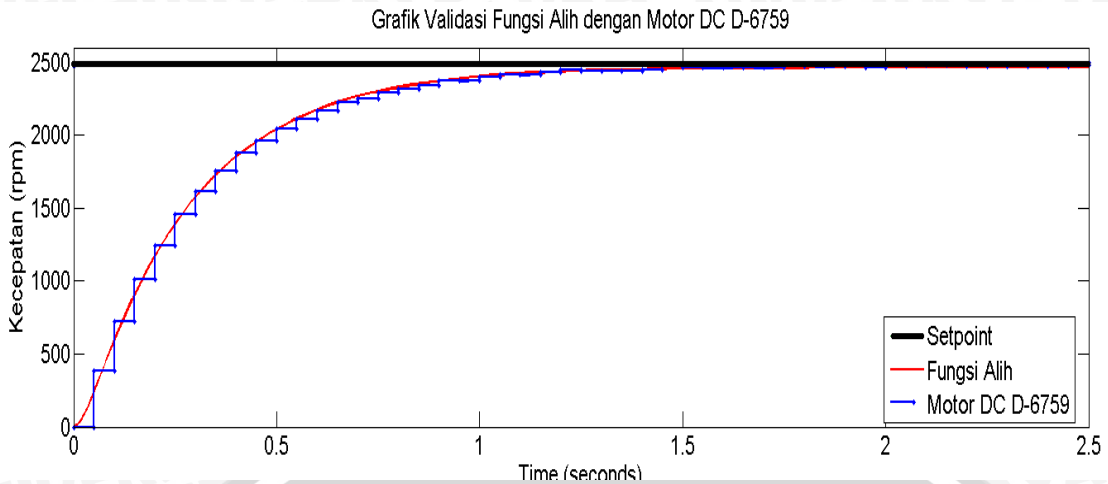
$$\frac{C_{(s)}}{R_{(s)}} = \frac{\frac{4.84725s^2 + 327.1683s + 1.124}{s^3 + 41.79s^2 + 142.4s}}{1 + \frac{4.84725s^2 + 327.1683s + 1.124}{s^3 + 41.79s^2 + 142.4s}}$$

$$\frac{C_{(s)}}{R_{(s)}} = \frac{4.84725s^2 + 327.1683s + 1.124}{s^3 + 41.79s^2 + 142.4s + 4.84725s^2 + 327.1683s + 1.124}$$

$$\frac{C_{(s)}}{R_{(s)}} = \frac{4.84725s^2 + 327.1683s + 1.124}{s^3 + 46.63725s^2 + 470.5683s + 1.124}$$

4.5 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 4.14).



Gambar 4. 14 Validasi fungsi alih dengan respon motor DC D-6759

Dari Gambar 4.14 dapat dilihat bahwa respon 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.

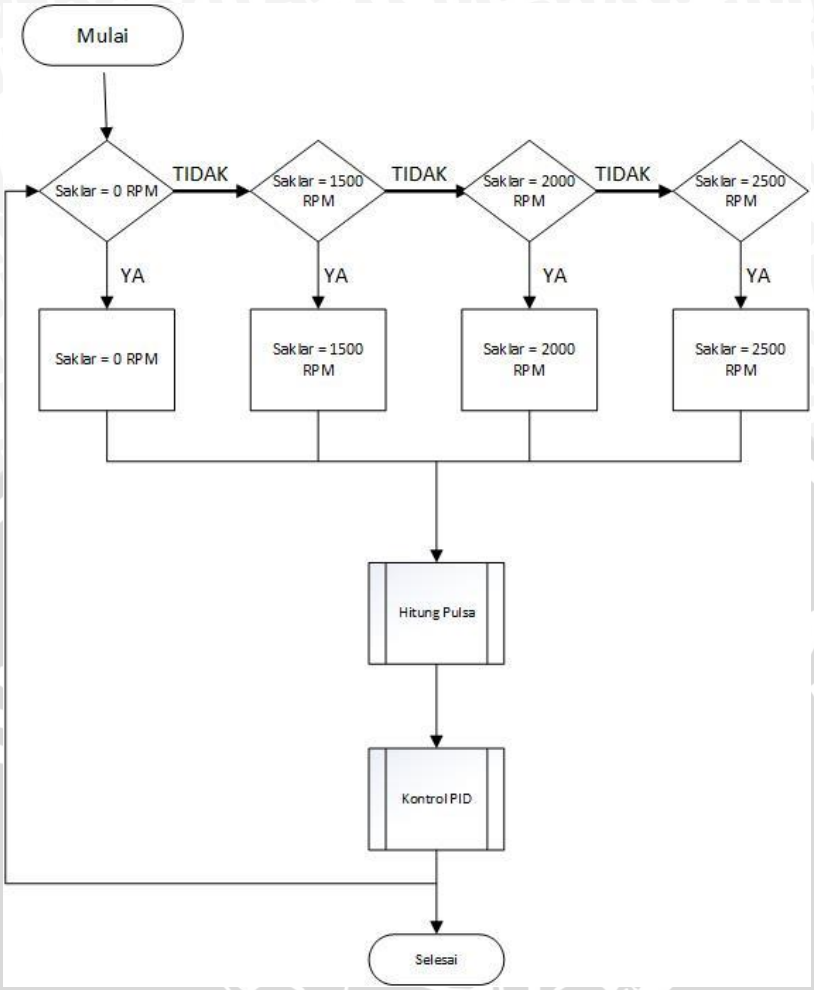
4.6 Perancangan Perangkat Lunak (*Software*)

Perancangan perangkat lunak terbagi menjadi 2 bagian, yaitu:

1. *Flowchart* sistem secara keseluruhan
2. *Flowchart* hitung pulsa.
3. *Flowchart* PID.

4.6.1 Flowchart Sistem Secara Keseluruhan

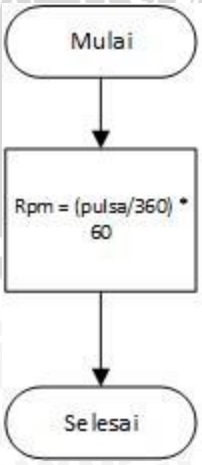
Flowchart keseluruhan sistem ditunjukkan dalam Gambar 4.15



Gambar 4. 15 Flowchart sistem secara keseluruhan

4.6.2 Flowchart Hitung Pulsa

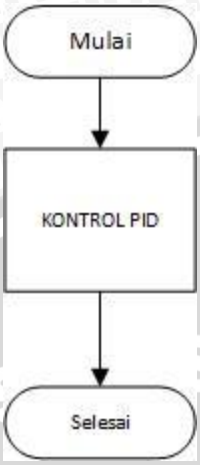
Flowchart perhitungan pulsa kecepatan putar motor dapat dilihat pada Gambar 4.16



Gambar 4.16 *Flowchart* Perhitungan Pulsa Kecepatan Putar Motor.

4.6.3 Flowchart *Flowchart* Kontrol Proporsional Integral Differensial (PID).

Flowchart Kontrol Proporsional Integral Differensial (PID) dapat dilihat pada Gambar 4.17.

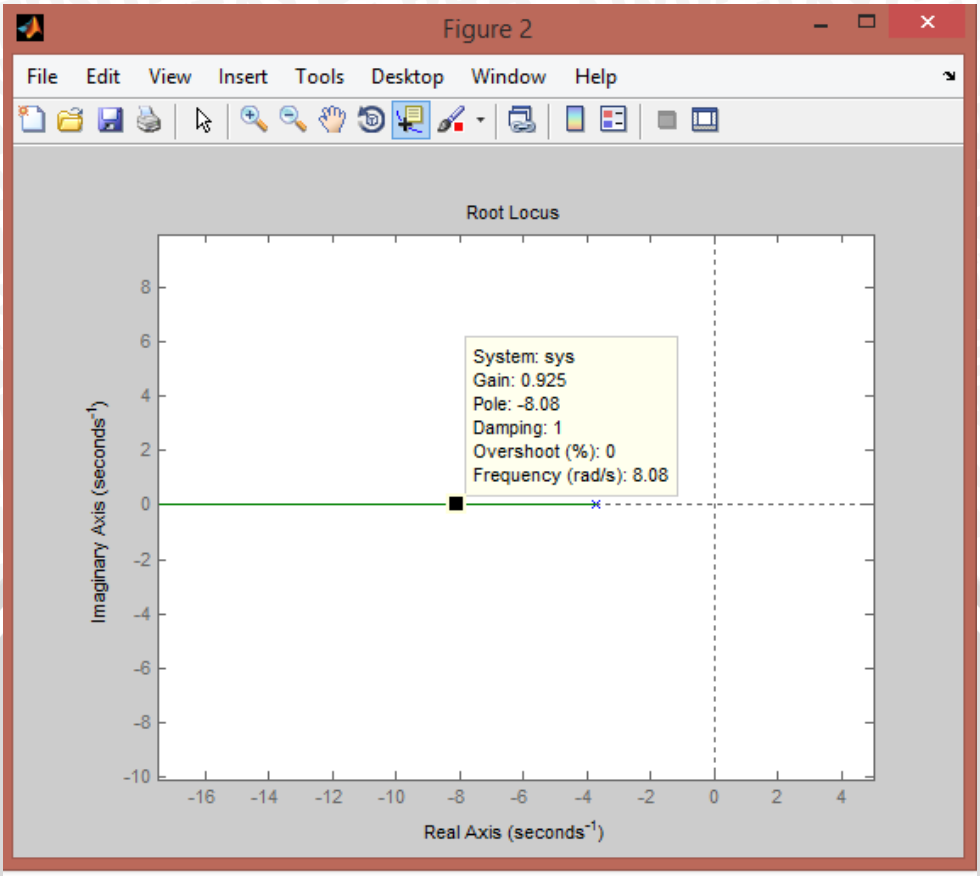


Gambar 4.17 *Flowchart* Kontrol Proporsional Integral Differensial (PID).

