**TUGAS AKHIR – TE 141599**

# PERANCANGAN PENGATURAN KECEPATAN PADA MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN *NEURAL NETWORK* BERBASIS *PARTICLE SWARM OPTIMIZATION*.

Irwan Eko Prabowo
NRP. 2211 100 085

Dosen Pembimbing
Ir. Rusdhianto Effendi AK.,MT.
Ir. Ali Fatoni, MT.

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

*FINAL PROJECT* – **TE 141599**

# *DESIGN OF SPEED CONTROL FOR BRUSHLESS DIRECT CURRENT MOTOR ( BLDC ) USING NEURAL NETWORK BASED PARTICLE SWARM OPTIMIZATION*

Irwan Eko Prabowo
NRP. 2211 100 085

*Supervisor*
Ir. Rusdhianto Effendi AK.,MT.
Ir. Ali Fatoni, MT.

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

**PERANCANGAN PENGATURAN KECEPATAN PADA MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN *NEURAL NETWORK* BERBASIS *PARTICLE SWARM OPTIMIZATION***

**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., MT.** | **Ir. Ali Fatoni,MT** |
| NIP. 196210051990031003 | NIP. 19781201200212200 2 |

**SURABAYA**
**JANAURI, 2016**

**PERANCANGAN PENGATURAN KECEPATAN PADA MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN NEURAL NETWORK BERBASIS PARTICLE SWARM OPTIMIZATION**

Irwan Eko Prabowo
2211 100 085

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

# ABSTRAK

*Brushless Direct Current Motor* ( Motor BLDC ) merupakan pengembangan dari teknologi motor DC yang telah ada. Motor ini dapat berfungsi tanpa menggunakan sikat pada komutatornya. Sikat / *brush* yang ada, merupakan salah satu kelemahan dari motor DC dikarenakan perlunya perawatan pada sikat secara berkala. Motor BLDC sendiri memiliki berbagai keunggulan dibandingkan dengan motor DC konvensional. Pada tugas akhir ini, saya merancang sebuah sistem motor BLDC, kemudian digunakan kontroler *Neural Network* memaksimalkan efektifitas kerja motor pada segala kondisi. Pada *Neural Network* sendiri memiliki kemampuan untuk memperbaiki kinerja sistemnya melalui proses *learning* dengan melakukan perubahan pada nilai bobotnya. Pada proses *learning*, untuk optimisasi perubahan nilai bobot maka dapat digunakan *Particle Swarm Optimization* ( PSO ). Dari Hasil *Learning* yang didapat diperoleh parameter nilai bobot $w_1$=0,136782366779, $w_2$= 0,56795706196, $w_3$= 0,00235166854, dan $w_4$=1

**Kata Kunci :** Motor arus searah tanpa sikat, *Brushless DC*, *PSO-NN , Neural Network*

# *Design of speed control for Brushless Direct Current Motor ( BLDC ) using Neural Network based Particle Swarm Optimization*

Irwan Eko Prabowo
2211 100 085

*Supervisor* I : Ir. Rusdhianto Effendie A.K., MT.
*Supervisor* II : Ir. Ali Fatoni, MT.

## *ABSTRACT*

*Brushless Direct Current Motor (BLDC Motor) is a development of the DC motor technology. This motor can function without the use of brushes in its comutator. Brush that exist, is one of the disadvantages of DC motor due to the need for care in the brush on a regular basis. BLDC motor itself has various advantages compared to the conventional DC motors. In this final task, I designed a system of BLDC motor controller, then used Neural Network to maximize the effectiveness of the motor work in all conditions. On Neural Network alone has the ability to improve the performance of the system through the process of learning by doing it, the weight on the value of Neural Network change. On the process of learning, for optimization changes the value of weight then it can be used a Reinforced learning using Particle Swarm Optimization (PSO) Algorithm.From the Learning process we get the parameter of weight* $w_1$=0.13678236677, $w_2$= 0.56795706196, $w_3$= 0.00235166854, *and* $w_4$=1.

***Keywords*** *: Brushless DC, Reinforced Learning, Neural Network, PSO-NN*

# KATA PENGANTAR

Alhamdullilah, puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan buku tugas akhir dengan judul **" PERANCANGAN PENGATURAN KECEPATAN PADA MOTOR ARUS SEARAH TANPA SIKAT MENGGUNAKAN *NEURAL NETWORK* BERBASIS *PARTICLE SWARM OPTIMIZATION* ".**

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, Bapak Agus Purwanto dan Ibu Irawati yang selalu memberikan dukungan, semangat, dan doa kepada penulis.
2. Bapak Rusdhi dan Bapak Ali selaku Dosen Pembimbing atas segala bantuan, perhatian, dan arahan selama pengerjaan tugas akhir ini.
3. Bapak Rusdhi selaku Koordinator Bidang Studi Sistem Pengaturan Jurusan Teknik Elektro ITS.
4. Bapak Ardyono selaku Ketua Jurusan Teknik Elektro ITS.
5. Rekan-rekan e51 khususnya bidang studi Sistem Pengaturan.
6. Teman-teman seperjuangan, dan teman satu lab.

Penulis berharap tugas akhir ini dapat bermanfaat bagi yang membutuhkannya.

Surabaya, 13 Januari 2016

Penulis

# DAFTAR ISI

**BAB 3 PERANCANGAN SISTEM**



**BAB 4 PENGUJIAN DAN ANALISA**



**BAB 5 PENUTUP**

# DAFTAR GAMBAR

# DAFTAR TABEL

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Motor listrik banyak digunakan dalam berbagai peralatan seperti: *air conditioning*, *vacuum cleaner*, *conveyor*, lemari pendingin, dan lain sebagainya. Dan yang terutama motor listrik ini juga mulai dipakai pada aplikasi mobil listrik. Motor *universal* dan motor *direct current* (DC) banyak digunakan dalam aplikasi tersebut. Akan tetapi penggunaan motor DC konvensional menimbulkan masalah diakibatkan oleh penggunaan sikat. Penggantian sikat secara periodik untuk menjaga kinerja serta busur api adalah masalah yang umum disoroti pada motor DC. Oleh karena itu pada tugas akhir ini dipilihlah *plant* motor *Brushless Direct Current* (BLDC) yang merupakan alternatif pengganti motor DC. Motor ini adalah salah satu jenis motor yang popularitasnya mulai naik. [1]

Untuk sistem kontrol dari motor BLDC yang paling umum dipergunakan adalah kontrol PI dan PID. Dalam tugas akhir ini dipergunakan kontrol *Neural Network* untuk mengatur kecepatan motor. *Neural Network* merupakan suatu sistem komputasi yang beroperasi secara prosedural. Pada *Neural Network* sendiri memiliki kemampuan untuk memperbaiki kinerja sistemnya melalui proses *learning* dengan melakukan perubahan pada nilai bobotnya. Pada proses *learning*, untuk optimisasi perubahan nilai bobot maka dapat digunakan *Particle Swarm Optimization* (PSO).

## 1.2 Perumusan Masalah

Permasalahan pada tugas akhir ini adalah kondisi nyata penggunaan motor tidak selalu sama, banyak terdapat kondisi lain seperti pada pengereman dan lainnya, sehingga yang menjadi masalah pada tugas akhir ini adalah bagaimana mengatur torsi motor BLDC agar dapat menanggung atau mengatasi beban yang berubah-ubah secara tidak menentu. Oleh karena itu, dibutuhkan suatu kontroler untuk melakukan pengaturan kerja dari motor listrik agar bekerja sesuai dengan kebutuhan yang diinginkan.

## 1.3 Batasan Masalah

Permasalahan pada tugas akhir ini dibatasi oleh beberapa hal antara lain:

a. Pada Tugas Akhir ini tidak dilakukan pembahasan mengenai elektronika motor BLDC
b. Kontroler yang dipergunakan adalah *Neural Network* berbasis PSO sebagai algoritma *learningnya*
c. Pada tugas akhir ini kecepatan yang dikontrol dimulai dari 1000 hingga 1500 rpm

## 1.4 Tujuan Penilitian

Tujuan dari pelaksanaan tugas akhir ini adalah:

a. Membuat *plant* berupa motor BLDC yang terangkai dengan rem magnetik yang berfungsi sebagai pembebanan pada motor BLDC.
b. Merancang kontroler *Neural Network berbasis PSO* untuk mengatasi permasalahan adanya efek pembebanan yang berubah-ubah pada motor BLDC.
c. Implementasi pada Motor BLDC yang telah dirancang menggunakan *software* dan mikrokontroler Arduino dengan harapan dapat memperbaiki performansi kerja pada motor BLDC dan mendapatkan performansi terbaik untuk pengendalian torsi motor BLDC pada pembebanan yang berbeda-beda.

Hasil yang diperoleh dari pelaksanaan tugas akhir ini diharapkan dapat memberikan manfaat dan kontribusi bagi dunia pendidikan, industri dan masyarakat agar dapat dijadikan referensi bagi peneliti lainnya dan sebagai ilmu pengetahuan.

## 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: DASAR TEORI

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 DAN KONTROLER

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: KESIMPULAN DAN SARAN

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

## 1.6 Relevansi

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

# BAB 2
# TINJAUAN PUSTAKA

Bab ini menjelaskan tentang landasan teori yang digunakan sebagai acuan dan referensi pada serangkaian proses pelaksanaan Tugas Akhir ini. Materi yang dijelaskan antara lain teori mengenai motor BLDC beserta perangkat keras pendukungnya serta perangkat lunak dan metode kontrol yang digunakan untuk pengaturan kecepatan motor BLDC.

## 2.1 Motor *Brushless* DC [1]

Motor arus searah adalah sebuah motor yang membutuhkan tegangan searah untuk menjalankannya. Pada umumnya motor jenis ini menggunakan sikat dan mengoperasikannya sangat mudah tinggal dihubungkan dengan sumber DC sehingga motor langsung bekerja. Jenis motor ini memerlukan perawatan pada sikatnya serta banyak terjadi rugi tegangan pada sikat. Sehingga pada era sekarang ini motor DC dikembangkan tanpa menggunakan sikat yang dikenal dengan Motor BLDC (*Brushless Direct Current Motor*) [1]. Dibandingkan dengan motor DC, BLDC memiliki biaya perawatan yang lebih rendah dan kecepatan yang lebih tinggi akibat tidak digunakannya *brush*.

*Brushless* DC Motor termasuk kedalam jenis motor sinkron. Artinya medan magnet yang dihasilkan oleh stator dan medan magnet yang dihasilkan oleh rotor berputar pada frekuensi yang sama. Motor BLDC tidak mengalami *slip* seperti yang terjadi pada motor induksi biasa. Motor jenis ini mempunyai magnet permanen pada bagian rotor dan elektromagnet pada bagian stator. Setelah itu, dengan menggunakan sebuah rangkaian sederhana (*simple computer system*), maka kita dapat merubah arus di elektromagnet ketika bagian rotornya berputar.

Walaupun merupakan motor listrik sinkron AC 3 fasa, motor ini tetap disebut dengan BLDC karena pada implementasinya BLDC menggunakan sumber DC sebagai sumber energi utama yang kemudian diubah menjadi tegangan AC dengan menggunakan *inverter* 3 fasa.

Tujuan dari pemberian tegangan AC 3 fasa pada stator BLDC adalah menciptakan medan magnet putar stator untuk menarik magnet rotor. Oleh karena tidak adanya *brush* pada motor BLDC, untuk menentukan *timing* komutasi yang tepat pada motor ini sehingga didapatkan torsi dan kecepatan yang konstan, diperlukan 3 buah *sensor hall* dan atau *encoder*. Pada *sensor hall*, *timing* komutasi ditentukan dengan cara mendeteksi

medan magnet rotor dengan menggunakan 3 buah *sensor hall* untuk mendapatkan 6 kombinasi *timing* yang berbeda, sedangkan pada *encoder*, *timing* komutasi ditentukan dengan cara menghitung jumlah pola yang ada pada *encoder*. Pada umumnya *encoder* lebih banyak digunakan pada motor BLDC komersial karena *encoder* cenderung mampu menentukan *timing* komutasi lebih presisi dibandingkan dengan menggunakan *hall sensor*. Hal ini terjadi karena pada *encoder*, kode komutasi telah ditetapkan secara *fixed* berdasarkan banyak kutub dari motor dan kode inilah yang digunakan untuk menentukan *timing* komutasi. Namun karena kode komutasi *encoder* untuk suatu motor tidak dapat digunakan untuk motor dengan jumlah kutub yang berbeda. Hal ini berbeda dengan *hall sensor*. Apabila terjadi perubahan *pole* rotor pada motor, posisi sensor *hall* dapat diubah dengan mudah. Hanya saja kelemahan dari sensor *hall* adalah apabila posisi sensor *hall* tidak tepat akan terjadi keselahan dalam penentuan *timing* komutasi atau bahkan tidak didapatkan 6 kombinasi *timing* komutasi yang berbeda.

### 2.1.1 Cara Kerja Motor *Brushless* DC

Cara kerja pada motor BLDC cukup sederhana, yaitu magnet yang berada pada poros motor akan tertarik dan terdorong oleh gaya elektromagnetik yang diatur oleh *driver* pada motor BLDC. Hal ini membedakakn motor BLDC dengan motor DC yang menggunakan sikat mekanis yang berada pada komutator untuk mengatur waktu dan memberikan medan magnet pada lilitan. Motor BLDC ini juga berbeda dengan motor AC yang pada umumnya menggunakan siklus tenaga sendiri untuk mengatur waktu dan memberi daya pada lilitan. BLDC dapat memberikan rasio daya dan beban yang lebih tinggi secara signifikan dan memberikan efisiensi yang lebih baik dibandingkan motor tanpa sikat tradisional.

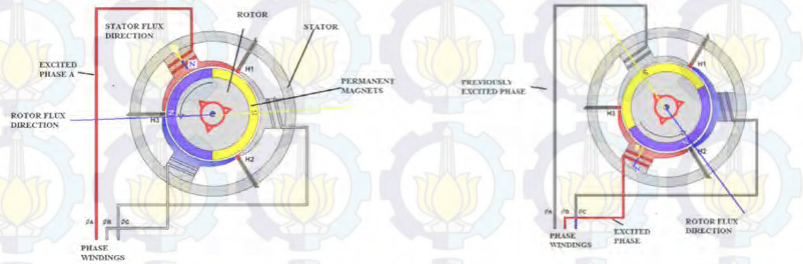
Pada prinsip dasar medan magnet adalah kutub yang sama akan saling tolak menolak sedangkan apabila berlainan kutub maka akan tarik menarik. Jadi jika kita mempunyai dua buah magnet dan menandai satu sisi magnet tersebut dengan *north* (utara) dan yang lainnya *south* (selatan), maka bagian sisi *north* akan coba menarik *south*, sebaliknya jika sisi *north* magnet pertama akan menolak sisi *north* yang kedua dan seterusnya apabila kedua sisi magnet mempunyai kutub yang sama [5].

