BAB II

TINJAUAN PUSTAKA

Pada bab ini dipaparkan beberapa dasar teori yang digunakan dalam penelitian. Dalam penelitian ini digunakan *fuzzy control system* pada Arduino untuk pengendalian kecepatan motor DC berbeban.

2.1. Sistem Pengendali

Sistem pengendali (atau sistem kendali, *control system*) adalah sebuah sistem yang terdiri dari sekumpulan perangkat yang mengelola, memerintah, mengarahkan, atau mengatur perilaku perangkat atau sistem lain untuk mencapai hasil yang diinginkan. Selain mendapatkan masukan berupa sekumpulan perintah dari operator, sistem pengendali juga menganalisa *feedback* (umpan balik) aksi objek untuk memperbaiki (memberikan koreksi) atas kesalahan yang timbul.

Sebuah sistem kendali bisa dikatakan memiliki empat fungsi, yaitu mengukur, membandingkan, menghitung, dan memperbaiki. Empat fungsi tersebut dilakukan oleh lima elemen, yaitu detektor, transduser, pemancar, pengendali, dan elemen kendali akhir. Fungsi pengukuran dilakukan oleh detektor, transduser, dan pemancar, yang biasanya dalam aplikasi praktis ketiga elemen tersebut berada dalam satu unit. Fungsi pembandingan dan penghitungan dilakukan oleh pengendali. Sedangkan fungsi perbaikan dilakukan terhadap elemen kendali akhir.

Keberhasilan sistem pengendali dapat diukur berdasarkan beberapa kriteria sebagai berikut:

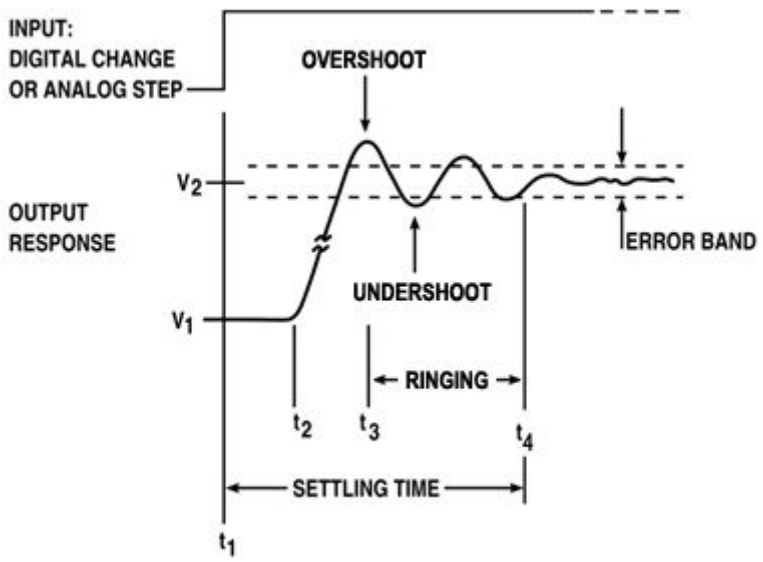
2.1.1. Overshoot dan Undershoot

Overshoot adalah satu kondisi di mana nilai sinyal atau fungsi melampaui *steady-state* (acuannya, atau *set-point* – nilai yang diinginkan). Lawannya, *undershoot*, terjadi ketika nilai sinyal atau fungsi lebih kecil dari acuannya. Baik *overshoot* maupun *undershoot* kebanyakan terjadi diikuti oleh ayunan (*ringing*, *oscillation*) yang diakibatkan oleh respon berlebihan berlawanan arah yang dilakukan oleh sebuah sistem kendali.

Dalam sebuah sistem kendali, *maximum overshoot* (atau*maximum undershoot*) didefinisikan sebagai nilai puncak maksimal dari kurva respon yang terukur terhadap respon yang diinginkan sistem. Baik *overshoot* maupun *undershoot* merupakan fenomena yang tidak diinginkan pada sebuah sistem kendali. Sistem kendali harus memenuhi syarat pengendalian di mana setiap *overshoot* (atau *undershoot*) yang terjadi harus memiliki nilai di bawah *maximum overshoot* (atau *maximum undershoot*). (Vukic, *et. al.*, 2003)

2.1.2. Ringing

Ringing adalah ayunan sinyal yang khususnya terjadi setelah ada perubahan masukan (*set-point*) yang mendadak. Pada beberapa kasus, *ringing* adalah fenomena yang tidak diinginkan, meskipun tidak selalu. Fenomena *ringing* sangat erat kaitannya dengan *overshoot*, yang dalam hal ini ringing terjadi mengikuti *overshoot*. (Vukic, *et. al.*, 2003)



Gambar 2.1 Sebuah ilustrasi pada fenomena *overshoot* yang diikuti oleh *ringing* dan *settling time*.
Sumber: Vukic, *et. al.*, 2003.