4.7 Penentuan Parameter Kontroller PID

Untuk memenuhi tujuan performansi loop yang diinginkan, maka perlu ditambahkan kontroler pada sistem tersebut. Kontroler yang dipilih ialah kontroler Proporsional Integral Differensial (PID). Setelah didapatkan fungsi alih sistem yaitu $F(s) = \frac{140.5}{s^2+41.79s+142.4}$. Selanjutnya adalah menentukan letak simpul *loop* tertutup.

Berdasarkan fungsi alih *loop* tertutup dapat diketahui sistem berorde dua. Nilai parameter PID ditentukan oleh pemilihan *pole* pada diagram *root locus*. Pada penelitian ini digunakan s_1= -8.08. Penentuan letak *pole* pada diagram *root locus* terlihat dalam Gambar 4.18.



Gambar 4.18 Letak *Pole* Pada Diagram *Root Locus* (Perancangan)

Setelah ditentukan letak *pole* yang diinginkan kemudian dengan mensubstitusi nilai s_1 dan nilai fungsi alih sistem dalam Persamaan 2.16 dan memvariasikan nilai Ki akan didapatkan parameter PID dalam Tabel 4.1. Pencarian parameter Kp, Ki, dan Kd dalam persamaan 2.17 dengan menggunakan MATLAB 2013 ditunjukkan pada *listing* program berikut :

```
%nilai pole yang ditentukan dari Gambarroot locus

s1=-8.08
KI=[1 2 3 5 8]

plant_num=[0 0 140.5];
plant_den=[1 41.79 142.4];

s1mag = abs(s1)
beta = angle(s1)
plant_a1 = polyval(plant_num,s1)/polyval(plant_den,s1);
plants1mag = abs(plant_a1)
psi = angle(plant_a1)
t=0:1:20:300;

for k =1:5
```

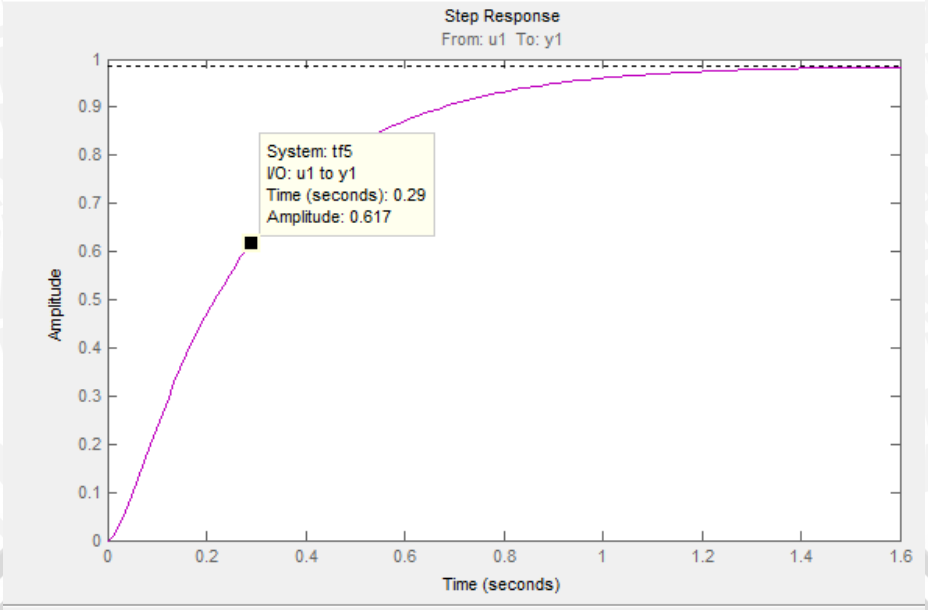
```
    KP=-sin(beta+psi)/(plants1mag*sin(beta))-
2*KI(k)*cos(beta)/s1mag
    nilai_KI= KI(k)
    KD = sin(psi)/(s1mag*plants1mag*sin(beta))+KI(k)/s1mag^2

    Gcnum = [KD KP KI(k)];
    Gcden = [0 1 0];

    Tnum = conv(plant_num,Gcnum);
    Tden = conv(plant_den,Gcden)+conv(plant_num,Gcnum);

    r = roots(Tden)
    step (Tnum,Tden,t)
    %step (plant_num,plant_den,t)
    hold on
end
hold off
figure, rlocus(Tnum,Tden)
```

Hasil pencarian parameter Kp, Ki dan Kd dari perhitungan pada program diatas ditunjukkan dalam Tabel 4.3.

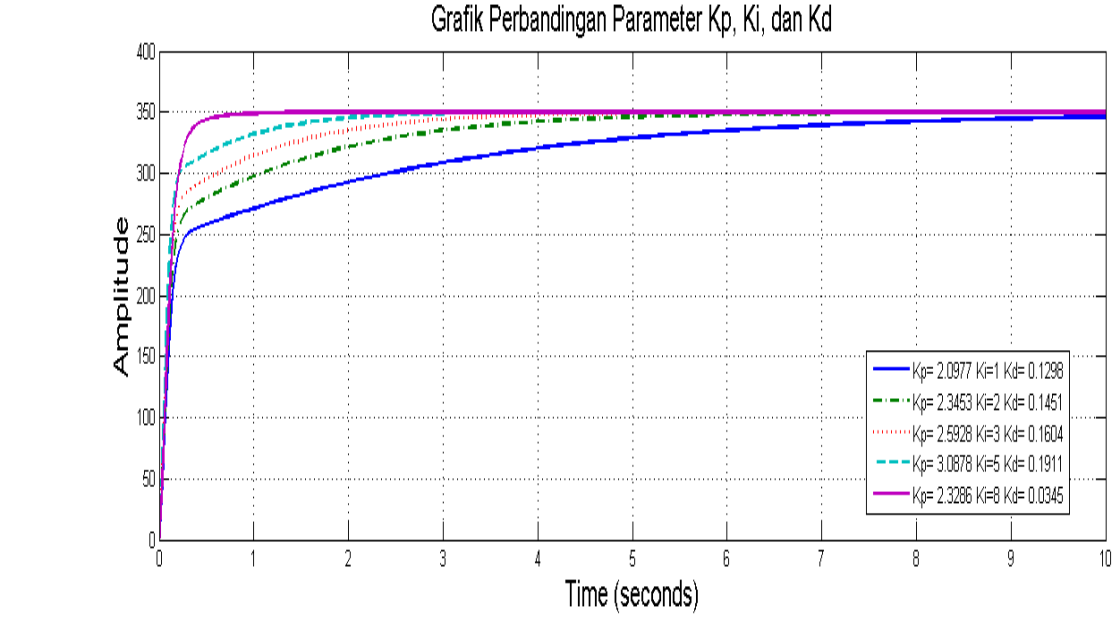
Tabel 4.1 Parameter PID dengan s_1= -8.08

No.	Kp	Ki	Kd	*Pole* 1	*Pole* 2	*Pole* 3
1	2.0977	1	0.1298	-51.6114	-8.08	-0.3369
2	2.3453	2	0.1451	-53.4497	-8.08	-0.6507
3	2.5928	3	0.1604	-55.3092	-8.08	-0.9432
4	3.0878	5	0.1911	-59.0850	-8.08	-1.4715
5	2.3286	8	0.0345	-64.8682	-8.08	-2.1445

Setelah didapatkan nilai Kp, Ki dan Kd hasil perhitungan, kemudian dilakukan pengujian terhadap sistem, dan parameter yang sesuai dengan sistem.



Gambar 4.19 Grafik respon sistem tanpa controller PID



Gambar 4.20 Grafik respon sistem dengan controller PID

Dari gambar 4.20 dapat diketahui bahwa respon sistem tanpa menggunakan PID lebih lama menuju steady state dan melebihi setpoint yang diinginkan. Dengan digunakannya parameter PID hasil tuning didapatkan respon yang lebih cepat dari pada respon tanpa menggunakan PID, serta dapat mencapai setpoint yang diinginkan seperti tertera dalam Gambar 4.19.