Prinsip mengenai kutub magnet tersebut dapat diterapkan dalam prinsip kerja motor BLDC. Secara umum motor BLDC memiliki medan magnet permanen pada rotor dan magnet yang berasal dari gaya

elektromagnet (magnet yang ditimbulkan dari pemberian *input* arus listrik) pada bagian kumparan stator.

Pada motor BLDC, kontroler berfungsi untuk mengatur arus masukan yang harus dialirkan ke kumparan stator untuk dapat menimbulkan medan elektromagnet yang sesuai untuk memutar rotor. Hal inilah yang menjadi pembeda dengan motor DC konvensional, dan menggantikan kerja komutasi mekanisnya.

Magnet permanen pada motor BLDC dilengkapi dengan kumparan tiga fase. Kumparan-kumparan tersebut terletak di bagian stator. Magnet bergerak terletak di stator. Fase kumparan diaktifkan dengan penyesuain gerakan rotor. Rotasi berbasis rotasi medan magnet diilustrasikan pada Gambar 2.8 (a) dan Gambar 2.8 (b) menjelaskan pergerakan dan eksitasi fase. Pada Gambar 2.8 (a) fase A dieksitasi, fluks stator dihasilkan oleh eksitasi fase A, fluks rotor dihasilkan oleh magnet permanen.



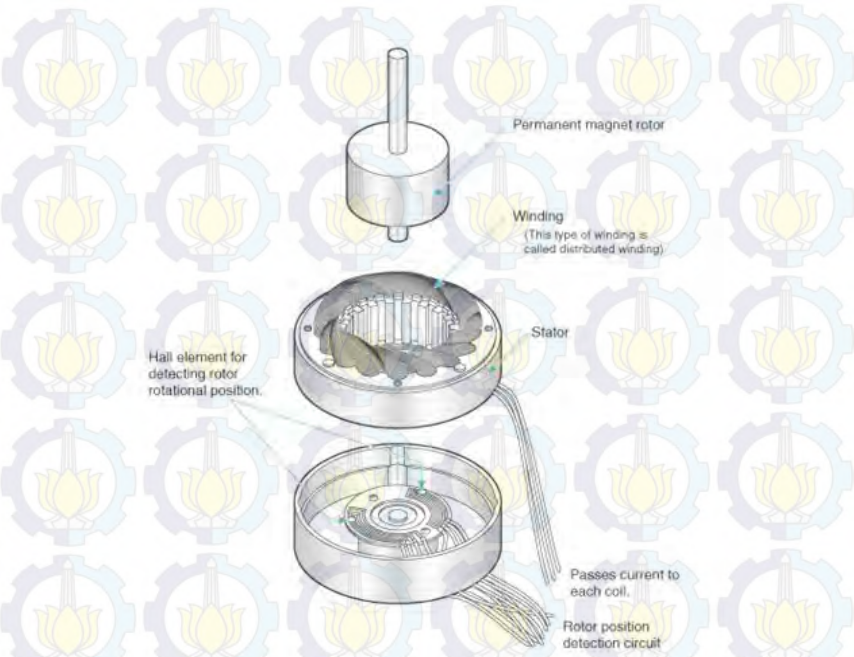

(a) Fase Pergerakan (b) Fase *Eksitasi*

**Gambar 2.1** Rotor Berbasis Medan Magnet

### 2.1.2 Konstruksi Motor Arus Searah Tanpa Sikat

Setiap motor BLDC memiliki dua bagian utama, rotor (bagian berputar) dan stator (bagian stasioner). Bagian penting lainnya dari motor adalah gulungan stator dan magnet rotor.

Adapun konstruksi dari motor BLDC dapat dilihat pada Gambar 2.2.




**Gambar 2.2** Konstruksi Motor BLDC

a. Rotor

Rotor adalah bagian pada motor yang berputar karena adanya gaya elektromagnetik dari stator, dimana pada motor DC *brushless* bagian rotornya berbeda dengan rotor pada motor DC konvensional yang hanya tersusun dari satu buah elektromagnet yang berada diantara *brushes* (sikat) yang terhubung.

Rotor dibuat dari magnet permanen dan dapat desain dari dua sampai delapan kutub Magnet Utara (N) atau Selatan (S). Bahan material magnetis yang baik sangat diperlukan untuk mendapatkan kerapatan medan magnet yang baik pula. Biasanya magnet permanen dibuat menggunakan magnet *ferrit*. Tetapi saat ini dengan kemajuan teknologi, campuran logam sudah kurang populer untuk digunakan. Meskipun dinilai lebih murah, magnet *ferrit* mempunyai kekurangan

yaitu kerapatan *fluks* yang rendah sebagai bahan material yang diperlukan untuk membuat rotor.

b. Stator

Stator adalah bagian pada motor yang diam/statis dan berfungsi sebagai medan putar motor untuk memberikan gaya elektromagnetik pada rotor sehingga motor dapat berputar. Pada motor DC *brushless* statornya terdiri dari 12 belitan (elektromagnet) yang bekerja secara elektromagnetik dimana stator pada motor DC *brushless* terhubung dengan tiga buah kabel untuk disambungkan pada rangkaian kontrol sedangkan pada motor DC konvensional stator-nya terdiri dari dua buah kutub magnet permanen.

### 2.1.3 Karakteristik Motor

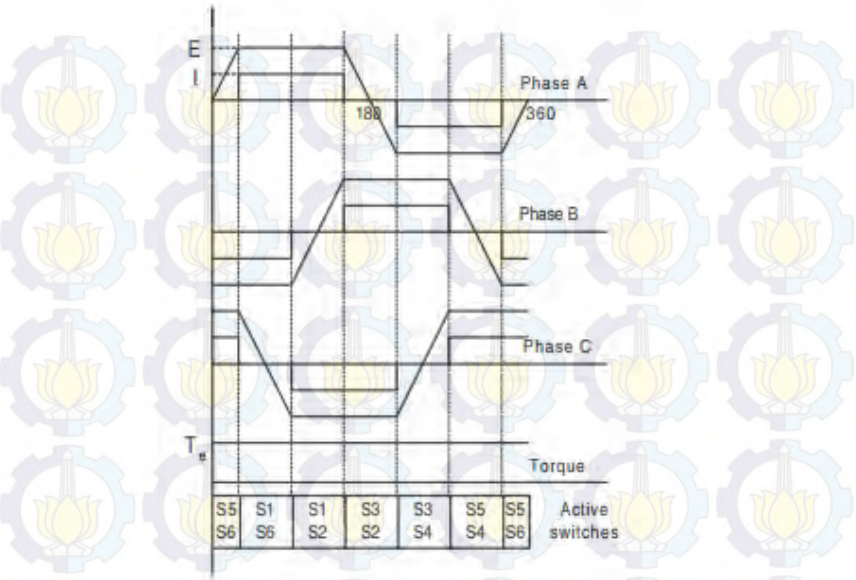
Pada motor BLDC terdapat bagian kosong yang ada (*air-gap*). Pada bagian kosong ini sendiri memiliki kepadatan medan tersendiri. Bentuk gelombang *air-gap-flux-density* pada dasarnya adalah sebuah gelombang persegi, tetapi karena adanya *fringing*, menyebabkan bentuk gelombangnya menjadi sedikit melengkung. Saat rotor berputar, bentuk gelombang tegangan yang terinduksi pada tiap fasa terhadap waktunya merupakan replika dari bentuk gelombang *air-gap-flux-density* terhadap posisi rotor. *Fringing* menyebabkan bentuk gelombang *back-emf* berbentuk *trapezoidal*. Bentuk gelombang *trapezoidal* inilah yang membedakan motor BLDC dengan *Permanent Magnet Synchronous Motor* (PMSM), yang memiliki bentuk gelombang *back-emf* berupa sinusoidal. Dengan memberikan arus *rectangular* pada tiap fasa yang *back-emf*-nya berada pada keadaan nilai penuh, dimungkinkan untuk mendapatkan torsi motor BLDC yang hampir konstan.

Tegangan *back-emf* dan gelombang arus fasa $120^{O}$ ideal untuk motor BLDC tiga fasa ditunjukkan pada Gambar 2.2. *Switch inverter* yang aktif di tiap interval $60^{O}$ ditunjukkan pada Gambar 2.3.

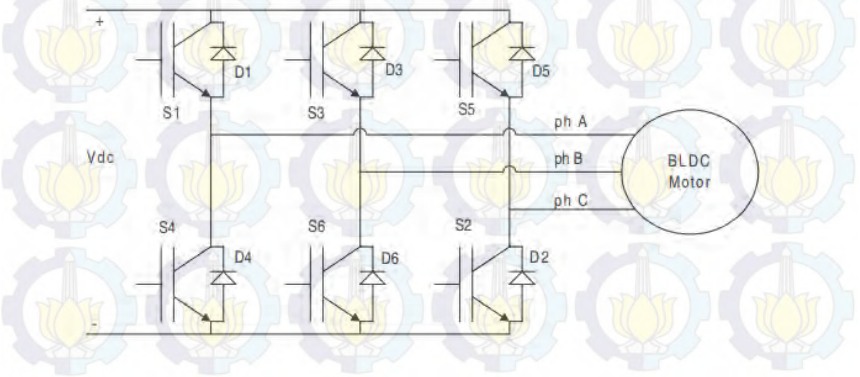
Amplitudo dari tegangan *back-emf* sebanding dengan kecepatan rotor dengan Persamaan 2.1.

$$E = k\phi\omega_m \tag{2.1}$$

Dimana nilai k merupakan konstanta yang bergantung pada jumlah lilitan setiap fasa, $\phi$ adalah nilai fluks magnet permanen, dan $\omega_m$ adalah kecepatan mekanis.




**Gambar 2.3** Gelombang *Back*-EMF dan Arus Fasa untuk Motor BLDC 3 Fasa dengan Arus *Bipolar* 120 $^{O}$




**Gambar 2.4** Skematik *Inverter* Berbasis IGBT

Selama tiap interval $120^{O}$, daya sesaat yang dikonversi dari elektris ke mekanis adalah

$$P_o = \omega_m T_e = 2EI \tag{2.2}$$

Di mana $T_e$ adalah torsi keluaran dan $I$ adalah amplitudo arus fasa. Dari persamaan 2.1 dan 2.2, persamaan torsi keluaran dapat ditulis

$$T_e = 2k\phi I = k_t I \tag{2.3}$$

Di mana $k_t$ adalah konstanta torsi.

## 2.2 Rem Elektromagnetik [2]

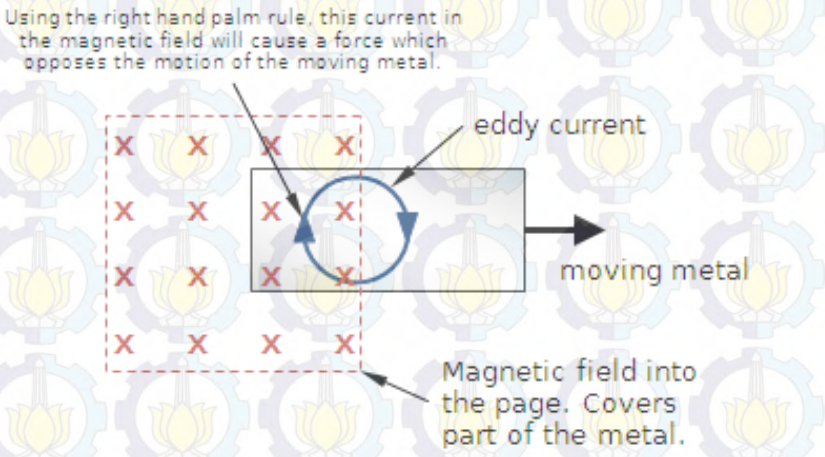
Rem elektromagnetik merupakan sistem pengereman yang tidak memanfaatkan prinsip gesekan dari medium pengereman dalam proses pengereman yang dilakukan. Proses pengereman dengan prinsip elektromagnetik memanfaatkan efek yang ditimbulkan dari arus *eddy* dan sesuai dengan prinsip hukum *Faraday* dan hukum *Lenz*.

Sistem pengereman ini menggunakan gaya elektromagnetik untuk memperlambat suatu gerakan, yang umumnya adalah gerakan poros. Sebuah piringan dengan bahan logam *nonferromagnetic* terpasang sebuah poros berputar. Piringan tersebut diapit oleh sisi stator berupa sistem lilitan elektromagnetik yang dapat membangkitkan medan magnet dari aliran listrik. Arus listrik menimbulkan medan magnet pada lilitan. Dan logam piringan yang memotong medan magnet tersebut akan menimbulkan arus *eddy* pada piringan itu sendiri. Arus *eddy* ini akan menimbulkan medan magnet yang arahnya berlawanan dengan medan magnet sebelumnya, sehingga menghambat gerakan putar dari poros tersebut.

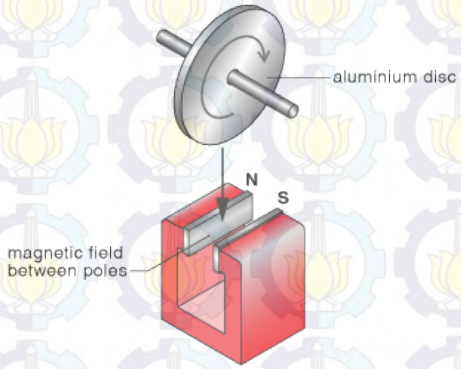
Arus *eddy* merupakan arus listrik yang timbul apabila suatu piringan logam *non-ferromagnetic* berada di sekitar medan magnet yang garis gayanya sedang berubah-ubah. Medan magnet yang dihasilkan oleh arus *eddy* ini berlawanan arah dengan arah gerakan piringan logam. Dengan begitu medan magnet yang ditimbulkan oleh arus *eddy* akan menghambat laju piringan logam tersebut.

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 2.6 akan memperlihatkan struktur dan konstruksi dari sebuah sistem rem elektromagnetik.