Urutan terjadinya fenomena tersebut adalah: *overshoot*, *ringing*, dan *settling time*, seperti pada Gambar 2.1.

2.1.3. Steady State

Sebuah sistem atau proses dikatakan pada keadaan *steady state* (stabil) jika variabel yang menentukan perilaku sistem atau proses dalam kondisi tidak berubah terhadap

waktu.Pada model waktu yang kontinyu, hal ini berarti variabelp dari sistem akan bernilai nol pada turunan parsial terhadap waktu, seperti pada Persamaan (2-1).

$$\frac{\partial p}{\partial t} = 0 \tag{2-1}$$

Dengan:

p = nilai proses

t = waktu (s)

Sedangkan pada model waktu diskrit, hal ini berarti selisih pertama pada setiap variabelpadalah nol dan seterusnya, seperti pada Persamaan (2-2).

$$p_t - p_{t-1} = 0 \tag{2-2}$$

Dengan:

p = nilai proses

t = waktu (s)

2.1.4. Steady StateError (Error Band)

Steady-state error didefinisikan sebagai selisih antara masukan (*set-point*) dan keluaran dari suatu sistem pada sebuah batas tertentu ketika besaran waktu menuju ketakhinggaan (yaitu ketika respon telah mencapai kondisi *steady-state*). *Steady state error* akan bergantung pada jenis masukan dan juga pada jenis sistem.*Steady-state error* disebut juga dengan *error band*.

2.1.5. Rising Time

Rising time (atau *rise time*, waktu naik) adalah waktu yang diperlukan oleh sebuah sinyal untuk mengubah nilainya dari suatu nilai rendah ke nilai tinggi yang diinginkan (*set-point*). Nilai ini dinyatakan sebagai rasio (perbandingan) atau persentase. Pada sebuah keluaran dari sistem, *rising time* yang dimilikinya secara umum bergantung pada *rising time* dari sinyal masukan dan pada karakteristik sistem itu sendiri. Pada dasarnya, sebuah sistem kendali harus bisa meminimalisasi efek *overshoot* sementara *rising time* yang dimiliki oleh objek yang dikendalikan memiliki nilai yang relatif besar.

Sebagai lawannya, *falling time* adalah sama dengan *rising time*, hanya saja berlaku dari suatu nilai tinggi ke nilai rendah yang diinginkan.

2.1.6. Settling Time

Settling time dari sebuah perangkat keluaran adalah waktu yang dibutuhkan mulai dari masukan acuan hingga sinyal yang ada memasuki tahap *steady-state* (atau sudah berada dalam kondisi *error steady state* yang ditoleransi). *Settling time* bergantung pada kemampuan respon sistem dan *time constant*.*Settling time* pada sebuah sistem yang dikedalikan harus dijaga sekecil mungkin oleh sistem kendali.

2.2. Fuzzy Logic

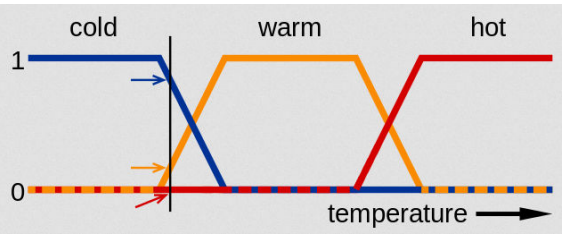
Fuzzy logic adalah sebuah bentuk dari logika bernilai banyak yang mana nilai kebenaran dari sebuah variabel jatuh antara 0 dan 1 sebagai bilangan nyata. Sebagai perbandingan, pada *boolean logic*, nilai kebenaran dari sebuah variabel jatuh hanya pada nilai 0 atau 1, sebagai bilangan bulat. Jadi sementara *boolean logic* hanya memiliki dua kemungkinan nilai, maka *fuzzy logic* memiliki semua nilai bilangan nyata yang mungkin antara 0 dan 1.

