SISTEM PENGONTROLAN GERAK *HOVER TRICOPTER* BERBASIS MIKROKONTROLER

SKRIPSI

KONSENTRASI TEKNIK KONTROL

Diajukan Untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Teknik

DISUSUN OLEH:

YONETH YONATHAN

NIM. 0710633044-63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2012

SISTEM PENGONTROLAN GERAK *HOVER TRICOPTER* BERBASIS MIKROKONTROLER

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

YONETH YONATHAN

NIM. 0710633044-63

Mengetahui dan menyetujui,
Dosen pembimbing :

Retnowati, Ir., MT.	**Tri Nurwati, ST., MT.**
NIP. 19511224 198203 2 001	**NIP. 19790615 200812 2 003**

LEMBAR PENGESAHAN

SISTEM PENGONTROLAN GERAK *HOVER TRICOPTER* BERBASIS MIKROKONTROLER

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

YONETH YONATHAN
NIM. 0710633044-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 13 Agustus 2012

Majelis Penguji

Fitriana Suhartati, ST., MT.	**Erni Yudaningtyas, Dr., Ir., MT.**
NIP. 19741017 199802 2 001	**NIP. 19650913 199002 2 001**

M. Aziz Muslim, ST., MT., PhD
NIP. 19741203 200012 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Ir. Sholeh Hadi Pramono, MS.
NIP. 19580728 198701 1 001

PENGANTAR

Puji dan syukur penulis panjatkan kepada Allah SWT atas limpahan rahmat dan karuniaNya, penulis dapat menyelesaikan skripsi yang berjudul "Sistem Pengontrolan Gerak *Hover Tricopter* Berbasis Mikrokontroler". Skripsi ini disusun sebagai persyaratan untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan bebagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada :

- Ibuku Susiati, Ayahku Purwanto, dan Mbah Kus atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Kakakku Radita Pravita Sari yang banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Bapak Dr. Sholeh Hadi Pramono, Ir., MT selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak M. Aziz Muslim, ST., MT, Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Purwanto Ir., MT selaku Ketua Kelompok Dosen Keahlian Sistem Kontrol Jurusan Teknik Elektro Universitas Brawijaya,
- Ibu Retnowati, Ir., MT sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, gagasan, ide, saran serta motivasi yang telah diberikan,
- Ibu Tri Nurwati, ST., MT sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, saran, kritik, dan masukan yang telah diberikan,
- Bapak Bambang Siswoyo, Ir., MT. sebagai Dosen Penelitian Bersama atas masukan yang telah diberikan.
- Staf Rekording, staf Pengajaran, dan staf Ruang Baca Jurusan Teknik Elektro yang telah membantu segala urusan penulis selama ini,
- Teman-teman yang menemani pengerjaan skripsi Oji, Enos, dan Adit yang selalu menemani disaat susah maupun senang,
- Sahabat-sahabat penulis Kas, Aboed, dan Eres yang selalu memberi semangat, bantuan dan saran dalam pengerjaan skripsi,
- Teman-teman CORE angkatan 2007 yang telah berbagi ilmu dengan penulis dan selalu memberikan semangat,

- Teman-teman geng belakang rumah sakit yang telah banyak memberi pengalaman yang tak terlupakan.
- Penghuni kos MT Haryono 9B 330,
- Teman-teman Laboratorium Siskon, Rio 08 dan Arif Blontang 08 yang sangat membantu skripsi ini.
- Seluruh teman-teman, senior serta semua pihak yang tidak mungkin untuk dicantumkan namanya satu-persatu, terima kasih banyak atas bantuan dan dukungannya.

Penulis menyadari bahwa tugas akhir ini masih belum sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis berharap, semoga tugas akhir ini bermanfaat bagi kita semua.

Malang, Juli 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Yoneth Yonathan**,** Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya,, Sistem Pengontrolan Gerak *Hover Tricopter* Berbasis Mikrokontroler, Dosen Pembimbing: Retnowati, Ir., MT dan Tri Nurwati, ST., MT.