Dari 5 jenis parameter PID yang didapat dipilih nilai PID yang memiliki respon terbaik yaitu: Kp = 2.3286, Ki = 8, Kd = 0.0345.

BAB V
PENGUJIAN DAN ANALISIS SISTEM

Pengujian dan analisis sistem dilakukan untuk mengetahui apakah sistem telah bekerja sesuai dengan perancangan. Pengujian yang dilakukan meliputi pengujian tiap-tiap blok maupun pengujian secara keseluruhan. Pengujian setiap blok ini dilakukan untuk menemukan letak kesalahan dan mempermudah analisis pada sistem apabila alat tidak bekerja sesuai dengan perancangan. Pengujian dibagi menjadi beberapa bagian, yaitu:

1. Pengujian motor DC D-6759.
2. Pengujian *driver* motor H-Bridge BTS7960.
3. Pengujian *rotary encoder* E50S8-360-3-N-24.
4. Pengujian keseluruhan sistem.
5. Membandingkan antara hasil pengujian keseluruhan pada aplikasi matlab dengan hasil implementasi pada alat.

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.

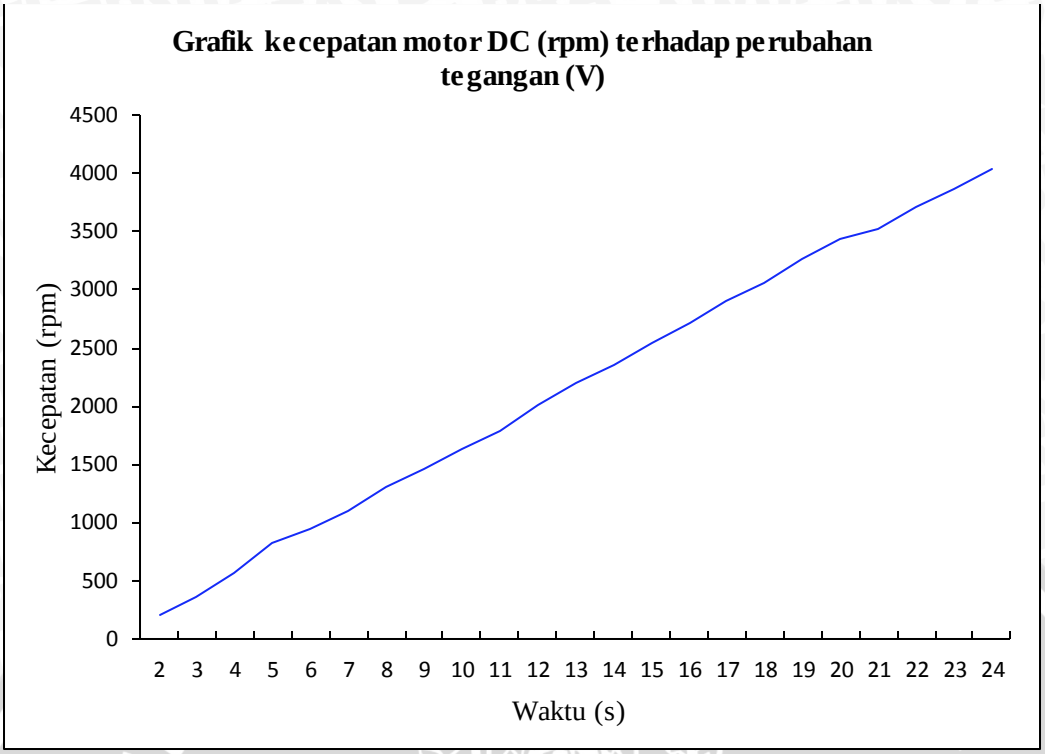
d. Hasil Pengujian

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 H-Bridge BTS7960

a. Tujuan

Mengetahui kinerja dan respon rangkaian *driver* motor H-Bridge BTS7960 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.
3. Driver Motor H-Bridge BTS7960.
4. Multimeter.
5. Arduino Mega 2560
6. Kabel penghubung.

c. Langkah pengujian

1. Hubungkan *output* tegangan *Power Supply Unit* (PSU) pada *input* tegangan referensi *driver* motor H-Bridge BTS7960.
2. Hubungkan *input* tegangan *driver* motor H-Bridge BTS7960 dengan pin *output* PWM di Arduino Mega 2560.

3. Hubungkan *output* tegangan *driver* motor H-Bridge BTS7960 dengan multimeter.
4. Atur *duty cycle* sinyal PWM pada Arduino Mega 2560 dengan nilai 0%-100%.
5. Amati dan catat hasil pembacaan multimeter disetiap kenaikan 5%.

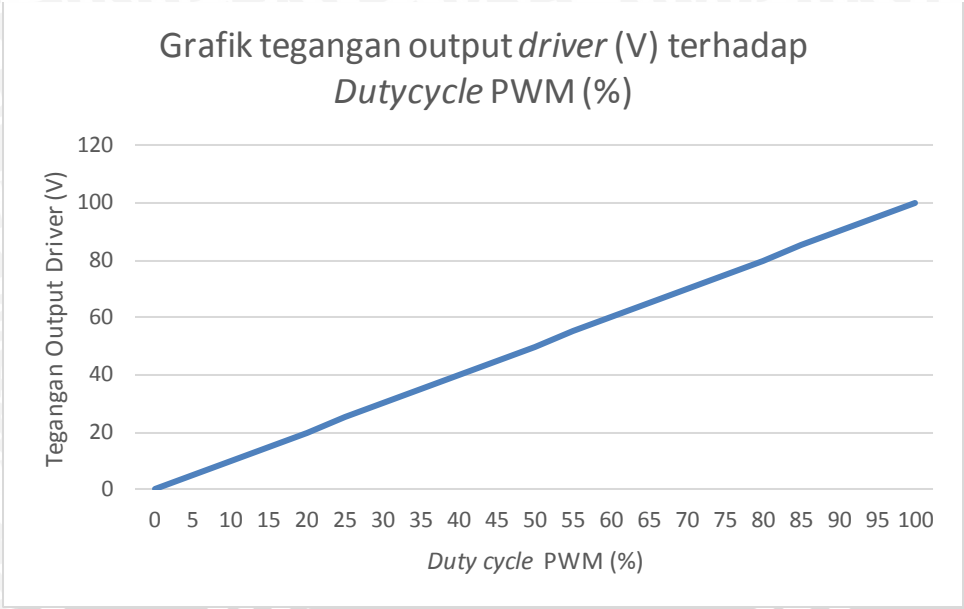
d. Hasil pengujian

Data pengujian *driver* motor H-Bridge BTS7960 ditunjukkan dalam Tabel 5.2.

Tabel 5.2 Data pengujian *driver* motor H-Bridge BTS7960

Duty Cycle (%)	Output Driver (V) dengan Multimeter
0	0
5	1.126
10	2.354
15	3.576
20	4.75
25	5.87
30	7.09
35	8.31
40	9.53
45	10.65
50	11.87
55	13.1
60	14.32
65	15.45
70	16.68
75	17.91
80	19.14
85	20.26
90	21.49
95	22.72
100	23.96

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 linier sehingga pemilihan range kerja sistem dapat bebas dipilih.

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 H-Bridge BTS7960.
5. Motor DC D-6759.
6. Arduino Mega 2560.
7. Kabel penghubung.

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 H-Bridge BTS7960.
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%.