Fuzzy logic telah banyak digunakan untuk menangani konsep tentang kebenaran sebagian (*partial truth*). Selain dari itu, ketika variabel linguistik digunakan, derajat kebenaran yang muncul sebagai nilai variabel *fuzzy* bisa diatur oleh fungsi keanggotaan tertentu. (Bojadziev, *et. al.*, 2005)

Proses pada *fuzzy logic* terdiri dari bagian-bagian berikut:

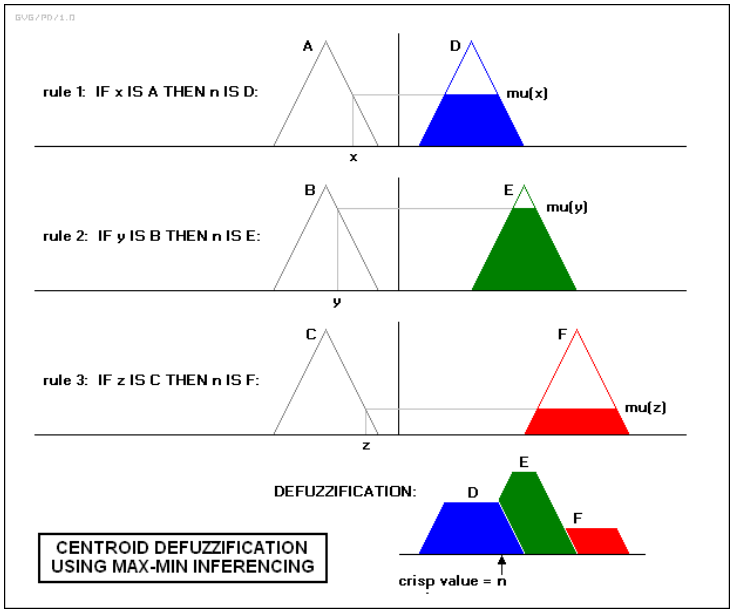
1. Fuzzifikasi semua nilai masukan pada fungsi keanggotaan.
2. Menjalankan semua aturan yang didefinisikan pada *rulebase* untuk menghitung fungsi keluaran *fuzzy*.
3. Defuzzifikasi fungsi keluaran *fuzzy* untuk mendapatkan nilai tegas.

Derajat keanggotaan harus didefinisikan terlebih dahulu pada sebuah fungsi keanggotaan yang dipetakan terhadap nilai masukan.



Gambar 2.2 Contoh fungsi keanggotaan pada temperatur.
Sumber: Bojadziev, *et. al.,* 2005.

Pada Gambar 2.2 ditunjukkan sebuah contoh fungsi keanggotaan pada temperatur, yang menjelaskan bahwa fungsi cold memiliki derajat 1 sampai pada temperatur tertentu, kemudian turun hingga 0 secara perlahan (tidak langsung) pada jangkauan tertentu sementara fungsi warm memiliki derajat dari 0 yang menaik hingga 1 pada jangkauan tersebut. Hal ini menunjukkan sifat derajat keanggotaan dari sebuah fungsi pada nilai masukan tertentu bisa berbagi dengan fungsi yang lainnya.



Gambar 2.3 Contoh max-min inferencingdancentroid defuzzification.
Sumber: Bojadziev, *et. al.*, 2005.

Gambar 2.3 menunjukkan contoh *max-min inferencing* yang menerjemahkan nilai variabel x, y, dan z menuju ke fungsi mu(x), mu(y), dan mu(z) sebagai derajat keanggotaan, kemudian melakukan *centroid defuzzification* untuk mendapatkan nilai tegas (*crisp value*).