Tricopter merupakan salah satu jenis kegiatan *aeromodelling*. *Tricopter* masuk dalam jenis *multicopter* yaitu pesawat model dengan jumlah *rotor* banyak. *Tricopter* mempunyai tiga buah mesin yang berupa motor *brushless* dan sebuah motor servo agar dapat terbang *hover* di udara. Faktor yang sangat penting dari *tricopter* adalah pengontrolan otomatis. Tanpa adanya pengontrolan otomatis, *tricopter* akan sulit dikendalikan ketika *hover* . Dalam skripsi ini, pengontrolan otomatis *tricopter* menggunakan Mikrokontroler ATMega168. Sensornya menggunakan *gyro* ENC-03M digunakan untuk mendeteksi gerakan dari *tricopter* saat gerak *hover*. *Tricopter* ini menggunakan 3 sensor *gyro*. Sensor *gyro pitch* dan sensor *gyro roll* umpan balik ke ke mikrokontroler untuk mengontrol motor *brushless.* Sedangkan sensor *gyro yaw* untuk mengontrol motor servo. Saat pengujian *board* mikrokontroler diberi gerakan *pitch, roll,* dan *yaw* , terjadi kesesuaian antara *input* yang berupa gerakan *pitch, roll* , dan *yaw* yang diberikan dengan *output* yang dihasilkan. Hasil pengujian untuk respon sensor *pitch, error steady state*-nya sebesar 0,74%, respon sensor *roll, error steady state*-nya sebesar 1,48% dan respon sensor *yaw, error steady state*-nya sebesar 2.22%. Parameter PID pada pengontrolan *tricopter* menggunakan metode *trial and error* dengan parameter Kp=50%, Ki=60%, dan Kd=50% dari tuas potensiometer *board* KKmulticopter *tricopter*. Dari hasil pengujian keseluruhan tersebut memiliki respon yang baik.

Kata kunci: mikrokontroler Atmega 168, sensor *gyro* ENC-03M, PID

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi di dunia yang semakin pesat ini membuat manusia semakin kreatif dalam membuat sesuatu. Termasuk munculnya hobi-hobi yang berhubungan dengan teknologi, salah satu contoh adalah miniatur pesawat yang dibuat seperti bentuk aslinya. Ditambah dengan begitu pesatnya perkembangan di bidang elektronika tentang sinyal-sinyal radio kontrol, maka diciptakanlah miniatur pesawat yang dapat digerakkan dengan menggunakan tenaga listrik yang salah satunya adalah kegiatan *aeromodelling*.

Aeromodelling adalah suatu kegiatan yang mempergunakan sarana miniatur (model) pesawat terbang untuk tujuan rekreasi, edukasi, dan olahraga. Dalam kegiatan *aeromodelling*, pengontrolan otomatis dan penentuan komponen yang tepat dari pesawat model merupakan faktor yang sangat penting dalam pembuatan pesawat model. Tanpa faktor tersebut, akan sangat sulit mengendalikan pesawat model yang bukan tidak mungkin dapat menyebabkan kecelakaan yang dapat berakibat kerusakan pada pesawat model.

Tricopter juga merupakan salah satu jenis kegiatan *aeromodelling*, *tricopter* masuk dalam jenis *multicopter* yaitu pesawat model dengan jumlah *rotor* banyak. *Tricopter* mempunyai tiga buah mesin yang berupa motor *brushless* untuk menggerakkan masing-masing *propeller* (baling-baling). Pada bagian ekor *tricopter* terdapat motor servo yang berfungsi untuk mengatur gerakan *yaw* (memutar dalam bidang horizontal). Hal yang diinginkan dari *tricopter* adalah harus dapat *hover* (melayang) di udara dengan control secara otomatis, karena akan sulit mengontrol *tricopter* agar *hover* hanya dengan menggunakan *tuning* manual dari *radio control*. *Hover* merupakan gerak terbang ke arah vertikal dengan kecepatan tertentu. Tanpa adanya pengontrolan otomatis, *tricopter* tidak bisa *hover* stabil. Penyebab *tricopter* sulit *hover* (melayang) karena *tricopter* mempunyai tiga motor yang ketika terbang memperoleh besar gangguan yang berbeda-beda.

Sistem terbang dari *tricopter* itu sendiri adalah adanya gaya angkat yang dihasilkan dari ketiga *propeller* yang digerakkan oleh masing-masing motor. Dalam *tricopter* ini terdapat sensor *gyro* yang berfungsi membaca perubahan gerakan *tricopter* yang kemudian mengirimkan sinyalnya ke mikrokontroler agar mengontrol putaran masing-masing motor serta sudut motor servo agar *tricopter* dapat *hover* (melayang) dengan baik.

