TUGAS AKHIR – TE 141599

PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN KONTROLER *FUZZY* BERBASIS *LINEAR QUADRATIC REGULATOR*

Fairuzza Dinansyar
NRP. 2211 100 134

Dosen Pembimbing
Ir. Rusdhianto Effendie A.K., M.T.
Ir. Ali Fatoni, M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – **TE 141599**

SPEED CONTROL OF BRUSHLESS DC MOTOR USING FUZZY BASED ON LINEAR QUADRATIC REGULATOR CONTROLLER

Fairuzza Dinansyar
NRP. 2211 100 134

Supervisor
Ir. Rushdianto Effendie A.K., M.T.
Ir. Ali Fatoni, M.T.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Insitute of Technology
Surabaya 2016

PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN KONTROLER *FUZZY* BERBASIS *LINEAR QUADRATIC REGULATOR*

TUGAS AKHIR

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

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Rusdhianto Effendie A.K.,M.T.	**Ir. Ali Fatoni, M.T.**
NIP. 195704241985021001	NIP. 196206031989031002

SURABAYA
JANAURI, 2016

PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN KONTROLER *FUZZY* BERBASIS *LINEAR QUADRATIC REGULATOR*

Fairuzza Dinansyar
2211 100 134

Dosen Pembimbing I : Ir. Rusdhianto Effendie A.K., M.T.
Dosen Pembimbing II : Ir. Ali Fatoni, M.T.

ABSTRAK

Motor DC, sebagai salah satu mesin listrik yang banyak digunakan di industri, penggunaannya saat ini telah banyak digantikan dengan motor *Brushless DC* (BLDC). Motor BLDC ketika diberi pembebanan maka kecepatan motor BLDC akan berubah sehingga fungsi alih motor juga mengalami perubahan. Pada penelitian ini, digunakan kontroler *Fuzzy* LQR untuk mengatur kecepatan motor BLDC supaya respon yang dihasilkan adalah sama untuk setiap pembebanan sesuai dengan spesifikasi performansi yang diinginkan. *Fuzzy* digunakan sebagai *tuning* nilai *gain state feedback* dari kontroler LQR. Setelah melakukan implementasi kontroler pada motor BLDC didapatkan bahwa hasil implementasi sebesar 0,96 detik untuk nilai *time constant*, 2,88 detik untuk nilai *settling time* dan 2,33 detik untuk nilai *rise time* dengan nilai RMSE masing – masing sebesar 0,135%, 0,417% dan 0,682%.

Kata Kunci : *Brushless DC*, *Fuzzy LQR*

SPEED CONTROL OF BRUSHLESS DC MOTOR USING FUZZY CONTROLLER BASED ON LINEAR QUADRATIC REGULATOR

Fairuzza Dinansyar
2211 100 134

Supervisor I : Ir. Rusdhianto Effendie A.K., M.T.
Supervisor II : Ir. Ali Fatoni, M.T.

ABSTRACT

DC motors, as one of the electrical machines are widely used in industrial, are now replaced with Brushless DC Motors (BLDC). When BLDC motor is given a load then BLDC motor speed and transfer function will change.In this study, controller Fuzzy LQR are used to control speed of BLDC motor. Fuzzy is used to tune the value of state feedback from LQR controller so the response will give the same result for all the load corresponding to the performance spesification that we want. After this controller implemented on motor BLDC we got that implementation result 0.96 *second for time constant,* 2.88 *second for setlling time and* 2.33 *second for rise time with RMSE value respectively* 0.135%, 0.417% *and* 0.682%.

Keywords : *Brushless DC*, *Fuzzy LQR*

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan buku tugas akhir dengan judul **"PENGATURAN KECEPATAN MOTOR *BRUSHLESS DC* MENGGUNAKAN KONTROLER *FUZZY* BERBASIS *LINEAR QUADRATIC REGULATOR*"**. Tugas akhir merupakan salah satu syarat yang harus dipenuhi untuk menyelesaikan program studi Strata-1 pada Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa dalam penulisan tugas akhir ini banyak mengalami kendala, namun berkat bantuan, bimbingan, dan kerja sama dari berbagai pihak sehingga kendala-kendala tersebut dapat diatasi. Untuk itu pada kesempatan ini penulis menyampaikan banyak terimakasih dan penghargaan setingi-tingginya kepada :

1. Kedua orang tua yang selalu memberikan dukungan, semangat, dan doa kepada penulis.
2. Bapak Rusdi dan Bapak Ali selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan tugas akhir ini.
3. Bapak Rusdi selaku Koordinator Bidang Studi Sistem Pengaturan Jurusan Teknik Elektro ITS.
4. Bapak Ardyono Priyadi selaku Ketua Jurusan Teknik Elektro ITS.
5. Rekan-rekan e51 khususnya bidang studi Sistem Pengaturan.

Penulis berharap tugas akhir ini dapat bermanfaat bagi yang membutuhkannya.

Surabaya,13 Januari 2016

Penulis

TABLE OF CONTENT

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Aplikasi dari motor listrik telah menyebar ke berbagai macam bidang dalam kehidupan kita sebagai alat konversi energi utama listrik menjadi mekanik. Motor listrik banyak digunakan dalam berbagai peralatan seperti *air conditioning, vacuum cleaner, conveyor*, lemari pendingin, dan lain sebagainya. Motor DC banyak digunakan dalam aplikasi tersebut akan tetapi penggunaan motor DC membutuhkan sikat untuk melakukan komutasi mekanik yang menyebabkan gesekan mekanik yang dapat memperpendek umur dan membuat suara bising pada motor kemudian muncullah motor *Brushless Direct Current* (BLDC) sebagai pengganti motor DC yang tidak menggunakan sikat tetapi menggunakan komutasi elektris sehingga motor BLDC ini lebih efisien.[1]

Pada tugas akhir ini digunakan metode *fuzzy* LQR untuk mengendalikan kecepatan motor BLDC dan kontroler *fuzzy* di sini berfungsi sebagai pengatur nilai gain *state feedback* untuk kontroler LQR karena pada implementasinya digunakan pembebanan yang berubah-ubah.

1.2 Perumusan Masalah

Permasalahan disini ini adalah ketika motor BLDC tidak dibebani, maka kecepatan motor akan berjalan dengan kecepatan tetap. Namun, ketika motor BLDC diberikan suatu beban yang diatur sedemikian rupa, maka kecepatan motor akan berubah dan menyebabkan fungsi alih motor berubah. Permasalahan utama dari Tugas Akhir ini adalah bagaimana mengontrol kecepatan motor BLDC pada kondisi berbeban sehingga menghasilkan kinerja yang sesuai dengan spesifikasi performansi yang diinginkan.

1.3 Batasan Masalah

Ruang lingkup pembahasan tugas akhir ini dibatasi sebagai berikut,

a. *Plant* yang digunakan adalah motor arus searah tanpa sikat/ BLDC.
b. Perancangan dan implementasi kontroler untuk mengatur kecepatan motor BLDC menggunakan metode *Fuzzy* LQR.

c. Pada tugas akhir ini kecepatan yang dikontrol dalam rentang 1000 hingga 1600 rpm.

1.4 Tujuan Penelitian

Tujuan dari pelaksanaan tugas akhir ini adalah:

a. Merancang *plant* berupa motor BLDC yang terangkai dengan rem magnetik yang berfungsi sebagai pembebanan pada motor BLDC.
b. Merancang kontroler *Fuzzy Linear Quadratic Regulator* (LQR) untuk mengatasi permasalahan adanya efek pembebanan yang berubah-ubah pada motor BLDC.
c. Implementasi pada motor BLDC yang telah dirancang menggunakan *software* MATLAB dan mikrokontroler Arduino supaya mendapatkan performansi terbaik untuk pengendalian kecepatan motor BLDC pada pembebanan yang berbeda-beda.

1.5 Sistematika Penulisan

Buku tugas akhir ini terdiri dari lima bab dan disusun menurut sistematika penulisan berikut ini

BAB I: PENDAHULUAN

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, sistematika penulisan, dan relevansi.

BAB 2: TINJAUAN PUSTAKA

Bab ini berisi tentang teori yang menunjang penelitian, berupa teori tentang BLDC dan komponennya, serta metode yang digunakan untuk pengaturan kecepatan motor BLDC.

BAB 3: PERANCANGAN SISTEM

Bab ini berisi tentang perancangan perangkat keras, perangkat lunak, dan perancangan kontroler.

BAB 4: PENGUJIAN DAN ANALISA

Bab ini berisi tentang hasil simulasi kontroler dan analisanya. Selain itu berisi tentang hasil implementasi kontroler pada Simulator BLDC beserta analisa hasil implementasi.

BAB 5: PENUTUP

Berisi kesimpulan dan saran yang dapat dijadikan pertimbangan pengembangan berdasarkan hasil pengerjaan tugas akhir ini.

1.6 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat dalam pengembangan penelitian tentang motor BLDC khususnya strategi kontrol motor listrik dalam motor BLDC.

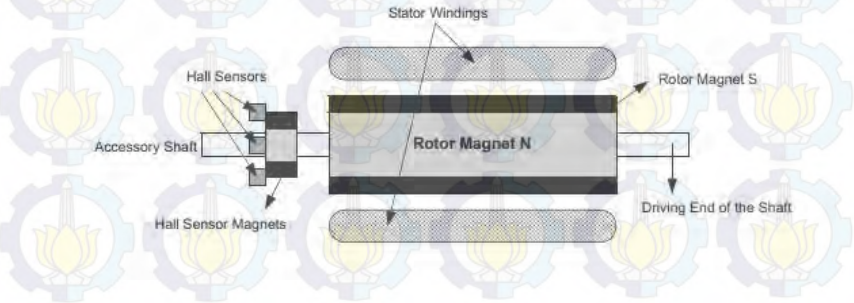
BAB 2
TINJAUAN PUSTAKA

2.1 Motor *Brushless* DC [2]

Motor BLDC adalah tipe dari motor sinkron. Hal ini berarti medan magnetik yang dihasilkan oleh *stator* dan medan magnetik yang dihasilkan oleh *rotor* berputar pada frekuensi yang sama. Motor BLDC tidak mengalami slip yang biasanya terjadi pada motor induksi. Motor BLDC terdapat dalam konfigurasi 1 fasa, 2 fasa, dan 3 fasa. Berdasarkan tipe *stator* yang memiliki jumlah kumparan yang sama terdapat dua tipe dari kumparan *stator* motor BLDC yaitu trapezoidal dan sinusoidal perbedaan ini dibuat berdasarkan dari hubungan kumparan *stator* yang menghasilkan tipe ggl balik yang berbeda.

Seperti namanya motor BLDC tidak menggunakan sikat untuk melakukan komutasi tetapi menggunakan komutasi elektris, untuk dapat memutar motor BLDC maka kumparan *stator* dari motor harus diberi tegangan sesuai urutan komutasi, Supaya dapat melakukan hal tersebut maka posisi *rotor* harus dapat diketahui. Posisi *rotor* ini dapat diketahui dengan menggunakan sensor *hall effect* atau dengan teknik *sensorless* yaitu dengan mendeteksi ggl balik pada kumparan *stator*. Pada umumnya motor BLDC memiliki tiga sensor *hall effect* yang terpasang dekat *stator* dan memiliki kontroler elektronik dalam motor untuk mengatur komutasi daya yang mengalir dalam kumparan stator.

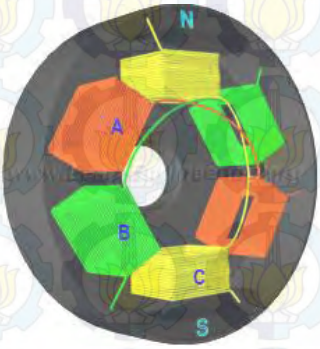
Pada tugas akhir kali ini motor yang kami gunakan adalah motor yang dipakai pada *air conditioner* (AC) dengan konstruksi motor seperti terlihat pada Gambar 2.1.



Gambar 2.1 Konstruksi Motor BLDC. [2]

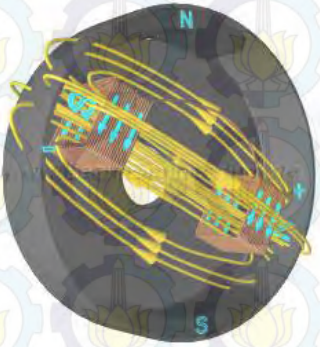
2.1.1 Prinsip Kerja Motor *Brushless* DC [3]

Prinsip kerja dari motor BLDC adalah gaya tarik menarik antara magnet permanen pada *rotor* dan elektromagnet pada *stator*, saat suatu kutub akan saling tolak menolak dengan kutub yang sejenisnya begitupun sebaliknya akan saling tarik menarik jika magnetnya berlawanan kutub. Gambar 2.2 memperlihatkan kumparan *stator* dalam motor BLDC tipe *outrunner* 3 fasa(A,B,C) dan 1 pasang kutub *rotor*. Setiap fase kumparan *stator* akan diberi daya DC secara bergantian supaya *rotor* dapat berputar.



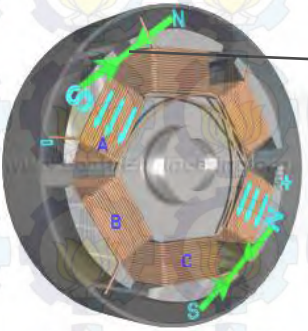
Gambar 2.2 Kumparan *Stator* 3 Fasa (A,B,C) Motor BLDC.

Ketika kumparan *stator* diberi daya DC maka kumparan *stator* akan menjadi elektromagnet seperti terlihat pada gambar 2.3.



Gambar 2.3 Kumparan *Stator* menjadi Elektromagnet.

Pada Gambar 2.4 ketika kumparan A diberi daya DC dengan arah aliran arus seperti pada gambar maka akan timbul gaya elektromagnetik sehingga kutub berbeda antara *rotor* dan *stator* akan menghasilkan gaya tarik menarik satu sama lain akibatnya kutub *rotor* akan bergerak menuju ke kumparan A.