**Gambar 2.5** Prinsip Arus *Eddy* Pada Logam yang Bergerak




**Gambar 2.6** Prinsip Kerja Rem Elektromagnetik

## 2.3 Arduino Uno [3]

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:

| | | |
|---|---|---|
| Mikrokontroler Arduino | : | Atmega328 |
| Tegangan operasi | : | 5V |
| Tegangan *input* (direkomendasikan) | : | 7-12V |
| Tegangan *input* (batasan) | : | 6-20V |
| Pin *input*/*output* digital | : | 14 (6 diantaranya *output* PWM) |
| Pin *input* analog | : | 6 |
| Arus DC tiap pin I/O | : | 40mA |
| Arus DC untuk pin 3,3 V | : | 50mA |

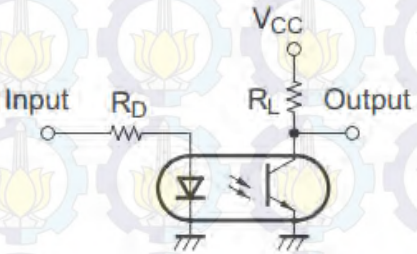
14 pin *input* dan *output* Arduino Uno dapat digunakan dengan fungsi *pinMode*( ), *digitalWrite*( ), dan *digitalRead*( ). Tiap pin memiliki tegangan operasi 5V dan dapat menerima arus maksimum 40mA. Beberapa pin memiliki fungsi khusus, antara lain :

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, walaupun begitu terdapat kemungkinan untuk mengubah batas atas dan bawah dengan menggunakan pin ARef dan fungsi *analogReference*( ).

## 2.4 Rangkaian *Optocoupler* [4]

Rangkaian *optocoupler* merupakan rangkaian yang dipergunakan untuk memisahkan perangkat Arduino dengan *plant*. Alat ini sendiri berguna untuk menghindari kerusakan pada Arduino karena arus berlebih dari *power supply* jika terjadi kesalahan.




**Gambar 2.7** Rangkaian *Optocoupler*

Rangkaian *isolator* yang dipergunakan terdiri dari *optocoupler* PIC817 dengan rangkaian yang dapat dilihat pada Gambar 2.7.

## 2.5 Sistem Pengaturan [4]

Sistem merupakan kumpulan beberapa komponen yang saling berhubungan dan bekerja sama untuk mencapai suatu tujuan tertentu. Pengaturan atau kontrol adalah suatu upaya yang dilakukan untuk menjaga atau mencapai kondisi yang diinginkan pada sistem. Pengaturan juga dapat berarti mengukur nilai dari variabel sistem yang dikontrol dan menerapkan variabel yang dimanipulasi ke sistem untuk mengoreksi atau membatasi perbedaan nilai keluaran dengan nilai yang dikehendaki.

### 2.5.1 Sistem Pengaturan *Loop* Terbuka

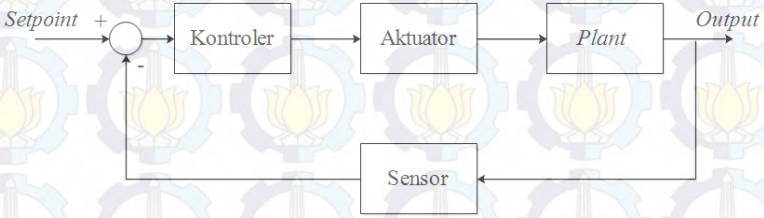
Sistem pengaturan *loop* terbuka ialah sistem yang keluarannya tidak memiliki pengaruh terhadap aksi kontrolnya. Dengan kata lain, pada sistem pengaturan *loop* terbuka, nilai *output* tidak diukur dan tidak memiliki *feedback* sebagai perbandingan dengan *input*. Oleh karena itu,

semua *input* referensi pada sistem ini berhubungan dengan kondisi operasi yang tetap. Hasilnya, akurasi dari sistem bergantung pada kalibrasi sistem. Ketika sistem pengaturan *loop* terbuka diberikan gangguan atau *disturbances*, maka sistem ini tidak akan bekerja sesuai dengan referensi yang ditentukan. Oleh karena itu, sistem pengaturan *loop* terbuka hanya bisa dijalankan ketika hubungan antara *input* dan *output* diketahui dan tidak terdapat gangguan internal maupun eksternal. Tampilan sistem pengaturan *loop* terbuka dapat dilihat pada Gambar 2.8.


**Gambar 2.8** Sistem Pengaturan *Loop* Terbuka

### 2.5.2 Sistem Pengaturan *Loop* Tertutup

Sistem pengaturan *loop* tertutup merupakan kebalikan dari sistem pengaturan *loop* terbuka. Sistem pengaturan *loop* tertutup memiliki *feedback* sebagai perbandingan dengan *output*-nya. Pada banyak buku, sistem pengaturan *loop* tertutup seringkali juga disebut sistem pengaturan umpan balik. Pada sistem pengaturan *loop* tertutup, sinyal *error*, yaitu perbedaan antara nilai *input* atau referensi (*set-point*) dan nilai *output* pada sistem, dimasukkan ke dalam kontroler untuk mengurangi nilai *error* dan mengembalikan nilai *output* sesuai dengan nilai referensi. Untuk lebih jelasnya, representasi atau diagram blok dari sistem pengaturan *loop* tertutup dapat ditemukan pada Gambar 2.9.




**Gambar 2.9** Sistem Pengaturan *Loop* Tertutup

## 2.6 Identifikasi Sistem [4]

Identifikasi sistem merupakan suatu langkah awal dalam menganalisa sistem dinamik. 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 cocok digunakan untuk analisa, prediksi, 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).

Model matematika dari sebuah sistem dapat diperoleh melalui dua cara yaitu permodelan fisik dan identifikasi sistem. Permodelan fisik merupakan pendekatan analitik berdasarkan hukum fisika seperti hukum Newton dan hukum kesetimbangan. Permodelan ini menjelaskan dinamika dalam sistem. Yang kedua adalah identifikasi sistem. Hal ini dilakukan dengan pendekatan eksperimental. Model ini berdasarkan data dari eksperimen yang kemudian didapatkan nilai parameter sistem. Pada beberapa kasus yang sangat kompleks, sangat sulit untuk menentukan model berdasarkan pemahaman fisik.

Identifikasi sistem pada suatu sistem dapat dilakukan dengan menggunakan metode identifikasi statis maupun dinamis. Identifikasi statis dilakukan untuk mendapatkan *gain* dan *time sampling* dari suatu sistem. Identifikasi statis pada suatu sistem dilakukan dengan memberikan suatu *input setpoint* yang bernilai konstan terhadap waktu. Sistem yang digunakan untuk identifikasi statis adalah sistem dengan *loop* terbuka tanpa kontroler. Sedangkan pada identifikasi dinamis, prosedur identifikasinya memiliki beberapa perbedaan mendasar. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan *input setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Pada Tugas Akhir kali ini, digunakan metode identifikasi dinamis. Pemilihan metode ini disebabkan karena respon yang sangat cepat dari motor BLDC, sehingga sangat sulit untuk mengamati respon transien dari sistem jika menggunakan identifikasi statis.

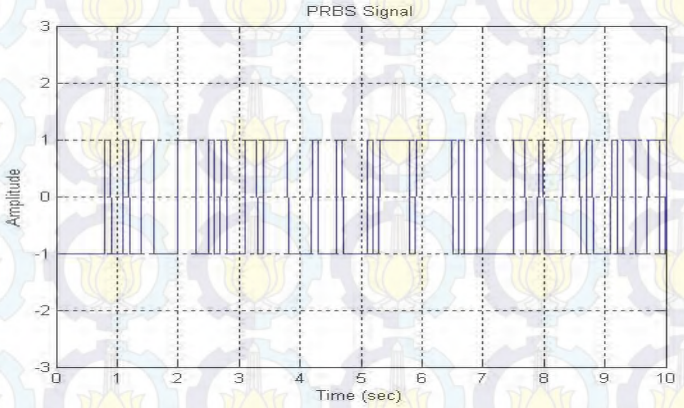
### 2.6.1 Identifikasi Dinamis

Identifikasi dinamis dilakukan untuk mendapatkan pemodelan dari suatu sistem atau *plant* BLDC. Identifikasi dinamis memiliki beberapa kelebihan dibandingkan identifikasi statis. Pada identifikasi dinamis, *input* yang diberikan pada sistem memiliki frekuensi yang berubah-ubah, sehingga karakteristik dan sifat dari sistem dapat diteliti dengan lebih cermat. *Input* yang diberikan ini dinamakan sinyal *Pseudo-Random Binary Sequence*.

Setelah respon dari sistem diperoleh, langkah selanjutnya adalah melakukan pendekatan fungsi alih dari sistem berdasarkan respon yang didapat. Ada beberapa metode yang bisa dilakukan untuk mendapatkan fungsi alih dari sistem. Penjelasan selanjutnya diberikan pada paragraf dibawah ini.

#### *2.6.1.1 Sinyal Pseudo Random Binary Sequence (PRBS)*

Terdapat beberapa perbedaan mendasar antara identifikasi statis dan dinamis. Diantaranya adalah perbedaan antara sinyal uji dan metode pemodelan yang digunakan. Jika identifikasi statis menggunakan *input setpoint* yang konstan terhadap waktu, maka identifikasi dinamis menggunakan sinyal acak atau *random* sebagai sinyal ujinya. Sinyal ini memiliki frekuensi yang berubah-ubah, sehingga memungkinkan karakteristik sistem dapat diketahui secara lebih teliti.




**Gambar 2.10** Tampilan Sinyal PRBS

Sinyal tersebut dinamakan sinyal *Pseudo-Random Binary Square* (PRBS). Sinyal PRBS seperti pada Gambar 2.10 mirip dengan bilangan acak secara nyata, tapi juga dapat disebut semu atau *pseudo* karena bersifat deterministik. Sinyal PRBS dapat dihasilkan dari penggunaan *shift register*. Salah satu contoh sinyal uji PRBS dapat dilihat pada Gambar 2.10

### 2.6.2 Validasi Model

Setelah fungsi alih didapat melalui proses identifikasi sistem, tahapan selanjutnya adalah tahapan validasi model. Validasi model diukur dengan cara mengukur nilai *error* setiap variabelnya. Tujuan dari validasi model adalah untuk mengukur apakah nilai pada pemodelan sudah mendekati dengan nilai sebenarnya dari sistem.

Dalam menggunakan proses validasi model, terdapat beberapa metode dan tolok ukur yang dapat digunakan. Diantaranya adalah metode *Root Mean Square Error* (RMSE). RMSE mengukur akurasi pada nilai deret waktu secara statistik seperti halnya regresi. RMSE dapat merepresentasikan ukuran dari *error* rata-rata karena RMSE membandingkan hasil data pengukuran dan data pemodelan pada skala yang sama antara kedua data tersebut. Formulasi perhitungan RMSE yang mempresentasikan nilai *error* dalam bentuk presentasi dapat dilihat pada Persamaan 2.5 dan Persamaan 2.6.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2}; \quad i = 1,2,3,...,n \tag{2.5}$$

$$e_i = \frac{A_i - M_i}{A_i} \times 100\%; \quad i = 1,2,3,...,n \tag{2.6}$$

$n$ : Jumlah data
$i$ : Urutan data
$e$ : Nilai *error*
$A$ : Nilai data hasil pemodelan
$M$ : Nilai data hasil pemodelan

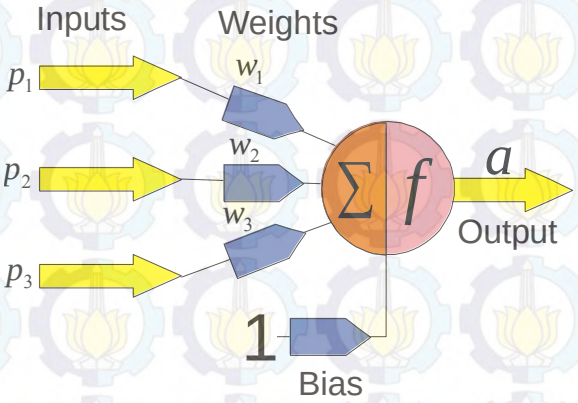
Semakin kecil nilai pengukuran RMSE suatu pemodelan, semakin baik pula model yang diberikan.

## 2.7 Jaringan Syaraf Tiruan (*Neural Network*) [5]

*Neural Network* (NN) merupakan suatu teknik pengolahan informasi yang terinspirasi oleh cara pengolahan informasi pada sistem saraf biologis, seperti pada otak. Elemen utama pada teknik pengolahan informasi adalah struktur baru pada sistem pengolahan informasi. Teknik ini terbentuk dari sejumlah besar elemen jaringan proses (*neuron*) yang bekerja sama untuk memecahkan problem secara spesifik.

### 2.7.1 Konsep Dasar Pemodelan *Neural Network*

Dimisalkan suatu *neuron* dibentuk dengan n *input*. Masing-masing saluran dapat mengirimkan sejumlah nilai riil. Fungsi $f$ yang dihitung dalam *neuron* dapat dipilih secara bebas. Biasanya saluran masukan memiliki nilai beban yang terkait, yang berarti bahwa setiap informasi masuk dikalikan dengan yang sesuai nilai bobot $w_i$. Informasi yang dikirimkan terintegrasi di *neuron* (biasanya hanya dengan menambahkan sinyal yang berbeda) dan fungsi kemudian dievaluasi.




**Gambar 2.11** Struktur Dasar Jaringan Syaraf Tiruan

Dari pemodelan diatas diperoleh persamaan *output* sebagai berikut:

$$a = f(p_1w_1 + p_2w_2 + p_3w_3 + b) = f(\sum p_iw_i + b) \qquad (2.7)$$

### 2.7.2 *Learning* pada *Neural Network*

*Neural Network*, seperti pada manusia, belajar dari contoh. Sebuah NN dapat dikonfigurasi secara khusus untuk aplikasi tertentu, seperti pengenalan pola atau klasifikasi data, melalui suatu proses belajar (*learning*). Proses belajar pada sistem biologis melibatkan perubahan pada koneksi pangkal (*synaptic*) yang ada antara *neuron*. Hal ini juga terjadi pada *Neural Network* juga.

Seperti yang telah diketahui salah satu elemen utama dari *neural network* adalah kemampuan untuk belajar (*learn*). Suatu *neural network* bukan hanya sebuah sistem kompleks, melainkan suatu sistem kompleks yang *adaptif*, yang artinya sistem ini dapat merubah struktur dasar berdasarkan dengan informasi yang masuk melalui sistem ini. Biasanya, hal ini dicapai melalui penyesuaian nilai bobot. Dalam diagram di dibawah, setiap baris mewakili hubungan antara dua *neuron* dan menunjukkan jalur untuk arus informasi. Setiap sambungan memiliki nilai bobot, nilai yang mengontrol sinyal antara dua *neuron*. Jika jaringan menghasilkan *output* yang "baik" (yang akan kita mendefinisikan kemudian), maka tidak perlu dilakukan penyesuaian bobot. Namun, jika jaringan menghasilkan *output* yang "kurang baik", maka sistem akan mengubah nilai bobot untuk meningkatkan hasil keluaran pada iterasi selanjutnya.