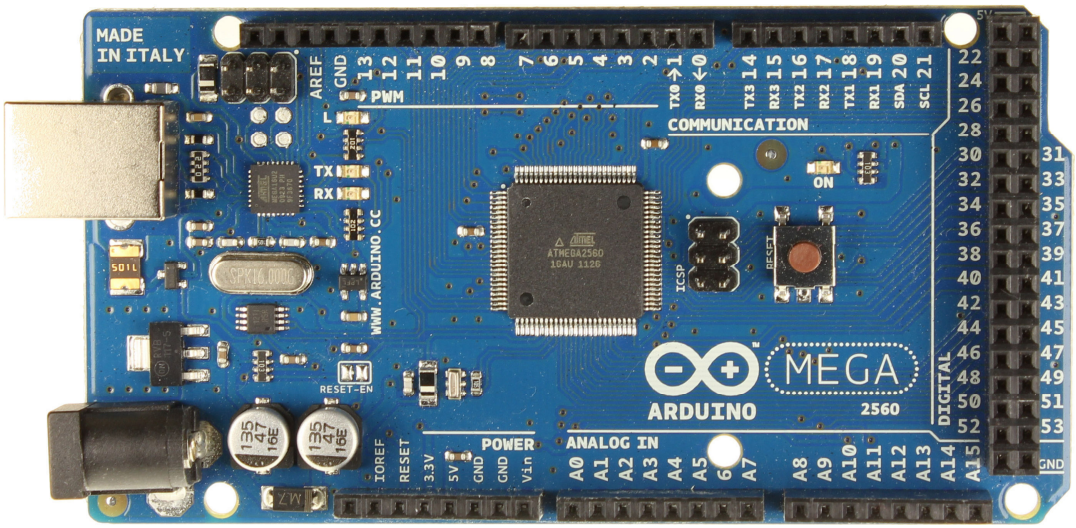
Ketika variabel dalam matematika biasanya mengambil nilai-nilai numerik, dalam aplikasi *fuzzy logic* sering digunakan nilai-nilai non-numerik untuk memfasilitasi ekspresi aturan dan fakta. Sebuah variabel linguistik seperti usia bisa jadi memiliki nilai seperti muda atau tua secara bersamaan. Nilai variabel linguistik dapat dimodifikasi melalui aturan linguistik diterapkan. Aturan linguistik dapat dikaitkan dengan fungsi-fungsi tertentu.

Operasi *fuzzification* dapat memetakan nilai masukan matematika dalam fungsi keanggotaan *fuzzy*. Dan sebaliknya proses *defuzzifying* dapat digunakan untuk memetakan fungsi keanggotaan keluaran yang bersifat kontinyu menjadi bersifat tegas yang dapat kemudian digunakan untuk tujuan keputusan atau pengendalian.

Sistem kendali yang berbasis pada *fuzzy logic* disebut dengan *fuzzy control system* (FCS), di mana kaidah-kaidah proses yang digunakan dalam melakukan pengendalian berdasarkan pada *fuzzy logic*. (Bojadziev, *et. al.*, 2005)

2.2.1. Arduino

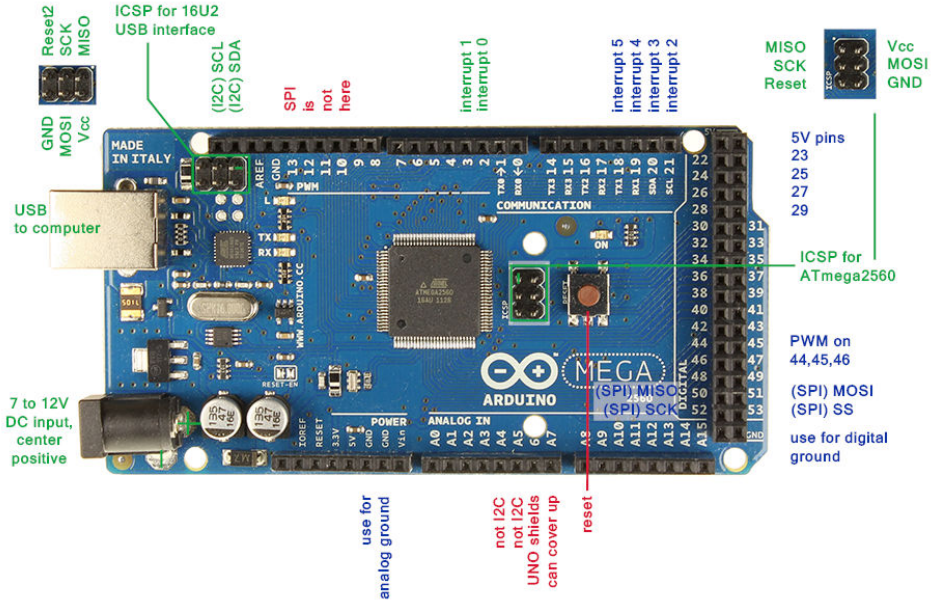
Arduino adalah perusahaan, proyek, dan komunitas pengguna perangkat keras komputer dan perangkat lunak yang mendesain dan memproduksi kit mikrokontroler untuk membangun perangkat digital dan objek interaktif yang memiliki kemampuan sensor dan kemampuan kendali fisik. Produk proyek Arduino didistribusikan sebagai perangkat keras dan perangkat lunak*open-source*, dengan lisensi menggunakan GNU *Lesser General Public License* (LGPL) atau GNU *General Public License* (GPL).Hal ini memungkinkan pembuatan papan rangkaian Arduino secara bebas dan pendistribusianperangkat lunak oleh siapa saja. Papan rangkaian Arduino tersedia secara komersial dalam bentuk terakit, atau sebagai kit yang bisa dibuat sendiri.



