PROYEK AKHIR – VE 180626

PENGATURAN KECEPATAN MOTOR *BRUSHLESS* DC MENGGUNAKAN KONTROLER BERBASIS PID

Baharuddin Baharsyah
NRP 10311500010040

Dosen Pembimbing
Ir. Josaphat Pramudijanto, M.Eng.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – VE 180626

BRUSHLESS DC MOTOR SPEED SETTINGS USING PID BASED CONTROLLER

Baharuddin Baharsyah
NRP 10311500010040

Supervisor
Ir. Josaphat Pramudijanto, M.Eng.

DEPARTMENT ELECTRICAL AND AUTOMATION ENGINEERING
Vocational Faculty
Sepuluh Nopember Institute of Technology
Surabaya 2019

-----Halaman ini sengaja dikosongkan-----

PERNYATAAN KEASLIAN PROYEK AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Proyek Akhir saya dengan judul "**Pengaturan Kecepatan Motor *Brushless* DC Menggunakan Kontroler Berbasis PID**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 18 Januari 2019

Baharuddin Baharsyah
NRP 10311500010040

-----Halaman ini sengaja dikosongkan-----

PENGATURAN KECEPATAN MOTOR *BRUSHLESS* DC MENGGUNAKAN KONTROLER BERBASIS PID

PROYEK AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing

Ir. Josaphat Pramudijanto M.Eng
NIP. 19621005 199003 1 003

SURABAYA
JANUARI, 2019

D-ii

-----Halaman ini sengaja dikosongkan-----

PENGATURAN KECEPATAN MOTOR *BRUSHLESS* DC MENGGUNAKAN KONTROLER BERBASIS PID

Nama : Baharuddin Baharsyah
NRP : 10311500010040
Pembimbing : Ir. Josaphat Pramudijanto, M.Eng.
NIP : 19621005 199003 1 003

ABSTRAK

Dalam proyek akhir ini, untuk mengetahui kecepatan dari motor *brushless* DC diperlukan sebuah modul yang bisa memberikan data yang tepat. Selain itu, untuk mengontrol kecepatan dari motor *brushless* DC diperlukan pula sebuah modul kontrol. Namun, modul khusus untuk kontrol kecepatan motor *brushless* DC tidak mudah untuk dibuat.

Oleh karena itu untuk mempermudah proyek akhir ini, diperlukan suatu modul yang praktis dan efisien. Sehingga muncul ide untuk membuat suatu modul yang digunakan untuk mengontrol kecepatan motor *brushless* DC dengan mikrokontroler. Modul ini terdiri dari 4 komponen utama, yaitu arduino sebagai mikrokontroler, motor *brushless* DC dan sensor kecepatan. Modul ini kemudian dioperasikan melalui *software* Lab*View* dengan metode kontroler PID.

Pada Proyek Akhir ini, dihasilkan bahwa metode kontrol dari motor *brushless* DC dengan kontroler PID dapat dikendalikan menggunakan Lab-*View* dengan metode coba-coba sehingga menghasilkan parameter kontroler PID yaitu K_p=1 T_i=0 T_d=0.

Kata Kunci : **Motor *Brushless* DC, LabView, PID.**

-----Halaman ini sengaja dikosongkan-----

BRUSHLESS DC MOTOR SPEED SETTINGS USING PID BASED CONTROLLER

Name **: Baharuddin Baharsyah**
NOR : 10311500010040
Supervisor **: Ir. Josaphat Pramudijanto, M.Eng.**
NIP : 19621005 199003 1 003

ABSTRACT

In this final project, we need a module that can provide the right data to find out the speed of a brushless motor DC. In addition, to control the speed of a DC brushless motor, a control module is also needed. However, special modules for DC brushless motor speed control are not easy to make.

Therefore to facilitate this final project, a module that is practical and efficient is needed. So the idea arises to make a module that is used to control the brush-less DC motor speed with a microcontroller. This module consists of 4 main components, namely Arduino as a microcontroller, DC brushless motor and speed sensor. This module is then operated through LabView software using the PID controller method.

In this Final Project, it is produced that the control method of the Brushless DC motor with the PID controller can be controlled using LabView with trial methods so as to produce the PID controller parameter, namely Kp = 1 Ti = 0 Td = 0.

Keywords :Brushless DC Motor, LabView, PID.

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena atas limpahan rahmat dan karunia-Nya, penulis dapat menyelesaikan Proyek Akhir ini dengan judul :

"PENGATURAN KECEPATAN MOTOR *BRUSHLESS* DC MENGGUNAKAN KONTROLER BERBASIS PID"

Proyek Akhir ini merupakan sebagian syarat untuk menyelesaikan mata kuliah dan memperoleh nilai pada Proyek Akhir.

Dengan selesainya Proyek Akhir ini penulis menyampaikan terima kasih sebesar-besarnya kepada:

1. Orang Tua atas limpahan doa, kasih sayang, dukungan dan dorongan baik berupa moral atau materil bagi penulis.
2. Bapak Ir. Joko Susila, MT. selaku Ketua Departemen Teknik Elektro Otomasi, FV-ITS Surabaya.
3. Bapak Ir. Josaphat Pramudijanto, M.Eng. selaku Dosen Pembimbing.
4. Seluruh staf pengajar dan administrasi Departemen Teknik Elektro Otomasi FV-ITS.
5. Seluruh Mahasiswa Teknik Elektro Otomasi khususnya angkatan 2015.
6. Semua pihak yang telah banyak membantu untuk menyelesaikan Proyek Akhir ini yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari masih banyak kekurangan dalam Proyek Akhir ini. Kritik dan saran untuk perbaikan tugas ini sangat diperlukan. Akhir kata semoga tugas ini dapat bermanfaat bagi kita semua.

Surabaya, 18 Januari 2019

Penulis

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

DAFTAR GAMBAR

-----Halaman ini sengaja dikosongkan-----

DAFTAR TABEL

-----Halaman ini sengaja dikosongkan-----

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring dengan perkembangan zaman, Motor *Brushless Direct Current* (BLDC) adalah alternatif pengganti motor DC. Sesuai namanya motor BLDC tidak memiliki sikat pada komutatornya. Sebagai gantinya digunakan komutator elektrik. Motor ini biasanya banyak digunakan dalam *automotive*, *aerospace*, *medical*, industri, dan instrumentasi.

Pada dunia industri biasanya menggunakan sistem kendali pada motor BLDC menggunakan pengendali PID. Keberhasilan pengendali PID tergantung ketepatan dalam menentukan konstanta (penguatan) PID. Dalam suatu sistem kendali kita mengenal adanya beberapa macam aksi kendali, yaitu aksi kendali proporsional, aksi kendali integral, dan aksi kendali derivatif. Masing-masing aksi kendali ini mempunyai keunggulan tertentu, dimana aksi kendali proporsional mempunyai keunggulan *risetime* yang cepat, aksi kendali integral mempunyai keunggulan untuk memperkecil *error* dan aksi kendali derivatif keunggulannya untuk memperkecil *error* atau meredam *overshoot/undershoot*.

Dengan menggunakan arduino tidak perlu perangkat *chip* programmer karena di dalamnya sudah ada *bootloader* yang akan menangani upload program dari komputer. Hal ini bertujuan mempermudah pengguna mengetahui hasil pengamatan yang telah mereka lakukan.

1.2. Tujuan Dan Manfaat

Tujuan kami menuliskan Proyek Akhir ini adalah:

1. Merancang kontrol kecepatan motor *brushless* DC.
2. Mengimplementasikan kontrol kecepatan motor *brushless* DC.

Harapan dirancangnya alat ini adalah agar alat ini dapat membantu sebagai metode pembelajaran yang berhubungan dengan kontrol kecepatan.

1.3. Permasalahan

Permasalahan yang diangkat pada Proyek Akhir ini adalah belum adanya modul pembelajaran praktis guna mempermudah proses pembelajaran mengenai kontrol kecepatan motor *brushless* DC.

1.4. Batasan Masalah

Pembatasan pada Proyek Akhir ini meliputi :

1. Arah putaran motor *brushless* DC tidak diperhatikan.

2. Kontroler yang digunakan adalah kontroler PID dengan metode *trial and error*.
3. Hanya mengatur kecepatan motor *brushless* DC menggunakan *driver* ESC.

1.5. Sistematika Penulisan

Dari proses pembuatan alat Proyek Akhir ini dimulai dari studi literatur, menentukan gambar desain alat, membuat alat, membuat program, menguji alat secara keseluruhan, analisa data, serta dapat menyusun laporan akhir dengan sistematika penulisan yaitu pendahuluan, teori penunjang, perancangan alat, pengukuran dan analisa serta penutup.

BAB I PENDAHULUAN
Berisi latar belakang pembuatan alat, permasalahan, pembatasan masalah, tujuan, metodologi, sistematika penulisan serta manfaat Proyek Akhir.

BAB II TEORI PENUNJANG
Menjelaskan teori yang berisi teori-teori penunjang yang dijadikan landasan prinsip dasar dan mendukung dalam perencanaan dan pembuatan alat yang dibuat.

BAB III PERANCANGAN ALAT
Membahas tentang tahap-tahap perancangan mekanik dan perancangan sistem kontrolernya.

BAB IV PENGUKURAN DAN ANALISA
Membahas tentang pengukuran dan pengujian alat atau rangkaian yang digunakan pada Proyek Akhir ini.

BAB V PENUTUP
Berisikan tentang kesimpulan atas hasil yang diperoleh serta saran-saran atas kekurangan dan kelemahan Proyek Akhir ini.

1.6. Relevansi

Membuat modul pembelajaran praktis yang mudah digunakan sebagai sarana pembelajaran kontrol kecepatan motor *brushless* DC.

BAB II
TEORI PENUNJANG

Sistem kendali atau sistem kontrol adalah suatu alat (kumpulan alat) untuk mengendalikan, memerintah, dan mengatur keadaan dari suatu sistem. Banyak contoh dalam bidang ah satunya adalah sistem kendali/kontrol kecepatan pada motor *brushless* DC.

Pada bab ini akan dibahas mengenai industri/instrumentasi dan dalam kehidupan kita sehari-hari di mana sistem ini dipakai, salah satu teori yang berkaitan dengan peralatan yang akan dirancang. Teori yang mendukung penyelesaian Proyek Akhir ini diantaranya adalah mengenai : motor *brushless* dc, sensor *rotary encoder*, mikrokontroler Arduino UNO REV3, Lab-*View*, dan kontroler PID.

2.1. Motor *Brushless* DC [1]

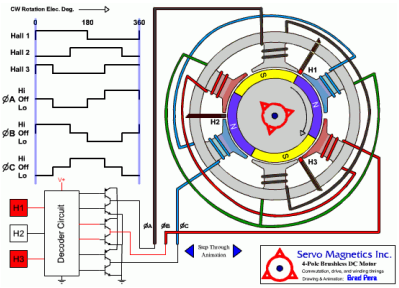
Salah satu jenis motor listrik yang paling banyak digunakan akhir-akhir ini adalah motor *brushless* DC dimana motor DC ini tidak menggunakan *brush* (sikat) untuk proses komutasi. Motor *brushless* DC sangat cocok untuk diaplikasikan pada produk yang menuntut reliabilitas dan efisiensi yang tinggi. Secara umum, dapat dikatakan bahwa motor *brushless* DC dapat menghasilkan torsi yang besar dan mempunyai *range* RPM yang tinggi. Motor *brushless* DC merupakan salah satu jenis motor sinkron dimana medan magnet yang dihasilkan oleh stator dan medan magnet yang dihasilkan oleh rotor berputar pada frekuensi yang sama. Untuk lebih jelasnya mengenai motor ini dapat dilihat pada Gambar 2.1 dan Tabel 2.1.

Gambar 2. 1 Salah Satu Contoh BLDC

Tabel 2. 1 Spesifikasi Motor *Brushless* DC Tipe *Outrunner* RC *Timer* BL2830-11 1000KV [1]

No.	Parameter	Nilai	
1.	Berat Motor	52 gram	
2.	KV	1000 rpm/Volt	
3.	Tegangan	Tegangan Minimal	7 Volt
		Tegangan Maksimal	15 Volt
4.	*Input* Arus	Arus Nominal	4 Ampere
		Arus Maksimal	14.5 Ampere
5.	*Maximal Power*	210 Watt	
6.	*Input* Baterai	Lithium Polimer 2S-4S	

2.1.1 Cara Kerja Motor *Brushless* DC [1]



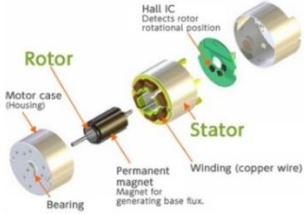
Gambar 2. 2 Skema Kerja Motor Brushless DC [1]

Cara kerja pada motor *brushless* DC cukup sederhana, yaitu magnet yang berada pada poros motor akan tertarik dan terdorong oleh gaya elektro-magnetik yang diatur oleh *Electronic Speed Kontroller* (ESC). Hal ini yang membedakan motor *bushless* DC dengan motor DC dimana motor DC menggunakan sikat mekanis yang berada pada komutator untuk mengatur waktu dan memberikan medan magnet pada lilitan. Motor *brushless* DC ini juga berbeda dengan motor AC yang pada umumnya menggunakan siklus tenaga sendiri untuk mengatur waktu dan memberi daya pada lilitan. Motor *brushless* DC dapat memberikan rasio daya dan beban yang lebih tinggi secara signifikan dan memberikan efisiensi yang lebih baik dibandingkan motor tanpa sikat tradisional.

2.1.2 Konstruksi Motor *Brshless* DC [1]

Motor BLDC terdir dari 2 jenis yaitu *outruneer* dan *inrunne. Outrunner* merupakan jenis motor BLDC yang memiliki stator terletak di dalam motor dan rotor terletak diluar sehingga bagian dari motor yang berputar merupakan bagian luar dari motor. Sedangkan *inrunner* merupakan jenis motor yang memiliki stator di bagian luar motor dan rotor berada pada bagian dalam motor sehingga bagian motor yang beruputar adalah bagian dalam motor.

Gambar 2. 3 *Outruneer Construction* [1]



Gambar 2. 4 *Inrunner Construction* [1]

Setiap motor *brushless* DC memiliki dua bagian utama, rotor (bagian berputar) dan stator (bagian stasioner). Bagian penting lainnya dari motor adalah sebagai berikut :

1. Rotor

   Rotor adalah bagian pada motor yang berputar karena adanya gaya elektromagnetik dari stator, dimana pada motor *brushless* DC bagian rotornya berbeda dengan rotor pada motor DC konvensional yang hanya tersusun dari satu buah elektromagnet yang berada diantara *brushes* (sikat) yang terhubung pada dua buah motor hingga delapan pasang kutub magnet permanen berbentuk persegi pajang yang saling direkatkan menggunakan semacam "*epoxy*" dan tidak ada *brushes*-nya.