Ada beberapa strategi untuk proses *learning* pada *neural network*, beberapa yang paling umum dipergunakan adalah sebagai berikut :

- *Supervised Learning*: Pada dasarnya, sebuah strategi yang melibatkan seorang guru yang lebih pintar daripada jaringan itu sendiri. Sebagai contoh, misalnya pada pengenalan wajah (*face recognition*). Guru menunjukkan jaringan sekelompok wajah, dan guru sudah tahu nama yang terkait dengan setiap wajah. Jaringan membuat dugaan untuk nama yang dipilih, kemudian guru menyediakan jaringan jawaban yang benar. Jaringan kemudian dapat membandingkan jawaban untuk yang "benar" yang dikenal dan membuat penyesuaian berdasarkan kesalahan nya.
- *Unsupervised Learning*: Diperlukan ketika tidak ada contoh data yang sesuai dengan jawaban. Bayangkan mencari pola tersembunyi di sebuah *set* data. Aplikasi ini adalah pengelompokan, yaitu membagi serangkaian elemen ke grup menurut beberapa pola yang tidak diketahui.

- *Reinforced Learning*: Algoritma pembelajaran dari *mapping input-output* secara terus menerus berinteraksi dengan lingkungan, dimana hal ini dapat memberikan pengaruh dari setiap langkah yang diambil, memberikan *feedback* kepada model.

## 2.8 *Particle Swarm Optimization* ( PSO ) [6]

*Particle Swarm Optimization*, disingkat sebagai PSO, didasarkan pada perilaku sebuah kawanan serangga, seperti semut, rayap, lebah atau burung. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung. Setiap individu atau partikel berperilaku secara terdistribusi dengan cara menggunakan kecerdasannya (*intelligence*) sendiri dan juga dipengaruhi perilaku kelompok kolektifnya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka berjauhan.

Metode optimasi yang didasarkan pada *swarm intelligence* ini disebut algoritma *behaviorally inspired* sebagai alternatif dari algoritma genetika, yang sering disebut *evolution-based procedures*. Dalam konteks optimasi *multivariable*, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/*space* tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi bagusnya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut. Sebagai contoh, misalnya perilaku burung-burung dalam dalam kawanan burung. Meskipun setiap burung mempunyai keterbatasan dalam hal kecerdasan, biasanya ia akan mengikuti kebiasaan (*rule*) seperti berikut :

1. Seekor burung tidak akan terlalu dekat dengan burung yang lain.

2. Burung tersebut akan mengarahkan terbangnya ke arah rata-rata keseluruhan burung.
3. Akan memposisikan diri dengan rata-rata posisi burung yang lain dengan menjaga sehingga jarak antar burung dalam kawanan itu tidak terlalu jauh.

Dengan demikian perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor sebagai berikut:

1. Kohesi - terbang bersama.
2. Separasi – jangan terlalu dekat.
3. Penyesuaian (*alignment*) – mengikuti arah bersama.

Jadi PSO dikembangkan dengan berdasarkan pada model berikut:

Model ini akan disimulasikan dalam ruang dengan dimensi tertentu dengan sejumlah iterasi sehingga di setiap iterasi, posisi partikel akan semakin mengarah ke target yang dituju (minimasi atau maksimasi fungsi). Ini dilakukan hingga maksimum iterasi dicapai atau bisa juga digunakan kriteria penghentian yang lain.

### 2.8.1 Implementasi PSO

Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan. Setiap partikel bergerak dalam ruang/*space* tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi bagusnya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut.

Setiap partikel dalam PSO harus mempertimbangkan posisi saat ini, kecepatan saat ini, jarak ke *pbest*, dan jarak ke *gbest* untuk mengubah posisinya. PSO matematis dimodelkan sebagai berikut:

$$v_i^{t+1} = w.v_i^{t+1} + c_1 \times rand \times (pbest_i - x_i^t) + c_2 \times rand \times (gbest - x_i^t) \quad (2.8)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \quad (2.9)$$

Dimana $v_i^t$ merupakan kecepatan dari suatu partikel i pada iterasi t, $w$ merupakan fungsi pembobotan dari PSO. $c_j$ merupakan koefisien percepatan, *rand* merupakan nilai *random* dari 0 hingga 1, $x_i^t$ merupakan posisi partikel saat ini pada iterasi t, $pbest_i$ merupakan nilai dari *pbest* anggota i pada iterasi t, dan *gbest* merupakan solusi terbaik hingga iterasi t.

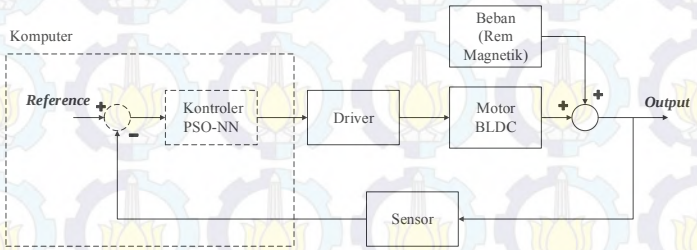
# BAB 3
# PERANCANGAN SISTEM

Bab ini menjelaskan segala hal terkait perancangan perangkat keras, perangkat lunak yang akan digunakan, serta perancangan kontroler menggunakan metode *Neural Network berbasis PSO*.

## 3.1 Gambaran Umum Sistem

Sistem yang dibahas dalam pada Tugas Akhir ini ialah sebuah sistem yang terdiri dari beberapa komponen yang dirangkai membentuk suatu *plant* BLDC P-1 terpadu untuk menjalankan objektif tertentu sesuai dengan tujuan dilaksanakan Tugas Akhir ini. *Plant* BLDC P-1 ini dibuat secara berkelompok yang terdiri dari penulis, M.Safrurriza,Tri Wahyu Kurniawan,Guntur S.P, Fairuzza Dinansyar, dan M. Iqbal Fauzi. Secara umum *plant* motor BLDC P-1 tersusun dari komponen penggerak utama yang sekaligus menjadi objek kontrol utama pada Tugas Akhir ini yaitu Motor Arus Searah Tanpa Sikat (BLDC Motor). Latar belakang penggunaan motor ini ialah karena saat ini perkembangannya dan penggunaannya semakin luas khususnya pada kendaraan listrik (mobil listrik) yang saat ini sedang banyak dikembangkan.

Di samping itu sistem pada *plant* BLDC ini juga menyertakan rem elektromagnetik yang bertujuan untuk memberikan efek pembebanan pada kinerja motor. Pembebanan yang dimaksud ialah representasi beban yang akan diterima motor pada penggunaan secara *real* ketika digunakan sebagai penggerak utama pada kendaraan listrik (mobil listrik). Adapun blok diagram sistem yang digunakan pada Tugas Akhir ini seperti ditunjukkan oleh Gambar 3.1 .




**Gambar 3.1** Diagram Blok Sistem Pengaturan Kecepatan Motor *Brushless* DC (BLDC)

Untuk mendukung kinerja *plant* BLDC tersebut dibutuhkan komponen pendukung lain berupa sub sistem yang mendukung kerja dan fungsi masing-masing perangkat yang telah disebutkan sebelumnya. Perangkat pendukung tersebut antara lain *driver* motor BLDC yang digunakan untuk mengatur kecepatan, rangkaian sensor kecepatan, rangkaian pendukung rem elektromagnetik untuk membantu menyearahkan sumber listrik yang didapat dari sumber listrik PLN (220V) dan juga rangkaian pengatur PWM (*Pulse Width Modulation)* pada rem yang digunakan untuk mengatur besarnya sinyal kontrol yang diberikan pada rem guna membantu besarnya pembebanan.

Pengaturan kecepatan motor BLDC yang sekaligus sebagai objek kontrol utama Tugas Akhir ini menggunakan metode kontrol *Neural Network* berbasis *PSO* yang merupakan kombinasi antara jaringan saraf tiruan (*Neural Network*) dan optimasi PSO. Penggunaan metode kontrol ini diharapkan dapat membantu meningkatkan performa motor untuk menghasilkan respon performansi yang diinginkan.

## 3.2 Perancangan Perangkat Keras

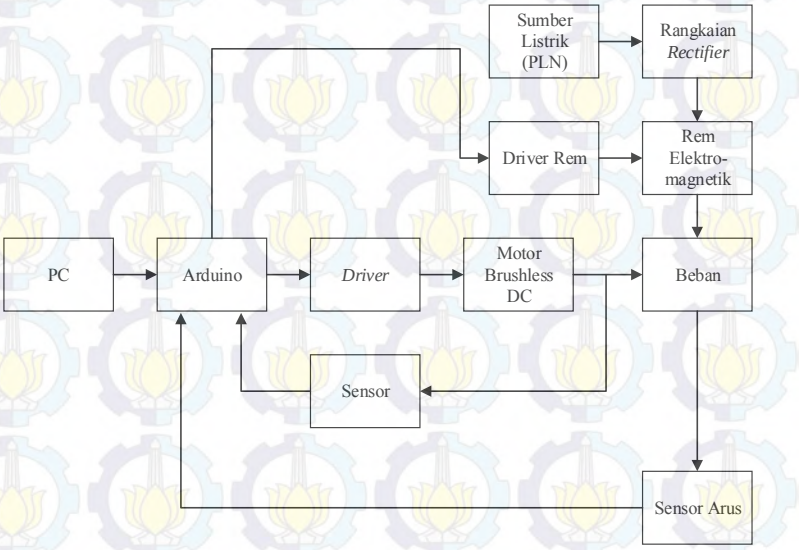
Pada tahap perancangan keras, terdapat 2 jenis perancangan yang dilakukan, yaitu perancangan mekanik dan elektronik. Perancangan mekanik merupakan perancangan komponen utama pada *plant*, yaitu berupa motor BLDC dan rem elektromagnetik. Sedangkan perancangan elektronik merupakan perancangan untuk kontroler, *driver* dan rangkaian sensor yang akan digunakan pada *plant* ini. Kontroler akan diprogram didalam PC dan sebagai perantara komputer dengan *plant*, digunakan mikrokontroler Arduino yang juga berfungsi sebagai perangkat akuisisi data.

Arduino akan menerima data dari sensor dan mengirimkannya kepada komputer. Selain itu, digunakan digunakan pula rangkaian *driver* untuk menggerakan motor BLDC dan memberi *input* arus pada rem elektromagnetik. Terdapat pula rangkaian sensor yang berfungsi mengukur *input* arus yang masuk ke dalam rem elektromagnetik dan sensor kecepatan motor BLDC. Konfigurasi perangkat keras pada *plant* dapat dilihat pada Gambar 3.2.

### 3.2.1 Perancangan Mekanik

Pada *plant* sistem pengaturan kecepatan motor BLDC P-1, terdapat beberapa komponen utama yang digunakan, antara lain motor BLDC sebagai tenaga penggerak dan objek yang dikontrol. Selain itu, terdapat

rem elektromagnetik yang digunakan untuk memberikan efek pembebanan pada motor BLDC. Rem elektromagnetik diberikan *input* arus DC yang berbentuk PWM. Untuk lebih jelasnya, penjelasan mengenai konstruksi motor BLDC dan rem elektromagnetik dapat dilihat pada subbab di bawah ini



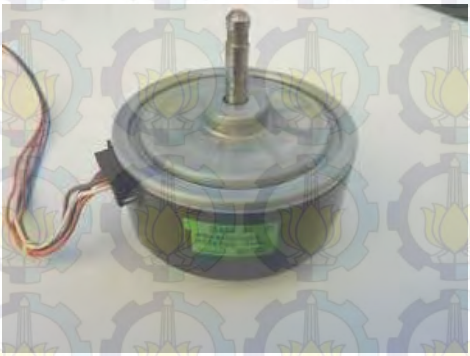

**Gambar 3.2** Konfigurasi Perangkat Keras Sistem Pengaturan Kecepatan Motor *Brushless* DC (BLDC)

### *3.2.1.1 Motor Brushless DC*

Motor *Brushless* DC (BLDC) dipilih karena efisiensi dan konstruksinya yang tidak menggunaakan *brush* atau sikat. Motor BLDC yang digunakan merupakan motor BLDC yang digunakan pada *air conditioner inverter* Daikin. Motor BLDC jenis ini merupakan miniatur dari motor BLDC penggerak kendaraan listrik dikarenakan konstruksinya yang lebih kecil. Bentuk fisik motor DC ditunjukkan pada Gambar 3.3.

| | |
|---|---|
| Tipe | : Daikin D43F |
| Tipe | : *inrunner* |
| Tegangan Nominal | : 307 VDC |

Arus
Tanpa Beban : 5,46 mA
Beban Maksimal : 24V
Kecepatan putar
Tanpa Beban : 1282 rpm
Beban Maksimal
Daya Keluaran
Tanpa Beban :1,734 Watt
Beban Maksimal



**Gambar 3.3** Bentuk Fisik Motor *Brushless* DC ( BLDC )

### *3.2.1.2 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, akan menyebabkan kerusakan pada *bearing*. Bahkan *shaft* bisa bengkok jika kedua *shaft* diputar pada keadaan tidak lurus. Untuk menjaga kedua *shaft* 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.

### *3.2.1.3 Rem Elektromagnetik*

Rem elektromagnetik pada *plant* ini berguna sebagai pembebanan pada motor. Medan elektromagnetik dari rem ini dihasilkan oleh beberapa

kumparan yang dihubungkan secara seri dan paralel dan diberikan masukan arus DC. Arus DC yang masuk ke dalam rem elektromagnetik berbentuk PWM yang *duty cycle*-nya diatur oleh *driver* dan Arduino.

Rem elektromagnetik ini terbuat dari 8 kumparan, terdiri dari 4 kumparan di setiap sisinya, yang disusun secara seri dan paralel. Maksudnya 4 kumparan pada satu sisi akan dirangkai secara seri begitu juga sisi lainnya. Lalu kedua sisi tersebut dihubungkan secara paralel. Diantara celah kedua sisi kumparan dipasang piringan aluminium. Piringan aluminium tersebut lalu dipasang ke dalam sebuah *shaft* yang selanjutnya dikopel dengan motor BLDC. Perbandingan *gear* yang digunakan untuk mengkopel motor BLDC dengan *shaft* tersebut.