Berdasarkan permasalahan di atas maka skripsi ini akan dirancang sebuah *tricopter* dengan pengontrolan otomatis sehingga dapat *hover* (melayang) dengan baik di udara, agar

memudahkan operator dalam mengendalikan *tricopter* sehingga mengurangi resiko jatuhnya *tricopter* ketika terbang di udara.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya , maka rumusan masalah dalam skripsi ini ditekankan pada:

1. Bagaimana merancang bentuk mekanik *tricoper* serta pemilihan komponen elektronika *tricopter* agar mampu melakukan gerakan *hover* (melayang) di udara?
2. Bagaimana memanfaatkan pemrograman Mikrokontroler ATMega168 dengan menggunakan program CV-AVR sehingga dapat digunakan sebagai pengontrolan *tricopter* dengan menggunakan kontroler PID?

1.3 Batasan Masalah

Untuk menekankan pada objek pembahasan yang ada maka penelitian ini diberikan batasan masalah sebagai berikut:

1. *Tricopter* yang dibuat merupakan hasil desain dengan ukuran sesuai perancangan.
2. Pembahasan hanya pada proses pengontrolan *tricopter* untuk gerakan *hover*.
3. Kecepatan motor dalam *tricopter* tidak dibahas mendalam karena pembahasan ditekankan pada pengontrolan gerakan *hover tricopter*.
4. Ketinggian *tricopter* ketika terbang berdasarkan gaya dorong dihasilkan dari ketiga motor.
5. Gangguan berupa distribusi beban yang tidak merata dan perubahan aerodinamis secara alami.
6. Kinerja driver dan elektronika tidak dibahas mendalam.
7. Menggunakan *board* KKmulticopter168 PA.

1.4 Tujuan

Tujuan penyusunan skripsi ini adalah merealisasikan pengontrolan otomatis sebuah *tricopter* agar dapat melakukan gerakan *hover*, dengan demikian dapat memudahkan operator dalam mengendalikan *tricopter* sehingga mengurangi resiko jatuhnya *tricopter* ketika terbang di udara.

1.5 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Menjelaskan dasar teori penunjang penelitian yang ada pada alat ini, yang terdiri dari teori Mikrokontroler ATmega168 sebagai sistem otomatisasi, kontroler PID, sensor *gyro*, motor servo, dan motor *brushless*.

BAB III Metodologi

Berisi tentang metode penelitian dan perencanaan alat serta pengujian.

BAB IV Perancangan Sistem

Membahas perancangan alat yang meliputi spesifikasi, perencanaan blok diagram, prinsip kerja dan pembuatan alat. Setelah itu, bagaimana penerapannya dalam sistem secara keseluruhan.

BAB V Pengujian Alat

Memuat hasil pengujian terhadap alat yang telah dibuat.

BAB VI Kesimpulan dan Saran

Memuat kesimpulan dan saran-saran.

BAB II
TINJAUAN PUSTAKA

Untuk memudahkan dalam memahami cara kerja rangkaian maupun dasar-dasar perencanaan dari sistem yang dibuat, maka perlu adanya penjelasan dan uraian mengenai teori penunjang yang digunakan dalam penulisan skripsi ini. Teori penunjang yang akan dijelaskan dalam bab ini adalah:

- *Tricopter*
- Gaya aerodinamika pesawat
- Motor DC *Brushless*
- Motor servo
- Sensor *Gyro*
- Kontroler PID (*Proportional Integral Derivative*)
- Mikrokontroler ATMega168
- *Radio Control* (RC)

2.1 *Tricopter*

2.1.1 *Multicopter*

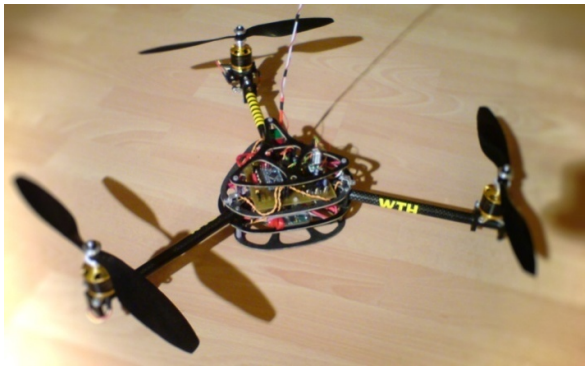
Multicopter (*multirotor*) adalah jenis pesawat dengan lebih dari dua *rotor* . *Multicopter* menggunakan *fixed-pitch blades*, dimana *rotor pitch* tidak bervariasi sebagai *blades rotate*, kontrol gerak kendaraan dicapai dengan memvariasikan kecepatan relatif dari masing-masing *rotor* untuk mengubah tujuan dan torsi yang dihasilkan masing-masing *rotor*.