2. Stator

Stator adalah bagian pada motor yang diam/statis dimana fungsinya adalah sebagai medan putar motor untuk memberikan gaya elektromagnetik pada rotor sehingga motor dapat berputar. Pada motor *brushless* DC statornya terdiri dari 12 belitan (elektromagnet) yang bekerja secara elektromagnetik dimana statornya terhubung dengan tiga buah kabel untuk disambungkan pada rangkaian kontrol sedangkan pada motor DC konvensional statornya terdiri dari dua buah kutub magnet permanen. Belitan stator pada motor *brushless* DC terdiri dari dua jenis, yaitu belitan stator jenis *trapezoidal* dan jenis *sinusoidal*. Yang menjadi dasar perbedaan kedua jenis belitan stator tersebut terletak pada hubungan antara koil dan belitan stator yang bertujuan untuk memberikan EMF (*Electro Motive Force*) balik yang berbeda.

Berikut ini adalah bagian penting pada stator :

a. *Axle*

Axle atau sumbu adalah batang yang berfungsi sebagai sumbu putar motor, terpusat pada rotor dan dirangkai bersama rotor.

b. Sensor *Hall*

Tidak seperti motor DC, *brushed* komutasi dari motor *brushless* DC diatur secara elektronik agar motor dapat berputar, stator harus di-*energize* secara berurutan dan teratur. Sensor *hall* inilah yang berperan dalam mendeteksi pada bagian rotor mana yang ter-*energize* oleh fluks magnet sehingga proses komutasi yang berbeda (enam langkah komutasi) dapat dilakukan oleh stator dengan tepat karena sensor *hall* ini dipasang menempel pada stator. *Hall sensor* ini ditempatkan setiap 120˚ pada jarak antar kutub stator hal ini bertujuan agar deteksi terhadap *vector* fluks stator yang dihasilkan akurat setiap perpindahan komutasi, arus yang mengalir tetap terjaga konstan pada setiap fasa. Prinsip kerja *hall sensor* sendiri membutuhkan arus yang mengalir terus jika ingin digunakan sebagai pendeteksi fluks magnet.

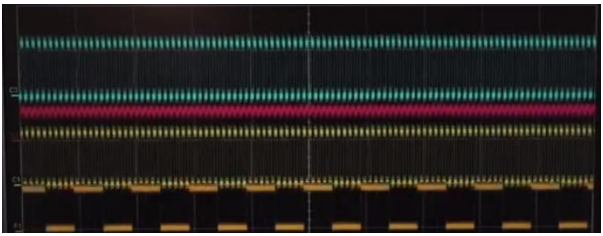
2.1.3 ESC (*Electronic Speed Controller*) [1]

Gambar 2. 5 ESC

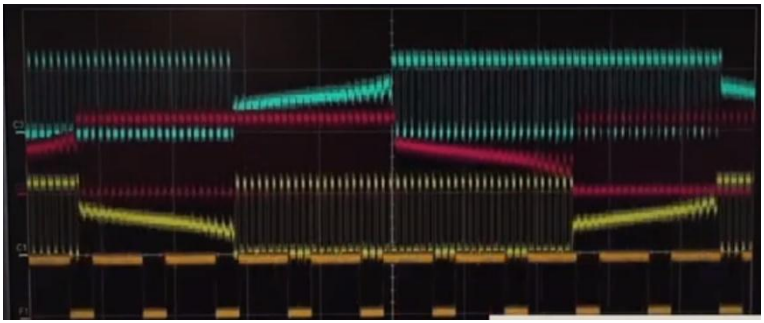
ESC merupakan rangkaian elektronik yang berfungsi sebagai pengatur kecepatan putaran motor pada motor RC (Radio *Control*), cara kerjanya yaitu dengan cara menterjemahkan sinyal yang diterima *receiver* dari *transmitter*. Di pasaran terdapat berbagai *merk* ESC dengan kekuatan arus (*current rating*) dan kekuatan Voltase (*voltage rating*) serta *feature* yang ditawarkan.

Penjelasan yang cepat mengenai ESC adalah ESC harus secara akurat menghubungkan dan memutuskan koneksi antara 3 masukan *input* dan 3 belitan pada stator agar rotor dapat berputar.

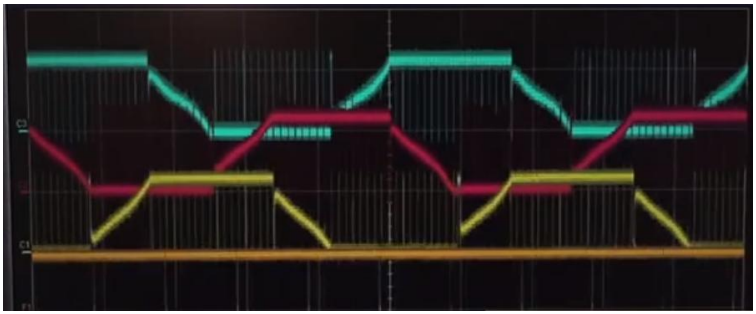
Untuk menentukan ESC yang akan kita gunakan sangatlah penting untuk mengetahui kekuatan (*peak current*) dari motor. Pilihlah ESC yang kekuatannya melebihi kekuatan motor. Misalnya, dari data kita dapatkan kekuatan motor adalah 12A (sesuai dengan *datasheet* motor) pada saat *throttle* terbuka penuh. Sebaiknya ESC yang akan kita gunakan adalah ESC yang berkekuatan 30A. Jika kita paksakan menggunakan ESC 10A kemungkinan pada saat *throttle* dibuka penuh, ESC akan panas bahkan terbakar.



Gambar 2. 6 Sinyal *Output* ESC Ketika *Duty Cycle* 50% [2]



Gambar 2. 7 Sinyal *Output* ESC Ketika *Duty Cycle* 75%[2]



Gambar 2. 8 Sinyal *Output* ESC Ketika *Duty Cycle* 99%[2]

Pada Gambar 2.6, Gambar 2.7, dan Gambar 2.8 adalah gambar sinyal dari ESC. Sinyal warna biru, merah, dan kuning adalah sinyal *output* ESC. Sinyal warna oranye adalah sinyal input ESC. Terlihat perbedaan dari Input 50%, 75%, dan 99%, yaitu ketika input PWM dinaikkan maka sinyal *output* ESC semakin rapat, dan sebaliknya.

2.1.4 Prinsip Kerja Motor *Brushless* DC [1]

Motor BLDC ini dapat bekerja ketika stator yang terbuat dari kumparan diberikan arus 3 fasa. Akibat arus yang melewati kumparan pada stator timbul magnet :

$$B = \frac{\mu N i}{21} \quad \text{.....(2.1)}$$

Dimana N merupakan jumlah lilitan, I merupakan arus, 1 merupakan Panjang lilitan dan μ merupakan permeabilitas bahan.

Karena arus yang diberikan berupa arus DC 3 fasa sinusoidal, nilai medan magnet dan polarisasi setiap kumparan akan berubah-ubah setiap saat. Akibat yang ditimbulkan dari adanya perubahan polarisasi dan besar magnet tiap kumparan adalah terciptanya medan putar magnet dengan kecepatan yang tertera pada persamaan berikut ini :

$$n_s = \frac{120f}{p} \quad \text{.....(2.2)}$$

Dimana f merupakan frekuensi arus *input* dan p merupakan jumlah *pole* stator. Sedangkan untuk mendapatkan total daya (*power)* yang dihasilkan, maka persamaannya adalah :

$$W = V\ x\ A \quad \text{.....(2.3)}$$

Jadi sebenarnya W adalah daya (*power*) yang dihasilkan oleh tegangan dan arus listrik untuk menggerakkan *propeller*. Dan untuk nilai kV yang terdapat pada spesifikasi motor RC (Radio *Control*) yaitu menjelaskan berapa kali motor berputar dalam satu menit dengan tegangan yang berikan, tanpa motor tersebut diberi beban. Sebagai contoh apabila kita mempunyai 11,1V baterai dan nilai kV dari motor adalah 1600, maka motor dapat berputar 1600 x 11,1 = 17760 kali per menit. Nilai kV pada motor yang tinggi biasanya cocok untuk pesawat yang lebih kecil dengan baling-baling standar.

2.2. Sensor *Rotary Encoder* [1]

Gambar 2. 9 *Rotary Encoder* [1]

Rotary encoder adalah suatu komponen elektro mekanis yang memiliki fungsi untuk memonitoring posisi anguler pada suatu poros yang berputar. Dari perputaran benda tersebut data yang termonitoring akan diubah

ke dalam bentuk data digital oleh *rotary encoder* berupa lebar pulsa kemudian akan dihubungkan ke mikrokontroler.

Konstruksi *rotary encoder* berupa piringan tipis yang biasanya di kopel dengan poros yang berputar. Piringan tipis tersebut terdapat lubang di sepanjang pinggir lingkarannya. Di bagian sisi-sisi piringan terdapat sebuah led dan *phototransistor* di bagian bersebrangan. Fungsi dari lubang-lubang yang berada di sepanjang pinggir lingkaran tersebut akan menghantarkan cahaya led ke *phototransistor*, sebaliknya jika cahaya led tidak menembus lubang piringan maka cahaya akan tertahan. Piringan tersebut akan berputar sesuai dengan kecepatan putaran motor sehingga *phototransistor* akan saturasi ketika cahaya LED menembus lubang-lubangnya.

Pada saat saturasi *phototransistor* akan menghasilkan pulsa dengan *range* +0,5 V s/d +5 V.

Tabel 2. 2 Deskripsi Arduino UNO REV3

Mikrokontroler	ATmega 328P
Operating Voltage	5 V
Input Voltage (recommended)	7 – 12 V
Input Voltage (limit)	6 – 20 V
Digital I/O Pins	16 (6 diantaranya menyediakan PWM)
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3,3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) (0.5 KB sebagai *bootloader*)
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
USB *Host Chip*	MAX3421E
Length	68,6 mm
Width	53,4 mm
Weight	25 g

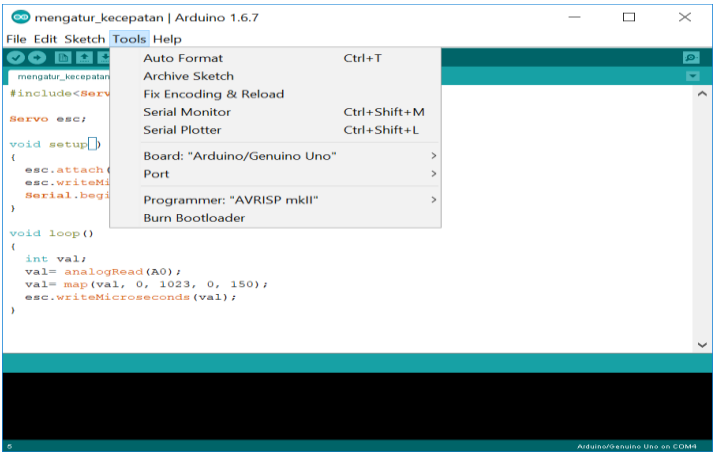
2.3. Mikrokontroler Arduino UNO REV3 [3]

Arduino Uno R3 adalah papan pengembangan (*development board*) mikrokontroler yang berbasis chip ATmega328P. Disebut sebagai papan pengembangan karena *board* ini memang berfungsi sebagai arena *prototyping* sirkuit mikrokontroler. Dengan menggunakan papan pengembangan, anda akan lebih mudah merangkai rangkaian elektronika mikrokontroler dibanding jika anda memulai merakit ATMega328 dari awal di *breadboard*. Tabel 2.2 merupakan spesifikasi arduino UNO REV3.

Gambar 2. 10 Arduino UNO REV3

Untuk memprogram arduino juga harus dilakukan beberapa tahapan sebagai berikut :

1. *Setting Board Arduino*
   Dalam pemprograman *software* arduino harus di *setting* terlebih dahulu *board* arduino agar penggunaan arduino cocok. Dalam purwarupa kali ini arduino menggunakan arduino UNO REV3. Untuk *setting board* arduino bisa masuk ke *tools – board –* setelah itu pilihlah *board* arduino yang sesuai.
2. *Setting Serial*
   Serial ini merupakan kabel arduino yang dihubungkan kepada komputer atau laptop. *Serial* ini mempunyai dua fungsi yang bisa digunakan. Pertama *serial port* digunakan untuk men-*download* program dari arduino yang kedua *serial* digunakan sebagai komunikasi *serial* pada arduino dengan komputer. *Setting serial* bisa masuk *tools – serial* - lali pilih COM yang sesuai dengan arduino yang terpasang. Untuk lebih jelasnya dapat dilihat pada Gambar 2.11.



Gambar 2. 11 *Setting Serial Port*

Apabila *program* tidak dapat di *download* karena *serial port*, maka cek terlebih dahulu *serial* yang benar pada *device manager*. Lalu dalam *software* arduino untuk memilih *serial port*nya samakan dengan *serial port* untuk arduino dalam *device manager* tersebut. Untuk masuk ke *device manager* dapat masuk start windows – lalu ketika *device manager* klik dua kali dan masuk ke dan COM.

2.4. Lab*View* [3]

Lab*View* adalah sebuah *software* pemograman yang diproduksi oleh *National instruments* dengan konsep yang berbeda. Seperti bahasa pemograman lainnya yaitu C++, MATLAB atau *Visual basic*, Lab*View* juga mempunyai fungsi dan peranan yang sama, perbedaannya bahwa Lab*View* menggunakan bahasa pemrograman berbasis grafis atau blok diagram sementara bahasa pemrograman lainnya menggunakan *basis text*. Program Lab*View* dikenal dengan sebutan VI atau *Virtual instruments* karena penampilan dan operasinya dapat meniru sebuah *instrument*. Pada Lab*View*, *user* pertamatama membuat *user interface* atau *front panel* dengan menggunakan kontrol dan indikator, yang dimaksud dengan kontrol adalah *knobs*, *push buttons*, *dials* dan peralatan *input* lainnya sedangkan yang dimaksud dengan indikator adalah *graphs*, LEDs dan peralatan *display* lainnya. Setelah menyusun *user interface*, lalu *user* menyusun blok diagram yang berisi kode-kode VIs untuk mengontrol *front panel*.

2.5. Kontroler PID [4]

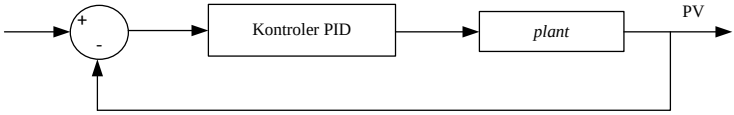
PID dari singkatan (*Propotional-Integral-Derivative Controller*) merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik pada sistem tersebut. Komponen control PID ini terdiri dari tiga jenis yaitu *Propotional*, *integral*, dan *derivative*. Ketiganya dapat dipakai bersamaan maupun sendiri-sendiri tergantung dari respon yang kita inginkan terhadap suatu *plant*.

Efek dari setiap pengendali dalam sistem *close loop* dapat dilihat pada Tabel 2.3 berikut.