Gambar 2.4 Contoh sebuah perangkat keras Arduino Mega 2560.
Sumber: Banzi, 2011.

Papan rangkaian Arduino sendiri merupakan perangkat keras pengendali yang menggunakan *microcontroller*. Pada Arduino, rancangan papan rangkaian yang ada menggunakan berbagai *microprocessor* dan pengendali. Pada papan rangkaian ini disediakan serangkaian pin *input-output* baik yang digital maupun analog yang bisa dihubungkan menuju beragam papan rangkaian ekspansi, rangkaian lainnya, komputer, atau langsung pada perangkat sensor dan motor. Papan rangkaian ini menyajikan beberapa

antarmuka komunikasi serial, termasuk USB *Port* untuk berhubungan dengan komputer dalam pemrograman.



Gambar 2.5 Keterangan pin Arduino Mega 2560.
Sumber: Banzi, 2011.

Microcontroller pada Arduino dapat diprogram menggunakan C atau C++, dengan kemudahan yang disediakan oleh komunitas berupa IDE (*integrated development environment*) untuk melakukan pemrograman. Sekali program dituliskan ke papan rangkaian menggunakan USB *communication*, maka setiap kali papan rangkaian diberi daya program akan berjalan sampai daya diputus.(Banzi, 2011)

Tabel 2.1
Spesifikasi singkat Arduino Mega 2560.

Microcontroller	Atmega2560
Operating Voltage	5 V
Input Voltage (recommended)	7 V – 12 V
Input Voltage (limit)	6 V – 20 V
Digital I/O Pins	54 (14 with PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3 V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

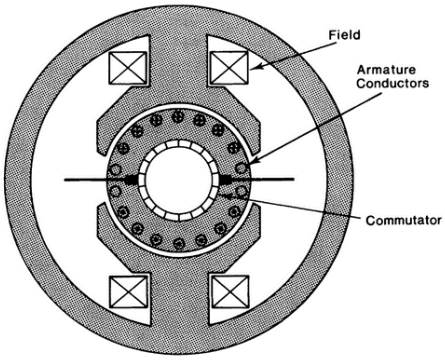
Sumber: http://www.robotshop.com/, 2016.

Arduino ada berbagai macam, bervariasi menurut tegangan pin *input-output*, *bit depth*, jumlah pin, dan *clock speed*. Pada penelitian ini digunakan Arduino Mega 2560, seperti

ditunjukkan pada Gambar 2.4 dan Gambar 2.5. Spesifikasi singkat dari Arduino Mega 2560 ditunjukkan pada Tabel 2.1.

2.2.2. Motor DC

Motor DC adalah peranti elektronik yang fungsinya mengubah arus listrik DC (searah) menjadi perwujudan putaran mekanik. Pada dasarnya, motor DC umumnya berbekal medan magnet untuk penggerak. Hampir semua jenis motor DC memiliki mekanisme di dalam, yang berupa elektromekanikal ataupun elektronik, yang secara ajeg mengubah arah arus pada interior motor DC. Kebanyakan jenis motor DC menghasilkan gerakan putar, jenis yang lain merupakan motor DC linier yang menghasilkan gaya dan gerakan dalam garis lurus (biasanya disebut dengan aktuator).



Gambar 2.6 Skematik 3D dari sebuah motor DC.
Sumber: Hughes, 1990.