Gambar 2.4 Gaya Tarik Menarik antara Kumparan *Stator* dengan Medan Magnet pada *Rotor*.

Gambar 2.5 memperlihatkan bahwa saat *rotor* mendekati kumparan A maka kumparan B akan diberi daya DC sedangkan aliran daya DC ke kumparan A diputus akibatnya sama seperti yang terjadi pada kumparan A ketika diberi daya DC, *rotor* akan bergerak mendekati kumparan B begitu seterusnya lalu ke kumparan C setelah itu kumparan A akan diberi daya DC dengan polaritas yang berbeda dengan sebelumnya sehingga *rotor* akan terus berputar.

Gambar 2.5 Urutan Komutasi Motor BLDC.

2.1.2 Konstruksi Motor *Brushless* DC

Secara umum konstruksi motor BLDC seperti yang terlihat pada Gambar 2.1 terdiri dari *rotor*,*stator*,sensor posisi *rotor* dan kontroler elektronik.Berikut penjelasan dari masing-masing komponen diatas.

a. *Rotor*

Rotor merupakan bagian yang bergerak dalam motor BLDC. *Rotor* pada motor BLDC umumnya seperti terlihat pada Gambar 2.6 terbuat dari magnet permanen dan memiliki jumlah kutub yang bervariasi dari dua sampai delapan pasang kutub utara dan selatan. Berdasarkan pada kebutuhan kerapatan medan magnetik dalam *rotor* maka material magnetik pada *rotor* harus dipilih dengan baik. Pada umumnya magnet *ferrite* digunakan sebagai magnet permanen.[2]

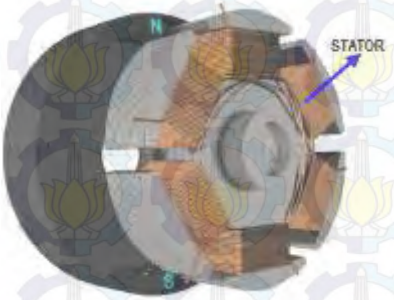
Terdapat dua konstruksi dasar motor BLDC yaitu *outrunner* dan *inrunner*. Motor *outrunner* merupakan jenis motor BLDC yang memiliki *rotor* terletak di luar sehingga bagian dari motor yang berputar merupakan bagian luar dari motor sedangkan *inrunner* merupakan jenis motor yang memiliki *rotor* berada pada bagian dalam motor.

Gambar 2.6 Konstruksi *Rotor* Motor BLDC Tipe *Outrunner*.[3]

b. *Stator*

Konstruksi *stator* dari motor BLDC ditunjukkan pada Gambar 2.7 terdiri dari susunan baja yang dilaminasi dan memiliki kumparan di setiap slotnya. Motor BLDC pada umumnya menggunakan kumparan *stator* dengan pola bintang. Terdapat dua tipe kumparan *stator* dari motor BLDC yaitu trapezoidal dan sinusoidal. *Stator* dalam motor BLDC digunakan untuk menghasilkan elektromagnet

sehingga terjadi gaya tarik menarik antara *rotor* dan kumparan *stator* yang menyebabkan *rotor* dapat berputar.[2]



Gambar 2.7 Konstruksi *Stator* Motor BLDC. [3]

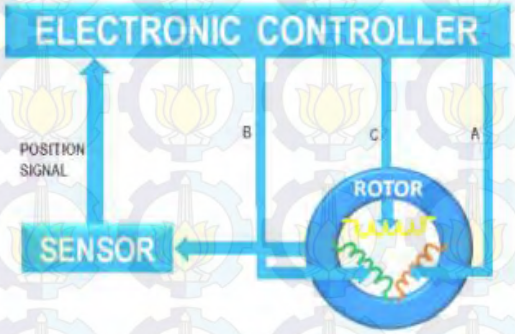
c. Sensor Posisi *Rotor*

Motor BLDC dikontrol secara elektris, untuk memutar motor BLDC maka kumparan *stator* harus diberi daya sesuai urutan komutasi. Sangat penting untuk mengetahui posisi *rotor* dengan tujuan untuk mengerti kumparan mana yang akan diberi daya mengikuti urutan komutasi. Posisi *rotor* ini dideteksi menggunakan sensor *hall effect* yang terpasang dalam *stator*. Pada umumnya seperti ditunjukkan pada Gambar 2.8 sensor *hall effect* terletak dekat *stator* jadi ketika kutub magnetik *rotor* mendekati sensor *hall effect* maka sensor akan memberikan sinyal *high* atau *low* yang mengindikasikan kutub N atau S melewati sensor. Berdasarkan pada sinyal dari sensor *hall effect* tersebut maka *driver* dapat menentukan urutan komutasi secara tepat.[2]

Gambar 2.8 Posisi Sensor *Hall Effect* dalam Motor BLDC. [3]

d. *Driver* Motor BLDC

Driver/kontroler elektronik adalah rangkaian yang digunakan untuk mengontrol aliran arus pada kumparan *stator* motor BLDC. Motor BLDC tidak memiliki *brush* untuk mengatur aliran arus pada kumparan *stator* tetapi menggunakan suatu *driver* untuk mengatur aliran arus tersebut. *Driver* berfungsi untuk mengatur kumparan mana yang akan dialiri arus agar motor dapat berputar berdasarkan pada sinyal posisi *rotor* dari sensor. Gambar 2.9 menunjukkan konfigurasi *driver* motor BLDC.



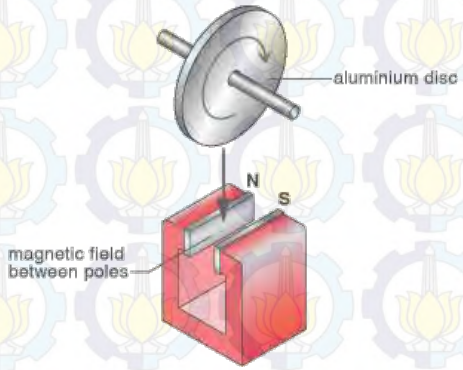
Gambar 2.9 Konfigurasi *Driver* Motor BLDC. [3]

2.2 Rem Elektromagnetik

Rem elektromagnetik merupakan sistem pengereman yang menggunakan gaya elektromagnetik yang timbul dari suatu magnet permanen tetap atau kumparan yang diberikan arus listrik untuk memperlambat suatu gerakan. Konstruksi dasarnya berupa suatu piringan logam non-feromagnetik yang terpasang pada suatu poros yang berputar. Piringan logam tersebut diapit oleh kumparan yang dialiri arus listrik hingga menimbulkan medan magnet yang kutubnya saling berlawanan. Logam piringan tersebut akan memotong medan magnet yang ditimbulkan oleh kumparan tersebut sehingga menimbulkan *eddy current* atau arus *eddy*.

Arus *eddy* merupakan arus listrik yang timbul bilamana suatu piringan logam berada di sekitar medan magnet yang garis-garis gayanya sedang berubah-ubah. Arus *eddy* ini mempunyai medan magnet yang arahnya berlawanan dengan arah gerak piringan logam sehingga laju piringan logam akan tertahan akibat dari adanya arus *eddy* ini.

Rem elektromagnetik biasa diletakkan dekat dengan bagian yang bergerak. Rem ini bekerja pada kondisi yang dingin dan memenuhi persyaratan energi pengereman kecepatan tinggi karena tanpa adanya gesekan. Selain pada kendaraan listrik, aplikasi rem elektromagnetik juga dapat ditemukan pada sistem pengereman kereta api. Gambar 2.10 memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.



Gambar 2.10 Struktur dari Sebuah Rem Elektromagnetik.[4]

2.3 Arduino Uno [5]

Arduino Uno adalah *board* mikrokontroler berbasis Atmega328. Alat ini mempunyai 14 pin *input*/*output* digital dan 6 diantaranya dapat digunakan sebagai *output* PWM, 6 *input* analog, resonator keramik 19MHz, koneksi USB, soket listrik, ICSP *header,* dan tombol *reset*. Dengan kabel USB alat ini mudah dihubungkan ke komputer dan dapat diaktifkan dengan baterai atau adaptor AC ke DC. Spesifikasi Arduino Uno adalah sebagai berikut :

a.	Mikrokontroler	:	Atmega328
b.	Tegangan operasi	:	5V
c.	Tegangan *input* (direkomendasikan)	:	7-12V
d.	Tegangan *input* (batasan)	:	6-20V
e.	Pin *input*/*output* digital	:	14 (6 diantaranya *output* PWM)
f.	Pin *input* analog	:	6

g. Arus DC tiap pin I/O : 40mA
h. Arus DC untuk pin 3,3 V : 50mA

14 pin *input* dan *output* Arduino Uno dapat digunakan dengan fungsi *pinMode(), digitalWrite(),* dan *digitalRead*(). Tiap pin memiliki tegangan operasi 5V dan dapat menerima arus maksimum 40mA. Beberapa pin memiliki fungsi khusus

a. *Serial* : 0 (RX) dan 1 (TX). Digunakan untuk menerima (RX) dan mengirim (TX) data *serial* TTL. Pin ini terhubung dengan pin pada Atmega8U2 USB ke cip USB ke TTL serial.
b. *External Interrupts* : 2 dan 3.pin ini digunakan sebagai *trigger* gangguan pada nilai yang rendah, menaikkan dan menurunkan nilai, atau merubah nilai.
c. PWM : 3,5,6,9,10, dan 11. Menyediakan *output* PWM 8bit dengan fungsi *analogWrite().*
d. SPI (*Serial Peripheral Interface*) : 10 (*Slave Select*), 11 (*Master Out Slave In*), 12 (*Master In Slave Out*), 13 (*Serial Clock*). Pin ini mendukung komunikasi SPI.
e. LED : 13, LED ini terhubung dengan pin 13. Ketika pin memiliki nilai yang tinggi LED akan *on* dan ketika pin memiliki nilai yang rendah, LED akan *off*

Pada Arduino Uno terdapat 6 *input* analog, dengan nama A0 hingga A5. Pin tersebut memiliki resolusi 10-bit (1024 nilai yang berbeda). *Input* analog ini berupa tegangan dari *ground* ke 5V.

2.4 MATLAB [6]

MATLAB merupakan paket program dengan bahasa pemrograman yang tinggi untuk mengembangkan algoritma, visualisasi data, dan komputasi numerik. Program MATLAB ini dapat digunakan untuk menyelesaikan masalah komputasi dengan lebih cepat dibandingkan dengan bahasa pemrograman tradisional, seperti C, C++, dan Fortran. MATLAB digunakan untuk banyak aplikasi seperti *signal and image processing*, desain kontrol, pengujian dan pengukuran, permodelan, dan analisis.

Simulink merupakan bagian dari MATLAB untuk memodelkan, mensimulasikan, dan menganalisa sistem dinamik. Simulink dapat membentuk model dari awal atau memodifikasi model yang sudah ada sesuai dengan apa yang diinginkan. Selain itu simulink juga mendukung sistem linier dan non-linier, permodelan waktu kontinyu atau diskrit, atau gabungan. Simulink ini dapat digunakan sebagai media untuk

menyelesaikan masalah dalam industri nyata meliputi kedirgantaraan dan pertahanan, otomotif, komunikasi, elektronik dan pemrosesan sinyal,

Salah satu modul dalam Simulink yang dapat digunakan untuk komunikasi perangkat keras adalah *Instrument Control Toolbox*. Modul ini merupakan kunpulan fungsi *m-file* yang dibangun pada lingkungan komputasi teknis MATLAB. *Toolbox* ini menyediakan kerangka kerja untuk komunikasi instrumen yang mendukung GPIB *interface,* standar VISA, TCP/IP, dan protokol UDP. *Toolbox* ini memperluas fitur dasar *serial port* yang ada dalam MATLAB. Selain itu *toolbox* ini berfugsi untuk komunikasi data antara *workspace* MATLAB dan peralatan lainnya. Data tersebut dapat berbentuk biner atau *text*.

Komunikasi *serial* merupakan protokol dasar tingkat rendah untuk komunikasi antara dua peralatan atau lebih. Pada umumnya satu komputer dengan modem, *printer*, mikrokontroler, atau peralatan lainnya. *Serial port* mengirim dan menerima informasi *bytes* dengan hubungan seri. *bytes* tersebut dikirimkan menggunakan format biner atau karakter ASCII (*American Standard Code for Information Interchange*). Dalam komunikasi serial MATLAB, agar data ASCII dapat diproses *real time*, maka digunakan ASCII *encode* dan *decode* yang terdapat pada *xPC Target Library for RS232.* ASCII *encode* merupakan blok dalam simulink yang digunakan untuk mengubah data *bytes* menjadi karakter ASCII. Sedangkan ASCII *decode* merupakan blok Simulink yang digunakan untuk mengubah karakter ASCII menjadi data *bytes* yang kemudian dapat dikonversi sesuai kebutuhan.

2.5 Identifikasi Sistem [7]

Dalam mendesain suatu kontroler dan melakukan simulasi dibutuhkan suatu model matematika dari *plant* oleh karena itu dilakukan identifikasi sistem. Menurunkan suatu model matematika yang baik dan sesuai merupakan salah satu bagian terpenting dalam proses menganalisa sebuah sistem secara keseluruhan. Model matematika yang baik dan sesuai dibutuhkan untuk analisa, prediksi, dan desain sistem, *regulator*, dan *filter*. Model matematika memiliki bentuk yang bermacam-macam. Salah satu bentuknya adalah *transfer function* yang cocok untuk permasalahan analisa transien dan sebuah sistem LTI (*Linear Time Invariant*) dan sistem dengan *Single-Input Single-Output* (SISO). Disisi lain, model matematika dengan bentuk *state space* sangat cocok untuk menganalisa suatu sistem dengan *Multiple-Input Multiple-Output* (MIMO).