Tabel 2. 3 Pengaruh Nilai Kp, Ki, Kd Pada Respon Sistem

Respon *Close Loop*	*Rise Time*	*Overshoot*	*Settling Time*	*Error Steady State*
K_p	Turun	Naik	Perubahan Kecil	Turun
K_i	Turun	Naik	Naik	Hilang
K_d	Perubahan Kecil	Turun	Turun	Perubahan Kecil

Blok diagram dari kontroler PID dengan sistem *close loop* digambarkan pada Gambar 2.12 dibawah ini.



Gambar 2. 12 Blok Diagram Kontroler PID Sistem *Close Loop*

Kontroler PID dibagi menjadi dua, yakni PID *Continous* dan PID *Discrete*.

-----Halaman ini sengaja dikosongkan-----

BAB III
PERANCANGAN DAN IMPLEMENTASI

Dalam bab ini akan dibahas mengenai perancangan alat yang meliputi perencanaan perangkat keras (*hardware*) dan perangkat lunak (*software*). Hal tersebut guna mewujudkan Proyek Akhir yang berjudul "Pengaturan Kecepatan Motor *Brushless* DC Menggunakan Kontroler Berbasis PID". Perancangan alat akan dibahas perbagian yang disertai dengan gambar skematik.

Untuk memudahkan dalam pembahasan bab ini akan dibagi menjadi 3 yaitu:

1. Blok fungsional sistem
2. Perancangan perangkat keras yang terdiri dari perancangan mekanik dan elektrik, yaitu:
    a. Perancangan mekanik meliputi desain alat secara keseluruhan
    b. Perancangan elektrik meliputi rangkaian sensor *rotary encoder*
3. Perancangan perangkat lunak yang terdiri dari :
    a. Perancangan perangkat lunak dengan *software* Arduino Uno REV3 yang menggunakan bahasa C sebagai bahasa pemrogramannya.
    b. Perancangan perangkat lunak Lab*View*

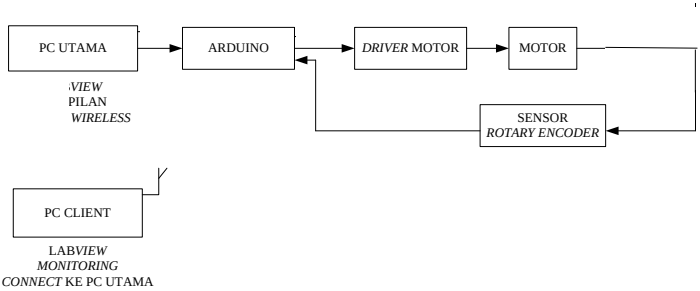
3.1. Blok Fungsional Sistem

Perancangan sistem dalam pembuatan alat ini secara garis besar disertai urutan dan cara kerja alat ini di ilustrasikan pada Gambar 3.1 dan Gambar 3.2.

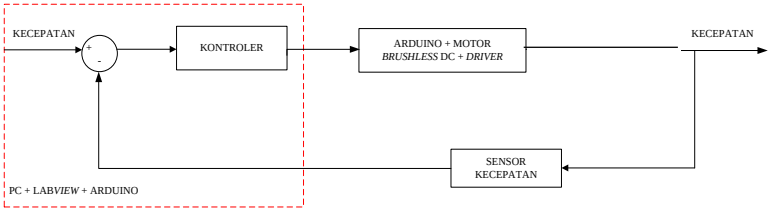
Dari Gambar 3.1 dapat dilihat bahwa sistem tersebut terdiri dari beberapa blok fungsional yaitu;

1. Arduino Uno REV3, merupakan mikrokontroler yang berfungsi sebagai *interface* dari perangkat elektronik dan dapat menyimpan program didalamnya.
2. *Driver* motor yaitu ESC berfungsi sebagai penggerak motor dan sensor.
3. Motor *brushless* DC, digunakan sebagai objek yang akan dikontrol kecepatannya.
4. Sensor kecepatan (*rotary encoder*), digunakan untuk mengetahui sampai seberapa kecepatan motor yang terjadi.

5. Lab*View*, merupakan perangkat lunak yang digunakan sebagai media untuk memberi masukan atau interupsi ke kontroler.
6. Tampilan (*Display*), untuk tampilan digunakan PC, PC ini digunakan untuk menampilkan kecepatan motor *brushless* DC.



Gambar 3. 1 Blok Fungsional Perancangan



Gambar 3. 2 Blok Diagram Sistem *Close Loop* PID

3.2. Perancangan Perangkat Keras (*Hardware*)

Pada perancangan perangkat keras ini, prosesnya dibagi menjadi 2 bagian, yaitu perancangan mekanik dan elektrik. Masing-masing perancangan tersebut selanjutnya akan dibahas lebih mendalam pada sub bab berikutnya.

3.2.1 Perancangan Mekanik

Perancangan mekanik ini terdiri dari desain alat secara keseluruhan, motor *brushless* DC, *rotary encoder*, dan *driver* motor *brushless* DC.

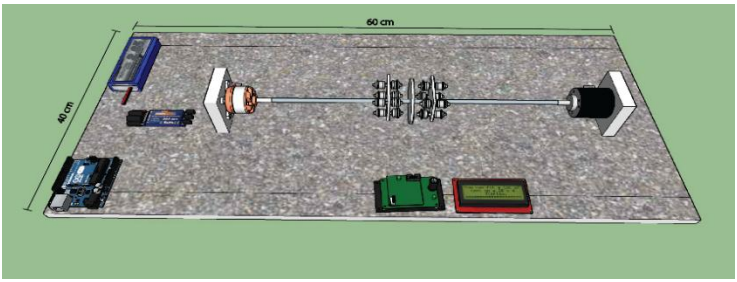
3.2.1.1. Motor *Brushless* DC

Motor BLDC dipilih karena memiliki berbagai kelebihan, yaitu, dapat menggerakkan dengan daya yang besar, suaranya halus, torsi besar, efisiensi tinggi, mudah di control, dan perawatan yang rendah. Motor BLDC yang digunakan merupakan motor BLDC yang biasa digunakan untuk *aero-modelling* sehingga ukurannya lebih kecil. Motor *brushless* DC merupakan salah satu jenis motor sinkron dimana medan magnet yang dihasilkan oleh stator dan medan magnet yang dihasilkan oleh rotor berputar pada frekuensi yang sama. Untuk lebih jelasnya mengenai motor ini dapat dilihat pada Gambar 2.1 dan Tabel 2.1.

3.2.1.2. *Electronic Speed Control* (ESC)

ESC atau disebut juga *Electronic Speed Control* adalah *driver* penggerak untuk jenis motor *brushless*, biasanya digunakan pada bidang *aeronautical* atau RC (Radio *Control*). *Driver* ini bertujuan untuk mengubah sinyal kontrol yang dikirimkan Arduino menjadi tegangan 3 fasa beda 120 derajat yang digunakan untuk menggerakkan motor BLDC. Untuk menentukan ESC yang akan kita gunakan sangatlah penting untuk mengetahui kekuatan (*peak current*) dari motor. Pilihlah ESC yang kekuatannya melebihi kekuatan motor. Misalnya, dari data kita dapatkan kekuatan motor adalah 12A (sesuai dengan *datasheet* motor) pada saat *throttle* terbuka penuh. Sebaiknya ESC yang akan kita gunakan adalah ESC yang berkekuatan 30A. Jika kita paksakan menggunakan ESC 10A kemungkinan pada saat *throttle* dibuka penuh, ESC akan panas bahkan terbakar.

3.2.1.3. Desain Alat Secara Keseluruhan



Gambar 3. 3 Desain Modul Kontrol Kecepatan Motor *Brushless* DC Dengan Mikrokontroler

3.2.2 Perancangan Elektrik

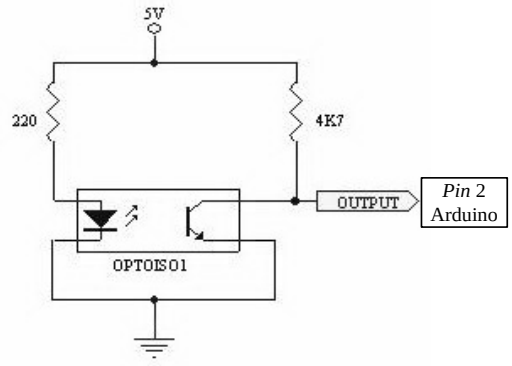
Perancangan elektronik ini meliputi desain layout PCB serta pengkabelan. Rangkaian elektrik pada *plant* ini meliputi rangkaian sensor *rotary encoder*, dan Arduino Uno REV3 .

3.2.2.1. Rangkaian Sensor *Rotary Encoder*

Rotary encoder pada Proyek Akhir ini akan digunakan untuk menentukan banyaknya putaran poros tiap menit (RPM) yang kemudian akan menghasilkan gelombang kotak yang frekuensinya akan bertambah bila kecepatan putar poros bertambah. *Rotary encoder* ini diletakkan pada poros yang sudah dikopel dengan motor *brushless* DC.

Rotary encoder ini tersusun dari suatu piringan tipis yang memiliki lubang-lubang yang terdapat pada piringan tersebut. Setelah itu akan ditempatkan LED pada salah satu sisi piringan. Hal ini akan membuat cahaya masuk menuju piringan.

Kemudian disisi lain dari piringan ini diletakan *phototrasnsistor* yang bertujuan untuk mendeteksi cahaya LED yang berseberangan. Piringan tipis ini yang nantinya akan dikopel dengan poros motor ataupun perangkat berputar lainnya yang ingin kita ketahui posisinya, hal ini akan membuat piringan berputar keika motor tersebut berputar. Apabila cahaya yang berasal dari LED mencapai *phototransistor*, maka *phototranstor* itu akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi.



Gambar 3. 4 Rangkaian *Rotary Encoder*

Pada gambar diatas LED Inframerah kita gunakan untuk menembakkan cahaya sedangkan disisi kanan *light receive* dapat kita gunakan sensor cahaya seperti *photodiode* atau *phototransistor*.

Pada *transmitter* terdapat sebuah LED inframerah (IR LED) yang berfungsi untuk mengirimkan sinyal kepada *receiver*. Cahaya yang dipancarkan oleh LED inframerah tidak terlihat oleh mata telanjang. Sedangkan pada *receiver* terdapat *phototransistor* (transistor yang peka terhadap perubahan cahaya). Untuk memicu *optocoupler* maka digunakan piringan hitam yang diberi celah di ujungnya dan di*couple* dengan poros *generator*. Prinsip kerjanya ketika generator berputar maka piringan akan ikut berputar, kemudian *optocoupler* menyensor ujung piringan. Ketika ada celah melewati *optocoupler*, maka cahaya dipancarkan *transmitter* menuju *receiver*, sehingga menghasilkan tegangan keluaran yang nilainya mendekati VCC, begitu juga sebaliknya, jika tidak ada benda diantara celah sensornya maka akan menghasilkan tegangan keluaran yang nilainya mendekati 0 Volt. Tujuannya adalah mendapatkan frekuensi putaran dari *generator*.

3.3. Perancangan Perangkat Lunak (*Software*)

Agar rem elektromagnetik dapat dikontrol dan hasil putaran dapat dilihat di LCD maka perlu dirancang di sebuah *software* yang mampu mengelola dan mengontrol data terhadap kinerja peralatan rem elektromagnetik. *Software* merupakan program berisi perintah perintah yang di eksekusi oleh Arduino Uno REV3 sehingga sistem dapat bekerja sesuai dengan alur dan tujuan yang dirancang.

3.3.1. Pemrograman *Software* Arduino

Dalam perancangan program pada *software* arduino dengan fungsi terkait yang dibutuhkan diperlukan beberapa tahapan yang harus dilakukan terlebih dahulu. Tahapan tersebut adalah membuat algoritma dari alat yang sudah kita jalankan.

Pembuatan algoritma ini dilakukan setelah membuat *flowchart*, dari algoritma kita ini maka diharapkan alat yang akan dibuat ini dapat terlebih lebih sederhana. Setelah tahapan tersebut terselesaikan barulah kita memprogram fungsi terkait yang dikodingkan dalam bahasa C.

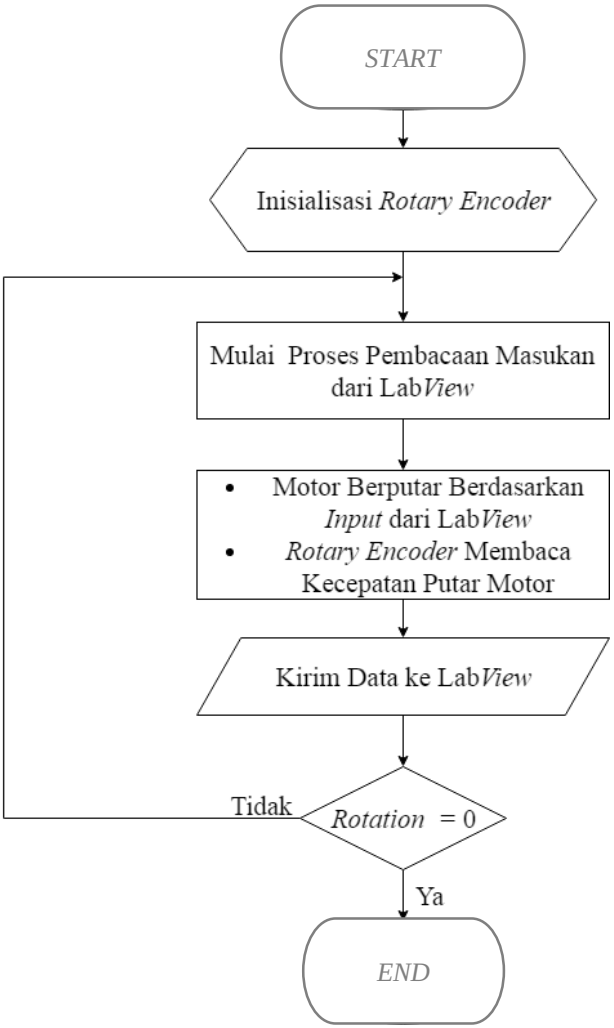
Berikut ini algoritma program utama dari kontrol kecepaan :

1. Modul dapat di operasikan setelah sistem terpasang dengan benar seperti motor *brushless* DC, *driver* motor *brushless* DC, catu daya dan *potensiometer*.
2. Modul dapat bekerja jika rangkaian kontrol sudah dijalankan dan sudah terpasang.
3. *Potensiometer* digunakan untuk mengatur *range* yang akan di suplai ke *driver* yang berupa tegangan dan akan di ubah menjadi digital melalui ADC Arduino.

Pada Proyek Akhir ini *software* yang digunakan adalah program Arduino Uno REV3 untuk membuat dan merencanakan program dalam bahasa C. Pemrograman *software* arduino dirancang dengan menggunakan *software* yang bernama Arduino IDE dengan menggunakan bahasa pemprograman C.