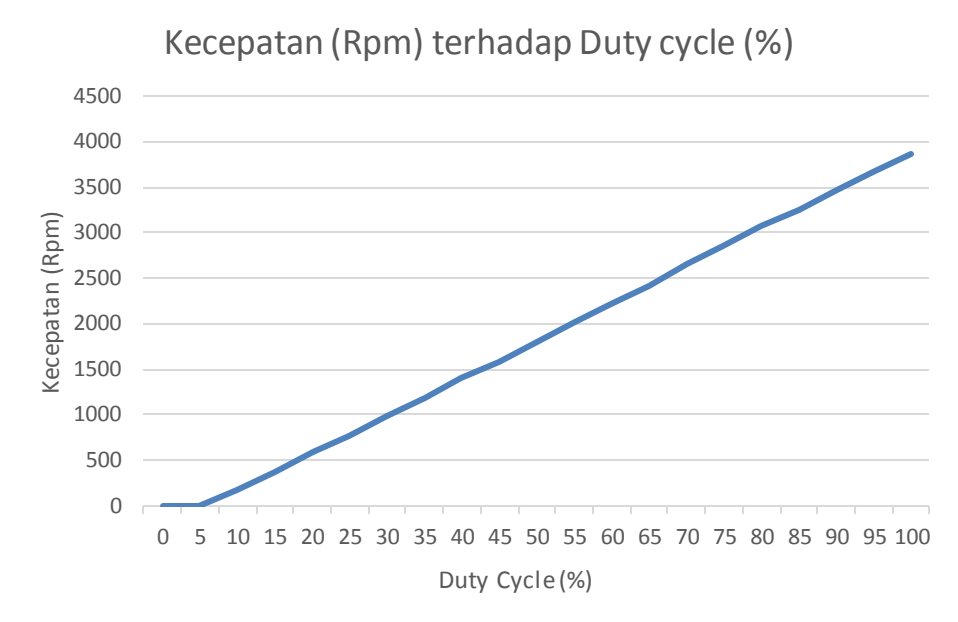
d. Hasil pengujian

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	166
15	380
20	583
25	776
30	976
35	1186
40	1393
45	1590
50	1810
55	2016
60	2226
65	2423
70	2646
75	2856
80	3070
85	3256
90	3480
95	3676
100	3863

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*

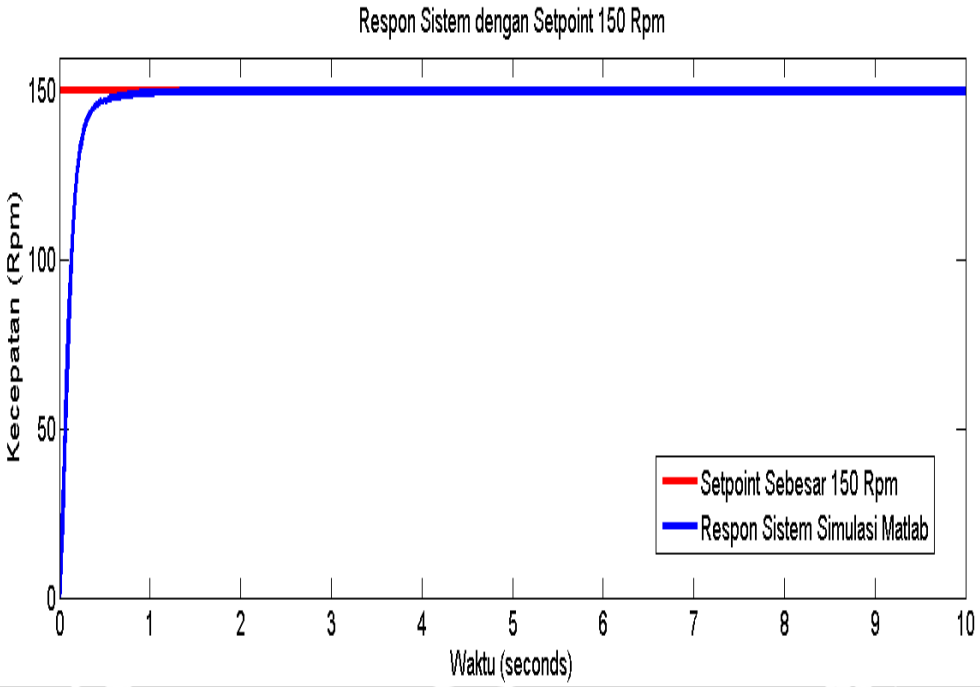
Pada Gambar 5.3, kurva respon linear, sehingga pemilihan daerah kerja sistem yang akan digunakan dalam proses pengontrolan dapat meggunakan semua range kecepatan yang ada. Namun, dalam skripsi ini pemilihan range kerja sistem berada pada range kecepatan 0-3500 rpm, yaitu 150, 250 dan 350 rpm.

5.2 Pengujian Sistem Keseluruhan

Pengujian sistem secara keseluruhan ini dilakukan untuk mengetahui seberapa besar nilai kontroler yang dibutuhkan agar sistem bekerja sesuai dengan *setpoint* yang diinginkan serta mengetahui hasil respon yang diimplementasikan pada alat.

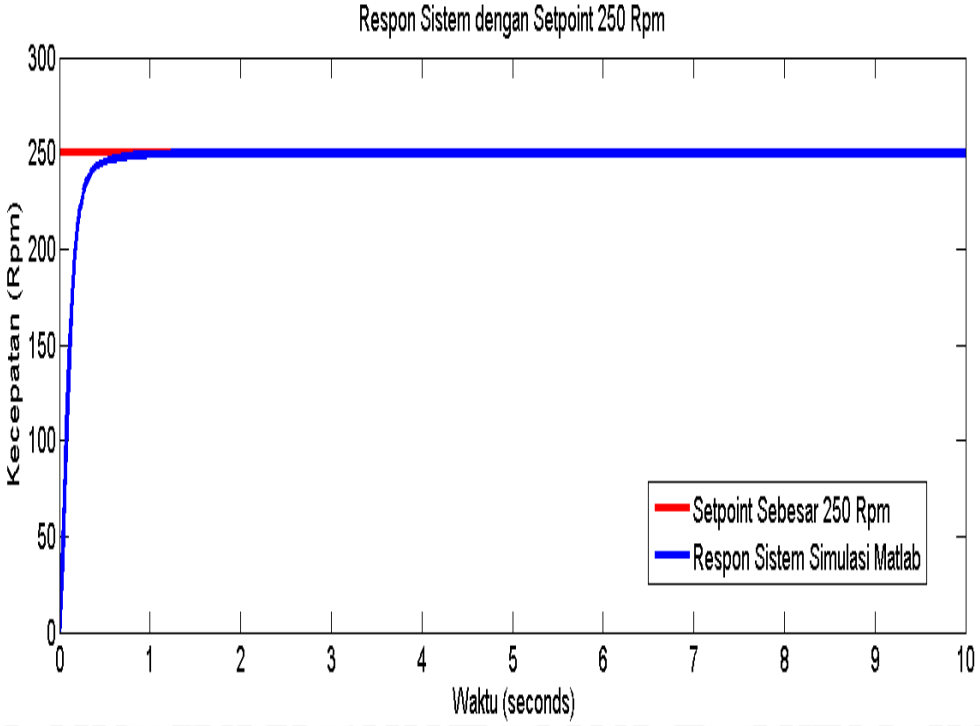
5.2.1 Pengujian sistem pada simulasi aplikasi Matlab

Pengujian ini dilakukan untuk mengetahui apakah nilai parameter yang didapat sudah sesuai dengan *setpoint* yang diinginkan. Pada pengujian ini diberi *setpoint* sebesar 150 Rpm, 250 Rpm, dan 350 Rpm, hasil pengujian ditunjukkan dalam Gambar 5.4, Gambar 5.5 dan Gambar 5.6.



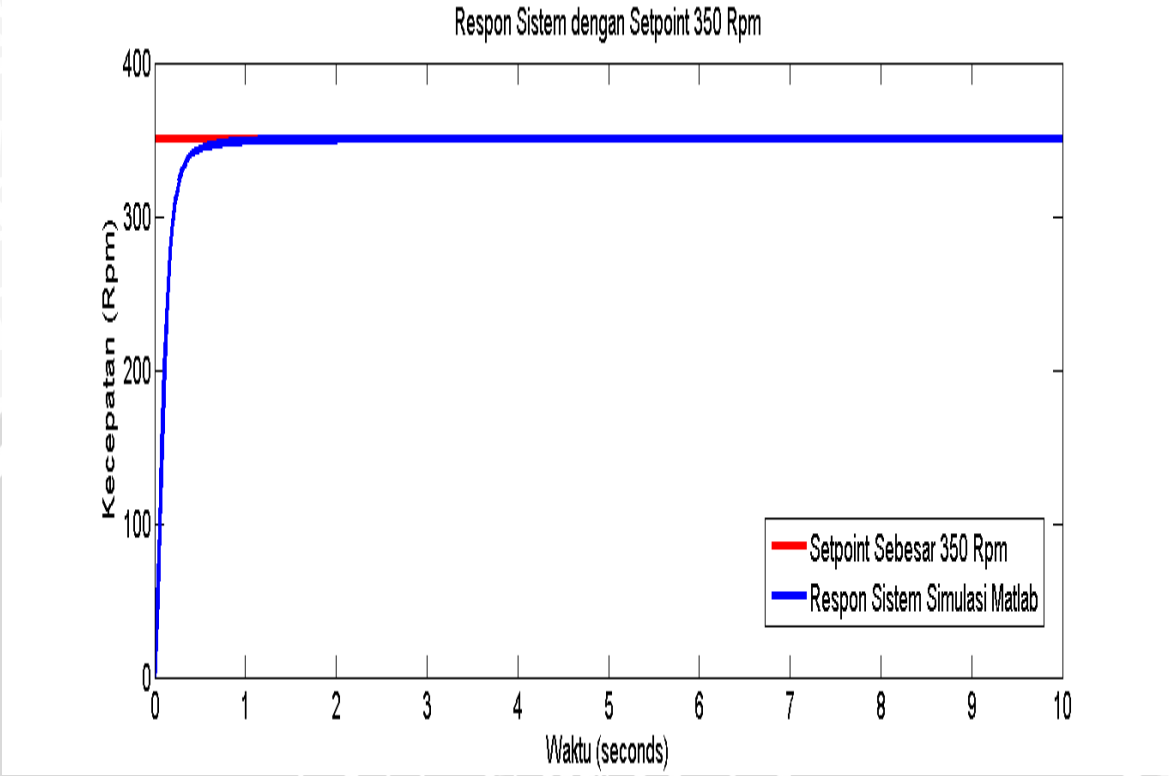
Gambar 5.4 Grafik respon sistem simulasi Matlab dengan *setpoint* sebesar 150 Rpm

Dari grafik yang ditunjukkan dalam Gambar 5.4 dapat disimpulkan bahwa sistem tidak terdapat *error steady state*, tidak ada *overshoot*, dan t_s sebesar 1 detik sehingga parameter PID yang didapat cukup sesuai dengan *setpoint* yang diinginkan.