Identifikasi sistem pada suatu *plant* dapat dilakukan dengan menggunakan metode identifikasi statis maupun dinamis. Identifikasi statis dilakukan untuk mendapatkan *gain* dan *time sampling* dari suatu *plant* sedangkan identifikasi dinamis digunakan untuk mendapatkan model matematika dari suatu *plant* yang menggambarkan hubungan antara *input* dan *output* pada kondisi berbeban.

2.5.1 Prosedur Identifikasi Sistem

Identifikasi sistem adalah pendekatan eksperimental untuk menentukan model dinamis dari sebuah sistem. Terdapat beberapa langkah dalam identifikasi sistem yang pertama adalah melakukan akuisisi data *input-output* kemudian menghitung estimasi dari struktur model dan mengestimasi parameter model lalu langkah terakhir yaitu melakukan validasi pada model yang telah teridentifikasi.

Pada akuisisi data *input-output*, sinyal *input* yang dipilih harus memiliki spektrum frekuensi yang kaya. Umumnya, sinyal *input* yang dipilih berupa sinyal PRBS. Masalah yang sering timbul saat melakukan identifikasi sistem adalah menentukan derajat dari polinomial (*numerator* dan *denumerator*) dari fungsi alih yang merepresentasikan model *plant*. Prosedur *trial* dan *error* banyak digunakan dalam penentuan kompleksitas model.

2.5.2 *Root Mean Square Error*

Persamaan permodelan digunakan pada banyak bidang ilmu, di mana variabelnya harus diukur dengan ada tidaknya *error*. Tujuannya adalah agar nilai pada permodelan mendekati nilai *real* sistem.

Untuk melakukan validasi model dapat digunakan beberapa tolak ukur. Salah satunya adalah RMSE. Dalam penggunaanya, harus ditentukan terlebih dahulu jumlah data dan interval toleransi RMSE. Jika nilai RMSE kecil, menunjukkan bahwa nilai parameter sesuai dengan apa yang diinginkan sebelumnya. [7]

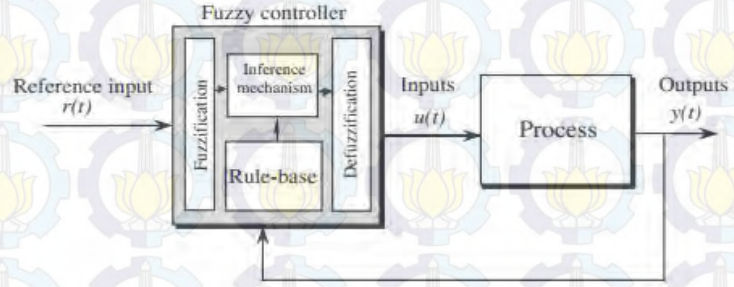
Dengan e adalah *error* antara nilai estimasi dan nilai sebenarnya dan n adalah jumlah data.

$$RMSE = \sqrt{\frac{\sum e^2}{n}} \tag{2.1}$$

2.6 *Fuzzy Logic* [8]

Dasar dari *fuzzy logic* adalah himpunan *fuzzy*. Zadeh memperkenalkan konsep himpunan *fuzzy* sebagai perluasan dari himpunan konvensional. Suatu himpunan *fuzzy* adalah koleksi dari bilangan real yang memiliki keanggotaan parsial dalam himpunan.

Saat ini logika *fuzzy* sudah banyak digunakan dalam berbagai bidang, salah satunya adalah dalam bidang kontrol (proses kontrol). Secara umum, sistem *fuzzy* terdiri dari beberapa komponen, yaitu *fuzzification*, *fuzzy rule base*, *fuzzy inference mechanism* atau *fuzzy inference engine*, dan *defuzzifier*, seperti diperlihatkan pada gambar berikut.



Gambar 2.11 *Fuzzy Controller*.

Yang menjadi inti dari logika *fuzzy* adalah *fuzzy rule base*, yang berisi pernyataan-pernyataan logika. *Fuzzy inference engine* merupakan komponen *fuzzy* yang menarik suatu kesimpulan berdasarkan pada *input* dan *rule base* yang dibuat. *Fuzzifier* digunakan untuk memetakan nilai/harga variabel di dunia nyata kedalam himpunan *fuzzy* (*fuzzy sets*), sedangkan *defuzzifier* mengembalikan hasil perhitungan *fuzzy* (himpunan *fuzzy*) menjadi *input* ke proses yang dikontrol.

2.6.1 *Fuzzy Rule Base*

Fuzzy rule base berisi pernyataan-pernyataan logika *fuzzy* (*fuzzy statement*), yang berbentuk pernyataan IF-THEN. Bentuk umum pernyataan *fuzzy* adalah :

$$\text{IF } x_1 \text{ is } A_1^1 \text{ and } \ldots \text{ and } x_n \text{ is } A_n^1 \text{ THEN } y \text{ is } B^1 \qquad (2.2)$$

A_1^l dan B^l adalah himpunan *fuzzy*, sedangkan $x = (x_1, x_2, \dots, x_n)^T$ dan y adalah *input* dan *output* dari *variable fuzzy*.

2.6.2 *Fuzzy Inference Mechanism*

Fuzzy inference mechanism merupakan suatu modul pembuat keputusan dalam kontroler *fuzzy* yang memiliki fungsi untuk menentukan suatu kesimpulan berdasarkan pada seberapa besar pengaruh setiap *rule* dalam *rule base* berdasarkan pada nilai *input* yang masuk.

2.6.3 *Fuzzification*

Fuzzifier digunakan untuk memetakan nilai/harga variabel di dunia nyata kedalam himpunan *fuzzy* (*fuzzy sets*). Pemetaannya dilakukan dengan menggunakan fungsi yang disebut fungsi keanggotaan. Terdapat beberapa metode *fuzzifier*, 3 diantaranya yaitu *singleton fuzzifier*, *gausian fuzzifier* dan *triangular fuzzifier*. Berikut adalah formulanya.

a. *Singleton fuzzyfier*

$$\mu A'(x) = \begin{cases} 1 \; if \; x = x* \\ 0 \; otherwise \end{cases} \tag{2.3}$$

b. *Gaussian fuzzifier*

$$\mu A'(x) = e^{-\left(\frac{x_1 - x_1^*}{a_1}\right)} * \dots . * e^{-\left(\frac{x_n - x_n^*}{a_n}\right)} \tag{2.4}$$

c. *Triangular fuzzyfier*

$$\mu A'(x) = \begin{cases} \left(1 - e^{-\left(\frac{x_1 - x_1^*}{b_1}\right)} * \dots . * e^{-\left(\frac{x_n - x_n^*}{b_n}\right)} \; if \; |x_n - x_n^*| \leq b_i, i = 1,2, \dots, n\right) \\ 0 \; otherwise \end{cases} \tag{2.5}$$

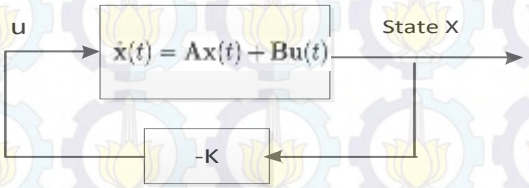
2.6.4 *Defuzzifier*

Defuzzifier mengembalikan hasil perhitungan *fuzzy* (himpunan *fuzzy*) menjadi variabel yang sesuai rentangnya di dunia nyata. Sama dengan *fuzzifier*, *defuzzifier* juga menggunakan fungsi keanggotaan untuk memetakan nilai himpunan *fuzzy* menjadi variabel nyata. Terdapat beberapa metode *defuzzifier*, 3 diantaranya yaitu:

a. *Center of gravity defuzzifier*. *Center of gravity* yang dinyatakan dengan y*, menunjukan pusat area yang diliputi oleh fungsi keanggotaan.
b. *Center average dufuzzifier*. *Center average* menunjukan *weight average* dari titik tengah (*center*) masing-masing fungsi keanggotaan.
c. *Maximum defuzzifier*. *Maximum defuzzifier* memilih nilai tertinggi sebagai y*. Ada 3 pilihan, *smallest of maxima*, *largest of maxima* atau *mean of maxima*.

2.7 Kontroler *Linear Quadratic Regulator* (LQR) [9]

LQR adalah suatu metode kontrol dalam teori kontrol *modern* yang menggunakan pendekatan persamaan *state* untuk menganalisa suatu sistem. Metode LQR menggunakan pendekatan *gain state feedback* (K) dalam mendesain suatu kontroler. Berikut ini merupakan diagram blok dari kontroler LQR.



Gambar 2.12 Diagram Blok Kontroler LQR.

Permasalahan kontrol optimal regulator adalah misal di berikan model *state* sistem :

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.7)$$
$$y(t) = Cx(t) + Du(t)$$

Dimana u(t) merupakan sinyal kontrol dan x(t) merupakan variabel *state* sistem lalu menentukan matriks *gain* K dari vektor kontrol optimal

$$u(t) = -Kx(t) \quad (2.8)$$

Sehingga meminimalkan indeks performansi pada Persamaan 2.9.

$$J = \int_0^t (x^T(t)\, Qx(t) + u^T(t) Ru(t))\, dt \quad (2.9)$$

Dimana Q adalah suatu matriks positif-semi definit dan R adalah suatu matriks definit positif. Matriks Q dan R sering disebut bobot matriks *state* dan bobot matriks kontrol. Elemen dari matriks K ditentukan sehingga meminimalkan indeks performansi pada Persamaan 2.9.

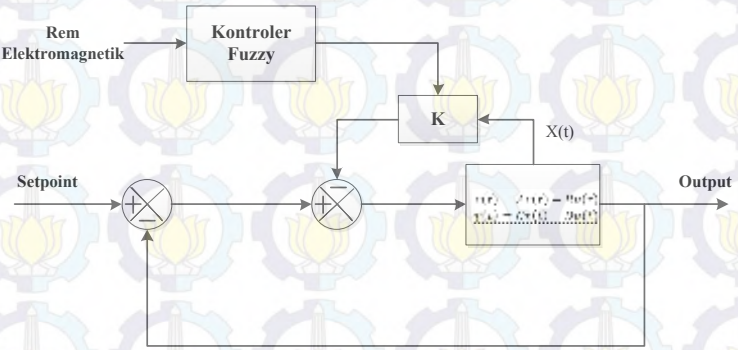
Maka $u(t) = -Kx(t) = -R^{-1}B^{T}Px(t)$ adalah optimal untuk segala *state* awal x(0) dimana P(t) adalah solusi dari persamaan riccati,K adalah matriks *feedback* optimal untuk mendapatkan matriks P(t) maka harus menyelesaikan persamaan riccati berikut :

$$A^{T}P + PA - PBR^{-1}B^{T}P + Q = 0 \qquad (2.10)$$

2.8 Kontroler *Fuzzy* LQR [10]

Dalam kontroler *fuzzy* LQR, kontroler *fuzzy* digunakan untuk mengatur nilai *gain state feedback* dari kontroler LQR berdasarkan *input* yang diberikan. Gambar 2.13 merupakan diagram kontrol sistem pengaturan *fuzzy* LQR. Pada diagram tersebut bisa dilihat bahwa struktur kontrol merupakan struktur konvensional tetapi bedanya kontroler ini dapat diubah parameter *gain state feedback* (K) dengan menggunakan kontroler *fuzzy*. Kontroler *fuzzy* ini mendapatkan *input* yang berupa informasi ekstra dari luar. Informasi ekstra ini dapat berupa informasi perubahan pada *plant* atau informasi gangguan pada *plant* yang dapat merubah fungsi alih dari sistem.

Kontroler *fuzzy* LQR ini didesain untuk memenuhi kebutuhan kriteria performansi yang berbeda ketika terjadi perubahan pada model *plant* misalnya pemberian beban pada motor BLDC. Ketika diberi beban maka model *plant* akan berubah sehingga dibutuhkan *gain state feedback* yang berbeda, disinilah kontroler *fuzzy* bekerja dengan mengubah nilai *gain* kontroler LQR sesuai dengan besar pembebanan yang diberikan.



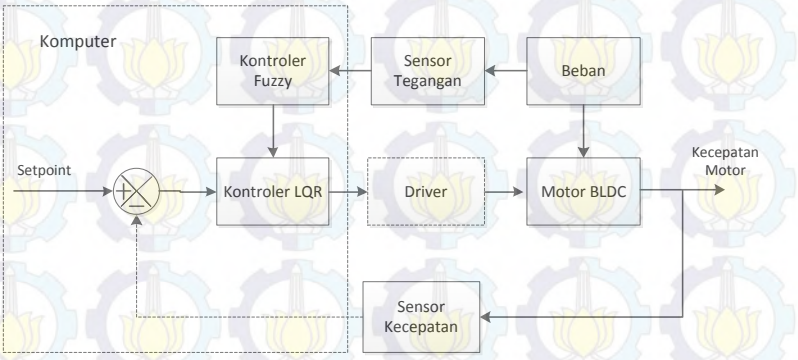
Gambar 2.13 Diagram Kontrol *Fuzzy* LQR.

Kontroler *fuzzy* LQR ini mengatur nilai *gain state feedback* berdasarkan pada *rule base* kontroler *fuzzy* yang mendapatkan *input* berupa besar pembebanan yang diberikan. Pada tugas akhir ini kontroler yang digunakan merupakan kontroler *fuzzy* LQR. Kontroler *fuzzy* LQR ini bekerja berdasarkan hubungan diantara *gain state feedback* dan *input* dari kontroler *fuzzy* yang merepresentasikan besar gaya rem magnetik yang diberikan sehingga berdasarkan nilai beban tersebut kontroler *fuzzy* dapat mengubah nilai dari *gain state feedback* supaya dapat memenuhi spesifikasi performansi yang diinginkan.