Arduino sangatlah berbeda sekali dengan mikrokontroler. Arduino merupakan sebuah *kit* mikrokontroler AVR yang dibuat dalam sebuah *board* (papan PCB). Dikembangkan di Italia sejak tahun 2005. Dalam 1 *board* sudah terdapat mikrokontroler lengkap dengan *pin/port* untuk koneksi serta sudah dilengkapi dengan *downloader*. Dalam segi bahasa pemrograman, arduino memiliki bahasa pemrograman yang lebih mudah dan sederhana terutama bagi pemula.

Alasan bahasa pemrograman arduino lebih mudah dan sederhana adalah karena didalam arduino sudah terdapat beberapa *library* yang dapat digunakan untuk merancang pemrograman yang diinginkan. Pada Gambar 3.5 berikut menunjukkan *flowchart* yang digunakan pada Arduino Uno REV3 pada Proyek Akhir ini.



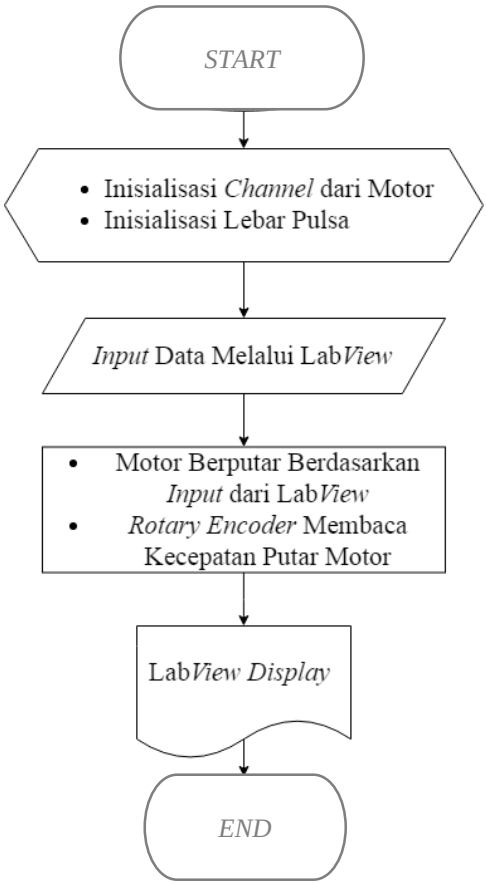
Gambar 3. 5 *Flowchart* Arduino Uno REV3

3.3.2 Perancangan Perangkat Lunak Lab*View*

Ketika awal memulai *software* Lab*View*, tahapan yang dilakukan adalah dengan membuka *File Menu* kemudian klik *New* VI. *Untitled* VI akan

mempunyai dua bagian yang disebut *Front Panel* dan *Block Diagram*. *Front Panel Window* merupakan *user interface window* pada saat VI dijalankan, sedangkan *Block Diagram Window* merupakan bagian yang akan melakukan kalkulasi dan mengeksekusi algoritma.

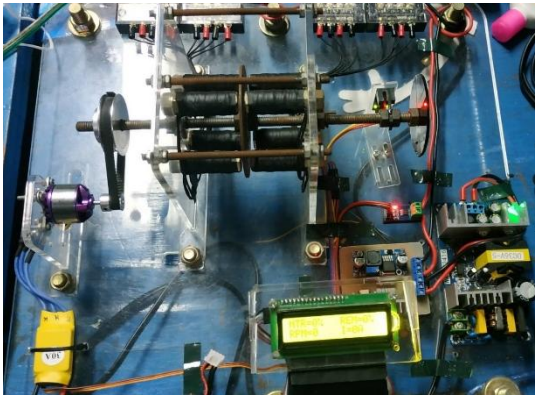
Gambar 3.6 menunjukkan *state diagram* yang difungsikan pada Lab-*View* untuk pengaturan kecepatan pada motor *brushless* DC.



Gambar 3. 6 *Flowchart* Lab*View*

BAB IV
PENGUJIAN DAN ANALISA ALAT

Pada bab ini akan dibahas mengenai pengujian alat dan analisa data dari hasil rancangan alat yang telah dibuat. Pengujian alat ini ditujukan untuk memastikan agar peralatan dapat berfungsi dengan baik.



Gambar 4. 1 Realisasi *Plant* Motor BLDC

Pada gambar 4.1 pengujian dan analisa dari alat ini meliputi analisa program arduino, pengujian sensor *rotary encoder*, pengujian kecepatan putaran motor *brushless* DC, serta pengujian motor *brushless* DC dengan kontroler PID Lab*View*. Setelah melakukan beberapa pengujian tersebut, data yang diperoleh akan dianalisa untuk mengetahui proses kerja dari seluruh sistem alat yang dibuat.

4.1. Analisa Program Arduino

Pada dasarnya program Arduino hanya berfungsi sebagai pengirim data dan penerima data dari komputer. Program Arduino yang dibuat kali ini memiliki memiliki 2 sistem utama yaitu inisialisasi dan program utama. Gambar 4.2 merupakan program inisialisasi yang dibuat.

Pada inisialisasi terdapat beberapa hal yang dilakukan yaitu menyebutkan jenis-jenis *library* yang dipakai, menyebutkan variabel-variabel yang dipakai, menyebutkan nilai *baudrate*, dan yang terakhir menyebutkan *pin-pin* yang dipakai pada Arduino.

Pada Gambar 4.2 bahwa dibutuhkan *library servo* untuk implementasi, *library servo* ini berfungsi untuk menjalankan motor sehingga motor hanya perlu dimasukkan nilai-nilai tertentu dan motor dapat langsung berputar.

Kemudian terdapat beberapa jenis variabel yang dipakai diantaranya *integer*, *float*, *unsigned long*, dan *unsigned int*. Dengan *baudrate* yang dipakai adalah bernilai 9600. Pada program utama Arduino akan terjadi program yang terus menerus dilakukan selama Arduino dalam kondisi menyala. Pada program utama terbagi menjadi 2 buah sistem, yang pertama merupakan program komunikasi data komputer (Lab*View*) melalui USB *serial*, kemudian yang kedua merupakan program untuk membaca nilai kecepatan motor dalam satuan RPM.

```
LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7);
PWMServo esc;

void counter()
{
  pulses++;
}

void setup()
{
  lcd.begin(16,2);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);
  esc.attach(9);
  Serial.begin(9600);
  pinMode(encoder_pin, INPUT);
  attachInterrupt(0, counter, FALLING);
  pinMode(11,OUTPUT);
  pulses = 0;
```

Gambar 4. 2 Potongan Program Inisialisasi

Pada Gambar 4.3 ditunjukkan program untuk kontrol motor melalui Arduino IDE. Kemudian pada Gambar 4.4 adalah program untuk membaca kecepatan motor *brushless* DC dalam satuan RPM.

```
//Note that this would be 60*1000/(millis() - timeold)*pulses i
//happened once per revolution
rpm = (20 * 1000 / pulsesperturn )/ (millis() - timeold)* pulse
timeold = millis();
pulses = 0;

 //Write it out to serial port
//Serial.print("RPM = ");
Serial.println(rpm,DEC);
//Restart the interrupt processing
attachInterrupt(0, counter, FALLING);

int val;
val= analogRead(A0);
setpoint = map(val, 0, 1023, 0, 255);
int percentsetpoint = map(val, 0, 1023, 0, 100);
int percentsetpoint2;
Serial.println(percentsetpoint);
esc.write(setpoint);
```

Gambar 4. 3 Potongan Program Motor

```
void loop()
{
  if (millis() - timeold >= 1000){  /*Uptade every one second, this will be equal to reading frecuency (Hz).*/

  //Don't process interrupts during calculations
   detachInterrupt(0);
   //Note that this would be 60*1000/(millis() - timeold)*pulses if the interrupt
   //happened once per revolution
   rpm = (60 * 1000 / pulsesperturn )/ (millis() - timeold)* pulses;
   timeold = millis();
   pulses = 0;

    //Write it out to serial port
   Serial.print("RPM = ");
   Serial.println(rpm,DEC);
   //Restart the interrupt processing
   attachInterrupt(0, counter, FALLING);
```

Gambar 4. 4 Potongan Program Sensor Kecepatan

Pada program sensor kecepatan, program akan membaca berapa banyak jumlah pulsa yang dihasilkan dalam suatu waktu tertentu yang telah ditentukan. Kemudian jumlah pulsa ini akan dihitung dalam suatu rumus dan hasilnya akan dikonversi menjadi dalam satuan RPM. Setelah itu program akan menuliskan nilai kecepatan motor tersebut pada *front panel* Lab*View*.

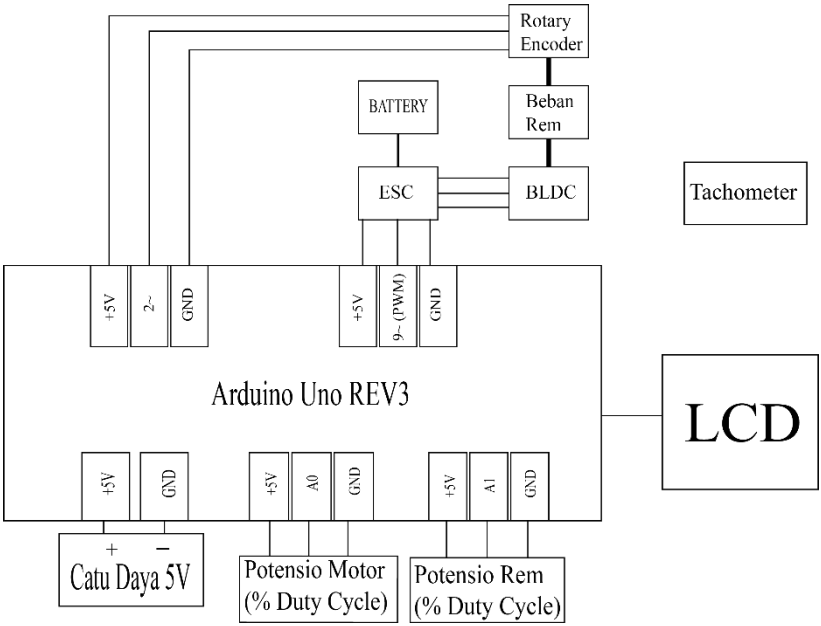
4.2. Pengujian Sensor *Rotary Encoder*

Pada pengujian sensor *rotary encoder*, terdapat 2 pengujian. Yang pertama adalah linierisasi sensor dan yang kedua adalah kalibrasi sensor. Linierisasi sensor dilakukan untuk mengetahui kondisi sensor, apakah sensor

dalam kondisi yang masih bagus atau sudah rusak. Sedangkan kalibrasi sensor dilakukan agar pembacaan kecepatan dari sensor *rotary encoder* akurat. Untuk pengujian linierisasi sensor, caranya yaitu dengan menjalankan motor *brushless* DC dengan sensor *rotary encoder* yang sudah terpasang. Hal ini dilakukan untuk mengetahui respon dan hasil pembacaan kecepatan yang terbaca sensor *rotary encoder* yang dipakai. Program dari pengujian ini menggunakan Lab*View* dengan blok diagram seperti tertera pada Lampiran A-5.

Hal ini menunjukkan bahwa sensor masih dalam kondisi yang cukup bagus. Selain itu, untuk mengetahui kondisi dari sensor *rotary encoder* dapat pula dilakukan pembacaan sinyal pulsa yang dihasilkan oleh sensor melalui osiloskop. Apabila hasil sinyal pulsa berupa kotak sempurna maka sensor dalam kondisi yang bagus. Seperti tertera pada Gambar 4.5 yang merupakan hasil keluaran sinyal pulsa dari sensor.

Gambar 4. 5 Sinyal Pulsa Ketika *Rotary Encoder* Membaca Kecepatan



Gambar 4. 6 Rangkaian Pengujian Sensor *Rotary Encoder*

Untuk melakukan pengujian terhadap data tabel dibawah ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di setting pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.

Tabel 4. 1 Pengujian Sensor *Rotary Encoder*

No	*Input* Potensio Motor (*Duty Cycle* %)	Kecepatan Mtr BLDC (tachometer) RPM	Kecepatan Mtr BLDC (Encoder) RPM	*Error*
1	10	570	541	29
2	20	864	838	26
3	30	1248	1225	23
4	40	1440	1417	23
5	50	1872	1823	49
6	60	2160	2068	92
7	70	2352	2194	158
8	80	2528	2471	57
9	90	2736	2614	122
10	100	3004	2983	21

Tabel 4.1 menunjukkan hasil dari pengujian kecepatan motor *brushless* DC. Penjelasan data pada Tabel 4.1 yaitu dari segi tegangan *input* potensiometer adalah input potensiometer motor. Dengan membandingkan motor berjalan tanpa rem dan menggunakan rem. Kemudian dalam segi keterangan adalah mencari korelasi dari pengukuran menggunakan sensor *rotary encoder* dan tachometer.

Pada pengujian yang tertera pada Tabel 4.1 dapat dilihat bahwa *output* hasil pembacaan pada sensor *rotary encoder* dan *digital tachometer* memiliki perbedaan hingga mencapai kurang lebih 10-40 RPM. Perbedaan kecepatan ini semakin besar pada saat kecepatan tinggi. Hal ini disebabkan karena getaran dari poros motor yang sudah tidak lurus lagi yang menyebabkan getaran piringan *encoder* saat diputar pada kecepatan tinggi.

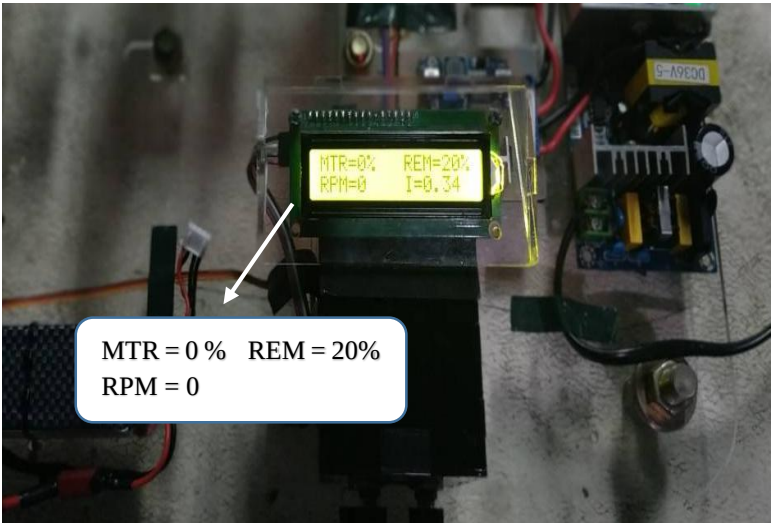
Adanya keterangan *error* dikarenakan pembacaan pada sensor tidak terlalu akurat seperti pada alat ukur sebenarnya.