Gambar 5.5 Grafik respon sistem simulasi Matlab dengan *setpoint* sebesar 250 Rpm

Dari grafik yang ditunjukkan dalam Gambar 5.5 dapat disimpulkan bahwa sistem tidak terdapat *error steady state*, tidak ada *overshoot*, dan t_s sebesar 1,2 detik sehingga parameter PID yang didapat cukup sesuai dengan *setpoint* yang diinginkan.

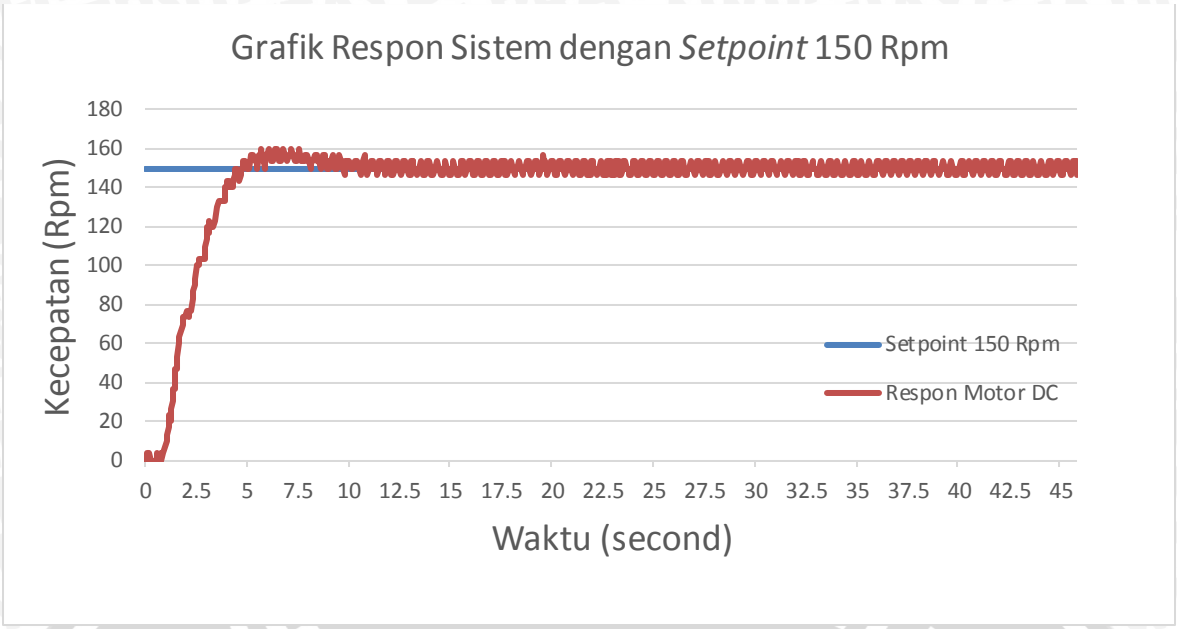


Gambar 5.6 Grafik respon sistem simulasi Matlab dengan *setpoint* sebesar 350 Rpm

Dari grafik yang ditunjukkan dalam Gambar 5.6 dapat disimpulkan bahwa sistem tidak terdapat *error steady state*, tidak ada *overshoot*, dan t_s sebesar 1,4 detik sehingga parameter PID yang didapat cukup sesuai dengan *setpoint* yang diinginkan.

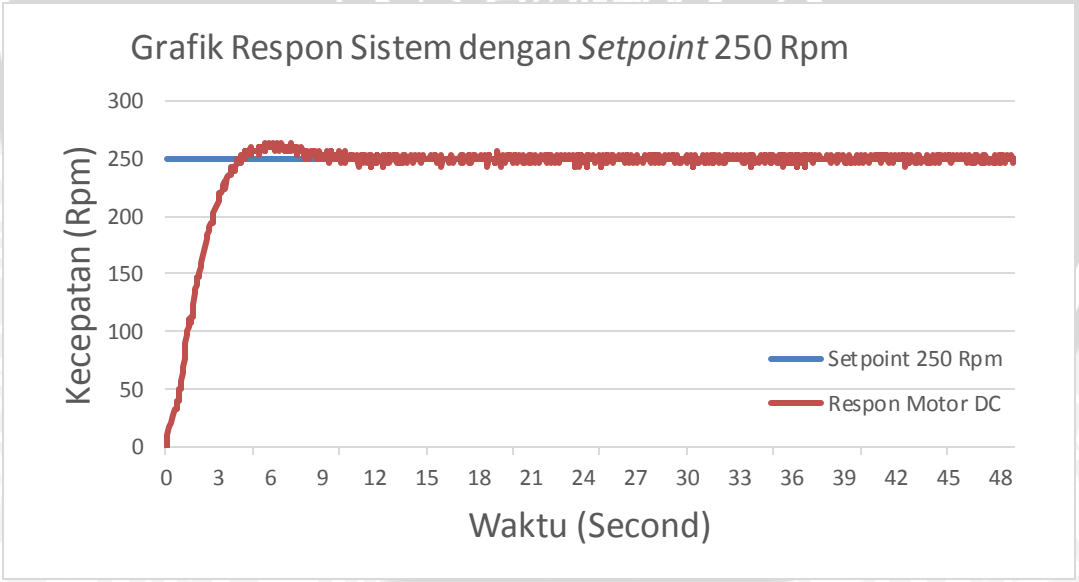
5.2.2 Pengujian sistem pada alat

Pengujian ini dilakukan untuk mengetahui apakah respon yang didapat dalam simulasi Matlab sama dengan respon yang didapat dari alat dengan nilai parameter kontroler yang sama. Pada pengujian ini *setpoint* yang diberikan sebesar 150 rpm, 250 rpm, dan 350 rpm.Hasil pengujian ditunjukkan dalam Gambar 5.7, Gambar 5.8, dan Gambar 5.9.



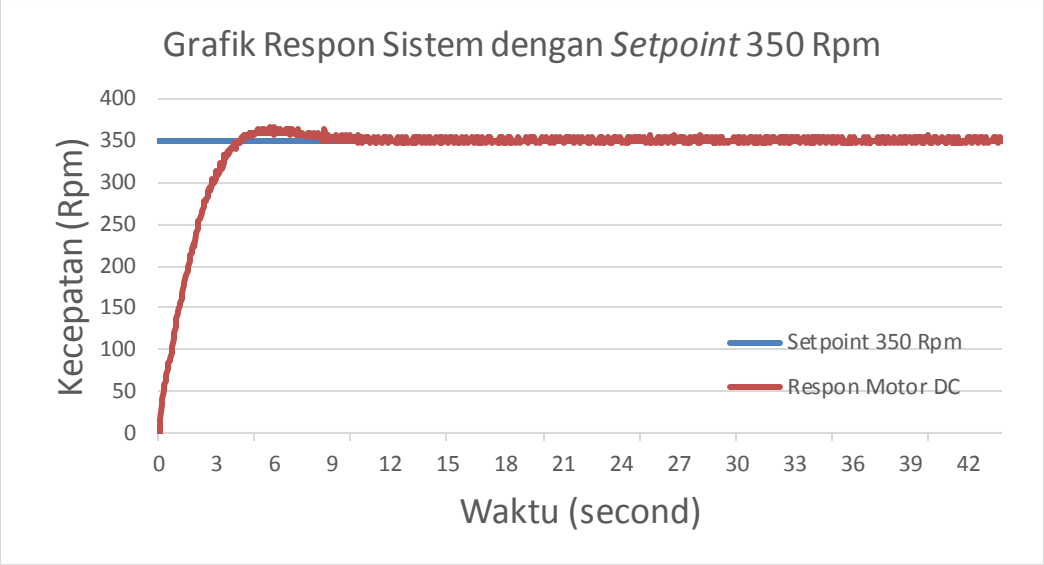
Gambar 5.7 Grafik respon Motor DC dengan *setpoint* sebesar 150 Rpm

Dari grafik respon sistem yang ditunjukkan dalam Gambar 5.7 dapat disimpulkan bahwa sistem terdapat osilasi namun masih dalam toleransi 2-5 %, t_s (waktu kerja) sebesar 8 detik, dan terdapat *overshoot* sebesar 6.667% sehingga parameter PID yang diimplementasikan pada alat dengan *setpoint* 150 rpm memiliki respon yang cukup baik, namun tidak sama dengan respon simulasi Matlab.