BAB 3
PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

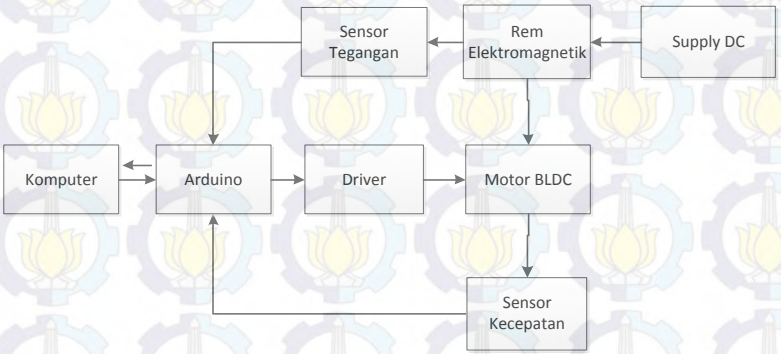
Dalam tugas akhir ini penulis merancang sistem pengaturan kecepatan motor BLDC seperti yang ditunjukkan pada Gambar 3.1. Sistem tersebut merupakan sistem kontrol negatif *feedback* yang tersusun atas komponen berikut motor BLDC sebagai *plant* yang akan dikontrol kecepatannya, beban mekanik berupa rem elektromagnetik yang diberikan pada motor untuk memberikan efek pembebanan pada motor BLDC selain itu terdapat *driver* motor BLDC yang digunakan sebagai aktuator yang menjembatani antara kontroler dengan *plant*. Sinyal kontrol dari kontroler berupa sinyal PWM yang digunakan sebagai masukan *driver* untuk mengontrol kecepatan motor BLDC, kontroler yang digunakan adalah kontroler *fuzzy* LQR bekerja melalui komputer dengan menggunakan *software* MATLAB dan menggunakan Arduino sebagai *interface* antara komputer dengan *driver* motor BLDC. Dalam sistem pengaturan kecepatan motor BLDC ini juga terdapat dua sensor yaitu sensor kecepatan yang digunakan sebagai negatif *feedback* untuk melihat kecepatan motor secara aktual dan sensor tegangan yang digunakan oleh kontroler *fuzzy* untuk membaca tegangan *supply* pada rem elektromagnetik. Sensor tegangan ini berupa rangkaian pembagi tegangan.



Gambar 3.1 Blok Diagram Sistem Pengaturan Kecepatan Motor BLDC.

3.2 Perancangan Perangkat Keras

Pada tahap perancangan simulator BLDC, perangkat keras dibagi menjadi 2 golongan yaitu perangkat mekanik dan perangkat elektronik. Komponen yang termasuk perangkat mekanik antara lain motor BLDC, kopel karet, dan rem elektromagnetik. Komponen yang termasuk perangkat elektronik antara lain, *driver* motor BLDC, *supply* DC untuk rem elektromagnetik, sensor tegangan rem elektromagnetik, Arduino, dan komputer. Secara umum susunan perangkat dalam simulator BLDC seperti terlihat pada Gambar 3.2.



Gambar 3.2 Konfigurasi Perangkat Keras Simulator BLDC.

Arduino bertugas untuk menerima data dari sensor kecepatan dan sensor tegangan kemudian mengirimkan data tersebut kedalam komputer lalu Arduino akan mengeluarkan *output* berupa sinyal PWM ke *driver* motor BLDC. Terdapat pula rangkaian sensor yang berfungsi mengukur *input* tegangan yang masuk ke dalam rem elektromagnetik dan sensor kecepatan motor BLDC yang menggunakan teknik *sensorless*.

3.2.1 Perancangan Mekanik

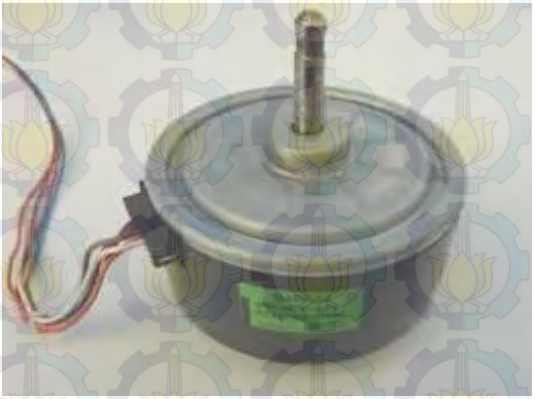
Pada *plant* sistem pengaturan kecepatan motor BLDC terdapat beberapa komponen utama yang digunakan antara lain motor BLDC sebagai objek yang dikontrol. Selain itu, terdapat rem elektromagnetik yang digunakan untuk memberikan efek pembebanan pada motor BLDC. Untuk lebih jelasnya, penjelasan mengenai konstruksi motor BLDC dan rem elektromagnetik dapat dilihat pada subbab di bawah ini.

3.2.1.1 Motor Brushless DC

Motor *Brushless* DC (BLDC) yang digunakan merupakan motor BLDC yang digunakan pada AC *inventer* Daikin. Motor BLDC jenis ini merupakan tipe dari motor BLDC yang menggunakan teknik *sensorless* yang memiliki 5 kabel untuk *supply* motor, *supply driver* motor, sinyal kontrol, sensor kecepatan dan *ground*. Bentuk fisik motor BLDC yang digunakan pada *plant* ini dapat dilihat pada Gambar 3.3. Sedangkan spesifikasi motor BLDC dapat dilihat pada Tabel 3.1 di bawah ini.

Tabel 3.1 Spesifikasi Motor BLDC Daikin D43F.

Parameter		Nilai
Tipe		Daikin D43F
Berat Motor		1200 gram
Tegangan Kerja		307 VDC
Kecepatan Motor	Beban Minimal	2410 rpm
	Beban Nominal	1502 rpm
	Beban Maksimal	1433 rpm



Gambar 3.3 Motor *Brushless* DC Tipe *Inrunning* dengan Tipe Daikin D43F.

3.2.1.2 Rem Elektromagnetik

Rem elektromagnetik pada *plant* ini terpasang pada poros motor BLDC yang berguna sebagai pembebanan pada motor. Medan elektromagnetik dari rem ini dihasilkan oleh delapan kumparan yang dihubungkan secara seri dan diberikan masukan arus DC. Daya DC yang masuk ke dalam rem elektromagnetik ini diberikan oleh *power supply* yang memiliki rentang tegangan antara 16-24V, resistansi rem elektromagnetik adalah tetap sehingga pengaturan besar magnetisasi yang dihasilkan dapat diatur melalui tegangan *supply* yang diberikan karena tegangan proporsional terhadap arus yang mengalir pada rem elektromagnetik tersebut.

Rem elektromagnetik ini terbuat dari 8 kumparan terdiri dari 4 kumparan di setiap sisinya yang disusun secara seri. Diantara celah kedua sisi kumparan dipasang piringan aluminium. Piringan aluminium tersebut kemudian dipasang ke dalam sebuah *shaft* yang selanjutnya dikopel dengan motor BLDC. Spesifikasi dari rem elektromagnetik adalah sebagai berikut :

a. Jumlah kumparan : 8 (masing-masing 400 lilitan)
b. Resistansi kumparan : 12 Ω
c. Diameter kumparan : 3 cm
d. Diameter kawat : 0,6 mm
e. Tegangan kerja : 0-30 V
f. Arus maksimal : 2 A

3.2.1.3 Kopel

Pada konstruksi simulator BLDC, *shaft* motor BLDC dipasang seporos dengan rem elektromagnetik. Oleh karena itu konstruksi penahan *shaft* pada rem elektromagnetik harus benar-benar lurus dengan shaft motor BLDC. Jika kedua *shaft* tidak lurus maka dapat menyebabkan kerusakan pada *bearing*. Bahkan *shaft* bisa bengkok jika kedua *shaft* diputar pada keadaan tidak lurus namun untuk menjaga kedua *shaft* ini agar tetap seporos sangat sulit. Hal ini dikarenakan saat motor berputar, akan timbul getaran pada motor sehingga menyebabkan kedua *shaft* tidak lurus. Untuk menghindari hal ini perlu dipasang kopel yang menghubungkan *shaft* motor BLDC dengan *shaft* rem elektromagnetik. Pemasangan kopel sendiri perlu diperhatikan dengan baik agar kopel terpasang dengan benar supaya *shaft* tidak bergetar saat motor berputar.

3.2.2 Perancangan Elektronik

Perancangan elektronik terdiri dari beberapa bagian, antara lain rangkaian sensor kecepatan motor BLDC, rangkaian sensor tegangan rem elektromagnetik, dan rangkaian *driver* motor BLDC selain itu akan dibahas juga mengenai Arduino yang berfungsi sebagai *interface*.

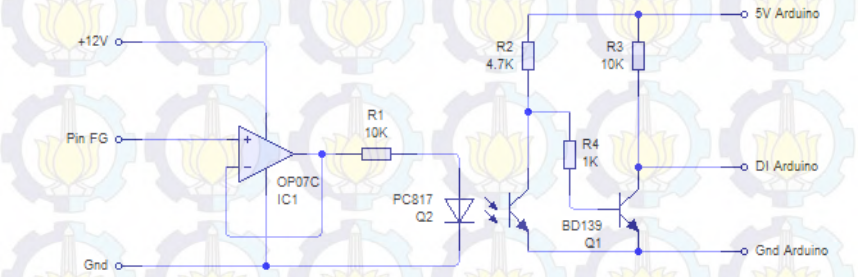
3.2.2.1 Rangkaian Sensor Kecepatan Motor BLDC

Rangkaian sensor kecepatan BLDC digunakan untuk membaca kecepatan motor BLDC. Pada motor BLDC yang digunakan pengukuran kecepatan menggunakan teknik *sensorless* yaitu dengan membaca ggl balik pada setiap belitan fasa untuk mendeteksi posisi *rotor*. Pada pin *frequency generator* (FG), rangkaian *driver* menghasilkan sinyal kotak dengan *duty cycle* tetap 50% dan frekuensi sinyal yang berubah-ubah proporsional terhadap kecepatan motor BLDC. Untuk membaca sinyal kotak pada pin FG maka digunakan Arduino dengan rangkaian *signal conditioning* seperti ditunjukkan pada Gambar 3.4.

Perhitungan kecepatan motor dapat dilakukan dengan menghitung frekuensi gelombang kotak yang dihasilkan pin FG. Untuk mendapatkan kecepatan motor dapat digunakan Persamaan 3.1 berikut.

$$N = \frac{60FG}{P} \qquad (0.1)$$

N : Kecepatan motor (rpm)
FG : *Frequency generator* (Hz)
P : Jumlah pasang kutub



Gambar 3.4 Rangkaian Sensor Kecepatan Motor BLDC.

Rangkaian pembaca sensor kecepatan ini terdiri dari *optocoupler* untuk isolasi antara pin FG dengan pin di Arduino dan *op amp* yang digunakan sebagai *buffer* untuk mengurangi pelemahan sinyal dari pin FG.

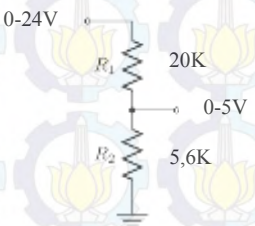
3.2.2.2 Mikrokontroler Arduino

Pada simulator BLDC ini, Mikrokontroler Arduino yang digunakan adalah tipe UNO R3. Arduino digunakan untuk mengakuisisi data kecepatan motor BLDC dan *supply* tegangan rem elektromagnetik. Tegangan yang masuk ke Arduino akan diubah menjadi data digital dengan resolusi 10 bit dan akan diolah oleh program kontroler pada komputer. Kemudian kontroler tersebut akan menghasilkan sinyal kontrol berbentuk pwm yang dikeluarkan melalui pin *output* digital pwm pada Arduino untuk menjadi sinyal kontrol pada *plant* motor BLDC. Pengiriman data antara komputer dan Arduino ini dilakukan melalui *port* komunikasi *serial*. Berikut ini pemilihan pin *input*/*output* pada Arduino,

a. Pin A0 : *input* analog dari rangkaian pembagi tegangan.
b. Pin 7 : *output* PWM untuk *driver* motor BLDC.
c. Pin 6 : *input* digital sensor kecepatan.
d. Pin +5V
e. Pin *Ground*

3.2.2.3 Rangkaian Pembagi Tegangan

Rangkaian pembagi tegangan pada Gambar 3.5 digunakan untuk membaca tegangan yang diberikan pada rem magnetik. Dalam pembebanan yang diberikan memiliki rentang tegangan dari 0-24V, tegangan ini nantinya akan diubah kedalam rentang tegangan 0-5V oleh rangkaian pembagi tegangan supaya dapat dibaca dengan mikrokontroler Arduino .

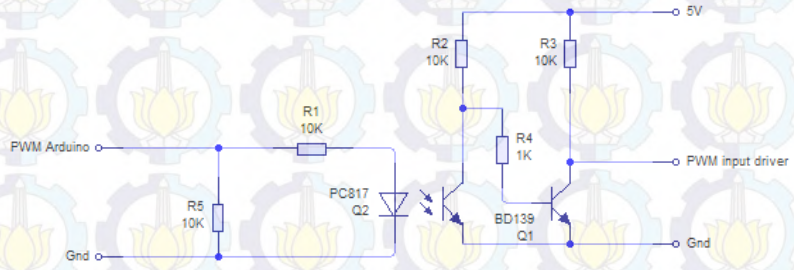


Gambar 3.5 Rangkaian Pembagi Tegangan.

3.2.2.4 Driver Motor BLDC [11]

Pada motor BLDC, *driver* berfungsi mengubah sinyal kontrol dari kontroler yang tegangan analog (PWM) menjadi urutan komutasi kumparan pada motor BLDC. Bentuk fisik dari rangkaian *driver* motor BLDC dapat ditunjukkan pada Gambar 3.6. Untuk dapat mengatur kecepatan motor BLDC dapat dilakukan dengan dua cara yaitu memberi sinyal PWM pada pin *input* PWM atau mengatur nilai tegangan pada pin *input* Vcc. Pada tugas akhir kali ini dipilih pengaturan kecepatan dengan memberikan sinyal PWM. Sinyal PWM ini akan diberikan oleh Arduino dengan rangkaian isolasi seperti pada Gambar 3.7.

Gambar 3.6 Rangkaian *Driver* Motor BLDC.