4.3. Pengujian Kecepatan Putaran Motor *Brushless* DC Tanpa Kontroler PID

Pengujian kecepatan motor dilakukan untuk menentukan kemampuan putaran dari motor *brushless* DC yang dipakai. Pengujiannya dilakukan sekali tetapi dengan set point yang berbeda-beda. Dan pengujian yang tertera pada Sub Bab 4.3.1 sampai dengan Sub Bab 4.3.5 ini bisa disebut pengukuran manual karena masih menggunakan potensiometer dan melihat hasil melalui LCD dan hasilnya untuk mencari *step* respon yang digunakan untuk mencari parameter PID.

4.3.1 Pengujian Kecepatan Motor Dengan 20% Pengereman

Untuk melakukan pengujian terhadap data tabel dibawah ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan battery untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di *setting* pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.



Gambar 4. 7 Pengereman 20% *Duty Cycle*

Pada Gambar 4.7 *input* potensiometer adalah *input* potensiometer motor, kami ambil per 10% *Duty Cycle*. Dan untuk segi beban rem yaitu kami atur pada potensiometer rem 20% dari *Duty Cycle* yang sudah di program di Arduino IDE, dengan membandingkan motor berjalan tanpa rem dan menggunakan rem.

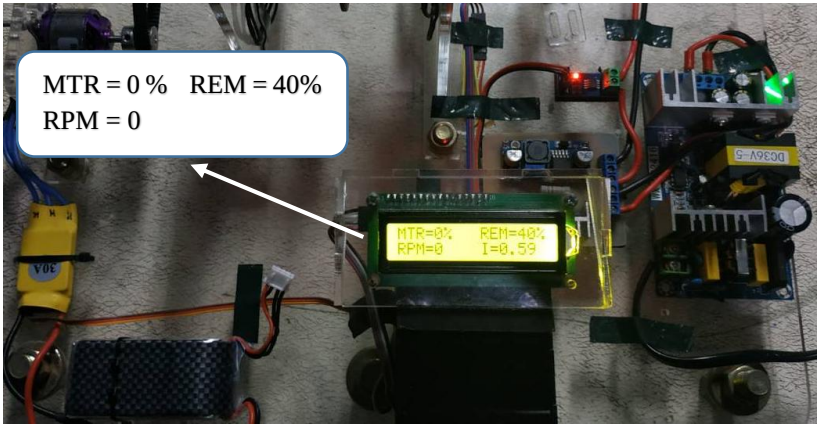
Tabel 4. 2 Kalibrasi Motor dengan Beban Rem 20%

No	*Input* Potensio Motor (*Duty Cycle* %)	Kecepatan Mtr BLDC (tachometer) RPM	Kecepatan Mtr BLDC (Encoder) RPM
1	10	902	873
2	20	1288	1275
3	30	1952	1927
4	40	2250	2238
5	50	2377	2359
6	60	2537	2514
7	70	2628	2592
8	80	2740	2710
9	90	2836	2804
10	100	2904	2873

Untuk melakukan pengujian terhadap data tabel diatas ini yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, kemudian di tampilkan di LCD, kemudian dengan menggunakan Tachometer dengan mengarahkan ke piringan yg terdapat pada modul tersebut.

4.3.2 Pengujian Kecepatan Motor Dengan 40% Pengereman

Untuk melakukan pengujian terhadap data Tabel 4.3 ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di *setting* pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.



Gambar 4. 8 Pengereman 40% *Duty Cycle*

Pada Gambar 4.8 *input* potensiometer adalah *input* potensiometer motor, kami ambil per 10% *Duty Cycle*. Dan untuk segi beban rem yaitu kami atur pada potensiometer rem 40% dari *Duty Cycle* yang sudah di program di Arduino IDE, dengan membandingkan motor berjalan tanpa rem dan menggunakan rem.

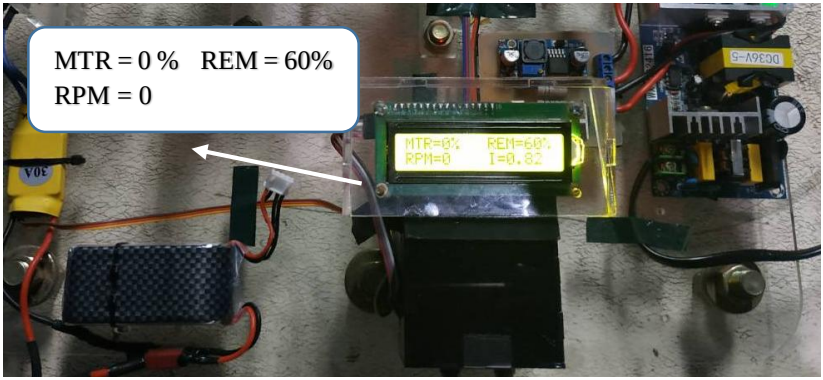
Tabel 4. 3 Kalibrasi Motor dengan Beban Rem 40%

No	***Input*** **Potensio Motor (*Duty Cycle* %)**	**Kecepatan Mtr BLDC (tachometer) RPM**	**Kecepatan Mtr BLDC (*En-coder*) RPM**
1	10	825	816
2	20	1096	976
3	30	1824	1805
4	40	2144	2104
5	50	2238	2201
6	60	2338	2306
7	70	2468	2416
8	80	2662	2627
9	90	2708	2691
10	100	2835	2807

Untuk melakukan pengujian terhadap data tabel diatas ini yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, kemudian di tampilkan di LCD, kemudian dengan menggunakan Tachometer dengan mengarahkan ke piringan yg terdapat pada modul tersebut.

4.3.3 Pengujian Kecepatan Motor Dengan 60% Pengereman

Untuk melakukan pengujian terhadap data tabel dibawah ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di *setting* pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.



Gambar 4. 9 Pengereman 60% *Duty Cycle*

Pada Gambar 4.9 *input* potensiometer adalah *input* potensiometer motor, kami ambil per 10% *Duty Cycle*. Dan untuk segi beban rem yaitu kami atur pada potensiometer rem 60% dari *Duty Cycle* yang sudah di program di Arduino IDE, dengan membandingkan motor berjalan tanpa rem dan menggunakan rem.

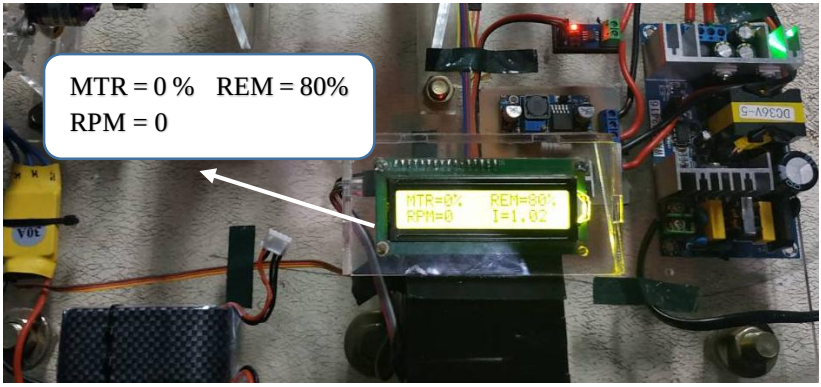
Tabel 4. 4 Kalibrasi Motor dengan Beban Rem 60%

No	*Input* Potensio Motor (*Duty Cycle* %)	Kecepatan Mtr BLDC (tachometer) RPM	Kecepatan Mtr BLDC (*Encoder*) RPM
1	10	656	642
2	20	871	816
3	30	1792	1750
4	40	2025	2016
5	50	2138	2124
6	60	2203	2176
7	70	2337	2304
8	80	2521	2480
9	90	2657	2640
10	100	2793	2768

Untuk melakukan pengujian terhadap data tabel diatas ini yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, kemudian di tampilkan di LCD, kemudian dengan menggunakan Tachometer dengan mengarahkan ke piringan yg terdapat pada modul tersebut.

4.3.4 Pengujian Kecepatan Motor Dengan 80% Pengereman

Untuk melakukan pengujian terhadap data tabel dibawah ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di *setting* pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.



Gambar 4. 10 Pengereman 80% *Duty Cycle*

Pada Gambar 4.10 *input* potensiometer adalah *input* potensiometer motor, kami ambil per 10% *Duty Cycle*. Dan untuk segi beban rem yaitu kami atur pada potensiometer rem 80% dari *Duty Cycle* yang sudah di program di Arduino IDE, dengan membandingkan motor berjalan tanpa rem dan menggunakan rem.

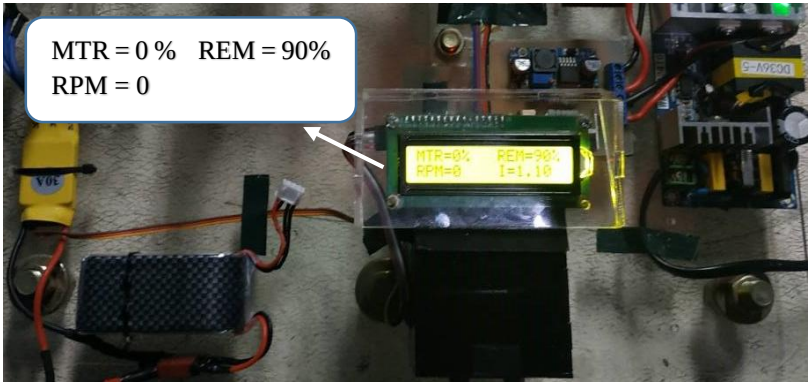
Tabel 4. 5 Kalibrasi Motor dengan Beban Rem 80%

No	***Input*** **Potensio Motor (*Duty Cycle* %)**	**Kecepatan Mtr BLDC (tachometer) RPM**	**Kecepatan Mtr BLDC (*Encoder*) RPM**
1	10	578	533
2	20	733	717
3	30	1627	1614
4	40	1907	1895
5	50	2002	1992
6	60	2107	2046
7	70	2215	2175
8	80	2405	2388
9	90	2577	2507
10	100	2759	2741

Untuk melakukan pengujian terhadap data tabel diatas ini yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, kemudian di tampilkan di LCD, kemudian dengan menggunakan Tachometer dengan mengarahkan ke piringan yg terdapat pada modul tersebut.

4.3.5 Pengujian Kecepatan Motor Dengan 90% Pengereman

Untuk melakukan pengujian terhadap data tabel dibawah ini yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan *ESC* agar dapat berjalan, setelah itu masukkan program ke Arduino uno REV3 untuk mengatur *Duty Cycle* pada potensiometer rem dan motor, setelah itu jalankan motor *BLDC* lewat potensiometer motor yang sudah di *setting* pada program, nanti hasil akan keluar pada LCD yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, agar hasil lebih tepat, maka mengambil data pada motor dilakukan lewat sensor *Rotary Encoder* dan alat pengukur kecepatan yaitu *Tachometer*.



Gambar 4. 11 Pengereman 90% *Duty Cycle*

Pada Gambar 4.11 *input* potensiometer adalah *input* potensiometer motor, kami ambil per 10% *Duty Cycle*. Dan untuk segi beban rem yaitu kami atur pada potensiometer rem 90% dari *Duty Cycle* yang sudah di program di Arduino IDE, dengan membandingkan motor berjalan tanpa rem dan menggunakan rem.

Tabel 4. 6 Kalibrasi Motor dengan Beban Rem 90%

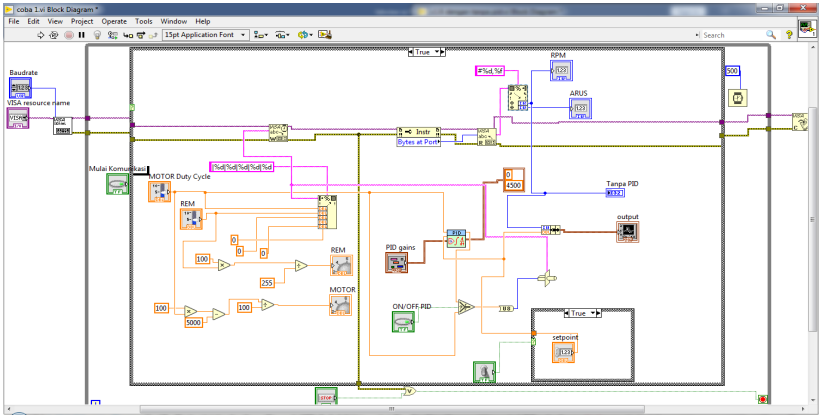
No	*Input* Potensio Motor (*Duty Cycle* %)	Kecepatan Mtr BLDC (tachometer) RPM	Kecepatan Mtr BLDC (*Encoder*) RPM
1	10	423	418
2	20	524	516
3	30	1476	1428
4	40	1858	1831
5	50	1944	1916
6	60	2057	2003
7	70	2155	2134
8	80	2357	2325
9	90	2456	2429
10	100	2630	2605

Untuk melakukan pengujian terhadap data tabel diatas ini yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, kemudian di tampilkan di LCD, kemudian dengan menggunakan Tachometer dengan mengarahkan ke piringan yg terdapat pada modul tersebut.

4.3 Pengujian *Software*

Pengujian *software* dilakukan untuk memastikan program-program perangkat lunak yang sudah dibuat dapat bekerja sesuai dengan yang diharapkan dan dapat bersinergi dengan *hardware* untuk menjalankan sistem sesuai dengan yang direncanakan. Diantaranya, dari pengujian *software* ini yaitu berupa pengujian melalui Lab*View* dari segi mengatur kecepatan sampai dengan melihat grafik respon yang dihasilkan sensor kecepatan yang sudah diberi dengan kontroler PID.

4.3.1 Kontroler PID Lab*View*



Gambar 4. 12 Blok Diagram Kontroler PID Lab*View*

Pada Gambar 4.12 menunjukkan bagaimana model dari kontroler PID yang dijalankan pada Proyek Akhir ini.

4.3.2 Pengujian Motor *Brushless* DC Dengan Kontroler PID

Pengujian motor *brushless* DC dengan kontroler PID ini dimaksudkan untuk menunjukkan apakah alat sudah berjalan sesuai fungsinya.

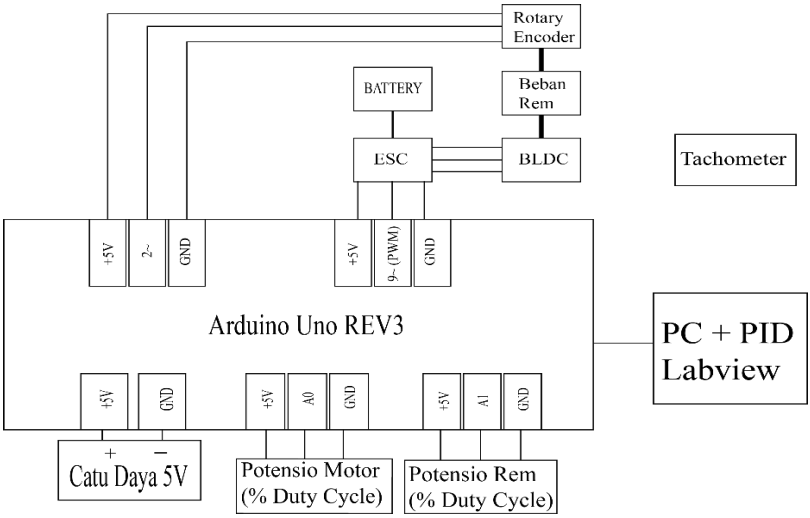
Pengujian ini dilakukan dengan melihat respon yang diberikan oleh motor *brushless* DC ketika motor *brushless* DC diberi *input* dari *set point* yang telah ditentukan. Pengujian ini juga memperlihatkan bagaimana pengaruh kontroler PID pada kecepatan putaran motor *brushless* DC.