Gambar 5.8 Grafik respon Motor DC dengan *setpoint* sebesar 250 Rpm

Dari grafik respon sistem yang ditunjukkan dalam Gambar 5.8 dapat disimpulkan bahwa sistem terdapat memiliki t_s (waktu kerja) sebesar 9 detik, dan terdapat *overshoot* sebesar 4% sehingga parameter PID yang diimplementasikan pada alat dengan *setpoint* 250 rpm memiliki respon yang cukup baik, namun tidak sama dengan respon simulasi Matlab.

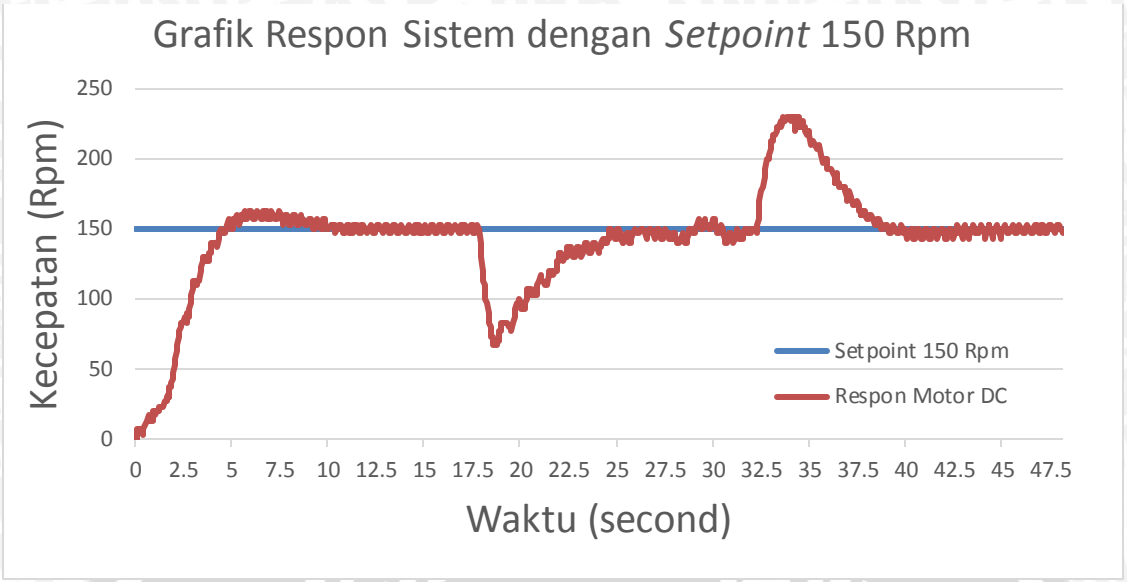


Gambar 5.9 Grafik respon Motor DC dengan *setpoint* sebesar 350 Rpm

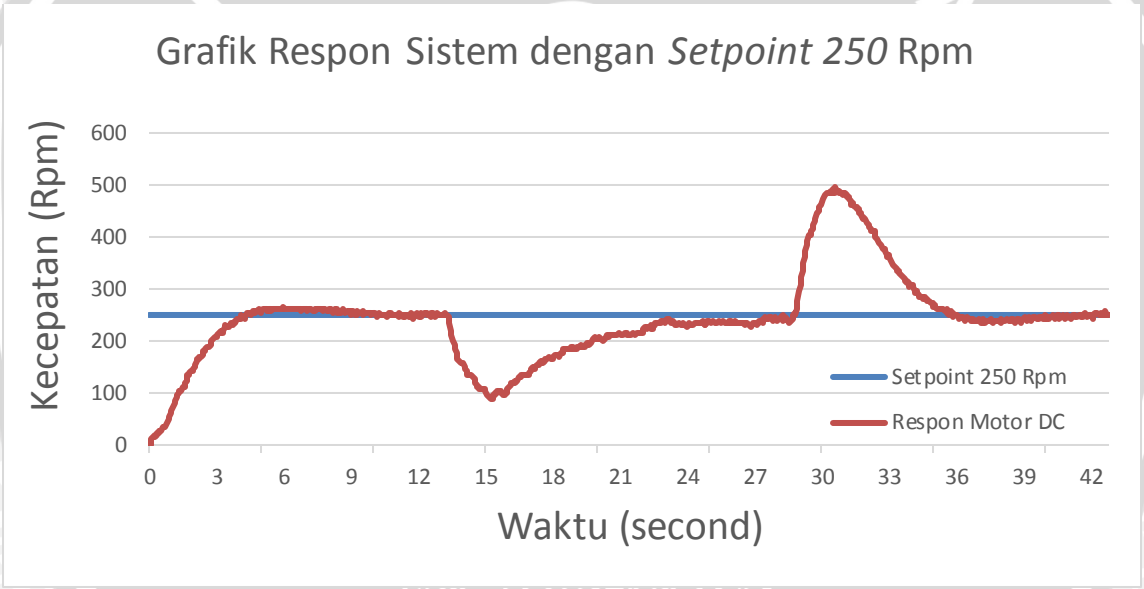
Dari grafik respon sistem yang ditunjukkan dalam Gambar 5.9 dapat disimpulkan bahwa sistem terdapat osilasi namun masih dalam toleransi 2-5 %, t_s (waktu kerja) sebesar 9.5 detik, dan terdapat *overshoot* sebesar 3.808% sehingga parameter PID yang diimplementasikan pada alat dengan *setpoint* 150 rpm memiliki respon yang cukup baik, namun tidak sama dengan respon simulasi Matlab.

5.2.3 Pengujian Sistem pada Alat dengan Diberi Gangguan

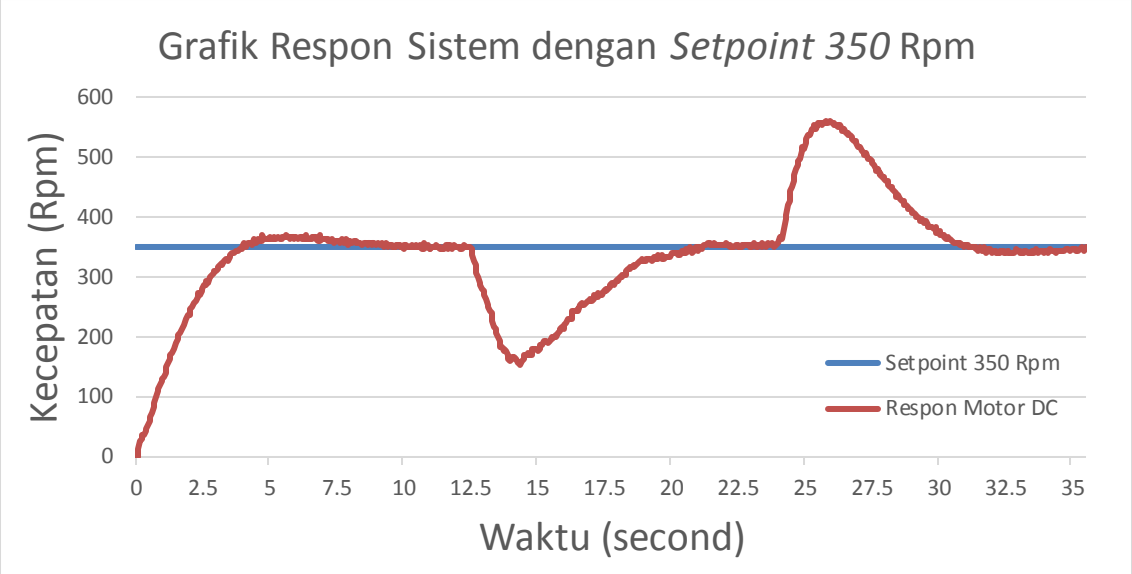
Pengujian ini dilakukan untuk mengetahui apakah respon dapat kembali dalam keadaan *steady state* ketika diberi gangguan. Pada pengujian ini *setpoint* yang diberikan sebesar 150 rpm, 250 rpm, dan 350 rpm dan gangguan yang diberikan berupa gesekan dari magnet U yang didekatkan pada motor DC D-6759 pada saat respon sudah mencapai keadaan *steady state*.Hasil pengujian ditunjukkan dalam Gambar 5.10, Gambar 5.11, dan Gambar 5.12.