Gambar 3.7 Rangkaian Isolasi Arduino dengan *Driver* BLDC.

Driver ini menggunakan teknik *sensorless* dalam menggerakkan motor BLDC jadi tidak menggunakan sensor *hall effect* maupun sensor kecepatan tetapi dengan mendeteksi sinyal ggl balik. Dengan mengamati

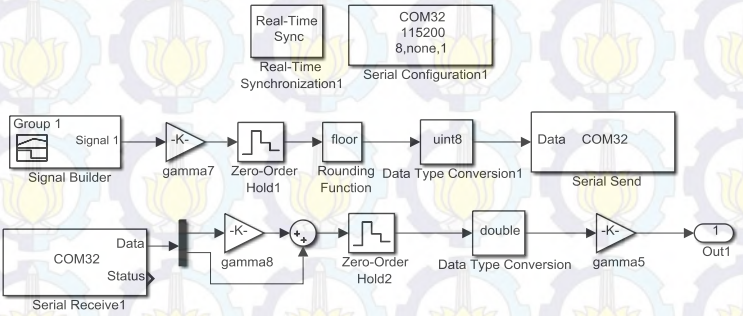
sinyal ggl balik yang timbul pada kumparan *stator* maka posisi dan kecepatan motor dapat diperkirakan.

3.3 Perancangan Perangkat Lunak

Dalam sistem pengaturan motor BLDC, beberapa perangkat lunak harus digunakan untuk proses pengambilan data, perancangan kontroler, dan pengiriman data. Perangkat lunak yang digunakan yaitu MATLAB dan *software* Arduino.

3.3.1 Perangkat Lunak MATLAB

Perangkat lunak ini digunakan pada proses pengambilan data dan pengiriman data. Dengan menggunakan komunikasi *serial* dengan Arduino, Simulink MATLAB dapat mengolah data dari blok *serial receive* dan mengirimkannya kembali melalui blok *serial send.* Contoh diagram blok Simulink MATLAB untuk pengambilan data *open loop* ditunjukkan pada gambar berikut.



Gambar 3.8 Blok Simulink *Open Loop.*

3.3.2 Perangkat Lunak Arduino

Mikrokontroler Arduino dalam sistem pengaturan motor BLDC ini digunakan sebagai *interface* antara sensor, dan *output* ke motor BLDC dengan Simulink MATLAB. Artinya, Arduino ini berfungsi sebagai media pengirim data yang menghubungkan antara sensor dan *plant* dengan Simulink MATLAB. Data yang diterima oleh *serial receive* pada Simulink MATLAB merupakan data *integer* 8 bit . Dari data integer yang dikirim ini terdiri dari 2 variabel yaitu nilai kecepatan pada sensor yang memiliki satuan rpm dan data pada rem magnetik. Dan Arduino akan menerima data sinyal kontrol yang dikirim dari MATLAB melalui *serial*

send dengan range sinyal kontrol 0 sampai 255 yang menunjukkan *duty cycle* sekitar 0% sampai 100%.

Selanjutnya data dari kecepatan motor BLDC yang berupa sinyal kotak dengan *duty cycle* 50% dan memiliki frekuensi yang berbeda – beda akan dibaca dengan menggunakan fungsi PulseIn( ). Fungsi ini akan mengembalikan sebuah nilai pada sebuah variabel yang telah di deskripsikan. Pertama-tama fungsi ini akan membaca nilai *high* atau *low* pada sebuah pin, pada sistem ini dipakai pin 8 dan yang dibaca pada pin merupakan *pulse high* lalu apabila pin 7 membaca nilai *high*, *timer* akan berjalan dan akan menunggu hingga pin 7 membaca nilai *low*. Apabila pin 7 membaca nilai *low* maka *timer* akan berhenti dan variabel yang telah disebutkan akan mendapatkan nilai waktu lamanya pulsa berada pada keadaan *high*. Disinilah data kecepatan motor bisa didapatkan, dari waktu tersebut didapatkan nilai kecepatan motor dalam satuan rpm. Dan yang terakhir data besar tegangan yang disuplai pada rem magnetik. Arduino dapat bekerja berdasarkan pada *script* yang telah dirancang dengan menggunakan Arduino IDE. Program yang telah dibuat bisa di *upload* pada Arduino dengan menggunakan Arduino IDE. Berikut tampilan dari Arduino IDE.

Gambar 3.9 Tampilan IDE Arduino.

3.4 Identifikasi dan Pemodelan Sistem

Pada pemodelan sistem di tugas akhir kali ini digunakan identifikasi dinamis. Metode dinamis ini dilakukan dengan cara memberikan *input* berapa sinyal *Pseudorandom Binary Sequence* (PRBS). Setelah itu *output* kecepatan akan diidentifikasi dengan menggunakan metode ARX yang ada pada *toolbox* MATLAB. Setelah hal ini dilakukan didapatkanlah model matematika dari *plant*.

3.4.1 Metode dan Pembebanan Sistem

Respons *plant* dapat dilihat dari beberapa cara. Salah satu cara yaitu dengan memberikan pembebanan. Pada *plant* motor BLDC pembebanan dilakukan dengan menggunakan rem elektromagnetik. Pembebanan dilakukan untuk mendapatkan model sistem dengan beban yang ditentukan. Pada sistem ini rem magnetik di berikan tegangan *supply* yang bervariasi mulai dari *input* 16V, 20V, dan 24V. Beban-beban tersebut disebut dengan beban minimal, nominal, dan maksimal.

3.4.2 Metode Identifikasi dan Pemodelan

Identifikasi pada motor BLDC dilakukan dengan menggunakan identifikasi dinamis. Hal ini dilakukan dengan cara memberikan *input* dengan menggunakan sinyal PRBS. Pada tugas akhir ini sinyal PRBS yang digunakan merupakan sinyal kotak yang memiliki batas bawah dan batas atas.

Setelah didapatkan data dari *input* dan *output* motor maka tahap selanjutnya adalah melakukan identifikasi dari data yang telah didapatkan. Metode ini dilakukan dengan *toolbox identification system* yang ada pada program MATLAB. Berikut hasil fungsi alih yang didapatkan setelah dilakukan identifikasi menggunakan metode ARX.

Tabel 3.2 Fungsi Alih *Plant* Motor BLDC.

Beban	Tegangan *input* beban	Fungsi Alih	RMSE (%)
Minimal	16 Volt	$\frac{98{,}578}{s^2 + 25{,}3125s + 98{,}578}$	5,31
Nominal	20 Volt	$\frac{122{,}2386}{s^2 + 28{,}9879s + 134{,}5795}$	4,89
Maksimal	24 Volt	$\frac{232{,}7394}{s^2 + 49{,}9617s + 273{,}8755}$	3,41

3.5 Perancangan Kontroler *Fuzzy* LQR

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk masing-masing pembebanan. Kontroler digunakan untuk mengembalikan respon ke nilai *set point* yang diinginkan sekalipun motor BLDC diberi beban. Tahapan desain kontroler ini meliputi perancangan kontroler LQR dan perancangan kontroler *fuzzy*.

3.5.1 Spesifikasi Performansi

Tahap awal dalam merancang suatu kontroler adalah menentukan spesifikasi performansi yang diinginkan. Secara umum spesifikasi performansi ada dua yaitu spesifikasi respon *transient* dan spesifikasi respon *steady state*, spesifikasi respon *transient* merupakan spesifikasi respon yang diamati mulai saat terjadinya perubahan *input*/gangguan sampai respon masuk dalam keadaan *steady state* dengan tolak ukur yang digunakan adalah nilai *settling time*, *rise time* dan persentasi *overshoot* lalu untuk spesifikasi respon *steady state,* spesifikasi respon sistem yang diamati mulai saat respon masuk dalam keadaan *steady state* sampai waktu tak terbatas tolak ukur yang digunakan adalah persentasi *error steady state.*

Dalam sistem pengaturan kecepatan motor BLDC ini ditetapkan spesifikasi respon seperti yang terlihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi Performansi yang Diinginkan.

Karakteristik Respon	Nilai yang Diinginkan
Time Constant	0,87 detik
Settling Time	2,6 detik
Rise Time	1,9 detik
Overshoot	0 %
Error Steady State	0 %

Spesifikasi respon diatas merupakan spesifikasi respon sistem orde 1 yang dapat dinyatakan dengan fungsi alih pada Persamaan 3.2.

$$\frac{Y(s)}{X(s)} = \frac{1}{0{,}87s + 1} \tag{3.2}$$

Setelah menentukan spesifikasi performansi yang diinginkan maka langkah selanjutnya adalah mendesain kontroler supaya dapat mencapai kriteria yang diinginkan tersebut.

3.5.2 Perancangan Kontroler LQR

Sebelum merancang kontroler *fuzzy* harus dicari dahulu kontroler LQR yang sesuai untuk setiap pembebanan. Dengan merancang kontroler LQR ini maka akan didapatkan nilai *gain state feedback* untuk setiap pembebanan, nilai ini nantinya akan dijadikan nilai fungsi keanggotaan untuk *output* pada kontroler *fuzzy*.

Pada tahap ini untuk dapat mencari *gain* dari kontroler LQR maka model *plant* yang telah didapatkan sebelumnya harus diubah kedalam bentuk persamaan *state*. Model *plant* dalam bentuk persamaan *state* adalah sebagai berikut.

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} 0 \\ k \end{bmatrix} u \qquad (3.3)$$

Kemudian didefinisikan suatu *state* baru (*augmented state*) $X_0 = \int X_1 \, dt$ atau $\dot{X}_0 = X_1$ maka persamaan *state* diatas menjadi seperti Persamaan 3.4.

$$\begin{bmatrix} \dot{X}_0 \\ \dot{X}_1 \\ \dot{X}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -b & -a \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k \end{bmatrix} u \qquad (3.4)$$

Penambahan *state* baru ini bertujuan agar sistem dapat menyerupai kerja dari kontroler PID karena terdapat *state* baru maka dalam perancangan kontroler LQR ini akan dicari 3 *gain state feedback* yaitu nilai *gain* K_1, K_2, dan K_3 supaya memenuhi spesifikasi peformansi yang diinginkan. Tabel 3.4 menujukkan nilai *gain state feedback* untuk setiap pembebanan.

Tabel 3.4 Nilai *Gain State Feedback* di Setiap Pembebanan.

Pembebanan	K_1	K_2	K_3
Minimal	1,348	0,633	0,024
Nominal	1,490	0,658	0,022
Maksimal	1,658	0,665	0,014

Gain K_1 memiliki cara kerja yang sama dengan kontroler integrator pada kontroler PID, *gain* K_2 sama dengan kontroler proporsional pada kontroler PID dan *gain* K_3 sama dengan kontroler derivatif pada kontroler PID. Hal ini memudahkan dalam melakukan *tuning* kontroler LQR. Untuk mencari nilai *gain state feedback* digunakan fungsi lqr pada MATLAB, Pemilihan matriks Q dan R dilakukan melalui pendekatan *trial and error* sampai respon dapat mencapai spesifikasi performansi yang diinginkan.

3.5.3 Perancangan Kontroler *Fuzzy*

Pada tahap perancangan kontroler *fuzzy* terdapat berberapa tahap lagi yaitu penentuan fungsi keanggotaan dan bentuk fungsi keanggotaan kemudian membuat *rule base* dan menentukan fungsi keanggotaan pada *output*.

Kontroler *fuzzy* pada sistem pengaturan motor BLDC ini berfungsi sebagai pengatur nilai *gain state feedback* K_1, K_2, dan K_3 pada kontroler LQR, sehingga pada sistem ini nilai *gain state feedback* akan menjadi fungsi keanggotaan pada *output* kontroler *fuzzy*. Pada *input fuzzy* terdapat 1 *input* yaitu besar beban pengereman yang diberikan.

3.5.3.1 Rule Base

Kontroler *fuzzy* digunakan untuk mengeluarkan nilai *gain state feedback* K_1, K_2, dan K_3 yang sesuai saat pembebanan kondisi tertentu maka dalam *rule base* yang dibuat terdapat 3 pernyataan di setiap *gain* K_1, K_2, dan K_3. Misalkan dipilih representasi fungsi keanggotaan *input* sebagai berikut:

Beban Minimal = BK
Beban Nominal = BM
Beban Maksimal = BH

Maka *rule base* untuk kontroler *fuzzy* akan terlihat seperti berikut.

If X = BK Then K_1 = 1,348
If X = BM Then K_1 = 1,490
If X = BH Then K_1 = 1,658

Pernyataan diatas merupakan *rule base* untuk pencarian nilai K_1. Bisa dilihat bahwa semakin besar pembebanan maka semakin besar nilai K_1 yang digunakan, *gain* K_1 ini bekerja seperti kontroler integrator pada kontroler PID. Selanjutnya adalah *rule base* pada pencarian *nilai* K_2.

If X = BK Then K_2 = 0,633
If X = BM Then K_2 = 0,658
If X = BH Then K_2 = 0,665

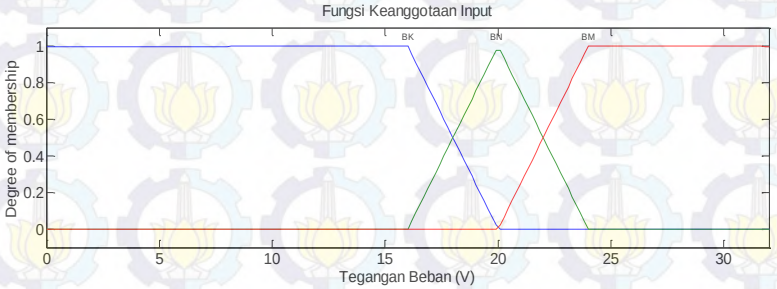
Dari *rule base* untuk pencarian nilai K_2 di setiap pembebanan ini dapat dilihat bahwa semakin besar nilai pembebanan maka semakin menurun nilai K_2 yang digunakan, *gain* K_2 ini bekerja seperti kontroler proporsional pada kontroler PID. Setelah mendapatkan *rule base* pada K_2 maka yang dilakukan selanjutnya adalah mendapatkan *rule base* untuk nilai K_3. Berikut ini merupakan *rule base* untuk nilai K_3.