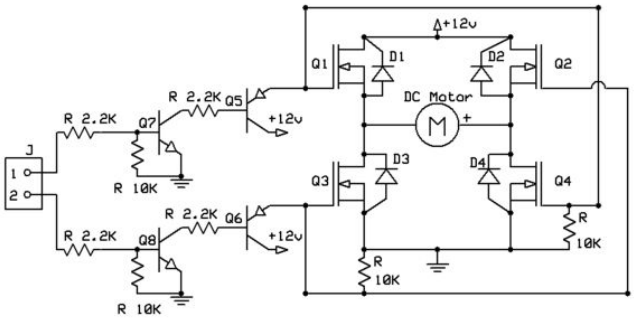
Putaran dari motor DC dapat dikontrol pada jangkauan yang cukup lebar, menggunakan variasi besarnya tegangan listrik atau variasi kekuatan arus listrik pada lilitan. Umumnya pengontrolan motor DC menggunakan variasi masukan pada tegangan listrik. (Hughes, 1990)

Gambar 2.7 Motor DC RS 555-555.
Sumber: Hughes, 1990.

Dalam penelitian ini digunakan RS 555-555 sebagai motor DC yang dikendalikan, seperti ditunjukkan pada Gambar 2.7, yang memiliki tegangan nominal 20 V, arus maksimal 220 mA, dan *output power* 4.4 W. Dalam kondisi tak berbeban, RS 555-555 mampu berputar hingga 3500 RPM.

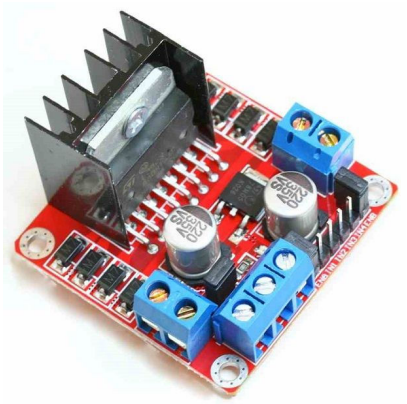
2.2.3. Driver Motor DC

Driver motor DC merupakan sebuah alat yang mengendalikan torsi motor DC. Sebuah *driver* motor DC merupakan penguat arus kecil (yang menjadi arus kendali) menjadi arus yang lebih besar untuk mencatu daya pada motor DC.



Gambar 2.8 Contoh sebuah skema rangkaian dari *driver* motor DC.
Sumber: Keljik, 2013.

Driver motor DC berupa rangkaian DIY (dikerjakan sendiri), dan disesuaikan dengan *control input*, tegangan, dan arus yang diperlukan oleh motor DC yang dikendalikan. *Driver* motor DC melakukan pengendalian dengan cara memvariasikan tegangan ke motor DC sesuai dengan kebutuhan torsi yang diinginkan.



*Gambar 2.9*L298N DC Motor Driver.
Sumber: https://www.instructables.com/, 2017.

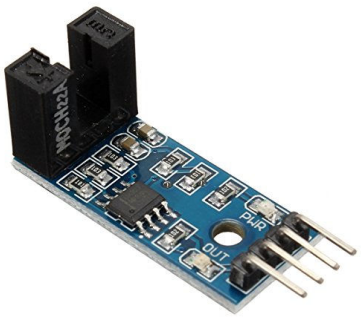
Pada penelitian ini, digunakan L298N sebagai *driver*, seperti yang ditunjukkan pada Gambar 2.9.

2.2.4. Speed Sensor

Speed sensor (sensor kecepatan) adalah sebuah alat yang mendeteksi kecepatan objek. Kecepatan yang dideteksi bisa berupa kecepatan putar, kecepatan linier, ataupun yang lainnya.

*Gambar 2.10*Optical chopper.
Sumber: https://www.thorlabs.com/,2017.

Dalam penelitian ini, digunakan detektor kecepatan putar untuk mendeteksi kecepatan putar motor DC yang dikendalikan.*Speed sensor* yang digunakan adalah*optical chopper* yang diapit oleh *opto switch*, yaitu sensor yang mengukur sinyal inframerah yang dipancarkan dari ujung satunya, tergantung pada sinyal yang diterima apakah terhalang atau tidak oleh *optical chopper*. Satuan untuk kecepatan putar yang digunakan adalah RPM (*revolutions per minute*), yaitu jumlah putaran yang terjadi setiap menitnya.