Nilai torsi pengereman berbanding lurus dengan fluks magnet, diameter cakram, ketebalan cakram, diameter inti magnet, konduktivitas cakram. Pengaturan torsi pengereman dapat dilakukan dengan mengatur tegangan keluaran *driver* rem yang berupa PWM.

$$\tau = n\frac{\pi\sigma}{4}D^2 dB^2 R^2 \dot{\theta}$$

Spesifikasi rem elektromagnetik:

- Jumlah kumparan : 8 (masing –masing 400 kumparan)
- Resistansi kumparan(seri) : 12 Ω
- Diamater kumparan : 3cm
- Diameter kawat : 0,6mm
- Tegangan kerja : 0-30V
- Arus maksimal : 3A

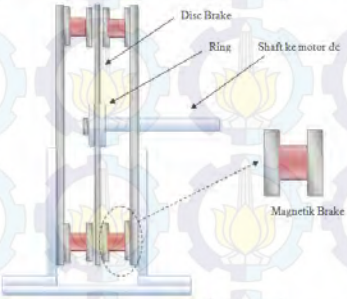
Kumparan ini terbuat dari 8 buah baut dan 16 buah mur. Lalu pada baut dilapisi kertas sebagai isolator. Selanjutnya kawat tembaga dililit pada kertas tersebut hingga jumlah lilitan yang telah ditentukan pada Persamaaan 3.1. Jumlah lilitan tiap kumparan dihitung berdasarkan rumus berikut ini:

$$N = \frac{44}{d} \times V = \frac{44}{1,4} \times 36 = 1131 \qquad (3.1)$$

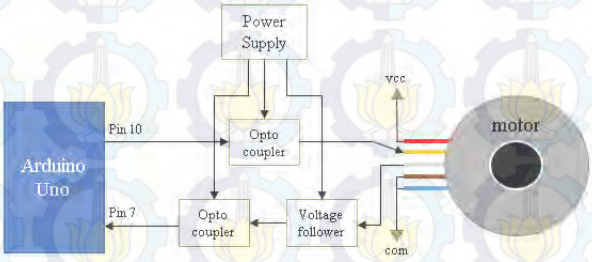
$N$ : Jumlah lilitan tiap sisi
$d$ : Diameter kumparan (cm)
$V$ : Tegangan *input* (Volt)

### 3.2.2 Perancangan Elektronik

Pada perancangan elektronik, dirancang berbagai macam komponen pendukung dalam *plant* motor BLDC ini. Perancangan elektronik ini meliputi desain rangkaian PCB komponen pendukung serta sistem pengkabelannya. Rangkaian elektronik pada *plant* ini meliputi rangkaian *driver* rem elektromagnetik serta sensor arus dapat dilihat pada Gambar 3.4. Selain itu, digambarkan pula rancangan pengkabelan pada mikrokontroler Arduino. Dengan konfigurasi perkabelan komponen elektrik *plant* yang dapat dilihat pada Gambar 3.5




**Gambar 3.5** Rancangan Konstruksi Rem Elektromagnetik




**Gambar 3.4** Konfigurasi Sistem Elektrik Motor BLDC

## 3.3 Perancangan Perangkat Lunak

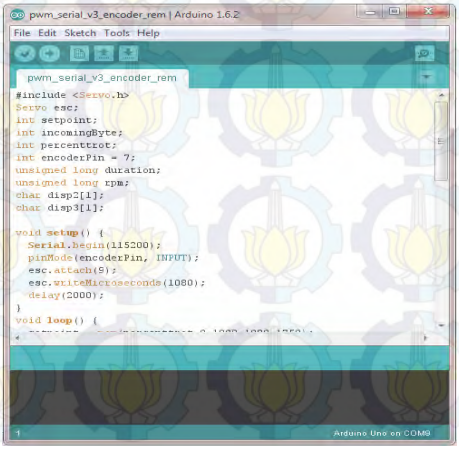
Perangkat lunak diperlukan dalam perancangan sistem sebagai *interface* antara *plant* BLDC dan komputer. Perangkat lunak yang digunakan dalam pengerjaan Tugas Akhir ini antara lain adalah

penggunaan perangkat lunak Arduino dan MATLAB. Arduino digunakan sebagai alat akuisisi data, sedangkan MATLAB digunakan untuk membaca nilai dari sensor untuk keperluan identifikasi sistem dan mengirimkan sinyal kontrol pada *driver*. Selain itu, MATLAB digunakan pula sebagai *software* untuk desain, simulasi dan implementasi kontroler.

### 3.3.1 *Software* Arduino

Seperti telah yang dijelaskan sebelumnya, mikrokontroler Arduino digunakan sebagai alat akuisisi data dari berbagai macam komponen pada *plant*. *Software* yang sering digunakan untuk meng-*compile* bahasa pemrograman pada Arduino bernama Arduino IDE. Program akuisisi data yang dikehendaki ditulis pada Arduino IDE di komputer, lalu di-*compile* dan di-*upload* menuju mikrokontroler via *serial* USB.

Secara garis besar, Pada tugas akhir ini, Arduino digunakan sebagai *interface* yang menghubungkan *plant* dengan laptop melalui komunikasi *serial*. Arduino menerima data sinyal kontrol yang dikirim oleh perangkat lunak Matlab dan mengeluarkan sinyal PWM sesuai dengan data tersebut untuk kemudian menjadi masukan dari *plant*. Kemudian, Arduino menghitung kecepatan putar motor dan mengirimkan nilainya ke Matlab. Matlab mengolah data kecepatan putar motor untuk kemudian mengirimkan kembali sinyal kontrol hasil pengolahan kecepatan putar motor ke Arduino dan begitu seterusnya. Tampilan dari program dapat dilihat pada Gambar 3.6




**Gambar 3.6** Tampilan *Interface* Arduino IDE

### 3.3.2 *Software* MATLAB

Selain Arduino IDE, *software* lain yang digunakan pada Tugas Akhir kali ini adalah *software* MATLAB. Versi MATLAB yang digunakan pada pelaksanaan Tugas Akhir kali ini mempunyai versi 2015a. *Software* MATLAB merupakan *software* yang sangat vital dan umum digunakan untuk desain kontroler beserta implementasinya. Pada *software* MATLAB, terdapat sub-program lainnya yang bernama Simulink. Software Simulink digunakan sebagai *Human Machine Interface* pada proses pengiriman dan penerimaan data melalui *serial* USB dari mikrokontroler Arduino

Dengan menggunakan komunikasi *serial* pada Arduino, Simulink MATLAB dapat mengolah data dari blok *serial receive* dan mengirimkannya kembali melalui blok *serial send.* Simulink juga digunakan untuk proses identifikasi sistem *open loop* maupun *closed loop* menggunakan blok *Instrument Control Toolbox*.

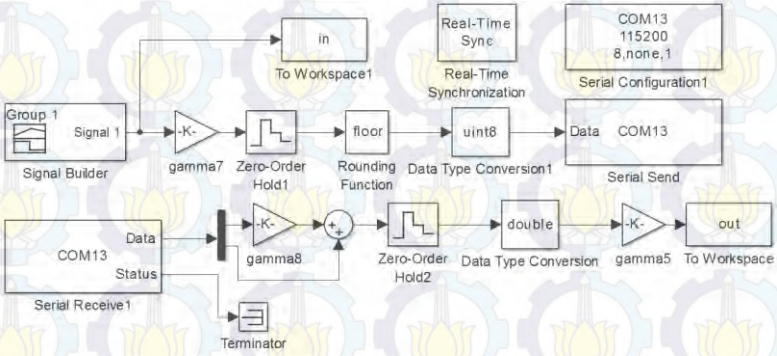
MATLAB juga digunakan untuk keperluan proses *learning* dan perancangan kontroler *Neural-Network* yang dipergunakan. Seperti yang telah diketahui sebelumnya bahwa pada proses *learning* kontroler ini digunakan metode *reinforced learning* yang menggunakan *algoritma* PSO, dimana nantinya dibuat *script* untuk mempermudah proses *learning* pada perubahan nilai bobot pada *neural network* untuk memperoleh hasil yang diharapkan. Setelah itu MATLAB digunakan untuk mensimulasikan hasil perancangan kontroler terhadap hasil pemodelan *plant* yang diperoleh. Terakhir, MATLAB digunakan sebagai antarmuka untuk melakukan implementasi kontroler yang telah dirancang sebelumnya untuk mengatur kecepatan motor BLDC.

## 3.4 Identifikasi dan Pemodelan Sistem

Pada Proses identifikasi yang dilakukan ini, digunakan *System Identification Toolbox* yang terdapat pada MATLAB. Dengan *Toolbox* ini, *user* dapat memperoleh model matematika *plant* dengan memasukkan data *input-output*, melakukan *preprocessing*, dan memilih bentuk model yang diinginkan. Pada tugas akhir kali ini, saya ingin mendapatkan model matematika *plant* dalam bentuk fungsi alih kontinyu. Diagram blok simulink identifikasi sistem dapat dilihat pada Gambar 3.7.

Dalam melakukan identifikasi menggunakan *System Identification Toolbox*, pertama-tama, data *input-output* yang telah didapatkan dimasukkan ke dalam aplikasi dengan memilih "*time domain data*" saat meng-klik *popout* bertuliskan "*import data*". Setelah itu, dilakukan *preprocessing*. Proses yang saya pilih pada tahap ini adalah

pembuangan nilai rata-rata atau dengan kata lain mengurangi seluruh nilai *input* dan *output* dengan rata-ratanya. Hal ini dapat dilakukan dengan memilih "*remove means*" setelah meng-klik *popout* bertuliskan "*preprocess*". Data yang telah melalui tahap *preprocess* kemudian dipilih sebagai *working data* sekaligus *validation data* dengan cara *drag-and-drop* kotak yang berisi data tersebut ke kotak *working data* dan *validation data*. Kemudian dilakukan identifikasi model dengan memilih "*transfer function*" setelah meng-klik *popout* bertuliskan "*estimate*". Fungsi transfer hasil identifikasi dibatasi hanya berupa fungsi transfer orde dua tanpa *zero*. Hasil identifikasi dapat dilihat pada Tabel 3.3.




**Gambar 3.7** Diagram Blok Simulink untuk Identifikasi Sistem

**Tabel 3.1** Fungsi *Transfer* Hasil Identifikasi

| Keadaan 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 *Neural Network* Berbasis PSO

Setelah fungsi alih dari sistem telah didapatkan, langkah selanjutnya adalah merancang kontroler untuk pengaturan kecepatan motor. Kontroler digunakan untuk mengembalikan respon ke nilai *setpoint* yang diinginkan sekalipun motor BLDC diberi beban. Tahapan desain kontroler ini meliputi perancangan *learning* pada kontroler NN dengan algoritma PSO dan perancangan kontroler NN.

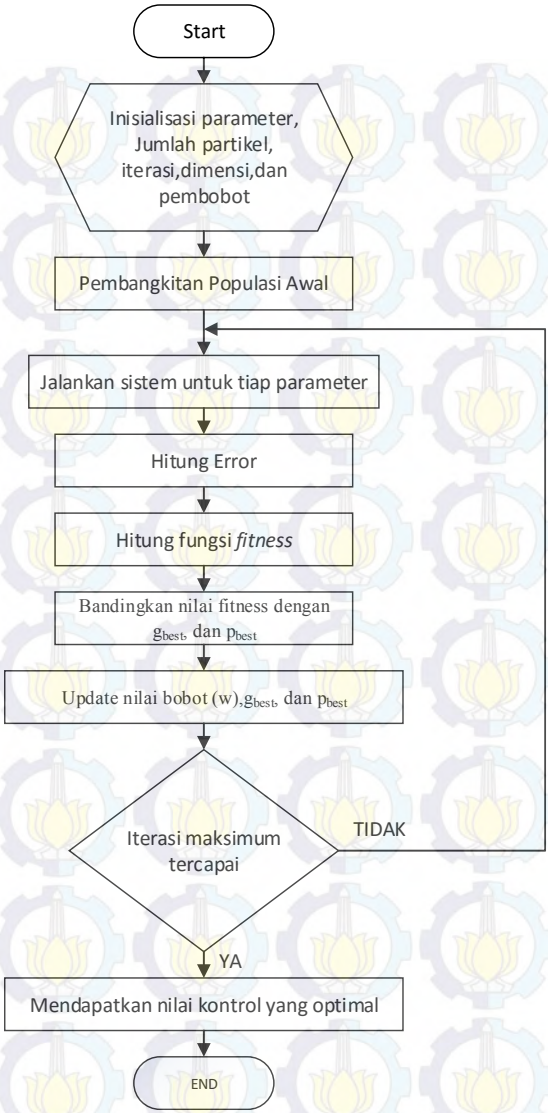
### 3.5.1 Perancangan *Learning* pada Kontroler *Neural Network* dengan Algoritma *Particle Swarm Optimization*

Sebelum merancang kontroler *Neural Network* diperlukan penyesuaian nilai bobot untuk kontroler NN. Dengan merancang *learning* ini maka akan didapatkan nilai bobot yang sesuai untuk *output* yang diharapkan dari kontroler *Neural Network*.

Pada tahap ini dibuat rancangan *learning* pada kontroler *Neural Network* dengan menggunakan algoritma PSO. Tipe *learning* yang dipergunakan termasuk kedalam tipe *reinforced learning*. Artinya, algoritma pembelajaran dari *mapping input-output* secara terus menerus berinteraksi dengan lingkungan, dimana hal ini dapat memberikan pengaruh dari setiap langkah yang diambil, memberikan *feedback* kepada model. Dengan *reinforced learning* ini, proses *learning* pada *plant* dapat disesuaikan dengan kebutuhan *user*. Selain itu dengan menggunakan algoritma *Particle Swarm Optimization*, nilai optimal dapat diperoleh dengan waktu yang lebih singkat. Adapun diagram alur perancangan *learning* pada *neural network* dengan algoritma PSO ada sebagai berikut:

Sesuai dengan Gambar 3.8 proses learning pada *Neural Network* dengan menggunakan *Particle Swarm Optimization* dilakukan melalui proses berikut:

1. Inisialisasi Paramater NN dan PSO
   Pada proses ini diberikan inisialisasi parameter untuk *Learning* NN dan PSO. Parameter untuk NN sendiri berupa jumlah nilai bobot yang dicari, *Input Data Neural Network*, Keluaran yang diharapkan dari NN (*desired value*), serta jumlah *epoch*.
   Untuk parameter PSO yang diberikan yaitu Jumlah *Training Sample*, nilai bobot *inersia* PSO, nilai konstanta percepatan PSO, nilai awal parameter posisi dan kecepatan, nilai awak $p_{best}$, dan $g_{best}$, Serta nilai *fitness* awal dari PSO.