If X = BK Then K_3 = 0,024
If X = BM Then K_3 = 0,022
If X = BH Then K_3 = 0,014

Pada *rule base* penentuan nilai *gain* K_3 untuk setiap pembebanan, *gain* K_3 ini bekerja seperti kontroler derivatif pada kontroler PID. Nilai *gain* K_3 menurun seiring naiknya nilai pembebanan yang diberikan.

3.5.3.2 Fungsi Keanggotaan

Pada tahap awal harus ditentukan fungsi keanggotaan untuk *input* pada kontroler *fuzzy* karena pada sistem pengaturan motor BLDC ini kontroler *fuzzy* digunakan untuk mengubah nilai *gain state feedback* K_1, K_2, dan K_3 pada setiap pembebanan, maka *input* fungsi keanggotaan harus berupa nilai yang merepresentasikan besarnya gaya rem yang dikeluarkan. Karena pada rem ini telah didapatkan data identifikasi pada pembebanan minimal, nominal, dan maksimal maka pada fungsi keanggotaan ini juga memiliki 3 anggota. Berikut ini fungsi keanggotaan dari *input kontroler fuzzy*.



Gambar 3.10 Fungsi Keanggotaan *Input*.

Pada Gambar 3.10 dapat dilihat bahwa jenis fungsi keanggotaan yang dipakai adalah jenis *triangular*. Jenis ini dipakai karena lebih mudah dalam perhitungannya. Pada gambar diatas merupakan fungsi

keanggotaan *input* kontroler *fuzzy* yang terdiri dari 3 *fuzzy set* yaitu BK, BN, dan BM.

Pada fungsi keanggotaan *output* terdapat 3 anggota pada setiap *gain* K_1, K_2, dan K_3. Bentuk dari fungsi keanggotaan hanya berupa konstanta atau *singleton* dari nilai *gain* K_1, K_2, dan K_3 yang telah didapatkan pada sub bab perancangan kontroler LQR. Untuk proses defuzzifikasi menggunakan metode *center average* yaitu nilai *gain* K_1, K_2, dan K_3 yang telah didapatkan dikalikan dengan nilai *weight* pada tiap *rule*-nya setelah itu hasil kali antara *weight* dengan nilai K_1, K_2, dan K_3 akan dijumlahkan dan dibagi dengan jumlah total *weight.*

BAB 4
PENGUJIAN DAN ANALISA

4.1 Gambaran Umum Pengujian Sistem

Pada tahapan ini akan dilakukan beberapa jenis pengujian yaitu pengujian sensor, pengujian *open loop* dari motor BLDC lalu pengujian kontroler yang disimulasikan pada hasil identifikasi model kemudian yang terakhir merupakan pengujian implementasi kontroler pada motor BLDC.

Pengujian pertama merupakan pengujian sensor kecepatan motor BLDC yang dilakukan dengan membandingkan hasil keluaran sensor kecepatan pada motor BLDC dengan hasil pengukuran dari tachogenerator. Pengujian kedua merupakan pengujian sistem *open loop* pada motor BLDC, hal ini dilakukan dengan memberi *input* berupa sinyal tangga pada motor BLDC lalu dilihat hasil respon kecepatan yang terbaca dan dilihat hubungannya. Pengujian ketiga merupakan simulasi sistem, hal ini dilakukan dengan cara memasang kontroler yang telah dibuat pada hasil identifikasi model dan yang terakhir merupakan implementasi kontroler pada motor BLDC.

4.2 Pengujian Sensor Kecepatan Motor BLDC

Pada motor BLDC yang digunakan untuk tugas akhir kali ini merupakan tipe motor *sensorless* sehingga tidak memerlukan sensor seperti sensor kecepatan dalam pengoperasiannya karena terdapat *feedback* dari motor yang memberikan informasi mengenai nilai kecepatan motor. Dalam tahap pengujian ini dilakukan pengecekan hasil pembacaan sensor kecepatan melalui kabel dari *driver* didalam motor BLDC yang mengeluarkan sinyal kotak dengan *duty cycle* tetap dengan frekuensi yang berbeda sesuai dengan kecepatan motor kemudian nilai ini akan dibandingkan dengan pengukuran menggunakan tachogenerator lalu dihitung nilai *error* untuk setiap kecepatan. Hasil pengujian sensor kecepatan ini dapat dilihat pada Tabel 4.1.

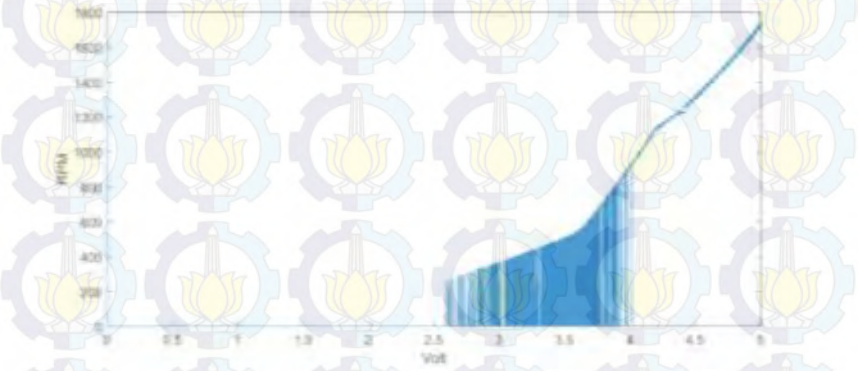
Dari hasil perbandingan pembacaan sensor kecepatan melalui Arduino dan tachogenerator didapatkan kesalahan pengukuran paling besar 1,84%. Karena *error* yang kecil masih dapat ditolerir maka disimpulkan bahwa pembacaan kecepatan dari sinyal pulsa yang dikeluarkan oleh *driver* motor dapat digunakan untuk menjadi sensor kecepatan motor BLDC.

Tabel 4.1 *Output* Pembacaan Kecepatan Motor.

Hasil Pembacaan		% *error*
Arduino	Tachogenerator	
972,36	984,6	1,25
1022,1	1035,41	1,30
1059	1070	1,11
1201,8	1218,04	1,35
1292,8	1310,42	1,36
1363,4	1382,67	1,41
1429,5	1448,82	1,35
1472,5	1491,69	1,30
1703,6	1726	1,31
2253	2290,45	1,67
2397	2441,21	1,84

4.3 Pengujian *Open Loop* Kecepatan Motor

Pengujian ini dilakukan untuk melihat hubungan antara *input* dan *output* dari motor BLDC. *Driver* BLDC diberikan masukan berupa sinyal tangga dan kemudian diukur kecepatannya. Berikut respon hasil pengujian *open loop* kecepatan motor.



Gambar 4.1 Respon Hubungan *Input Output* Kecepatan Motor.

Pada respon diatas, terdapat kesalahan pembacaan sensor pada saat tegangan *input driver* dibawah 4 Volt atau pada saat kecepatan motor dibawah 1000 rpm. Hal ini terjadi karena mekanisme *starting driver*

motor yang membutuhkan ggl balik untuk dapat berputar pada saat *starting* awal motor, ggl yang ditimbulkan masih rendah sehingga terjadi kesalahan pembacaan sensor karena untuk dapat mendeteksi kecepatan ini perlu untuk dapat mendeteksi ggl balik pada kumparan *stator*.

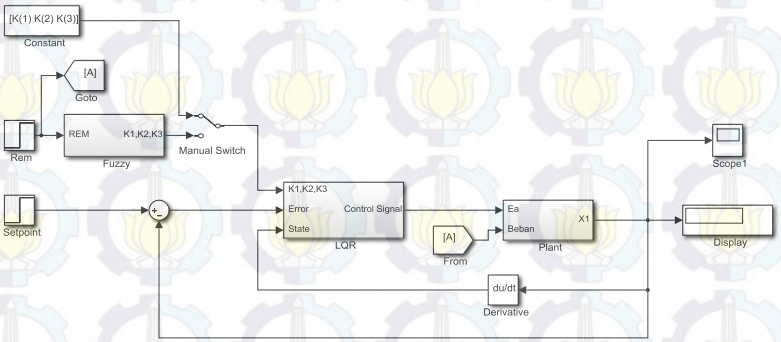
Pada kecepatan antara 1000-1600 nilai kecepatan bernilai linier terhadap besar tegangan yang diberikan oleh karena itu dalam tugas akhir ini dipilih rentang kerja motor diantara 1000-1600 rpm.

4.4 Simulasi Sistem

Pada tahapan ini dilakukan pengujian kontroler yang telah dirancang sebelumnya dengan menggunakan model *plant* hasil identifikasi. Simulasi merupakan salah satu hal yang harus dilakukan sebelum mengimplementasikan kontroler pada *plant*. Dengan hasil simulasi yang sesuai akan mempermudah dalam penerapan kontroler agar sistem mencapai kriteria yang diinginkan oleh perancang.

4.4.1 Diagram Blok Simulasi Sistem

Berikut ini merupakan diagram blok Simulink untuk simulasi dengan menggunakan kontroler.

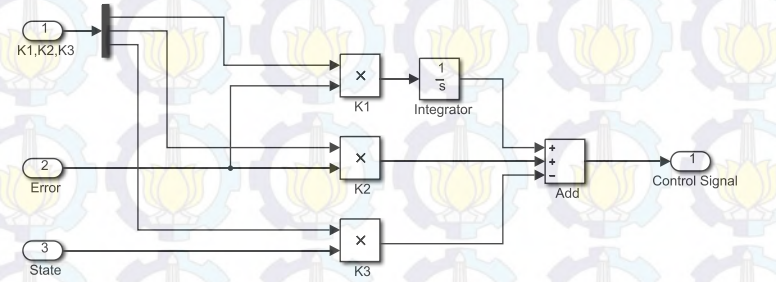


Gambar 4.2 Diagram Blok Simulink.

Pada Gambar 4.2 dapat dilihat bahwa program MATLAB untuk simulasi ini berupa sistem *close-loop* dengan *negative feedback*. Simulasi ini berupa *close-loop* karena terdapat *summing point* yang membandingkan nilai *set point* dengan nilai pembacaan kecepatan. Pada sistem ini terdapat dua kontroler yaitu kontroler *fuzzy* dan kontroler LQR. Kontroler *fuzzy* akan memberikan nilai *gain* pada kontroler LQR

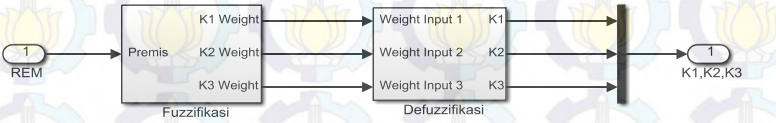
berdasarkan pada masukannya tegangan beban rem dengan model *plant* yang digunakan dalam bentuk persamaan *state*.

Pada Gambar 4.3 merupakan blok diagram untuk kontroler LQR. Bisa dilihat bahwa blok Simulink kontroler LQR pada gambar 4.3 memiliki 3 *gain* yang strukturnya dipisah antara *gain* K_1, *gain* K_2, dan *gain* K_3 hal ini digunakan untuk mempermudah kontroler *fuzzy* dalam mengubah parameter *gain* K_1, K_2, dan K_3.



Gambar 4.3 Blok Simulink Kontroler LQR.

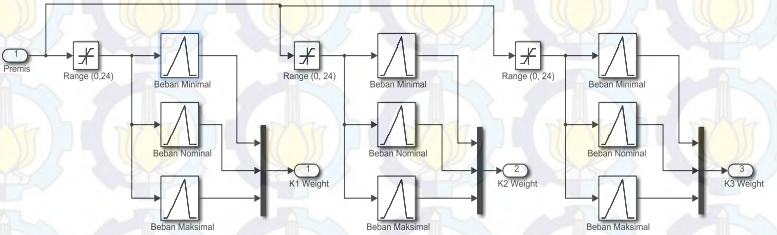
Lalu berikut ini merupakan blok diagram untuk kontroler *fuzzy*, pada blok diagram *fuzzy* terdiri atas dua tahapan yaitu fuzzifikasi yang mengubah nilai pembebanan yang masuk menjadi nilai bobot untuk setiap *gain* dan blok defuzzifikasi mendapatkan masukan berupa nilai bobot untuk setiap *gain* dan mengeluarkan *output* berupa *gain* K_1, K_2, dan K_3 untuk kontroler LQR.



Gambar 4.4 Blok Simulink *Fuzzy*.

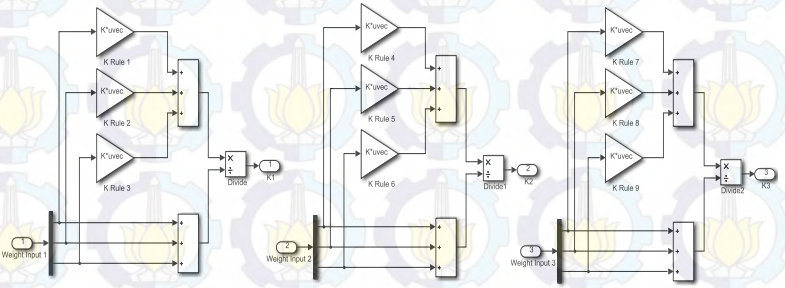
Seperti yang kita sudah ditulis sebelumnya *fuzzy* memliki 4 elemen yaitu *fuzzy rule base*, *fuzzy inference mechanism*, fuzzifikasi, defuzzifikasi. Pada blok Simulink di atas merupakan blok fuzzifikasi dan blok defuzzifikasi. Pada gambar berikutnya merupakan blok Simulink untuk fuzzifikasi dan defuzzifikasi.

Pada Gambar 4.5 terdapat tiga fungsi keanggotaan dalam blok fuzzifikasi yang masing-masing mewakili nilai dari beban. Pada setiap nilai K_1, K_2, dan K_3 terdapat tiga anggota yang merupakan fungsi keanggotaan untuk *input* rem.