Pada Proyek Akhir ini, motor *brushless* DC akan dikopel dengan *rotary encoder* yang berfungsi untuk mengukur seberapa banyak putaran yang dihasilkan oleh motor tersebut dalam suatu waktu. Banyaknya putaran yang dihasilkan oleh motor *brushless* yang dilakukan tersebut akan ditampilkan pada Lab*View*.

Oleh karena Proyek Akhir ini nantinya akan dijadikan sebagai modul praktikum, maka selain ditampilkan pada Lab*View* yang letaknya ada pada praktikan, banyaknya putaran yang dihasilkan oleh motor *brushless* DC.

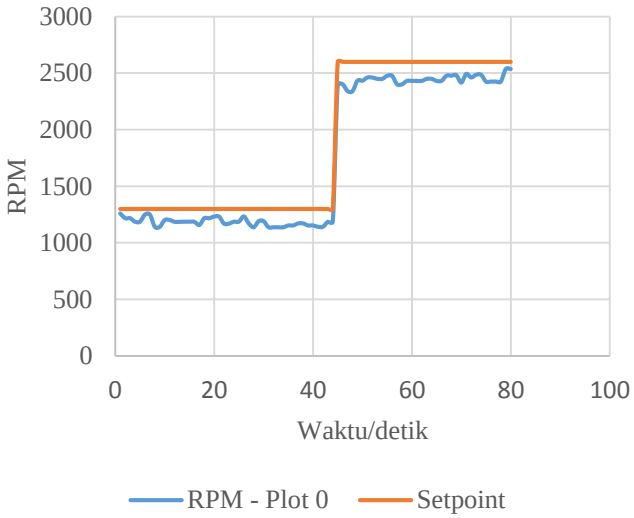
Untuk melakukan pengujian terhadap Gambar 4.13 yaitu dengan cara memprogram sensor kecepatan *Rotary Encoder* di aplikasi Arduino IDE, setelah itu dikomunikasikan ke *LabView* dengan menggunakan komunikasi

Visa pada *Software LabView*. Untuk mengatur kecepatannya nya melalui Lab*View*.



Gambar 4. 13 Rangkaian Pengujian Kecepatan Motor Dengan PID

Untuk melakukan pengujian terhadap Gambar 4.13 yaitu dengan menghidupkan Arduino Uno melalui PC kemudian nyalakan *battery* untuk mengaktifkan ESC agar dapat berjalan, setelah itu jalankan program Lab*View* untuk mengatur *Duty Cycle*, setelah itu jalankan motor *BLDC* melalui control yang sudah di *setting* pada program Lab*View*, nanti hasil akan keluar pada indikator grafik yang sudah di program berdasarkan apa yg telah di baca pada sensor *rotary encoder*, kemudian *setting* motor yang berisi PID agar kecepatan berputar dengan stabil.



Gambar 4. 14 Kecepatan Motor Dengan Kontroler PID

Pada Gambar 4.14 menunjukkan hasil dari respon menggunakan kontroler PID. Kemudian dilakukan pengujian pada kecepatan motor *brush-less* DC.

-----Halaman ini sengaja dikosongkan-----

BAB V
PENUTUP

Bab penutup ini berisi tentang kesimpulan yang diperoleh selama proses pembuatan modul kontrol kecepatan motor *brushless* DC, kesimpulan dari hasil pengujian dan analisa data, serta saran untuk modul kontrol kecepatan motor *brushless* DC ini kedepannya.

5.1. Kesimpulan

1. Sensor *rotary encoder* yang digunakan dalam kondisi *error* pembacaan kurang dari 100 RPM.
2. Hasil pengujian kecepatan putaran motor setelah menggunakan kontroler PID akan menghasilkan kecepatan yang sama dengan *setpoint* kecepatan yang diberikan. Misalnya dalam Proyek Akhir ini pada *setpoint* dari program Arduino 65 adalah sebesar 1300 RPM.
3. Parameter kontroler PID yang digunakan pada Proyek Akhir ini yaitu K_p=1 T_i=0 T_d=0.

5.2 Saran

1. Diharapkan kedepannya dapat memilih komponen-komponen yang dibutuhkan dengan lebih tepat. Karena selama perancangan alat yang lalu masih sering terjadi alat berfungsi tidak maksimal dikarenakan komponen penyusunnya kurang tepat.
2. Keseluruhan alat baik penyangga maupun tatanan lebih baik dibuat dari bahan yang kuat dan tahan terhadap getaran, agar sewaktu dijalankan tidak menggangu kecepatan putar.

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

[1] Hudaibiy Hibban. "*Desain Kontroler Fuzzy PID Gain Schedulling Untuk Pengaturan Kecepatan Motor DC Tanpa Sikat*", **Proyek Akhir**, Jurusan Teknik Elektro FTI-ITS, Surabaya, 2015.

[2] ……, **Allegro A4915 Demo Board**, https://www.youtube.com/watch?v=Ak26nkEpO4I&feature=youtu.be, (Diakses Pada Tanggal 12 February 2019).

[3] Bachtiar, Muhammad Fachri, Priyatna, Alif Gigah. **Perancangan Rem Magnetik Pada Motor DC Dengan Menggunakan Arduino**, Proyek Akhir, D3 Teknik Elektro, ITS Surabaya, 2015.

[4] Mihura, Bruce. **Learning LabView For The First Time**, National Instruments, Washington, 2001.

[5] Ogata, Katsuhiko. **Teknik Kontrol Automatik (Sistem Pengaturan) Jilid 1**, Penerbit Erlangga, Jakarta, 1991.

[6] Pramudijanto, Jos. **Catatan Kuliah Penulisan Ilmiah**, http://www.ee.its.ac.id/~jos, Jurusan Teknik Elektro ITS, 10 Januari 2016.

[7] Hanif Guntoro, **Dasar SCADA**, http://dunia-listrik.blogspot.co.id/2010/02/belajar-dasar-scada.html, 19 Desember 2017.

[8] Intan Nur Robi Annisa dan Zaka Perwira, "*Pembuatan Modul Kontrol Kecepatan Motor Brushless DC dengan Mikrokontroller*", **Proyek Akhir**, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2016.

[9] Wicaksono, Handy. **Programmable Logic Controllers Teori Pemrograman dan Aplikasinya dalam Otomasi Sistem**. Yogyakarta. GrahaIlmu. 2009.

[10] Wicaksono, Handy. **SCADA Software dengan Wonderware In Touch**. Yogyakarta. GrahaIlmu. 2011.

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN A
LISTING PROGRAM

LISTING PROGRAM ARDUINO UNO REV3 *Interface* Lab*View*

```
#include <PWMServo.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <PWM.h>

String myString;
String data1;

int setpoint;
int incomingByte;
int val;
signed int motorlSpeed;
int encoder_pin = 2;
int rpm;
volatile byte pulses;
volatile int rpmcount = 0;
unsigned long timeold;
unsigned int pulsesperturn = 1;
unsigned long duration;
unsigned long rpm2;
char disp2[1];
char disp3[1];

char c;
int Index1, Index2, Index3, Index4, Index5, Index6;
String secondValue, thirdValue, fourthValue, fifthValue, firstValue;

LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7);
PWMServo esc;

void counter()
{
  pulses++;
}
```

```
void setup()
{
  lcd.begin(16, 2);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);
  esc.attach(9);
  Serial.begin(9600);
  pinMode(encoder_pin, INPUT);
  attachInterrupt(0, counter, FALLING);
  pinMode(11, OUTPUT);
  pulses = 0;
  rpm = 0;
  timeold = 0;
}

void parsing()
{

  Index1 = myString.indexOf('|');
  Index2 = myString.indexOf('|', Index1 + 1);
  Index3 = myString.indexOf('|', Index2 + 1);
  Index4 = myString.indexOf('|', Index3 + 1);
  Index5 = myString.indexOf('|', Index4 + 1);
  Index6 = myString.indexOf('|', Index5 + 1);
  secondValue = myString.substring(Index1 + 1, Index2);
  thirdValue = myString.substring(Index2 + 1, Index3);
  fourthValue = myString.substring(Index3 + 1, Index4);
  fifthValue = myString.substring(Index4 + 1, Index5);
  firstValue = myString.substring(Index5 + 1, Index6);
  //Serial.println(Index1);
  //  Serial.print("data 1:"); Serial.println(secondValue);
  //  Serial.print("data 2:"); Serial.println(thirdValue);
  //  Serial.print("data 3:"); Serial.println(fourthValue);
  //  Serial.print("data 4:"); Serial.println(fifthValue);
  //  Serial.print("data 5:"); Serial.println(firstValue);
  myString = "";
}

void loop()
```

```
{
  if (millis() - timeold >= 1000) { /*Uptade every one second, this will be equal to reading frecuency (Hz).*/

    //Don't process interrupts during calculations
    detachInterrupt(0);
    //Note that this would be 60*1000/(millis() - timeold)*pulses if the interrupt
    //happened once per revolution
    rpm = (20 * 1000 / pulsesperturn ) / (millis() - timeold) * pulses;
    timeold = millis();
    pulses = 0;

    //Write it out to serial port
    //Serial.print("RPM = ");
    //Serial.println(rpm, DEC);
    //Restart the interrupt processing
    attachInterrupt(0, counter, FALLING);

    //    int val;
    //    val = analogRead(A0);
    //    setpoint = map(val, 0, 1023, 0, 255);
    //    int percentsetpoint = map(val, 0, 1023, 0, 100);
    //    int percentsetpoint2;
    //Serial.println(setpoint);
    //    esc.write(setpoint);
    //    if (percentsetpoint <= 20)
    //    {
    //      percentsetpoint2 = 0;
    //    }
    //    if (percentsetpoint > 20 && percentsetpoint <= 70)
    //    {
    //      percentsetpoint2 = (percentsetpoint - 20) * 2;
    //    }
    //    if (percentsetpoint > 70)
    //    {
    //      percentsetpoint2 = 100;
    //    }
```

```
//Serial.print(" ");
//sprintf(disp2,"%4d",rpm2);
//Serial.print(disp2);

float average = 0;
for (int i = 0; i < 1000; i++) {
  average = (average + (.0264 * analogRead(A2) - 13.42))/1000;
  delay(1);
}
// Serial.println(average/1000);
//    int sensorValue = analogRead(A1);
//    int rem = map(sensorValue, 0, 1023, 0, 255);
//    analogWrite(11, rem);
//    int percentrem = map(sensorValue, 0, 1023, 0, 100);
Serial.print("#");
Serial.print(rpm);//ganti rpm
Serial.print(",");
Serial.println(average / 1000); //ganti arus
//Serial.println(rem);
while (Serial.available())
{
  delay(3);
  char c = Serial.read();
  myString += c;

}

if (myString.length() > 0)
{
  //Serial.println(readString);
  parsing();

  esc.write(secondValue.toInt());//to esc
  analogWrite(11, thirdValue.toInt());//rem
  Serial.print("#");
  Serial.print(rpm);//ganti rpm
  Serial.print(",");
  Serial.println(average); //ganti arus
  // readString = "";
  //    Serial.print("#");
```

```
    //    Serial.print(data1);
    //    Serial.print(",");
    //    Serial.println(data2);
    //myString = "";
   }
   //    lcd.setCursor (0, 0);
   //    lcd.print ("RPM=");
   //    lcd.print (rpm, DEC);
   //    lcd.setCursor (9, 0);
   //    lcd.print ("Duty=");
   //    lcd.print (percentsetpoint2);
   //    lcd.print (" %");
   //    lcd.setCursor (0, 1);
   //    lcd.print ("REM=");
   //    lcd.print (percentrem);
   //    lcd.print (" %");
   //    lcd.setCursor (9, 1);
   //    lcd.print ("A=");
   //    lcd.print (average / 1000);
   //    delay(10);
  }
}
```