**Gambar 3.8** Diagram Alur Proses *Learning* NN berbasis PSO

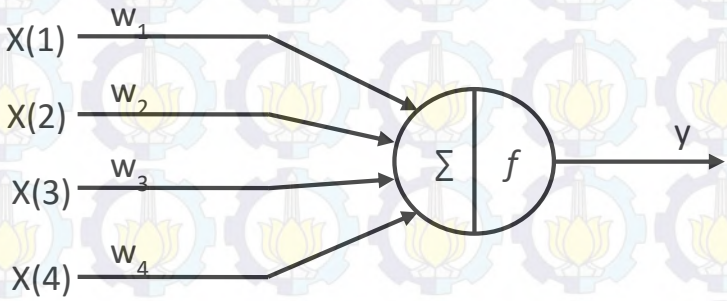
2. Pembangkitan Populasi Awal.
   Pembangkitan populasi awal sendiri dilakukan dengan memberikan nilai *random* pada setiap *particle* dari *swarm* yang dicari.
3. Jalankan Sistem untuk Setiap Parameter
   Setiap parameter yang sudah diberikan akan melakukan perhitungan sesuai dengan sistem perhitungan *feed-forward* pada NN.
4. Hitung Nilai *Error*
   Error dari proses learning ini merupakan perbedaan hasil dari keluaran *Neural Network* yang sebenarnya (*actual value*) dengan keluaran yang diharapkan (*desired value*).
5. Hitung Fungsi *Fitness*
   Perhitungan fungsi *fitness* dapat diperoleh dengan mengambil nilai rata-rata dari kuadrat error dari hasil *learning Neural Network.*

$$fitness = \frac{\sum(u_{actual} - u_{desired})^2}{jumlah\ partikel}$$

6. Bandingkan Nilai *fitness* dengan *pbest* dan *gbest*.
   Pada proses ini nilai fitness yang sudah diperoleh akan dibandingkan dengan nilai *pbest* dan *gbest* untuk memperoleh nilai optimalnya. Nilai *gbest* sendiri merupakan nilai minimal secara keseluruhan, nilai ini yang digunakan sebagai nilai bobot pada *Neural Network*.
7. *Update* nilai bobot (*Neural Network*), *pbest* dan *gbest*.
   Sesuai dengan proses sebelumnya, dari hasil pembandingan antara fitness dengan pbest dang best diambil nilai terendah yang kemudian diambil dan dipergunakan sebagai *pbest* ataupun *gbest*. Selanjutnya, perubahan nilai bobot dilakukan sesuai dengan perubahan dari nilai *gbest*. Nilai dari bobot sendiri menyesuaikan dengan masukan dan keluaran harapan dari NN. Diambil nilai bobot dengan *error* paling minimal sebagai solusinya.
8. Proses ini di-iterasikan hingga mencapai *epoch* yang diberikan

### 3.5.2 Perancangan Kontroler *Neural Network*

Setelah melalui proses learning, *neural network* memiliki nilai bobot yang optimal. Selanjutnya, kontroler bekerja secara *Feed-Forward*. *Controller* yang dibuat menggunakan 4 node pada layer *input* dan 1 node pada layer *output*. Pada struktur *Neural Network* yang dirancang tidak menggunakan *hidden layer* (*Single Layered Neural Network*). Struktur ini sendiri dibuat dengan harapan agar kontroler *Neural Network* yang dibuat dapat bekerja menyerupai kerja kontroler PID Diskrit. Struktur NN yang digunakan dapat dilihat pada Gambar 3.9.




**Gambar 3.9** Struktur *Neural Network*

Pemodelan ini sendiri dimodelkan sesuai dengan pemodelan PID Diskrit untuk memudahkan proses implementasi dan jika dilakukan *learning* secara *online*. Nilai dari PID diskrit sendiri dicari untuk memberikan batasan pada beban agar nilai yang ada tidak terlalu luar. Perhitungan dari PID Diskrit sendiri dapat dilihat pada Persamaan 3.2.

$$u(k) = K_p(e(k) + e(k-1)) + K_i\big(e(k)\big) + K_d\Big(e(k) - 2\big(e(k-1)\big) + e(k-2)\Big) + u(k-1) \quad (3.2)$$

Sesuai dengan Persamaan 3.2, Parameter nilai dari *neural network* dapat dilihat pada Tabel 3.2.

**Tabel 3.2** Parameter Nilai Variabel Kontroler *Neural Network*

| Variabel | Parameter |
|---|---|
| X(1) | $(e(k) + e(k-1))$ |
| X(2) | $\big(e(k)\big)$ |
| X(3) | $\big(e(k) - 2\big(e(k-1)\big) + e(k-2)\big)$ |
| X(4) | $u(k-1)$ |
| $w_1$ | $K_p$ |
| $w_2$ | $K_i$ |
| $w_3$ | $K_d$ |
| $w_4$ | 1 |

# BAB 4
# PENGUJIAN DAN ANALISA

Pada bab ini dijelaskan proses pengujian, hasil dan analisa dari data yang diperoleh, baik berupa simulasi *software* dan juga implementasi sistem pada pengaturan kecepatan motor BLDC.

## 4.1 Gambaran Umum Pengujian Sistem

Pada Tugas Akhir ini dilakukan pengujian sistem dengan dua cara yaitu simulasi dan implementasi. Pengujian sistem dimaksudkan untuk mengetahui performansi sistem dari *plant* motor BLDC terhadap adanya beban tambahan dengan menggunakan rem elektromagnetik.

Pada tahapan ini, terdapat beberapa hal yang diuji. Diantaranya adalah pengujian sensor kecepatan motor, pengujian *open-loop* kecepatan motor, pengujian kontroler dengan simulasi dan pengujian kontroler pada *plant real*.

Pengujian sensor kecepatan motor dilakukan dengan menghitung nilai frekuensi sinyal *output driver* motor, kemudian membandingkannya dengan hasil bacaan dari *tachogenerator*. Hal ini berguna untuk mengetahui hubungan frekuensi sinyal dengan kecepatan motor.

Pengujian *open-loop* kecepatan motor dilakukan dengan memberikan *input* tegangan pada motor dari 0 Volt sampai 5 Volt secara bertahap. Kecepatan motor kemudian dibaca pada tiap tahapan sehingga didapat grafik *input-output plant*.

Pengujian simulasi dilakukan dengan mensimulasikan kontroler yang telah dirancang di perangkat lunak Matlab. Simulasi ini menggunakan fungsi *transfer plant* hasil identifikasi yang telah diterangkan di Bab 3.

Pengujian terakhir adalah implementasi kontroler ke *plant* sebenarnya.

## 4.2 Pengujian Perangkat Keras

Pada pelaksanaan Tugas Akhir ini pengujian perangkat keras dilakukan untuk mengetahui apakah perangkat keras yang digunakan bekerja dan berfungsi pada sistem secara semestinya atau tidak. Pengujian perangkat keras yang dilakukan pada Tugas Akhir ini terdiri dari dua buah pengujian utama yakni pengujian sensor kecepatan, dan pengujian *open loop*.

### 4.2.1 Pengujian Sensor Kecepatan

Pada pengujian kecepatan motor, diukur kecepatan dengan menggunakan perhitungan frekuensi dari sinyal kotak oleh *driver* bawaan motor menggunakan fungsi *pulse-in* Arduino. Kecepatan motor dihitung dengan hubungan $\omega = 15 \times f$. Hasil pembacaan Arduino kemudian dibandingkan dengan hasil pembacaan dengan *tachogenerator*. Hasil pengujian sensor kecepatan ini dapat dilihat pada Tabel 4.1.

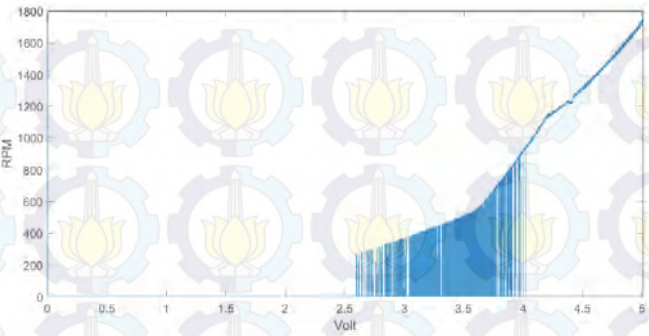
**Tabel 4.1** Hasil Pengujian Sensor Kecepatan

| Hasil Pembacaan (RPM) | | % 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 | 1,3 |
| 1703,6 | 1726 | 1,31 |
| 2253 | 2290,45 | 1,67 |
| 2397 | 2441,21 | 1,84 |

Dapat dilihat, dari hasil pengujian diketahui bahwa hasil pembacaan yang didapatkan oleh Arduino tidak berbeda jauh dengan hasil pembacaan dari *tachogenerator* dengan *error* berkisar antara 1,11-1,84 %. Sehingga dapat disimpulkan bahwa metode pengukuran kecepatan motor dengan menghitung frekuensi pulsa dari keluaran *driver motor* sudah benar.

### 4.2.2 Pengujian *Open-Loop* Kecepatan Motor

Pengujian ini dilakukan untuk mengetahui grafik hubungan *input-output plant*. Pengujian dilakukan pada saat beban nominal dengan memberikan sinyal tangga pada *input plant* dan diukur kecepatannya. Hasil dari pengujian dapat dilihat pada Gambar 4.1.




**Gambar 4.1** Hasil Pengujian *Open-Loop Plant*

Dari gambar tersebut, terlihat bahwa pada saat kecepatan dibawah 1000 RPM, kecepatan motor tidak terukur dengan baik. Hal ini menunjukkan kekurangan dari sensor kecepatan motor sehingga rentang kerja dari *plant* ini tidak menggunakan rentang kecepatan dibawah 1000 RPM.

Pada kecepatan di atas 1000 RPM, terlihat bahwa hubungan *input-output plant* relatif linier dengan batas maksimum kecepatan motor berkisar pada 1700 RPM. Maka, pada Tugas Akhir ini, dipilih rentang kerja motor antara 1000-1500 RPM.

## 4.3 Simulasi Sistem

Simulasi sistem akan dilakukan dengan bantuan program Simulink yang ada pada *software* MATLAB R2015a. Sistem yang ada pada Simulink akan dirancang sebagaimana dengan keadaan yang ada pada *plant* BLDC.

### 4.3.1 Simulasi dengan Menggunakan Kontroler *Neural-Network*

Proses kontrol yang ada pada kontroler ini terbagi menjadi dua tahap, yakni proses *learning* dan proses simulasi. Pada proses *learning*, struktur *Neural-Network* akan bekerja untuk meminimalisir *error* yang ada dengan cara melakukan perbaikan bobot yang ada pada struktur tersebut. Operasi yang dilakukan oleh kontroler *Neural-Network* bekerja dalam dua tahapan yakni *forward propagation* dan *Reinforced Learning*.

*Forward propagation* merupakan proses maju ke depan melalui struktur jaringan dengan hasil akhir sebuah sinyal kontrol. Di akhir

operasi, sinyal kontrol tersebut akan dibandingkan dengan target *output* sinyal kontrol yang telah ditentukan sebelumnya. Ketika target *output* sinyal kontrol belum sesuai dengan *output* yang diinginkan, maka sistem akan melakukan perbaikan bobot untuk meminimalisir *error* yang ada. Proses perbaikan bobot tersebut menggunakan metode *Reinforced learning* yang menggunakan algoritma PSO.

#### *4.3.1.1 Proses Learning Kontroler Neural-Network*

Sebelum melaksanakan simulasi terlebih dahulu dilakukan proses *learning* oleh program dari *Neural-Network* tersebut. Pada proses *learning* ini jumlah *epoch* yang diinginkan disesuaikan dengan banyaknya proses perulangan yang diinginkan. Jumlah *epoch* ini berpengaruh terhadap perubahan bobot yang terjadi. Semakin kecil toleransi *error* yang diinginkan disertai banyaknya *epoch* atau perulangan yang dilakukan akan membuat hasil semakin baik. Namun apabila jumlah *epoch* yang dilakukan semakin ditingkatkan maka proses *learning* yang berlangsung akan semakin lama.

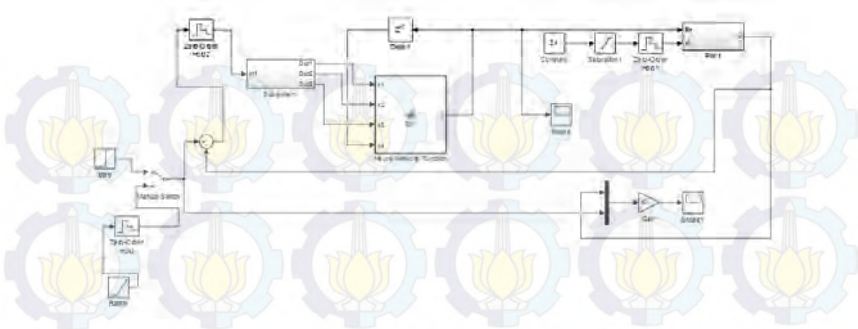
Setelah proses *learning* selesai, kemudian dilanjutkan dengan proses penggunaan nilai bobot yang telah didapat ke dalam simulasi *plant* BLDC untuk mendapatkan hasil respon *output* dari *plant* BLDC. Dari hasil *learning* diperoleh nilai bobot $w_1 = 0{,}13678236677$, $w_2 = 0{,}56795706196$, $w_3 = 0{,}00235166854$, dan $w_4 = 1$.

#### *4.3.1.2 Proses Simulasi Kontroler*

Dari hasil *learning* yang dilakukan sebelumnya didapat nilai bobot yang berasal dari bobot nilai *error* yang terkecil. Nilai bobot tersebutlah yang digunakan untuk menjalankan kontroler *Neural-Network*. Simulasi dilakukan dengan program Simulink yang terdapat pada *software* MATLAB 2015a, tampilan hasil perancangan ada pada Gambar 4.2

### 4.3.2 Pengujian Simulasi Respon dengan Kontroler

Pada tahapan ini kontroler yang telah dibuat akan dicoba dijalankan pada model hasil identifikasi sistem. Pengujian ini dilakukan dengan memberikan respon *step* di semua kondisi pembebanan mulai dari kondisi beban nominal (rem 16 V) sampai dengan kondisi beban maksimal. Setelah itu respon akan dianalisa untuk dicari nilai *rise time*, *settling time*, dan *error steady state*.