Gambar 4.5 Blok Simulink Fuzzifikasi.

Terdapat 3 fungsi keanggotaan yaitu beban minimal, beban nominal, dan beban maksimal yang diwakili dengan fungsi keanggotaan yang berbentuk *triangular* yang memiliki nilai tengah 16V, 20V, dan 24V. Dari keluaran subsistem ini akan dihasilkan tiga nilai *weight* untuk setiap *input* di setiap nilai *gain* K_1, K_2, dan K_3 oleh karena itu *output* dari subsistem ini terdapat 9 *weight*. 9 nilai *weight* ini akan diolah pada subsistem defuzzifikasi. Berikut blok Simulink untuk subsistem defuzzifikasi.



Gambar 4.6 Blok Simulink Defuzzifikasi.

Pada blok Simulink defuzzifikasi digunakan metode *center average* yang memiliki 9 buah nilai *weight* yang akan dikalikan dengan fungsi keanggotaan *output* yang telah dihitung sebelumnya pada subbab perancangan kontroler lalu nilai *weight* yang telah dikalikan dengan

fungsi keanggotaan *output* akan dijumlahkan semuanya dan akan dibagi dengan nilai *weight* yang telah dijumlahkan sebelumnya dan akan menghasilkan *output* nilai *gain* K_1, K_2, dan K_3 yang telah ditentukan pada subbab perancangan kontroler. Nilai K_1, K_2, dan K_3 disimpan pada blok *gain* yang terhubung dengan *workspace* MATLAB.

4.4.2 Pengujian Respon dengan Kontroler

Pada tahapan ini kontroler yang telah dirancang akan dicoba dijalankan pada model hasil identifikasi sistem. Pengujian ini dilakukan dengan memberikan *input step* di semua pembebanan mulai dari beban minimal sampai beban maksimal. Setelah itu respon akan dianalisa untuk mencari nilai *rise time*, *settling time*, *overshoot*, dan *error steady state*.

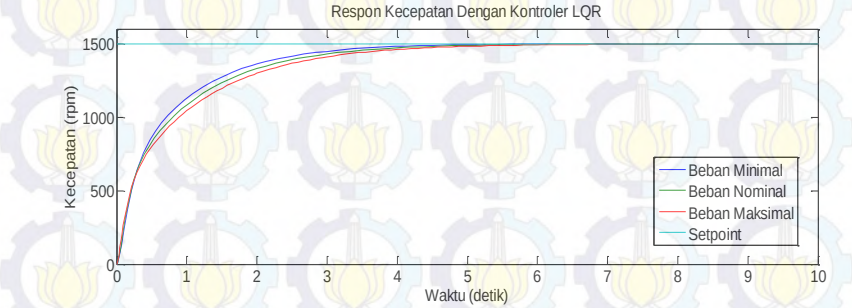
Pengujian dilakukan dengan memberikan sinyal *step* pada semua fungsi alih di setiap pembebanan. Sinyal *step* yang diberikan merupakan *setpoint* sebesar 0,97 yang seharusnya akan menghasilkan *output* kecepatan motor sebesar kisaran 1500 rpm.

Setalah melalukan simulasi maka respon yang dihasilkan akan dianalisa dengan menghitung *settling time*, *rise time*, *overshoot* dan *error steady state.* Untuk menghitung nilai *settling time* yang perlu dilakukan hanyalah menghitung nilai di mana respon *output* telah memasuki zona ±5% atau ±2% atau ±0,5% dari nilai *steady state* respon lalu untuk *rise time* yang perlu dilakukan adalah menghitung waktu yang dibutuhkan oleh respon dari mulai 10% dari nilai *steady state* hingga ke 90% dari nilai *steady state* dan untuk nilai *overshoot* yang perlu dilakukan adalah menghitung nilai nilai maksimum respon lalu dikurangi nilai *steady state* respon lalu dibagi dengan nilai *steady state* respon. Berikut karakteristik respon hasil simulasi.

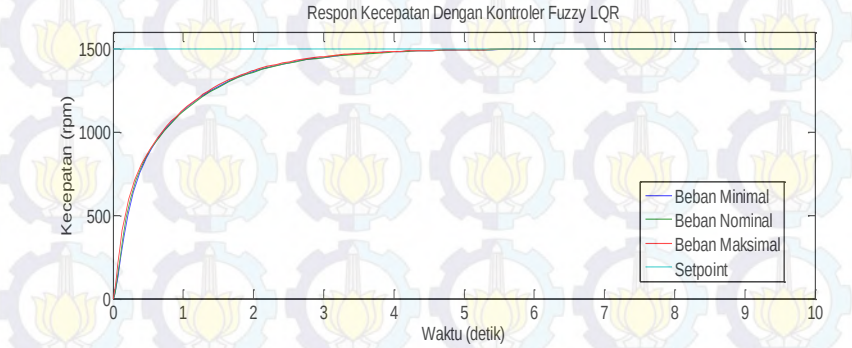
Tabel 4.2 Karakteristik Respon Hasil Simulasi.

Pembebanan	*Settling Time* (detik)	*Rise Time* (detik)	*Overshoot*	*Error Steady State*
Minimal	2,60	1,90	0	0
Nominal	2,63	1,95	0	0
Maksimal	2,54	1,87	0	0

Pada pengaturan kecepatan motor dengan menggunakan kontroler LQR saja sebenarnya sudah cukup tetapi tujuan dari kontroler ini adalah membuat respon kecepatan motor memiliki nilai *settling time* yang sama pada setiap kondisi beban oleh karena itu dalam simulasi ini dilakukan perbandingan, ketika sistem menggunakan kontroler LQR saja dan kontroler menggunakan kontroler *Fuzzy* LQR. Berikut ini merupakan hasil respon kecepatan yang membandingkan nilai *settling time* di setiap kondisi beban dengan kontroler LQR dan dengan kontroler *Fuzzy* LQR.



Gambar 4.7 Respon dengan Kontroler LQR.



Gambar 4.8 Respon dengan Kontroler *Fuzzy* LQR.

Dari gambar diatas dapat dilihat bahwa respon sistem dengan menggunakan kontroler *fuzzy* LQR dapat menghasilkan respon dengan *settling time* yang hampir sama yaitu sekitar 2,6 detik ketika diberi pembebanan yang berbeda-beda sedangkan kontroler LQR mengalami

perubahan respon ketika diberi beban dengan *settling time* dan *rise time* yang naik seiring dengan bertambahnya beban. Hal ini terjadi karena pada kontroler *fuzzy* LQR *gain state feedback* dapat berubah ketika terjadi perubahan pembebanan sehingga spesifikasi performansi yang diinginkan tetap dapat tercapai.

Berikut ini merupakan tabel yang membandingkan nilai *settling time* di setiap kondisi beban dengan kontroler LQR dan dengan kontroler *fuzzy* LQR.

Tabel 4.3 Tabel Perbandingan *Settling Time*.

Pembebanan	*Settling Time Fuzzy* LQR (detik)	*Setting Time* LQR (detik)
Minimal	2,61	2,61
Nominal	2,63	3,17
Maksimal	2,54	3,25

4.5 Implementasi Sistem

Pada tahapan ini kontroler yang telah dibuat akan dicoba dijalankan pada model hasil identifikasi sistem. Pengujian ini dilakukan dengan memberikan respon *step* di semua pembebanan mulai dari kondisi tanpa beban hingga beban sangat berat.

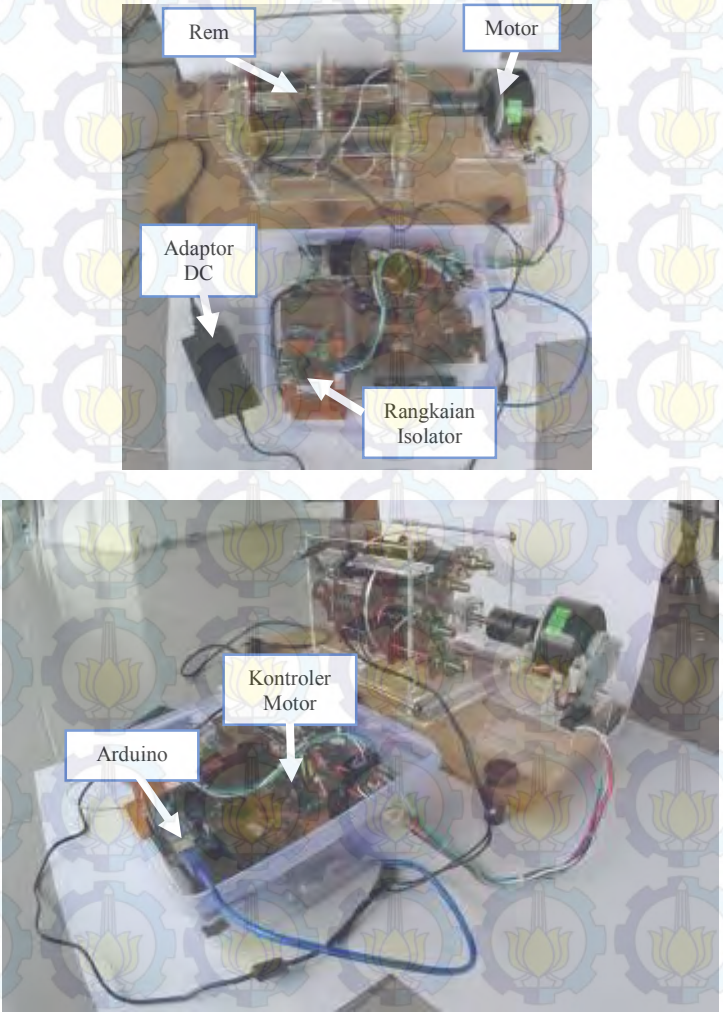
4.5.1 Realisasi *Plant*

Berikut ini merupakan *plant* dari sistem pengaturan motor BLDC yang telah di realisasikan. *Plant* ini terbagi menjadi 3 sistem yaitu sistem yang bekerja menjalankan motor, lalu sistem yang menjalankan rem, dan satu lagi sistem yang membaca sensor. Dari 3 sistem ini untuk bisa terhubung dengan kontroler harus memiliki *interface*. *Interface* yang kami gunakan adalah Arduino Uno.

4.5.1.1 Realisasi Fisik Plant

Pada Gambar 4.9 menunjukan realisasi fisik *plant*. Pada *plant* ini terdiri dari motor BLDC, rem elektromagnetik, rangkaian pembaca tegangan *supply* untuk rem magnetik, rangkaian pembaca kecepatan motor BLDC yang terdiri dari rangkaian isolator dan rangkaian *voltage*

follower, adaptor DC sebagai *supply* tegangan untuk rem elektromagnetik kemudian terdapat kontroler motor yang berfungsi untuk memberikan *supply* daya pada motor BLDC dan didalam motor juga terdapat *driver*, Arduino juga digunakan sebagai *interface*.



Gambar 4.9 *Plant* Motor BLDC.

4.5.1.2 *Program Arduino*

Pada dasarnya program Arduino hanya berfungsi sebagai pengirim data dan penerima data dari komputer. Program Arduino yang dibuat kali ini memiliki 2 sistem utama yaitu inisialisasi dan program utama. Berikut ini merupakan program inisialisasi yang digunakan saat implementasi.

```
#include <Wire.h>
int pin = 7;
int out = 6;
byte u = 255;
int ul;
unsigned long duration;
unsigned long spd=0;
word kec;
word arus;
char i[1];
char w[1];
byte w1;
byte w2;
int b1;
int b2;
unsigned long t;
unsigned long dt;

void setup()
{
  Serial.begin(115200);
  pinMode(pin,INPUT);
  pinMode(out,OUTPUT);
  analogWrite(out,u);
}
```

Gambar 4.10 Potongan Program Inisialisasi.

Pada program inisialisasi terdapat beberapa hal yang dilakukan yaitu menyebutkan jenis-jenis *library* yang dipakai, variabel-variabel yang dipakai, nilai *baudrate*, dan yang terakhir menyebutkan pin-pin yang dipakai pada Arduino. Terdapat beberapa jenis variabel yang dipakai diantaranya *integer*, *character*, dan *unsigned long* lalu *baudrate* yang dipakai bernilai 115200, pin yang digunakan pada *plant* adalah pin 7 sebagai membaca sensor kecepatan dan pin 6 yang mengeluarkan *output* PWM.

Pada program utama Arduino akan terjadi program yang terus menerus dilakukan selama Arduino dalam kondisi menyala. Pada program utama terbagi menjadi 2 buah sistem, pertama merupakan

program yang membaca angka yang dikirim dari Simulink MATLAB, lalu yang kedua merupakan program untuk membaca nilai kecepatan motor dalam satuan rpm dan nilai tegangan *supply* pada rem. Gambar 4.11 merupakan program untuk menerima sinyal kontrol dari Simulink.Sinyal kontrol yang masuk akan dikurangi dengan 255 dan nilai hasil pengurangan ini akan menjadi nilai *duty cycle* yang dikeluarkan oleh Arduino melalui pin 6.

```
void loop()
{
  //sinyal kontrol
  if (Serial.available()) {
    u=255-Serial.read();
    analogWrite(out,u);
    baca();
    }
}
```

Gambar 4.11 Potongan Program Sinyal Kontrol.