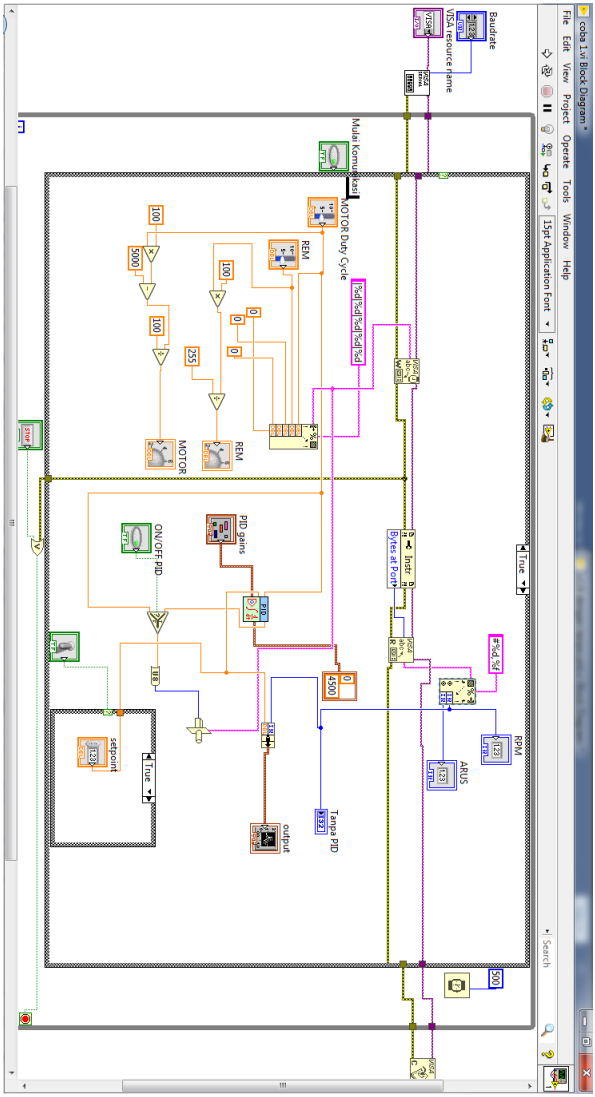
LISTING PROGRAM ARDUINO UNO REV3

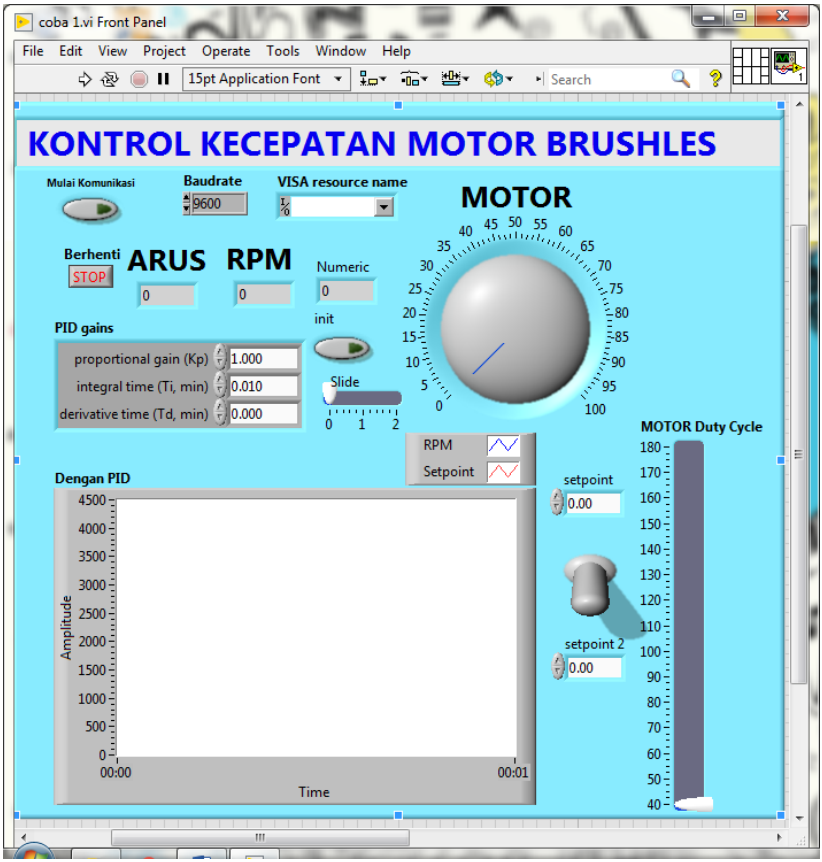
```
#include <PWMServo.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <PWM.h>

int setpoint;
int incomingByte;
int val;
signed int motorlSpeed;
int encoder_pin = 2;
unsigned int rpm;
volatile byte pulses;
volatile int rpmcount = 0;
unsigned long timeold;
unsigned int pulsesperturn = 1;
unsigned long duration;
unsigned long rpm2;
char disp2[1];
char disp3[1];

LiquidCrystal_I2C lcd(0x3F,2,1,0,4,5,6,7);
PWMServo esc;

void counter()
{
  pulses++;
}

void setup()
{
  lcd.begin(16,2);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);
  esc.attach(9);
  Serial.begin(9600);
  pinMode(encoder_pin, INPUT);
  attachInterrupt(0, counter, FALLING);
  pinMode(11,OUTPUT);
```

```
  pulses = 0;
  rpm = 0;
  timeold = 0;
}

void loop()
{
  if (millis() - timeold >= 1000){   /*Uptade every one second, this will be
equal to reading frecuency (Hz).*/

  //Don't process interrupts during calculations
   detachInterrupt(0);
   //Note that this would be 60*1000/(millis() - timeold)*pulses if the interrupt
   //happened once per revolution
   rpm = (20 * 1000 / pulsesperturn )/ (millis() - timeold)* pulses;
   timeold = millis();
   pulses = 0;

    //Write it out to serial port
   //Serial.print("RPM = ");
   Serial.println(rpm,DEC);
   //Restart the interrupt processing
   attachInterrupt(0, counter, FALLING);

  int val;
  val= analogRead(A0);
  setpoint = map(val, 0, 1023, 0, 255);
  int percentsetpoint = map(val, 0, 1023, 0, 100);
  int percentsetpoint2;
  Serial.println(percentsetpoint);
  esc.write(setpoint);
  if(percentsetpoint<=20)
  {
    percentsetpoint2=0;
  }
  if(percentsetpoint>20 && percentsetpoint<=70)
  {
    percentsetpoint2=(percentsetpoint-20)*2;
  }
  if(percentsetpoint>70)
```

```
 {
  percentsetpoint2=100;
 }

 duration=pulseIn(encoder_pin, HIGH, 100000);
 rpm2=(79550000/3)/duration;
 if(rpm2>10000){
  rpm2=0;}
  //Serial.print(" ");
  //sprintf(disp2,"%4d",rpm2);
  //Serial.print(disp2);

 float average = 0;
 for(int i = 0; i < 1000; i++) {
   average = average + (.0264 * analogRead(A2) -13.42);
 delay(1);
}

 int sensorValue=analogRead(A1);
 int rem=map(sensorValue,0,1023,0,255);
 analogWrite(11,rem);
 int percentrem=map(sensorValue,0,1023,0,100);

  lcd.setCursor (0,1);
  lcd.print ("RPM=");
  lcd.print (rpm,DEC);
  lcd.setCursor (0,0);
  lcd.print ("MTR=");
  lcd.print (percentsetpoint2);
  lcd.print ("%");
  lcd.setCursor (9,0);
  lcd.print ("REM=");
  lcd.print (percentrem);
  lcd.print ("%");
  lcd.setCursor (9,1);
  lcd.print ("I=");
  lcd.print (average/1000);
  delay(10);
 }
}
```

BLOK DIAGRAM LAB*VIEW*




coba 1.vi Front Panel
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
KONTROL KECEPATAN MOTOR BRUSHLES
Mulai Komunikasi
Baudrate
9600
VISA resource name
MOTOR
Berhenti
STOP
ARUS
RPM
Numeric
0
0
0
init
PID gains
proportional gain (Kp)
1.000
integral time (Ti, min)
0.010
derivative time (Td, min)
0.000
Slide
RPM
Setpoint
Dengan PID
Amplitude
Time
00:00
00:01
setpoint
0.00
setpoint 2
0.00
MOTOR Duty Cycle

LAMPIRAN B
DATASHEET

1. *Datasheet* Arduino UNO

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 BOARD

The Arduino Uno is one of the most common and widely used Arduino processor boards. There are a wide variety of shields (plug in boards adding functionality). It is relatively inexpensive (about $25 - $35). The latest version as of this writing (3/2014) is Revision 3 (r3):

- Revision 2 added a pull-down resistor to the 8U2 HWB line, making it easier to put into DFU (Device Firmware Update) mode
- Revision 3 added
  - SDA and SCL pins are now brought out to the header near the AREF pin (upper left on picture). SDA and SCL are for the I2C interface
  - IOREF pin (middle lower on picture that allows shields to adapt to the voltage provided
  - Another pin not connected reserved for future use

The board can be powered from the USB connector (usually up to 500ma for all electronics including shield), or from the2.1mm barrel jack using a separate power supply when you cannot connect the board to the PC's USB port.

Links:

- Arduino web site: http://www.arduino.cc/
- Arduino Uno overview and image source: http://arduino.cc/en/Main/arduinoBoardUno#.UxNpBk2YZuG
- DFU Mode (Device Firmware update) explanation: http://arduino.cc/en/Hacking/DFUProgramming8U2#.UxNqXE2YZuE
- Arduino Uno schematic: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf
- Arduino Uno Eagle PCB Files: http://arduino.cc/en/uploads/Main/arduino_Uno_Rev3-02-TH.zip
- Eagle PCB PCB design software (use Licesnse = "Run as Freeware"): https://www.cadsoftusa.com/download-eagle/
- Hardware Index – past and present boards: http://arduino.cc/en/Main/Boards#.UxNq9U2YZuE
- Specifications comparison chart: http://arduino.cc/en/Products.Compare#.UxOJGk2YZuF
- Board comparison chart: http://arduino.cc/en/Products.Compare#.UxN6oE2YZuE
- Sources
  - MP3Car: http://store.mp3car.com/SearchResults.asp?Search=arduino
  - Sparkfun: https://www.sparkfun.com/
  - Adafruit: http://www.adafruit.com/category/17
  - Amazon: http://www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Daps&field-keywords=Arduino
  - Pololu: http://www.pololu.com/search?query=Arduino

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 Specifications

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Uno Board Recommended Input Voltage: 7 – 12 V
- Uno Board Input Voltage Limits: 6 – 20 V
- Digital I/O Pins: 14 total – 6 of which can be PWM
- Analog Input Pins: 6
- Maximum DC Current per I/O pin at 5VDC: 40ma
- Maximum DC Current per I/I pinat 3.3 VDC: 50ma
- Flash Memory: 32KB (0.5KB used by bootloader)
- SRAM Memory: 2KB
- EEPROM: 1KB
- Clock Speed: 16 MHz

Links:

- Arduino specifications and image page: http://arduino.cc/en/Main/arduinoBoardUno#.UxOOLk2YZuH

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 Processor Peripherals (Atmel ATmega 328)

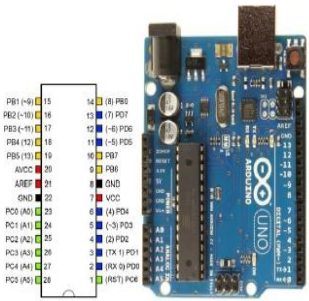
- Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
- One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
- Real Time Counter with Separate Oscillator
- Six PWM channes
- Six channel 10 bit ADC including temperature measurement
- Programmable Serial USART
- Master/Slave SPI Serial Interface
- Byte-oriented 2 wire Serial Interface (Philips I2C compatible)
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator

Links:

- Source of above diagram: http://tekkpinoy.com/wp-content/uploads/2013/10/1.jpg
- AT Mega 328 datasheet: http://www.atmel.com/Images/doc8161.pdf

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 and ATmega328 processor



The Arduino board makes it very easy to use the ATmega328 processor by providing easy access to most of the pins via the headers, In addition, it provides:

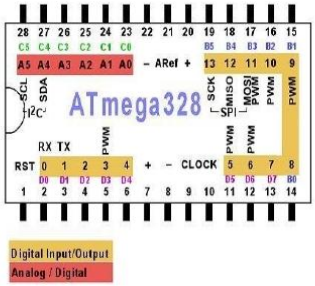
- 5 VDC regulated power from the 6 – 20 VDC input jack
- 3.3 VDC regulated power available for other electronics
- The crystal oscillator
- A reset switch
- USB access to the serial port
- Headers for connection and for shields

Links:

- Arduino specifications and image page: http://arduino.cc/en/Main/arduinoBoardUno#.UxOOLk2YZuH
- ATmega328 processor image modified from image found at: http://www.protostack.com/microcontrollers/atmega328-pu-atmel-8-bit-32k-avr-microcontroller

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 Processor Pinout (Atmel ATmega 328) – Commonly Used



Pin Definition

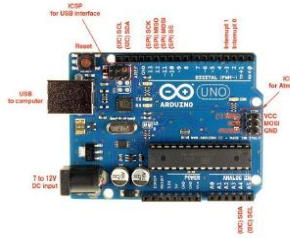
- PORT B (PB0 – PB7) is an 8 bit bidirectional I/O port with internal pull-ups. Processor pins 14 – 17 bring PB0 to PB5 out
  - PB0 – PB5 are also interrupts 0-5 respectively
  - PB1 can also be used as a PWM output
  - PB2 can also be SPI Bus Master Slave Select (*SS) or PWM output
  - PB3 can also be or SPI Bus Master Out/Slave In (MOSI) or PWM output
  - PB4 can also be SPI Bus Master In/Slave Out (MISO)
  - PB5 can also be SPI Bus Master Clock Input (SCK)
  - PB6 and PB7 are brought out on Processor pins 9 and 10 for the crystal clock oscillator
- PORT C (PC0 – PC5) is a 7 bit bidirectional I/O port with internal pull-up resistors. Processor pins 23 – 28 bring PC0 to PC5 out.
  - PC0 – PC5 are also interrupts 8-13 respectively
  - PC0 – PC5 can also be used as A/D inputs
  - PC4 and PC5 can also be used as SDA and SCL for I2C
  - PC6 is brought out on processor pin 1 as reset
- PORT D (D0 – D7) is an 8 bit bidirectional I/O port with internal pull-ups. Processor pins 2 – 6 and 11 – 13 bring all pins out
  - PD0 can also be USART Input (RXD)
  - PD1 can also be USART Output (TXD)
  - PD3 can also be used as a PWM output
  - PD5 can also be used as a PWM output
  - PD6 can also be used as a PWM output

Links:

- Source of above diagram: http://www.hobbytronics.co.uk/arduino-atmega328-pinout
- AT Mega 328 datasheet: http://www.atmel.com/Images/doc8161.pdf

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

ARDUINO UNO Revision 3 Pinout (Uno PCB) – Commonly Used Features are printed on Silkscreen



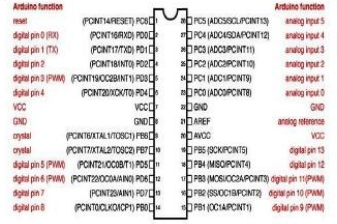
The Arduino Uno pinout is printed in the silkscreen on the top of the part. While this pinout is a good start, it does not explain the complete story – but it does give a good beginning. At first you use mainly the pins in the female headers at the edge of the board (top and bottom in the photo), plus USB and maybe power

- Tx and Rx are serial UART pins used for RS-232 and USB communications
- I2C is another serial communications method using a bidirectional data line (SDA) and a clock line (SCL)
- SPI is another serial communications method using one line for the master to transmit (MOSI – Master Out Slave In), another for the master to receive (MISO), and a third as a clock (SCK)
- A/D in Analogue to Digital this input converts an analogue voltage in to a digital representation
- PWM (Pulse Width Modulator) is used to create a square wave with a specific duty cycle (high time vs low time)
- ICSP is the In Circuit Serial Programming – another way to program the processor
- Vcc is the voltage supplied to the processor (+5VDC regulated from the higher input voltage)
- 3.3VDC is a regulated voltage (from the higher input voltage) for peripherals needing that voltage – 50ma maximum
- IOREF provides a voltage reference so shields can select the proper power source
- AREF is a reference INPUT voltage used by the A/Ds
- GND is the ground reference
- RESET resets the processor (and some peripherals)

Links:

- Source of above diagram: http://www.adafruit.com/blog/2012/05/25/handy-arduino-r3-pinout-diagram/
- Description of pin usage: http://www.gammon.com.au/forum/?id=11473
- Arduino Uno Pin Mapping: http://arduino.cc/en/Hacking/PinMapping168#.UxOJik2YZuE
- Description of Arduino Serial: http://arduino.cc/en/reference/serial#.UxOMKk2YZuE
- Description of the Arduino SPI functions and library: http://arduino.cc/en/Reference/SPI#.UxOPLk2YZuE
- Description of Arduino A/D: http://arduino.cc/en/Tutorial/AnalogInputPins#.UxOM7k2YZuE
- Description of Arduino PWM: http://arduino.cc/en/Tutorial/PWM#.UxOLz02YZuE
- Tutorial on ISP: http://arduino.cc/en/Tutorial/ArduinoISP#.UxOUSk2YZuE
- Tutorial on the AREF pin: http://tronixstuff.com/2013/12/12/arduino-tutorials-chapter-22-aref-pin/