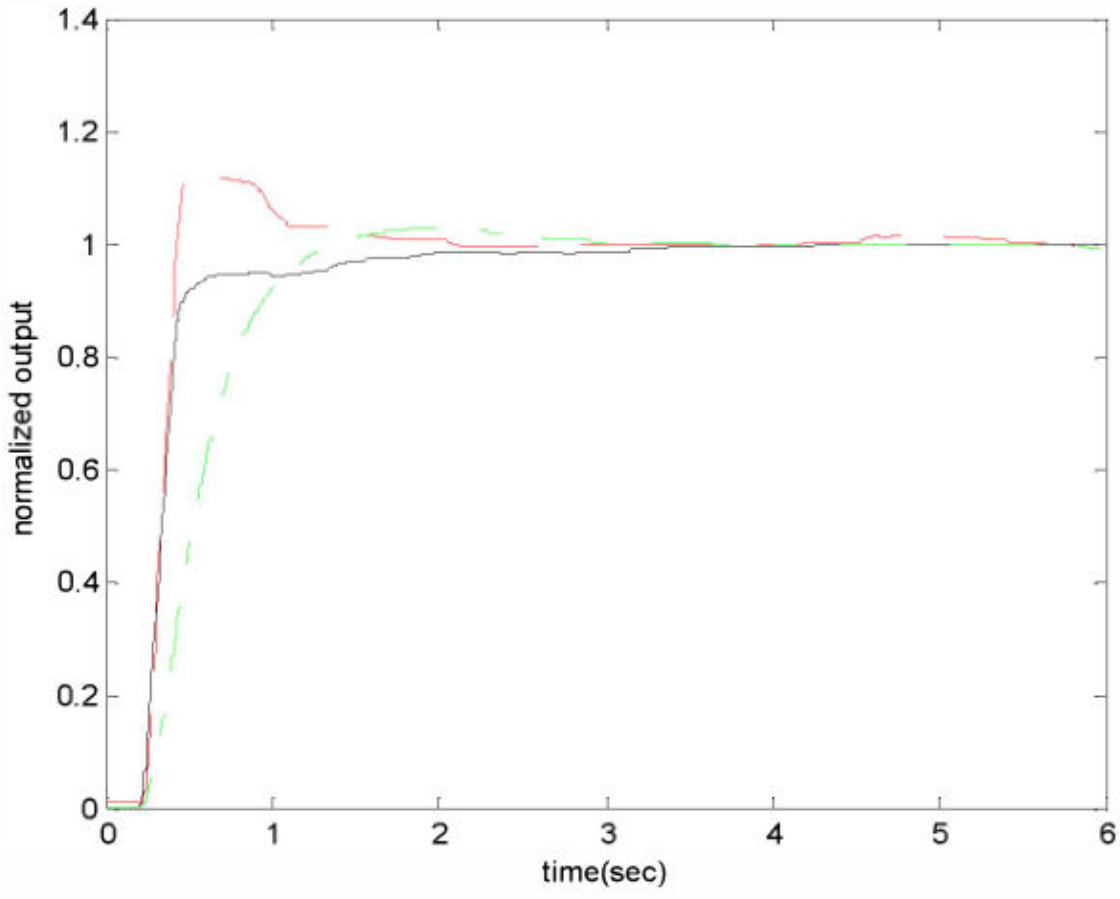


Gambar 2.11 Opto Switch LM393.
Sumber: https://www.banggood.com/, 2016.

Optical chopper yang digunakan adalah yang memiliki 20 lubang, sedangkan *opto switch* yang digunakan adalah LM393. Dengan 20 lubang yang ada, maka kecepatan transisi antara lubang dengan non-lubang harus dibagi dengan 20 untuk mendapatkan kecepatan putar dalam CPS.

2.3. Penelitian Sebelumnya

Beberapa penelitian terdahulu menunjukkan hasil kualitatif, tetapi belum menunjukkan kuantitatif. Seperti pada penelitian oleh Azadi (2012), yang menyatakan Pengendali Azadi mampu menstabilkan putaran motor DC dengan meminimalkan *overshoot* dan *undershoot*, tetapi belum menunjukkan besaran *overshoot* yang dihasilkan oleh sistem (Azadi, *et. al*., 2012).



Gambar 2.12 Hasil simulasi *Azadi Controller*.
Sumber: Azadi, *et. al.*,2012.

Dari hasil simulasi pada Gambar 2.12, ditunjukkan perilaku sistem *Azadi Controller* tetapi tidak menunjukkan besaran nominal *rising time*, *overshoot*, dan *oscillation*.

Penelitian tentang *neuro-fuzzy controller* menggunakan *particle swarm optimization* juga tidak memberikan hasil nominal tetang besaran *overshoot* dan *undershoot* serta ayunan, dan hanya memberikan hasil kualitatif serta digunakan sebagai sistem yang fleksibel (Farid, *et. al*., 2014).