Prinsip dasar terbang dari pesawat bersayap tetap (*fixed wing*) dengan *multicopter* yang dikenal juga pesawat bersayap putar pada dasarnya sama. Kuncinya ada pada dua kekuatan besar yang bekerja terpadu, menghasilkan gaya angkat dan daya dorong yang besar. Pada *multicopter*, fungsi sayap digantikan oleh *propeller* (baling-baling). Meskipun setiap baling-balingnya berukuran lebih kecil dari sayap pesawat biasa, namun ketika diputar, curvanya relatif sama dengan sayap pesawat. Untuk mendapatkan gaya angkat, baling-baling *rotor* harus diarahkan pada posisi tertentu sehingga dapat membentuk sudut datang yang besar. Aliran udara yang bergerak ke depan baling-baling menekan baling-baling sehingga bilah baling-baling terdorong balik ke belakang menghasilkan suatu gaya angkat kecil. Tetapi ketika aliran udara bergerak cepat melewati bagian atas dan bawah bilah baling-baling, tekanan udara yang besar di antara baling-baling otomatis akan mengembang ke seluruh permukaan yang bertekanan lebih rendah, menyebabkan baling-

baling terdorong ke atas dan *multicopter* pun terangkat. (shrediquette.blogspot.com, 1 juni 2012).

2.1.2. *Tricopter*

Tricopter adalah salah satu jenis *multicopter* yang menggunakan tiga motor dan baling-baling yang diatur dalam segitiga horizontal agar tetap bisa terbang di udara. Kecepatan dari masing-masing motor dapat dikontrol secara terpisah. Selain itu, motor pada bagian belakang bisa diputar ke kiri dan ke kanan dengan menggunakan motor servo. Salah satu baling-baling berputar searah jarum jam sedangkan dua baling-baling lainnya berputar berlawanan arah jarum jam agar meminimalkan torsi. Dalam *tricopter* terdapat sensor *gyro* yang berfungsi mengukur kecepatan sudut digunakan untuk menstabilkan sistem yang tidak stabil. Pilot juga dapat mengontrol *tricopter* dengan menggunakan *remote control*. Gambar 2.1 menunjukkan bentuk salah satu bentuk dari *tricopter* (shrediquette.blogspot.com, 2 juni 2012)



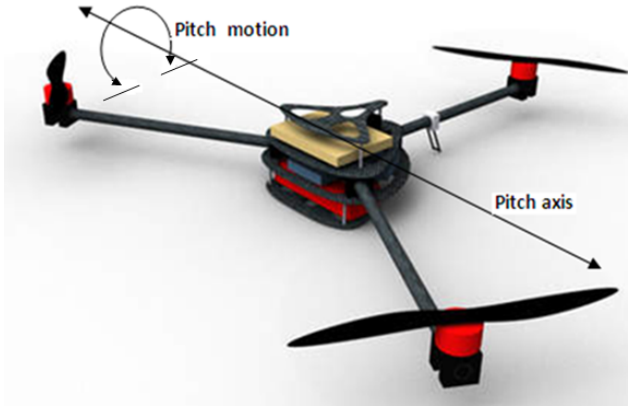
Gambar 2. 1 *Tricopter*

Sumber: www.fuelyourproductdesign.com

Dalam *tricopter* ada beberapa jenis gerakan, gerak-gerak adalah:

1. *Pitch*

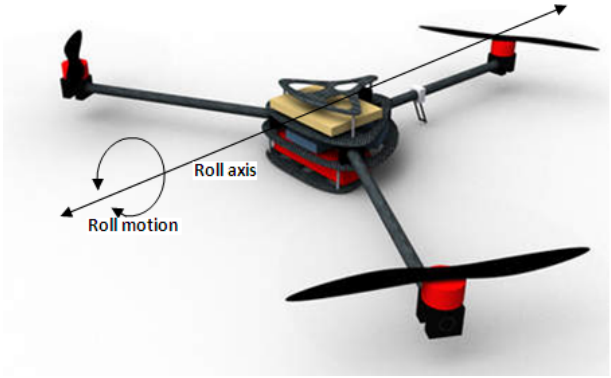
Pitch merupakan gerakan mengganguk atau gerakan ke atas dan ke bawah dari badan *tricopter*, *pitch* bergerak terhadap sumbu *pitch axis*. Untuk dapat melakukan gerakan *pitch*, yaitu dengan mengurangi atau menambah kecepatan motor bagian depan atau belakang . Gambar 2.2 Menunjukkan gerakan *pitch* pada *tricopter*