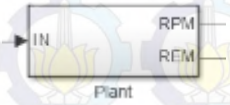
Pada Gambar 4.12 adalah program untuk membaca data dan mengirim data ke dalam Simulink, data yang dibaca ada dua data yaitu nilai tegangan beban yang diberikan dan nilai kecepatan motor yang didapatkan dengan membaca frekuensi dari kabel sensor dan menghitung nilai kecepatan dengan rumus $\omega = 15 \times f$.

```
void baca()
{
  //baca kecepatan motor
  duration = pulseIn(pin, LOW, 30000);
    if (duration == 0)
    {
      spd = 0;
    }
    else{
      spd=(15*1000000/(2*duration));
      kec=int(spd);
      }

  //baca tegangan supply rem
  int sensorValue = analogRead(A0);
  int beban = map(sensorValue,0,1023,0,255);

  //kirim ke serial
  //data kecepatan dan tegangan supply rem
  w1=kec/256;
  w2=kec%256;
  Serial.write(w1);
  Serial.write(w2);
  Serial.write(beban);
}
```

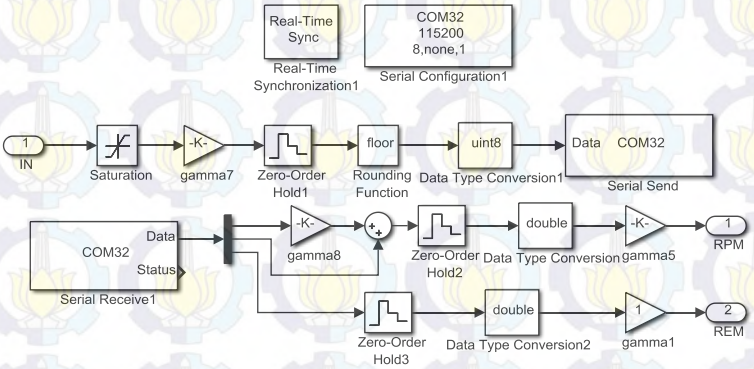
Gambar 4.12 Potongan Program *Serial*.

4.5.2 Blok Simulink Implementasi Sistem

Pada dasarnya blok Simulink pada simulasi dan implementasi memiliki struktur yang sama, perbedaannya hanya dibagian blok pada *plant* saja. Jika pada simulasi sistem blok diagram pada *plant* diisi dengan fungsi alih yang telah didapatkan dari identifikasi. Pada implementasi sistem, blok dari *plant* berisi blok komunikasi *serial*.



Gambar 4.13 Blok *Plant* untuk Implementasi.



Gambar 4.14 Blok Simulink untuk Komunikasi *Serial*.

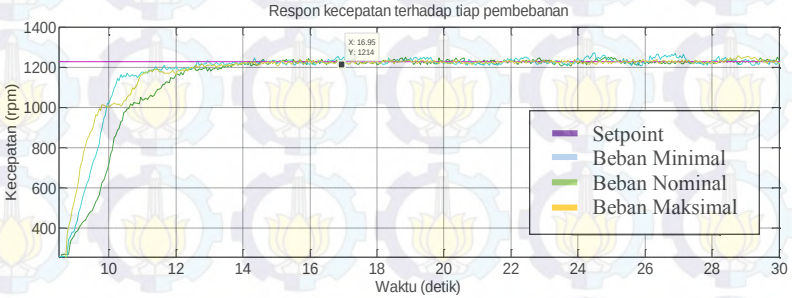
Blok Simulink untuk komunikasi *serial* ditunjukkan pada Gambar 4.14, pada gambar diatas sinyal kontrol yang diberikan pada *plant* akan menjadi *input* yang dikirim ke Arduino melalui blok *serial send*. Pada blok *serial receive* akan menerima data kecepatan dan nilai tegangan beban yang dikirim oleh Arduino.

4.5.3 Pengujian Respon Motor BLDC dengan Kontroler

Setelah melakukan simulasi hal yang selanjutnya dilakukan adalah implementasi kontroler yang telah didesain sebelumnya pada *plant* motor BLDC. Pada implementasi ini dipilih *setpoint* berupa sinyal *step* bernilai 0,8 yang merepresentasikan kecepatan 1228 rpm. Setelah itu respon dari

motor dapat dianalisa dan dicari nilai *rise time*, *settling time*, *overshoot*, dan *error steady state*.

Pada Gambar 4.15 menunjukan hasil respon kecepatan motor BLDC pada setiap beban dengan *setpoint* yang diberikan sebesar 1228 rpm.



Gambar 1.15 Respon Kecepatan Terhadap Setiap Pembebanan.

Tabel 4.4 menunjukkan karakteristik respon hasil implementasi. Dari data ini didapatkan bahwa respon dari hasil implementasi memiliki respon yang lebih lambat dibandingkan dengan respon pada hasil simulasi dengan *settling time* yang berubah sekitar 2,88 detik dan nilai *rise time* yang bernilai rata-rata 2,23 detik tanpa *overshoot* dan *error steady state*.

Tabel 4.4 Data Karakteristik Respon Hasil Implementasi.

Pembebanan	*Settling Time* (detik)	*Rise Time* (detik)	*Overshoot* (%)	*Error Steady State*
Minimal	2,50	1,55	0	0
Nominal	3,25	3,00	0	0
Maksimal	2,90	2,15	0	0

Setelah melakukan dua jenis pengujian yaitu simulasi dan implementasi, selanjutnya kedua data hasil pengujian dibandingkan untuk mengetahui bagaimana perbedaan perhitungan hasil simulasi dan implementasi. Hal ini dilakukan dengan membandingkan nilai *rise time*, *settling time*, *overshoot*, dan *error steady state*. Perbandingan ini akan

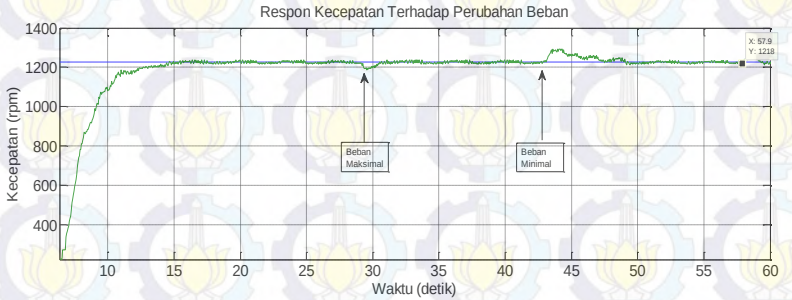
memperlihatkan bagaimana perbedaan respon kontroler yang dipasang pada pemodelan motor BLDC dengan respon kontroler yang langsung dipasang pada motor BLDC, karena respon yang dihasilkan tidak memiliki *overshoot* dan *error steady state* maka kedua karakteristik ini tidak dicantumkan dalam Tabel 4.5.

Berikut ini merupakan tabel perbandingan data *settling time* dan *rise time* antara hasil simulasi dengan implementasi.

Tabel 4.5 Data Hasil Simulasi dan Implementasi.

Pembebanan	*Settling Time* (detik)		*Error* (%)	*Rise Time* (detik)		*Error* (%)
	Sim.	Imp.		Sim.	Imp.	
Mininal	2,60	2,50	3,85	1,90	1,55	18,40
Nominal	2,63	3,25	23,57	1,95	3,01	53,84
Maksimal	2,54	2,90	14,17	1,87	2,15	14,97

Pada Tabel 4.5 bisa dilihat perbedaan nilai hasil simulasi dengan hasil implementasi. Pada saat motor berada pada kondisi beban nominal, respon kecepatan motor memiliki *error* terbesar yaitu sekitar 23,57 % untuk nilai *settling time* dan 53,84% untuk nilai *rise time.* Terdapat perbedaan hasil respon antara simulasi dengan implementasi hal ini dikarenakan perbedaan model hasil identifikasi dengan *plant* motor BLDC.



Gambar 4.16 Respon Kecepatan Terhadap Perubahan Beban.

Gambar 4.16 merupakan gambar hasil implementasi kontroler *fuzzy* LQR pada saat sistem diberi masukan *setpoint* 0,8 yang setara dengan 1228 rpm saat motor berada dalam kondisi beban nominal kemudian nilai tegangan rem diubah dengan menaikkan dan menurunkan nilai rem menjadi beban maksimal dan nominal. Dari hasil respon sistem dapat dilihat bahwa ketika nilai tegangan rem naik, respon sistem mengalami penurunan dan kemudian kembali ke keadaan *steady state* dan ketika nilai tegangan rem diturunkan hasil respon sistem akan naik dan kemudian secara kembali ke keadaan *steady state*.

Pada Tabel 4.6 ditunjukkan data perbandingan antara hasil implementasi dengan spesifikasi performansi yang diinginkan.

Tabel 4.6 Data Hasil Implementasi dan Spesifikasi Performansi.

Karakteristik Respon	Waktu (detik)		RMSE (%)
	Implementasi	Spesifikasi	
Time constant	0,96	0,87	0,135
Settling time	2,88	2,60	0,417
Rise time	2,33	1,90	0,682

Dari data diatas dapat dilihat bahwa terdapat perbedaan karakteristik respon antara hasil implementasi dengan spesifikasi performansi yang diinginkan, pada nilai *rise time* memiliki nilai *root mean square error* (RMSE) terbesar yaitu 0,682% dan untuk nilai *settling time* sebesar 0,417% dan 0,135% untuk nilai *time constant*. Hal ini terjadi karena kontroler *fuzzy* LQR bergantung pada model yang digunakan sehingga jika terdapat perbedaan antara model dengan *plant* motor BLDC maka respon hasil implementasi dengan hasil simulasi dapat berbeda. Hasil respon implementasi dapat dinyatakan dalam fungsi alih orde 1 pada persamaan berikut.

$$\frac{Y(s)}{X(s)} = \frac{1}{0{,}96s + 1} \tag{4.1}$$

BAB 5
PENUTUP

5.1 Kesimpulan

Setelah dilakukan pengujian dan analisa dapat ditarik beberapa kesimpulan yaitu setelah melakukan simulasi dan implementasi dengan kontroler *fuzzy* LQR didapatkan bahwa hasil respon pada implementasi lebih lambat daripada spesifikasi performansi yang diinginkan. Dari data hasil implementasi dan spesifikasi performansi yang diinginkan didapatkan selisih sebesar 0,09 detik untuk nilai *time constant*, 0,28 detik untuk nilai *settling time,* dan 0,43 detik untuk nilai *rise time* dengan RMSE masing – masing sebesar 0,135%, 0,417% dan 0,682%.

5.2 Saran

Apabila ada yang ingin menggunakan *plant* motor BLDC ini untuk dilakukan penelitian lebih lanjut atau menggunakan kontroler *Fuzzy* LQR, disarankan beberapa hal :

a. Jika ingin melakukan identifikasi dengan metode dinamis, perhatikanlah periode sinyal PRBS yang dimasukkan. Periode sinyal harus lebih besar dari *rise time plant.*
b. Dalam mendesain kontroler *fuzzy* LQR harus mendapatkan model yang sesuai dengan *plant* motor BLDC supaya hasil respon saat implementasi sesuai dengan hasil respon saat melakukan simulasi.
c. Dalam melakukan implementasi kontroler harus dilihat terlebih dahulu sinyal kontrol yang dikeluarkan kontroler Karena respon yang berlebih dapat menyebabkan terjadinya hentakan pada motor.

LAMPIRAN

Program Arduino

```
#include <Wire.h>
int pin = 7;
int out = 6;
byte u = 255;
int u1;
unsigned long duration;
unsigned long spd=0;
word kec;
word arus;
char i[1];
char w[1];
byte w1;
byte w2;
int b1;
int b2;
unsigned long t;
unsigned long dt;

void setup()
{
  Serial.begin(115200);
  pinMode(pin,INPUT);
  pinMode(out,OUTPUT);
  analogWrite(out,u);
}

void loop()
{
  //sinyal kontrol
  if (Serial.available()) {
    u=255-Serial.read();
    analogWrite(out,u);
    baca();
    }
}
```

```
void baca()
{
//baca kecepatan motor
  duration = pulseIn(pin, LOW, 30000);
    if (duration == 0)
     {
      spd = 0;
     }
     else{
      spd=(15*1000000/(2*duration));
      kec=int(spd);
      }

  //baca arus
  int sensorValue = analogRead(A0);
  int beban = map(sensorValue,0,1023,0,255);


  //kirim ke matlab
    //kecepatan
    w1=kec/256;
    w2=kec%256;
    Serial.write(w1);
    Serial.write(w2);
    Serial.write(beban);
}
```

DAFTAR PUSTAKA

[1] Xia, Chang Liang,*Permanent Magnet Brushless DC Motor Drives and Controls*, Singapore : John Wiley & Sons Singapore Pte. Ltd., 2012.

[2] Yedamale, Padmaraja,*Brushless DC (BLDC) Motor Fundamentals*,United States : Microchip Technology Inc, 2003.

[3] ……., "*Brushless DC Motor,How It Works?*", <URL: http://www.learnengineering.org/2014/10/Brushless-DC motor.html>,Oktober,2014.

[4] ……., "*Electromagnetic Braking*", <URL: http://lrrpublic.cli.det.nsw.edu.au/lrrSecure/Sites/Web/physics_explorer/physics/lo/induction_08/induction_08_02.html>, 2007.

[5] Blum, Jeremy,*Exploring Arduino Tools and Technique,*Indiana Polis : John Wiley & Sons Inc,2013.

[6] Elrosa, Ilmiyah, *"*Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control*"*, *Tugas Akhir*, Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember, Surabaya, Bab. 2, 2014.

[7] Ogata, Katsuhiko, *Modern Control Engineering*, New Jersey: Prentice Hall, 2002.

[8] Passino, Kevin M., Stephen Yurkovich, *Fuzzy Control*, California: Addison Wesley Longman, Inc, 1998.

[9] Naidu ,Desineni Subbaram,*Optimal Control System*, United States: CRC Press,2000.

[10] Mahjoob, M.J., "Fuzzy LQR Controller for Heading Control of an Unmanned Surface Vessel", *IEEE Control Syst. Magazine*, vol. 17, no. 2, pp. 43–51, Apr. 2001.

[11] ……., "*DRV10866 5-V, 3-Phase, Sensorless BLDC Motor Drier*",Texas Instrument,2015.

RIWAYAT HIDUP

Fairuzza Dinansyar lahir di Kupang tanggal 24 Oktober 1993, merupakan anak kedua dari Iriansi dan Kolik. Setelah lulus dari SMA Kemala Bhayangkari 1 Surabaya tahun 2011 kemudian melanjutkan studi di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun yang sama.