ARDUINO UNO Revision 3 Processor Pinout (Atmel ATmega 328) – Other functions



Pin Definition

- PORT B pins, in addition to digital I/O have other uses
  - PB0 can also be the divided system clock output (CLKO) or Timer/Counter 1 Input Capture (ICP1)
  - PB1 can also be Timer/Counter1 Output Compare Match A (OC1A) out
  - PB2 can also be Timer/Counter1 Output Compare Match B (OC1B)
  - PB3 can also be Timer/Counter2 Output Compare Match A out(OC2A)
- Port D pins, in addition to digital I/O have other uses
  - PD3 is also Timer/Counter2 Output Compare Match B Output (OC2B)
  - PD4 is also Timer/Counter0 External Counter Input (T0) or USART External Clock Input/Output (XCK)
  - PD5 is also Timer/Counter0 Ouoput Compare Match B Output (OC0B) and Timer/Counter 1 External Counter Input
  - PD6 can also be Analog Comparator Positive In (AIN0)
  - PD7 can also be Analog Comparator Negative In (AIN1)

Links:

- Source of above diagram: http://nearbus.net/wiki/index.php?title=Atmega_328_Pinout
- AT Mega 328 datasheet: http://www.atmel.com/Images/doc8161.pdf

NOTE: A single diagram showing all features of the Arduino Uno and the Atmel ATMega328 processor is shown in Appendix A

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

APPENDIX A

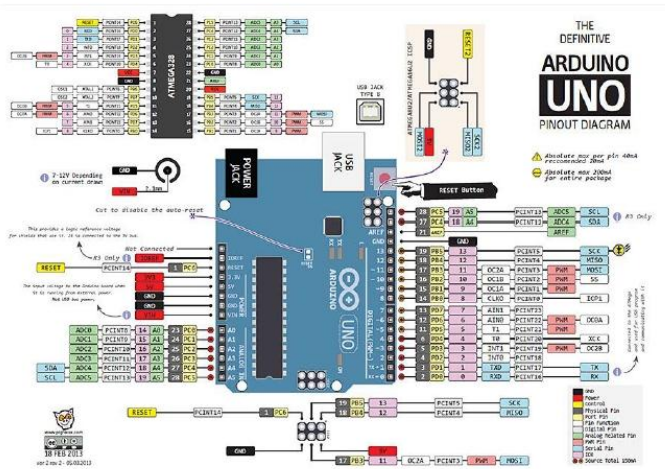


Diagram from: http://arduino-info.wikispaces.com/file/view/ArduinoUNO-900.jpg/421496636/ArduinoUNO-900.jpg

APPENDIX B

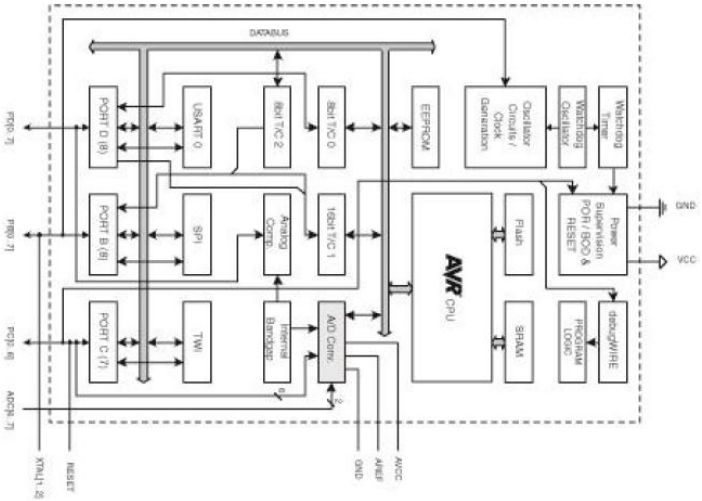


Diagram from document at: http://www.atmel.com/Images/doc8161.pdf

AIAA OC Rocketry (Revision 3 April 27, 2014 - http://aiaaocrocketry.org)

APPENDIX C

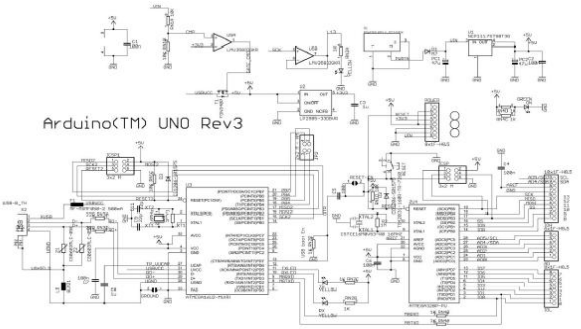
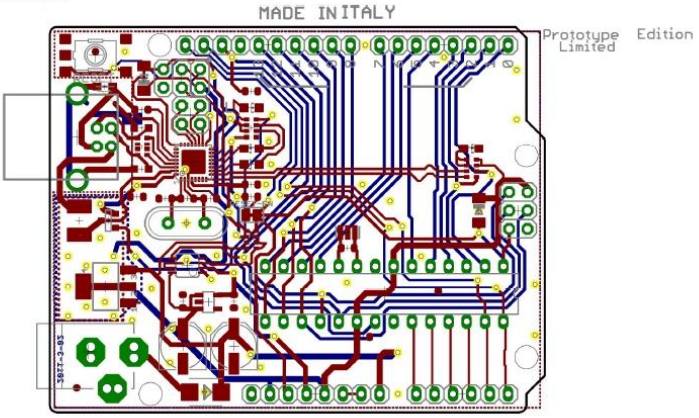


Diagram from document at: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf



From arduino_Uno_Rev3-02-TH.zip file at http://arduino.cc/en/Main/ArduinoBoardUno#.Uxk9qk2YYpA

Eagle PCB software: Eagle PCB PCB design software (use Licesnse = "Run as Freeware"): https://www.cadsoftusa.com/download-eagle/

2. *Datasheet* Driver Motor RCTimer ESC 30 A

Manual of RC Timer ESC 30A Brushless Motor Speed Controller

Thanks for purchasing **RC Timer** Electronic Speed Controller (ESC). High power system for RC model can be very dangerous, so we strongly suggest you read this manual carefully. In that we have no control over the correct use, installation, application, or maintenance of our products, no liability shall be assumed nor accepted for any damages, losses or costs resulting from the use of the product. Any claims arising from the operating, failure or malfunctioning etc. will be denied. We assume no liability for personal injury, property damage or consequential damages resulting from our product or our workmanship. As far as is legally permitted, the obligation to compensation is limited to the invoice amount of the affected product.

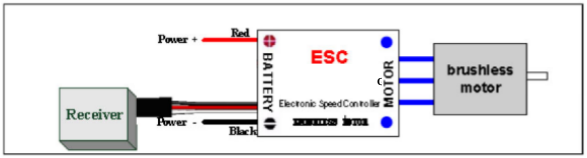
Specification

Input voltage: DC 6-16.8V(2-4S Lipo)
Running current:30A(Output: Continuous 30A, Burst 40A up to 10 Secs.)
Size: 26mm (L) * 23mm (W) * 11mm (H).
Weight: 32g.

Features

- Extreme low output resistance, super current endurance.
- Multiple protection features: Low-voltage cut-off protection / over-heat protection / throttle signal loss protection.
- 3 start modes: Normal / Soft / Super-Soft, compatible with fixed-wing aircraft and helicopter.
- Throttle range can be configured to be compatible with all transmitters currently available on market.
- Smooth, linear and precise throttle response.
- Separate voltage regulator IC for microprocessor (except Pentium-6A and Pentium-10A), providing good anti-jamming capability.
- Supported motor speed (Maximum): 210000 RPM (2 poles), 70000 RPM (6 poles), 35000 RPM (12 poles).

Wire Diagram



Very important: If you use banana-shape connectors on main power wires (input wires), please connect the black wire (negative polarity) BEFORE red wire (positive polarity). So the right

sequence is: BDMP Adapter → BLACK wire of main power → RED wire of main power

Programmable Items:

1. Brake Setting：Enabled / Disabled, default is Disabled

2. Battery Type：Li-xx(Li-ion or Li-poly) / Ni-xx(NiMH or NiCd)，default is Li-xx.

3. Low Voltage Protection Mode(Cut-Off Mode)： Soft Cut-Off (Gradually reduce the output power) or Cut-Off (immediately stop the output power). Default is Soft Cut-Off.

4. Low Voltage Protection Threshold(Cut-Off Threshold)：Low / Medium / High, default is Medium.

1) For lithium batteries, the number of battery cells is calculated automatically. Low / medium / high cutoff voltage for each cell is: 2.6V/2.85V/3.1V. For example: For a 3 cells lithium pack, when "Medium" cutoff threshold is set, the cut-off voltage will be: 2.85*3=8.55V.

2) For nickel batteries, low / medium / high cutoff voltages are 0%/45%/60% of the startup voltage (i.e. the initial voltage of battery pack), and 0% means the low voltage cut-off function is disabled. For example: For a 10 cells NiMH battery, fully charged voltage is 1.44*6=8.64V, when "Medium" cut-off threshold is set, the cut-off voltage will be:8.64*50%=4.3V。

5. Startup Mode：Normal /Soft /Super-Soft, default is Normal.

Normal is preferred for fixed-wing aircraft. Soft or Super-soft are preferred for helicopters. The initial acceleration of the Soft and Super-Soft modes are slower in comparison, usually taking 1 second for Soft startup or 2 seconds for Super-Soft startup from initial throttle advance to full throttle. If the throttle is closed (throttle stick moved to bottom) and opened again (throttle stick moved to top) within 3 seconds of the initial startup, the restart-up will be temporarily changed to normal mode to get rid of the chances of a crash caused by slow throttle response. This special design is very suitable for aerobatic flight when quick throttle response is needed.

6. **Timing**：Low / Medium / High, default is Low.

Usually, low timing value can be used for most motors. We recommend the Low timing value for 2 poles motor and Medium timing value for motors with more than 6 poles to get a high efficiency. For higher speed, High timing value can be chosen.

Special Note

Some high KV out-runner motors have very special construction, the space between each magnet is very large, and many ESCs can't drive these motors. After much testing, our ESCs have proven to work very well with these types of motors. Therefore, we have provided some suggestions as follows:

Programmable / Motor	Timing	Startup mode
Generic in-runner motor	Low	Usually,aircraft use "Normal" startup mode and helicopter use"super-soft" startup mode
Generic out-runner motor	Low or Medium	
Align 420LF（Made in TAIWAN，out-runner）	High（MUST）	
450TH（Made in TAIWAN）	Low	Soft(MUST

Begin To Use Your New ESC

Please start the ESC in the following sequences:

1. Move the throttle stick to the bottom position and then switch on the transmitter.
2. Connect the battery pack to the ESC, the ESC begins the self-test process, a special tone " 123" is emitted, which means the voltage of the battery pack is in normal range, and then N "beep" tones will be emitted, means the number of lithium battery cells.Finally a long "beep------" tone will be emitted, which means self-test is OK, the aircraft/

 If a special tone " 56712" is emitted after 2 beep tones ("beep-beep-"), means the ESC has entered the program mode, it is because the throttle channel of your transmitter is reversed, please set it correctly;

Program the ESC with your transmitter (4 Steps):

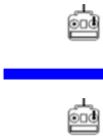
1. Enter program mode
2. Select programmable items
3. Set item's value (Programmable value)
4. Exit program mode

1. Enter program mode

1) Switch on transmitter, move throttle stick to top , connect the battery pack to ESC

2) Wait for 2 seconds, the motor should emit special tone like "beep-beep-"

3) Wait for another 5 seconds, special tone like "56712 " should be emitted, which means program mode is entered



2. Select programmable items:

After entering program mode, you will hear 8 tones in a loop with the following sequence. If you move the throttle stick to bottom within 3 seconds after one kind of tones, this item will be selected.

1. "beep" brake (1 short tone)
2. "beep-beep-" battery type (2 short tone)
3. "beep-beep-beep-" cutoff mode (3 short tone)
4. "beep-beep-beep-beep-" cutoff threshold (4 short tone)
5. "beep-----" startup mode (1 long tone)
6. "beep-----beep-" timing (1 long 1 short)
7. "beep-----beep-beep-" set all to default (1 long 2 short)
8. "beep-----beep-----" exit (2 long tone)

Note: 1 long "beep-----" = 5 short "beep-"

3. Set item value (Programmable value):

You will hear several tones in loop. Set the value matching to a tone by moving throttle stick to top when you hear the tone, then a special tone "1515 " emits, means the value is set and saved. (Keeping the throttle stick at top, you will go back to step 2 and you can select other items; Moving the stick to bottom within 2 seconds will exit program mode directly)

Tones / Items	"beep-" 1 short tone	"beep-beep-" 2 short tones	"beep-beep-beep" 3 short tones
Brake	off	on	
Battery type	LI-ion / LI-poly	NIMH / NICd	
Cutoff mode	Soft-Cut	Cut-off	
Cutoff threshold	Low	Medium	High
Start mode	Normal	Soft	Super soft
Timing	Low	Medium	High

4. Exit program mode

There are 2 ways to exit program mode:

1. In step 3, after special tone "1515 ", please move throttle stick to the bottom position within 2 seconds.

2. In step 2, after tone "beep-----beep-----"(ie. The item #8), move throttle stick to bottom within 3 seconds.

-- Halaman ini sengaja dikosongkan --

LAMPIRAN C
DOKUMENTASI

-- Halaman ini sengaja dikosongkan –

RIWAYAT PENULIS

Nama : Baharuddin Baharsyah
TTL : Banyuwangi, 23 Oktober 1996
Jenis Kelamin : Laki - Laki
Agama : Islam
Alamat Asal : Perum Berlian Citra Kertanegara A-19, Kebalenan, Banyuwangi
Telp/HP : 089665223215
E-mail : *baharsyah1623@gmail.com*

RIWAYAT PENDIDIKAN

- *2003 – 2009 : MI At-Taqwa Bondowoso*
- *2009 – 2012 : MTS Assalaam Sukoharjo*
- *2012 – 2015 : SMAN 1 Glagah Banyuwangi*
- *2015 – Sekarang : Departemen Teknik Elektro Otomasi Institut Teknologi Sepuluh Nopember.*

PENGALAMAN KERJA

- *Kerja Praktek PT PLN (Persero) Distribusi Jawa Timur Rayon Rungkut (Juni 2016 – Juli 2017)*
- *Kerja Praktek PT. Indocrane Surabay (Juli 2017 – Agustus 2017)*

PENGALAMAN ORGANISASI

- *Staff ITS Billiard 16/17*
- *Staff Ahli ITS Billiard 17/18*
- *KaBiro Pemetaan PSDM HIMAD3TEKTRO 17/18*
- *Ketua Departemen KOMINFO UKM Sepakbola ITS 17/18*
- *Ketua BEM Fakultas Vokasi 17/18*

--- Halaman ini sengaja dikosongkan ---