Gambar 2. 2 Gerakan *Pitch* pada *Tricopter*
Sumber : www.tomsguide.com

2. *Roll*

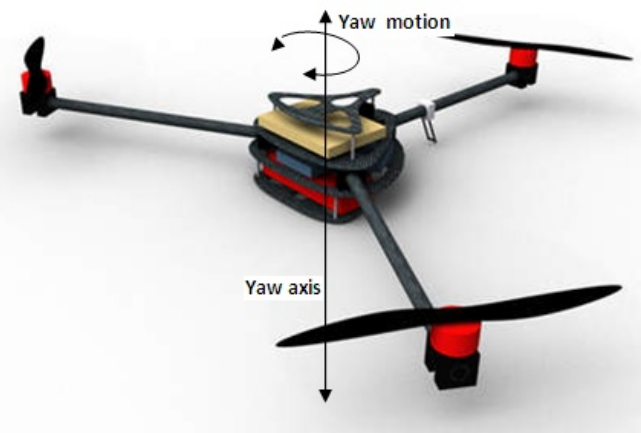
Roll merupakan gerakan memutar (*roll*) dari *tricopter* , *tricopter* bergerak terhadap sumbu *roll axis tricopter*. Gerakkan *roll* (memutar) terjadi karena perbedaan kecepatan motor antara sisi kanan dan sisi kiri motor. Untuk melakukan gerakan *roll* yaitu dengan menambah kecepatan motor pada sisi kiri maupun kanan. Gambar 2.3 Menunjukkan gerakan roll pada *tricopter*.



Gambar 2. 3 Gerakan *Roll* pada *Tricopter*
Sumber : www.tomsguide.com

3. *Yaw*

Yaw merupakan gerakan menggeleng atau *tricopter* bergerak ke kanan dan ke kiri terhadap sumbu *yaw axis* yang terletak pada *centre of gravity*. Gerakan *yaw* terjadi karena sudut motor bagian ekor yang bisa digerakkan miring ke kanan atau ke kiri dengan menggunakan motor servo. Gambar 2.4 Menunjukkan gerakan *yaw* pada *tricopter*.



Gambar 2. 4 Gerakan *Yaw* pada *Tricopter*
Sumber : www.tomsguide.com

2.2 Gaya aerodinamika pesawat

Gaya aerodinamika dapat muncul sebagai akibat dari aliran udara pada suatu permukaan dari suatu benda seperti pesawat, mobil, kereta api, helicopter dan sebagainya. Gaya-gaya aerodinamik dapat muncul karena adanya distribusi tekanan yang berbeda-beda pada permukaan dan tegangan geser pada permukaan yang berasal dari efek fluida yang melawan bidang permukaan benda.

Ada 4 gaya utama pada pesawat terbang:

1. *Thrust* adalah gaya dorong yang dihasilkan oleh mesin atau baling-baling.
2. *Drag* disebabkan oleh gangguan angin oleh sayap, *fuselage* (badan pesawat), dan objek-objek lain. *Drag* kebalikan dari *thrust*.
3. *Weight* (gaya berat) adalah kombinasi berat dari muatan. *Weight* menarik pesawat ke bawah karena gaya gravitasi. *Weight* melawan *lift* (gaya angkat) dan bereaksi secara *vertical* ke bawah melalui *center of gravity* dari pesawat.
4. *Lift* (gaya angkat) melawan gaya dari *weight*.

(http://panjicero.wordpress.com).

2.3 Motor DC Brushless

Motor DC *brushless* mempunyai 4 bagian dasar yaitu : rotor, stator, komutator elektronik dan sensor posisi. Pada motor DC konvensional, medan magnet stationer dibangkitkan oleh sebuah magnet permanen atau elektromagnetik. Medan ini disebut stator karena sifatnya stationer, sedangkan catu daya DC (*direct current*) diberikan pada *armature* (angker dinamo) yang bebas berputar. Pada motor DC *brushless* justru medannya yang berputar berupa magnet permanen dan medan ini disebut sebagai rotor. Sedangkan catu daya DC diberikan pada armature atau stator.