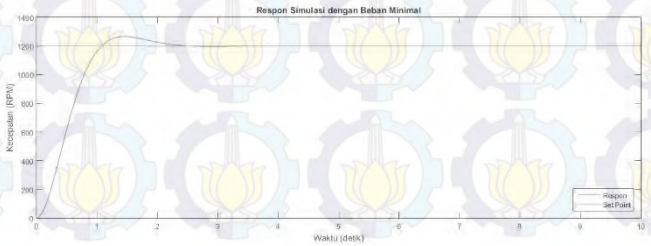
**Gambar 4.2** Blok Simulink Simulasi Sistem Motor BLDC

Pengujian dilakukan dengan memberikan sinyal *step* pada semua fungsi alih di setiap pembebanan.

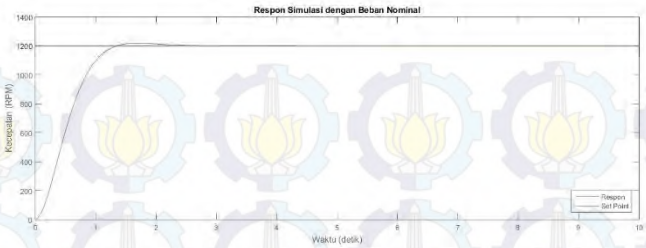
Setelah dilakukan simulasi akan dilakukan analisa terhadap respon hasil dari kontroler. Yang dilakukan selanjutnya adalah menghitung *settling time*, *rise time* dan *error steady state*.

Pada perhitungan nilai *settling time* dilakukan dengan cara menghitung nilai respon *output* yang berada pada ±5% atau ±2% atau ±0,5% dari nilai *steady state* respon. Sedangkan pada penghitungan *rise time* yang perlu dilakukan ialah menghitung waktu yang dibutuhkan oleh respon dari mulai 0% dari nilai *steady state* hingga ke 100% dari nilai *steady state* karena respon *underdamped* untuk beban minimal dan nominal.

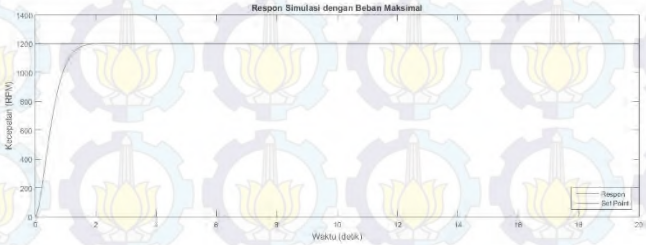
Dapat dilihat respon hasil simulasi dari pemodelan sistem dengan beban minimal, nominal, dan maksimal pada Gambar 4.3, Gambar 4.4 dan Gambar 4.5. Untuk lebih jelas mengenai analisa dapat dilihat pada Tabel 4.2




**Gambar 4.3** Respon Simulasi dengan Beban 16V




**Gambar 4.4** Respon Simulasi dengan Beban 20V




**Gambar 4.5** Respon Simulasi dengan Beban 24V

**Tabel 4.2** Indeks Performansi Respon Simulasi Sistem

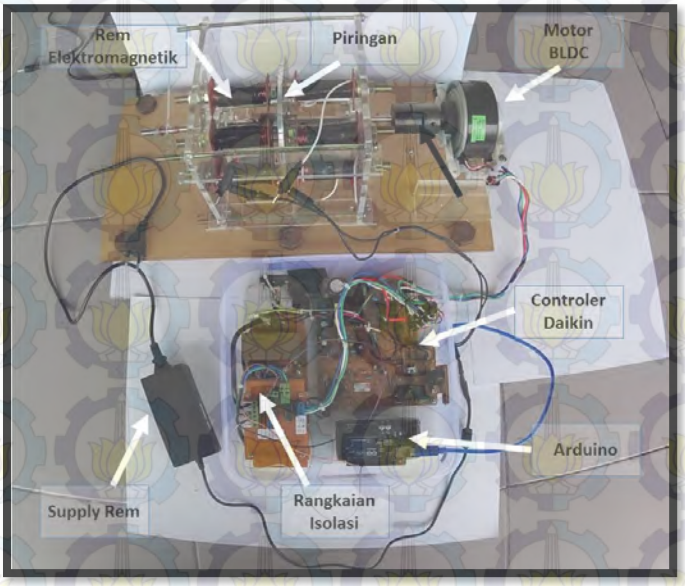
| Beban | Minimal (16V) | Nominal (20V) | Maksimal (24V) |
|---|---|---|---|
| *Overshoot* | 5,41% | 1,5% | 0,167% |
| *Settling Time* | 3,7 detik | 3,1 detik | 3 detik |
| *Rise Time* | 1,081 detik | 1,374 detik | 1,973 detik |
| Ess | 0 % | 0% | 0% |

## 4.4 Realisasi *Plant* Sistem Motor BLDC

Sub bab ini menjelaskan mengenai *realisasi plant* motor BLDC dari hasil perancangan yang telah dijelaskan sebelumnya. Secara umum *plant* sistem motor BLDC ini terdiri dari komponen penggerak utama berupa motor BLDC, komponen pembaca kecepatan yang menggunakan bantuan Arduino Uno, komponen pembebanan berupa rem elektromagnetik, serta rangkaian elektronik pendukung seperti *driver* rem elektromagnetik, rangkaian *signal conditioning*, dan Arduino Uno sebagai *interface* serta sebagai penghubung antara *plant* BLDC dengan

komputer. Hasil akhir realisasi *plant* BLDC dapat dilihat pada Gambar 4.6.

Komponen penggerak utama yang digunakan pada *plant* BLDC tersebut ialah motor jenis BLDC. Penggunaan motor jenis BLDC ini mengacu pada objektif penelitian terkait motor yang umum digunakan pada mobil listrik karena karakteristik dan kelebihan yang dimiliki motor jenis BLDC tersebut. Pada *plant* BLDC-P1 ini, motor BLDC yang digunakan ialah motor BLDC yang biasa digunakan pada perangkat *Air Conditioner* (AC/Pendingin Ruangan). Motor BLDC tersebut merupakan hasil pabrikan dari produsen keperluan pendingin ruangan yang dipergunakan pada *plant* ini yaitu Daikin D43F. Motor jenis ini memiliki karakteristik dengan tipe *inrunner*, Memiliki tegangan nominal 307 VDC, dengan kecepatan tanpa beban maksimal pada kisaran 1700 RPM. Dilihat dari struktur fisik motor yang cukup lebar, biasanya motor jenis ini memiliki kecepatan putar yang cukup rendah tetapi memiliki torsi yang lebih besar.

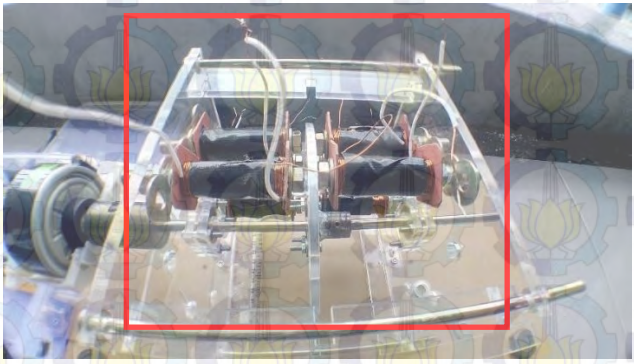



**Gambar 4.6** Bentuk Fisik Realisasi Motor BLDC

Pada Gambar 4.6 dapat dilihat bahwa motor BLDC dihubungkan dengan kopel karet agar sambungan antara motor dengan rem dapat terjaga. Motor BLDC ini digerakan oleh sinyal kontrol yang dihasilkan oleh *driver* motor BLDC .

*Driver* ini berfungsi untuk mengubah sinyal kontroler yang dikirimkan oleh mikrokontroler Arduino Uno menjadi tegangan 3 fasa yang digunakan motor BLDC untuk beroperasi.

Subsistem selanjutnya ialah rem elektromagnetik. Realisasi subsistem ini dibuat dengan beberapa komponen pendukung seperti kumparan dan piringan. Rem elektromagnetik ini terdiri dari dua sisi yang berhadapan dengan masing-masing sisinya memiliki 4 kumparan. Penggunaan dua sisi pada rem elektromagnetik ini merepresentasikan kutub magnet utara dan selatan. Kedua sisi tersebut masing-masing dirangkai secara seri dan paralel. Hal ini dimaksudkan untuk menghasilkan arus *eddy* yang dapat memberikan gaya lawan pada motor. Di antara kedua sisi kumparan tersebut dipasang lempengan alumunium yang bersifat non-*ferromagnetik* sebagai media pengereman yang akan mendapat efek langsung pengereman dari arus *eddy*. Lempengan alumunium tersebut terhubung langsung dengan *shaft* yang juga terhubung dengan motor BLDC. Kumparan pada rem elektromagnetik ini dibuat dari baut yang dililitkan kawat tembaga. Kumparan tersebut disusun pada dudukan yang terbuat dari bahan akrilik. Penampakan fisik realisasi rem elektromagnetik dapat dilihat pada Gambar 4.7 berikut.

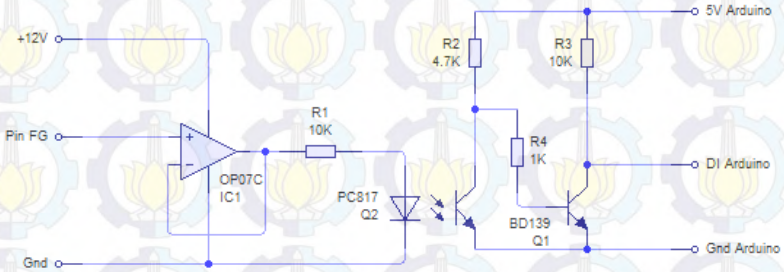


**Gambar 4.7** Bentuk Fisik Rem Elektromagnetik

Besar kekuatan medan magnet yang dihasilkan pada rem elektromagnetik ini diatur dengan cara mengatur tegangan yang masuk dari sumber menjadi PWM. Pengaturan PWM tersebut menggunakan rangkaian *driver* rem elektromagnetik.

Subsistem selanjutnya ialah sensor pembaca kecepatan yang memiliki peran *vital* dalam pelaksanaan Tugas Akhir. Untuk mendapatkan informasi seberapa besar kecepatan motor yang terjadi pada motor BLDC secara *realtime*, digunakan sebuah rangkaian yang dipergunakan sebagai pengganti *sensor hall* dan *encoder* untuk mengukur kecepatan motor.

Pada *driver* motor BLDC, digunakan ggl balik pada setiap belitan fasa untuk mendeteksi posisi rotor. Pada pin *Frequency Generator* (FG),*driver* menghasilkan sinyal kotak dengan *duty cycle* 50% . Frekuensi sinyal kotak merepresentasikan kecepatan motor BLDC. Sinyal kotak pada pin FG dimasukkan *pin digital input* Arduino. Untuk mengisolasi rangkaian Arduino dengan *plant* digunakan *optocoupler*.




**Gambar 4.8** Rangkaian Pembaca Kecepatan pada *Arduino*

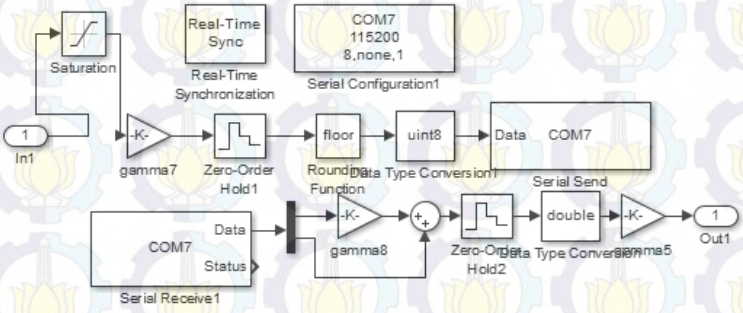
## 4.5 Implementasi Sistem

Pada Bab sebelumnya sudah dibahas mengenai simulasi dan analisa performa kontroler maka tahap selanjutnya adalah mengimplementasikan kontroler yang telah dirancang pada sistem secara langsung. Secara umum tahap yang dilakukan untuk melakukan implementasi mirip dengan tahap yang dilakukan untuk melakukan proses simulasi. Hanya saja pada tahap implementasi blok *plant* sistem motor BLDC yang digunakan pada diagram *simulink* tidak menggunakan fungsi alih hasil identifikasi dan pemodelan, melainkan menggunakan blok *plant* BLDC yang sesungguhnya yakni langsung dihubungkan ke motor BLDC menggunakan komunikasi *serial* beserta kondisi pembebanan yang

dikehendaki seperti penggunaan rem pada beberapa kondisi pembebanan. Hasil yang akan diperoleh kemungkinan berbeda karena adanya beberapa faktor yang tidak terhitung, tetapi diharapkan hasil yang akan didapat tidak jauh melenceng ada hasil simulasi sistem yang telah dilakukan.

### 4.5.1 Diagram Blok Implementasi Sistem

Seperti yang telah dibahas sebelumnya, perbedaan utama antara tahap simulasi dan implementasi ialah pada blok *plant* BLDC yang dipergunakan. Pada tahap simulasi, sinyal kontrol yang dihasilkan oleh kontroler dimasukkan ke blok *plant* BLDC yang berisi fungsi alih dari hasil identifikasi pada masing-masing kondisi pembebanan. Sedangkan pada tahap implementasi, sinyal kontrol yang dihasilkan oleh kontroler dimasukan ke blok *realisasi plant* BLDC yang berisi komunikasi *serial* dengan perantara Arduino Uno dan kemudian diolah oleh *driver* motor BLDC, yang selanjutnya menggerakan motor BLDC. Blok realisasi *plant* dapat dilihat pada Gambar 4.9




**Gambar 4.9** Diagram Simulink Komunikasi Data Implementasi

### 4.5.2 Pengujian Respon Implementasi Kontroler *Neural-Network*

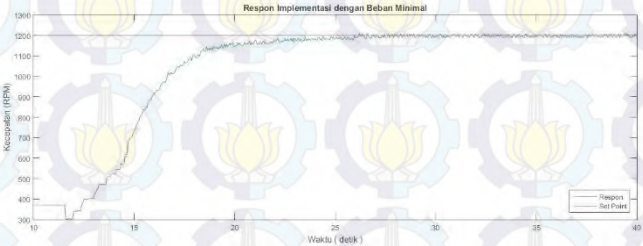
Sebelum melakukan tahap implementasi, terlebih dahulu dilakukan penentuan parameter yang akan digunakan pada kontroler *Neural-Network*. Penentuan parameter ini dilakukan sesuai dengan pengujian pada proses simulasi sebelumnya. Dari hasil *learning* tersebut digunakan parameter nilai bobot yang dipergunakan untuk parameter kontroler PID Diskrit.