Gambar 5.10 Grafik respon Motor DC dengan *setpoint* sebesar 150 Rpm dengan gangguan



Gambar 5.11 Grafik respon Motor DC dengan *setpoint* sebesar 250 Rpm dengan gangguan



Gambar 5.12 Grafik respon Motor DC dengan *setpoint* sebesar 350 Rpm dengan gangguan

Pada respon sistem (Gambar 5.10, 5.11 dan 5.12) dengan diberikan gangguan pada *setpoint* 150 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 6 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 5.2 detik. Pada *setpoint* 250 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 8.85 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 6 detik. Pada *setpoint* 350 rpm, respon akan mengalami perlambatan dan *recovery time* respon adalah 6 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon adalah 5 detik.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan data respon sistem yang diperoleh dari pengujian dengan menggunakan sinyal *Pseudo-Random Binary Sequence* (PRBS) didapat nilai fungsi alih $F(s) = \frac{140.5}{s^2+41.79s+142.4}$. dengan nilai *best fit* sebesar 96.15.Berdasarkan respon sistem yang diperoleh dari pengujian dengan menggunakan metode *root locus* didapat nilai parameter kontroler PID dengan penguatan sebesar K_p = 2.3286, K_i= 8, dan K_d= 0.0345.Berdasarkan hasil implementasi, respon motor DC dengan nilai *setpoint* 150 rpm, 250 rpm dan 350 rpm tidak memiliki nilai *error steady state*. Sedangkan *settling time* adalah 8 detik, 9 detik dan 9.5 detik. 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 6 detik, 8.85 detik dan 6 detik. Ketika gangguan yang diberikan dilepaskan, respon akan mengalami percepatan dan *recovery time* respon masing-masing adalah 5.2 detik, 6 detik dan 5 detik.

.

6.2 Saran

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

DAFTAR PUSTAKA

Amstrom, K. J., & Hagglund, Tore. 1995 *PID Controllers: Theory, Design and Tuning*. Instrument Society of America: Research Triangle Park.

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

Benjamin, C. 1995. *Teknik Kontrol Automatik*. Yogyakarta. Penerbit : PT Aditya Media.

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

Johnson, M. A., Moradi, 1984. Mohammad H. *PID Controler*. London: Springer-Verlag.

Kustanti, Ika. 2013. *Pengendalian Kadar Keasaman (pH) Pada Sistem Hidroponik Stroberi Menggunakan Kontroller PID Berbasis Arduino Uno*. Laporan Skripsi, Teknik Elektro Brawijaya.

Ikrom, H. 2008. *Perancangan Kontroler PID-Kaskade Dengan Metode Root Locus Untuk Kontrol Temperatur Dan Tekanan Pada Proses Evaporator*. Laporan Skripsi, Teknik Elektro Brawijaya.

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.

R, Hamdi. https://hamdi88.wordpress.com/2008/11/16/root-locus/.(diakses pada 26 November 2015).

Thomas, W. D. H., Prasetyo, Y. W. A. 2003. *Analisis dan Desain Sistem Kontrol dengan Matlab*. Yogyakarta. Penerbit : Andi.

Umam, Ardian. http://ardianumam.web.ugm.ac.id/2013/05/27.(diakses pada 26 November 2015).

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

LAMPIRAN 1

FOTO ALAT

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

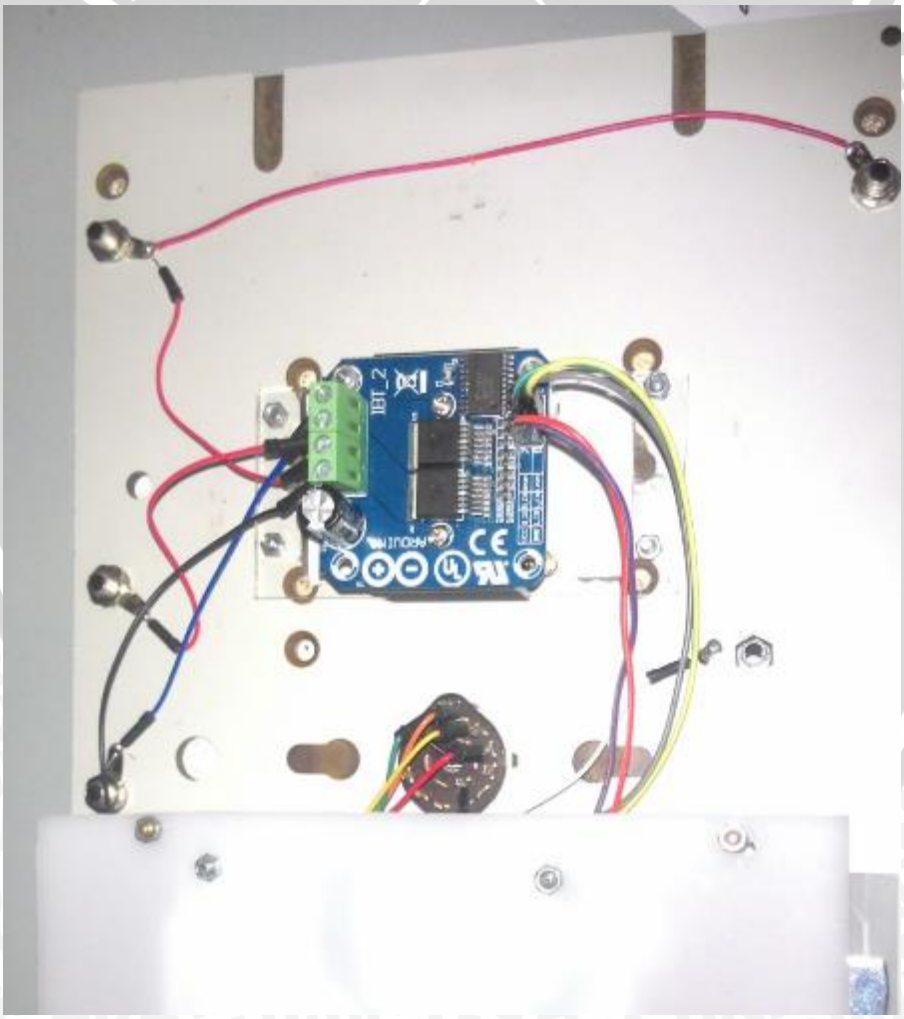


Foto *driver* motor IC L298N (*Dual H-Bridge*) di dalam modul

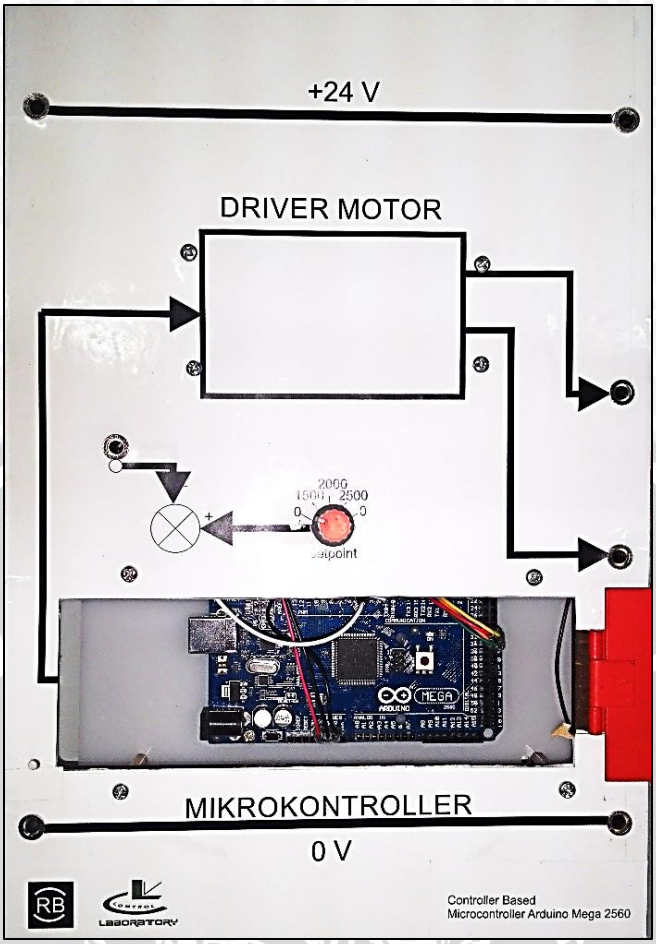


Foto mikrokontroler Arduino Mega 2560 di dalam modul

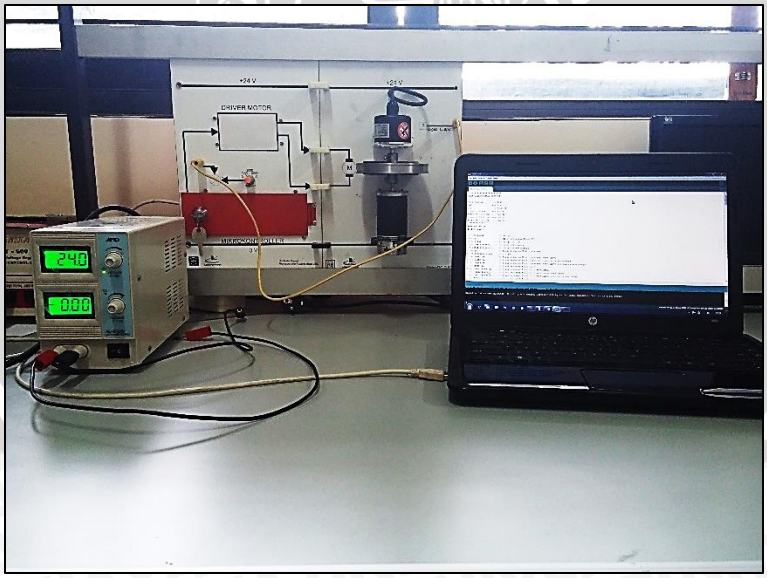


Foto pengujian keseluruhan

LAMPIRAN 2

LISTING PROGRAM

Listing Program Matlab Mencari s

```
num = [140.5];
den = [1 41.79 142.4];

step (num,den)
figure, rlocus (num,den)
```