Setelah parameter ditentukan, selanjutnya ialah menjalankan sistem dengan menggunakan kontroler *Neural-Network* yang sebelumnya telah melalui proses *learning* dan kemudian dianalisa hasil respon yang didapat

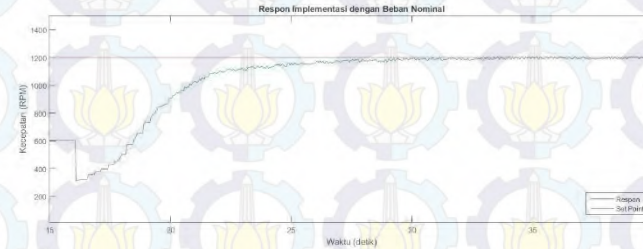
pada semua kondisi pembebanan. Seperti yang telah dijelaskan pada tahap perancangan bahwa *Neural Network* yang didesain menggunakan parameter yang menyerupai PID Diskrit. Karena adanya *delay* pada *input Neural Network*, mempengaruhi keseluruhan sistem. Sehingga dipergunakan struktur PID Diskrit sebagai kontroler pada Implementasi menggunakan nilai bobot yang diperoleh dari *tuning Neural Network*.

Dari hasil tuning diperoleh parameter $w_1$=0,13678236677, $w_2$= 0,567957061, $w_3$= 0,00235166, dan $w_4$=1, yang kemudian $w_1$ merupakan $K_p$, $w_2$ merupakan $K_i$, serta nilai $w_3$ merupakan $K_d$. Hasil dari *learning* sendiri menggunakan struktur PID diskrit yang dapat dilihat pada Sub-bab 3.5. Diagram Simulink serta *subsystem* dari implementasi dapat dilihat pada bagian Lampiran.

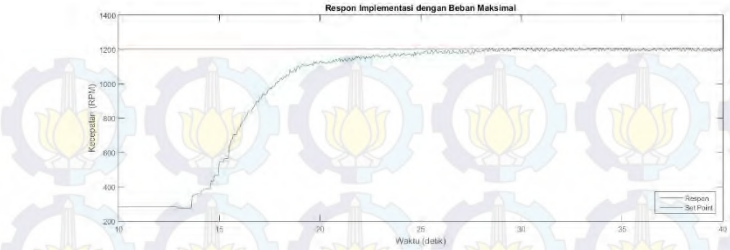
Dari hasil *tuning* tersebut dilakukan pengujian dengan *plant* yang telah dibuat dan diperoleh hasil yang terdapat pada Gambar 4.10 berupa hasil respon dengan beban minimal,Gambar 4.11 dengan beban nominal, Gambar 4.12 untuk kondisi beban maksimal, serta Gambar 4.13 untuk kondisi beban yang berubah ubah.



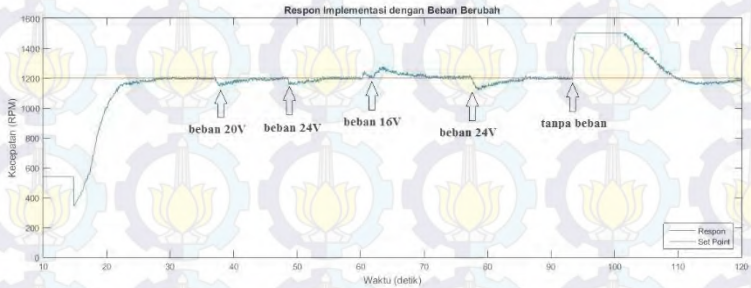

**Gambar 4.10** Respon Implementasi dengan Beban Minimal




**Gambar 4.11** Respon Implementasi dengan Beban Nominal




**Gambar 4.13** Respon Implementasi dengan Beban Maksimal




**Gambar 4.12** Respon Implementasi dengan Beban Berubah

### 4.5.3 Hasil Karakteristik Respon pada Semua Kondisi Pembebanan

Pada Sub Bab ini akan dipaparkan karakteristik dari respon hasil implementasi kontroler PSO-NN pada semua kondisi

#### *4.5.3.1 Kondisi Beban Minimal*

Pada kondisi tanpa beban, pengereman diatur sebesar 16 Volt tegangan masuk ke rem elektromagnetik. Pada tahap ini, diperhatikan bagaimana respon yang diperleh. Parameter kontrol yang dipergunakan sesuai dengan nilai bobot yang diperoleh dari hasil learning PSO-NN. Berikut hasil responnya dapat dilihat pada Tabel 4.3

**Tabel 4.3** Indeks Performansi untuk Kondisi Tanpa Beban

| Beban | 16V |
|---|---|
| *Overshoot* | 0,83% |
| *Settling Time* | 14,65 detik |
| *Rise Time* | 5,35 detik |
| *Error Steady State* | 0,64 % |

Dari Hasil Implementasi diatas, Pada *Neural Network* yang di didesain hanya menyesuaikan parameternya untuk mencapai kondisi error steady state 0,83 % . Pada hasil implementasi sendiri memiliki *Overshoot* yang paling tinggi di antara lainnya dikarenakan pada proses *learning* parameternya menggunakan beban maksimal sebagai fungsi alihnya.

#### *4.5.3.2 Kondisi beban nominal*

Selanjutnya tahap implementasi dilanjutkan untuk kondisi pembebanan nominal. Sebelum dilakukan implementasi, terlebih dahulu dilakukan proses *learning* dengan menggunakan fungsi alih hasil pemodelan pada kondisi beban nominal orde 1. Pemilihan orde ini berdasarkan pada nilai RMSE yang dihasilkan tiap orde. Didapatkanlah fungsi alih orde 1 pada tiap kondisi pembebanan sebagai fungsi alih dengan nilai RMSE yang paling kecil atau dengan kata lain paling mendekati hasil yang sebenarnya jika dibandingkan dengan fungsi alih orde lain. Hasil dari implementasi dengan beban nominal dapat dilhat pada Tabel 4.4.

**Tabel 4.4** Indeks Performansi untuk Kondisi Beban Nominal

| Beban | 20V |
|---|---|
| *Overshoot* | 0,46% |
| *Settling Time* | 13,2 detik |
| *Rise Time* | 4,62 detik |
| *Error Steady State* | 0,21 % |

#### *4.5.3.3 Kondisi Beban Maksimal*

Pada tahap implementasi kondisi ini, beban pengereman diatur pada kondisi beban maksimal. Pada kondisi ini beban yang dipergunakan menggunakan tegangan 24V. Selanjutnya dilakukan proses *learning* dengan menggunakan fungsi alih hasil pemodelan pada kondisi beban maksimal orde 1. Setelah proses *learning* pada kondisi ini selesai, dilanjutkan dengan implementasi dengan memberikan kecepatan referensi sama seperti pada kondisi pembebanan sebelumnya. Kemudian didapatkan hasil respon dan dilakukan analisa terhadap karakteristik yang ada pada kondisi beban maksimal. Indeks performansi untuk beban maksimal dapat dilihat pada Tabel 4.5.

**Tabel 4.5** Indeks Performansi untuk Kondisi Beban Maksimal

| Beban | 24V |
|---|---|
| *Overshoot* | 0,42% |
| *Settling Time* | 12,65 detik |
| *Rise Time* | 3,75 detik |
| *Error Steady State* | 0,26 % |

### 4.5.4 Pengujian Respon Implementasi Kontroler *Neural-Network* pada Kondisi Beban Berubah

Pada tahap ini dilakukan proses implementasi sama seperti tahap implementasi yang telah dilakukan sebelumnya. Namun, pada tahap ini kondisi pembebanan dirubah secara langsung ketika motor BLDC sedang beroperasi. Perubahan yang dilakukan bertahap dari kondisi beban maksimal hingga kondisi *steady state* dan setelah itu dilakukan perubahan beban secara acak dengan kondisi beban minimal, nominal, hingga tanpa beban. Setelah itu dilihat bagaimana pengaruh hasil kontroler *Neural Network* terhadap respon kecepatan yang dihasilkan akibat perubahan beban yang dilakukan secara langsung. Dapat dilihat pada Gambar 4.13 bahwa kontroler dapat mempertahankan diri pada kondisi *steady state*.

# BAB 5
# PENUTUP

Pada bab terakhir ini menjelaskan tentang penarikan kesimpulan pelaksanaan Tugas Akhir dan saran untuk penelitian selanjutnya tentang topik terkait serta lampiran berupa data pendukung Tugas Akhir.

## 5.1 Kesimpulan

Berdasarkan hasil yang didapat dari proses simulasi dan implementasi dapat disimpulkan bahwa:

a. Penggunaan kontroler *Neural Network* dapat membuat respon kecepatan Motor BLDC menyesuaikan dengan model referensi yang diberikan
b. Semakin banyak jumlah iterasi yang dilakukan dalam proses *learning* pada kontroler *Neural Network* akan menambah keakuratan hasil yang didapat
c. Dari Hasil *Learning* yang didapat diperoleh parameter nilai bobot $w_1$=0,1367823667, $w_2$= 0,567957061, $w_3$= 0,002351668, dan $w_4$=1

## 5.2 Saran

Apabila ada yang ingin menggunakan *plant* BLDC ini untuk dilakukan penelitian lebih lanjut atau menggunakan kontroler *Neural Network*, ada beberapa hal yang dapat dijadikan pertimbanagan, antara lain:

a. Faktor pendukung yang dapat mempengaruhi kecepatan harus diperhatikan seperti, kopel pada motor, dan permukaan alas. Faktor-faktor tersebut secara langsung maupun tidak langsung dapat mempengaruhi kecepatan putaran motor BLDC
b. Pada pengambilan data untuk *learning* pada *Neural Network* dengan algoritma PSO sebaiknya menggunakan data dengan sinyal PRBS sehingga data data yang diperoleh dapat lebih merata untuk berbagai kondisi kontrol.

# LAMPIRAN

- Script Algoritma *Learning* PSO-NN

```
I1=masukan1;
I2=masukan2;
I3=masukan3;
I4=datadelay;
T=data;

Dim=4;                  %dimensi dari partikel PSO
TrainingNO=2500;        %jumlah training samples
hiddennodes=0;

%Initial Parameter untuk PSO
noP=30;                 %jumlah partikel
Max_iteration=10000;%jumlah iterasi maksimal
iw=2;                   %bobot inersia
iwMax=2;               %bobot inersia maksimal
iwMin=1;               %bobot inersia minimal
c1=2;
c2=2;
dt=0.8;

vel=zeros(noP,Dim); %vektor kecepatan
pos=zeros(noP,Dim); %vektor posisi

%////////Komponen Kognitif/////////
pBestScore=zeros(noP);
pBest=zeros(noP,Dim);
%//////////////////////////////////////

%////////Social component//////////
gBestScore=inf;
gBest=zeros(1,Dim);
%//////////////////////////////////////
```

```
%Initialization
for i=1:size(pos,1) % Untuk Setiap Partiel
    for j=1:size(pos,2) % Untuk Setiap Dimensi
        pos(i,1)=0.1 +(0.2-0.1).*rand();
        pos(i,2)=0.5+(0.6-0.5).*rand();
        pos(i,3)=0.005+(0.008-0.005).*rand();
        pos(i,4)=1;
        vel(i,j)=0.3*rand();
    end
end

 %initialize gBestScore for min
 gBestScore=inf;


for Iteration=1:Max_iteration
    %Calculate MSE
    for i=1:size(pos,1)
        for ww=1:4;
            Weights(ww)=pos(i,ww);
        end
        fitness(1)=0;
        for pp=2:TrainingNO

actualvalue(pp)=((I1(pp)*Weights(1))+(I2(pp)*Wei
ghts(2))+(I3(pp)*Weights(3))+(I4(pp)*Weights(4))
);
            fitness(pp)=fitness(pp-1)+((T(pp)-
actualvalue(pp))^2);
            end
        end

fitness(Iteration)=fitness(TrainingNO)/TrainingN
O;

        pBestScore(1)=fitness(Iteration);

        if(pBestScore(i)>fitness(Iteration))
            pBestScore(i)=fitness(Iteration);
```

```
            pBest(i,:)=pos(i,:);
        end

        if(gBestScore>fitness(Iteration))
            gBestScore=fitness(Iteration);
            gBest=pos(i,:);
        end

        if(gBestScore==1)
            break;
        end
end

    %Update the w of PSO
    iw=iwMin-Iteration*(iwMax-
iwMin)/Max_iteration;

    %Update the velocity and position of
particles
    for i=1:size(pos,1)
        for j=1:size(pos,2)

vel(i,j)=iw*vel(i,j)+c1*rand()*(pBest(i,j)-
pos(i,j))+c2*rand()*(gBest(j)-pos(i,j));
            pos(i,j)=pos(i,j)+dt*vel(i,j);
        end
    end
```

# DAFTAR PUSTAKA


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

[2] R. Shemanske, "Electronic Motor Braking," *IEEE Trans. On Industry, vol. IA, no. 19, pp. 824-831, 1983.*

[3] .....,"*Arduino*",<URL:http://arduino.cc/>, Mei,2015

[4] Elrosa, Ilmiyah., " Traction Control pada Parallel Hybrid Electric Vehicle dengan Metode Generalized Predictive Control ", *Tugas Akhir*, Jurusan Teknik Elektro ITS Surabaya, 2014

[5] Gerard Dreyfus, *Neural Networks, Methodology and Applications*. Springer, 2005.

[6] Parsopoulos, V. *Particle Swarm Optimization and Intelligence* . New York: IGI Global,2010.

[7] Sayed Ali Mirjalli, S. Z.. "Training Feedforward Neural Network using Hybrid Particle Swarm Optimization" dalam *Sciencedirect Applied Mathematics and Computation*, 2012.

[8] K. Naga Sujatha. "Artificial Intelligence Based Speed Control of Brushless DC Motor," *IEEE Power and Energy Society General Meeting*, 2010.

[9] Ogata, K., *Modern Control Engineering (4th ed.)*, New Jersey: Prentice Hall, 2002.

[10] D.R. Hush, N.G. Horne,"Progress in Supervised Neural Network," *IEEE Signal process Mag, 10*, 1993.

# RIWAYAT HIDUP

Irwan Eko Prabowo lahir di Semarang tanggal 27 Desember 1992, merupakan anak pertama dari Agus Purwanto dan Irawati. Setelah lulus dari SMA Negeri 1 Cepu tahun 2011, melanjutkan studi di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS) Surabaya pada tahun yang sama.