Listing Program Matlab Mencari Parameter Kontroller PID dengan Root Locus

```
s1=-8.08
KI=[1 2 3 5 8]

plant_num=[0 0 140.5];
plant_den=[1 41.79 142.4];

s1mag = abs(s1)
beta = angle(s1)
plant_a1 = polyval(plant_num,s1)/polyval(plant_den,s1);
plants1mag = abs(plant_a1)
psi = angle(plant_a1)
t=0:1:20:300;

for k =1:5

    KP=-sin(beta+psi)/(plants1mag*sin(beta))-
2*KI(k)*cos(beta)/s1mag
    nilai_KI= KI(k)
    KD = sin(psi)/(s1mag*plants1mag*sin(beta))+KI(k)/s1mag^2

    Gcnum = [KD KP KI(k)];
    Gcden = [0 1 0];

    Tnum = conv(plant_num,Gcnum);
    Tden = conv(plant_den,Gcden)+conv(plant_num,Gcnum);

    r = roots(Tden)
    step (Tnum,Tden,t)
    %step (plant_num,plant_den,t)
    hold on
end
hold off
figure, rlocus(Tnum,Tden)
```

Listing Program keseluruhan dengan Arduino Mega 2560

```
/*************************
BISMILLAHIRROHMANIRROHIIM.

Baca Putaran      ==> PIN 2
PWM               ==> PIN 5
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;          // Set point
float y;               // Nilai kecepatan Motor DC
float error;            // set_point - y
float last1_error;      // error sebelumnya
float pulsa;           // pulsa rotary encoder
float Ts;              // Time Sampling Model Referensi
float pwmMotor;         // PWM Motor
float Prop;            // Sinyal kontrol Kontroler P
float Intg;            // Sinyal kontrol Kontroler I
float last1_Intg;       // Sinyal kontrol Kontroler I sebelumnya
float Dif;             // Sinyal kontrol Kontroler D
double MV;              // Sinyal Kontrol Kontroler PID
float Kp;              // Nilai konstanta Kp
float Ki;              // Nilai konstanta Ki
float Kd;              // Nilai konstanta Kd
int as,ap;

void setup()
{
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 digitalWrite(7,HIGH);
 digitalWrite(8,HIGH);
 pinMode(pwm,OUTPUT);
 pinMode(led,OUTPUT);
 pinMode(28, INPUT_PULLUP);
 pinMode(29, INPUT_PULLUP);
 pinMode(30, INPUT_PULLUP);
 pinMode(31, INPUT_PULLUP);

 //set_point=350;
 Ts = 0.01;
 Kp = 2.3286;
 Ki = 8;
 Kd = 0.0345;

 last1_error = 0; last1_Intg = 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/64/25KHz/10ms

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

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

}

// Timer Controller and Serial Monitor
ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  error =  set_point - y;
  Kontroler();

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

void Kontroler()
{
  Prop = Kp * error;
  Intg = last1_Intg + (Ki * Ts * error);
  Dif  = ((Kd/Ts) * error ) - ((Kd/Ts) * last1_error);
  MV = Prop + Intg + Dif;

  last1_Intg = Intg;
  last1_error = error;

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

void Rot_switch()
{
  if(digitalRead(29)==LOW)
  {
```

```
  set_point=0; pwmMotor=0;  MV=0;
 }
 else if(digitalRead(28)==LOW)
 {
  set_point=150;
 }
 else if(digitalRead(30)==LOW)
 {
  set_point=250;
 }
 else if(digitalRead(31)==LOW)
 {
  set_point=350;
 }
 else;
}

void loop()
{
 analogWrite(pwm,pwmMotor);
}

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

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

LAMPIRAN 3

BLOK DIAGRAM

Diagram Blok SIMULINK Sistem Tanpa Kontroller

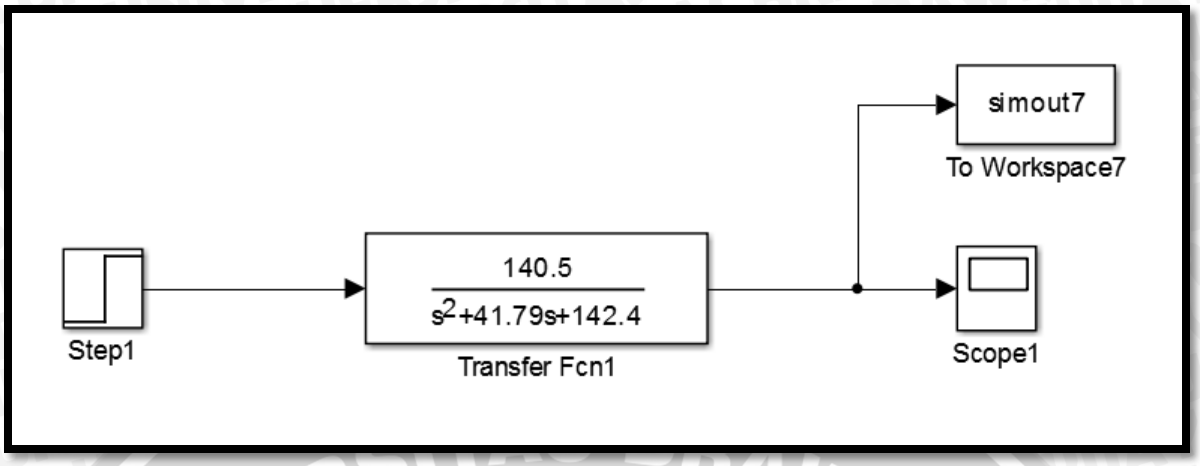
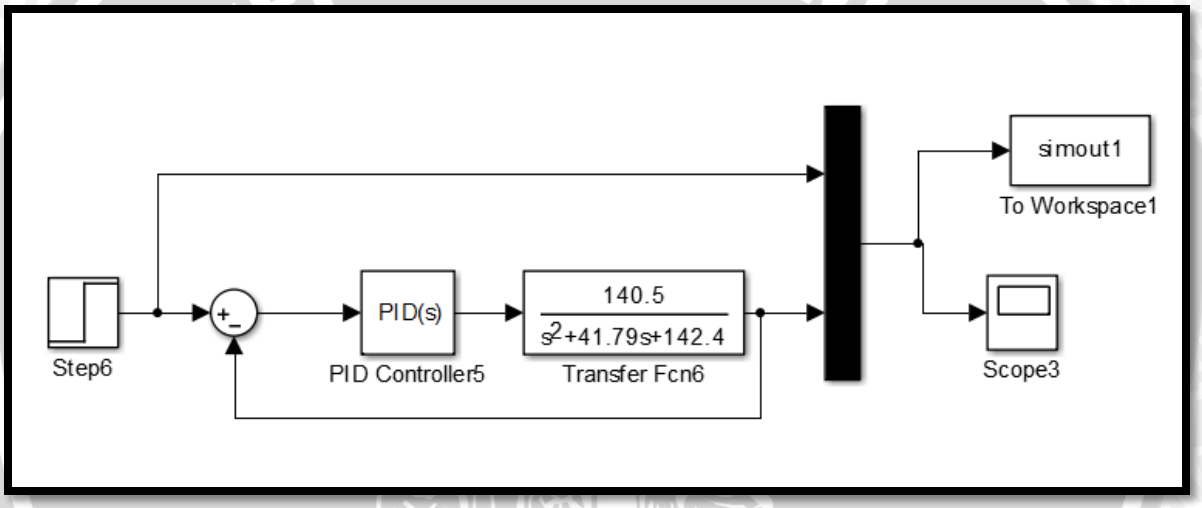


Diagram Blok SIMULINK Sistem dengan Kontroller PID



LAMPIRAN 4

DATASHEET