Rotor pada motor DC *brushless* berupa permanen magnet yang membentuk busur. Digunakannya permanen magnet pada motor ini bertujuan untuk memberikan efisiensi yang tinggi. Magnet permanen ini diletakkan menempel pada bagian permukaan motor. Hal ini dilakukan sehubungan dengan usaha untuk menghasilkan karakteristik yang mendekati linier. Lilitan stator pada motor DC *brushless* ini dapat disusun secara terkonsentrasi maupun secara terdistribusi. Rangkaian elektronika didesain untuk menghasilkan arus yang membentuk gelombang sinus maupun gelombang persegi. Pilihan konstruksi yang umumnya dipakai di industri adalah lilitan terkonsenstrasi dengan gelombang arus berbentuk persegi dengan alasan sebagai berikut:

1. Kombinasi lilitan terkonsentrasi ini secara teoritis menghasilkan *output power* yang lebih besar untuk arus rms (*root mean square*) yang sama besar.
2. Rangkaian kontrol menjadi lebih sederhana jika menggunakan desain dengan gelombang persegi yang mirip digital daripada gelombang sinusoida.

Untuk mengembangkan sebuah motor DC tanpa *brush*, sebagai dasarnya adalah motor AC, seperti motor induksi dengan rotor sangkar atau motor sinkron dengan magnet permanen. Dasar pemikirannya adalah motor-motor ini tidak memiliki komutator dan *brush* sehingga jika motor-motor ini dijalankan dengan sumber DC maka mereka dapat disebut dengan motor DC *brushless.* Jadi jika sebuah motor dengan permanen magnet menggunakan rangkaian elektronik sebagai pengontrol dan sensor posisi maka motor ini dapat disebut motor DC *brushless* dengan karakteristik yang hampir sama dengan motor DC konvensional. (Lucas Yuda Brata, 1997). Motor DC *brushless* ditunjukkan dalam Gambar 2.5.

Gambar 2. 5 Motor Dc Brushless
Sumber: http://miketigabelas.webs.com

2.4 Motor Servo

Berbeda dengan motor DC dan motor Stepper, motor servo adalah sebuah motor dengan sistem *closed feedback* di mana posisi dari motor akan diinformasikan kembali ke rangkaian kontrol yang ada di dalam motor servo. Motor ini terdiri atas sebuah motor, serangkaian *internal gear*, potensiometer dan rangkaian kontrol. Potensiometer berfungsi untuk menentukan batas sudut putaran servo. Sedangkan sudut sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor.

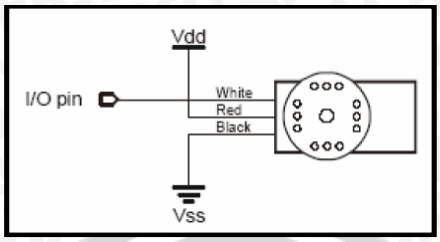
Motor servo mampu bekerja dua arah yaitu CW (*clockwise*) atau searah jarum jam dan CCW (*counterclockwise*) atau berlawanan arah jarum jam di mana arah dan sudut pergerakan rotornya dapat dikendalikan hanya dengan memberikan pengaturan *duty cycle* sinyal PWM *(pulse width modulation)* pada bagian pin kontrolnya. Gambar 2.6 menunjukkan gambar motor servo.

Gambar 2. 6 Motor Servo Parallax
Sumber : Parallax, Inc.

Motor servo merupakan motor yang berputar lambat, biasanya ditunjukkan oleh *rate* putarannya yang lambat, namun demikian memiliki torsi yang kuat karena *internal gear*-nya. Karakteristik motor servo adalah sebagai berikut :

1. Memiliki 3 jalur kabel : *power, ground, dan control* seperti ditunjukkan dalam Gambar 2.2.
2. Pin *Control* untuk mengendalikan posisi.
3. Operasional dari servo motor dikendalikan oleh sebuah pulsa selebar ± 20 ms, dimana lebar pulsa antara 0.5 ms dan 2 ms menyatakan akhir dari *range* sudut maksimum.
4. Konstruksi didalamnya meliputi *internal gear*, *potensiometer*, dan *feedback control.*

Gambar 2.7 menunjukkan konfigurasi pin pengkabelan motor servo.



Gambar 2. 7 Konfigurasi Pin Pengkabelan Motor Servo

Sumber : Parallax, Inc.

Secara umum terdapat dua jenis motor servo, yaitu :

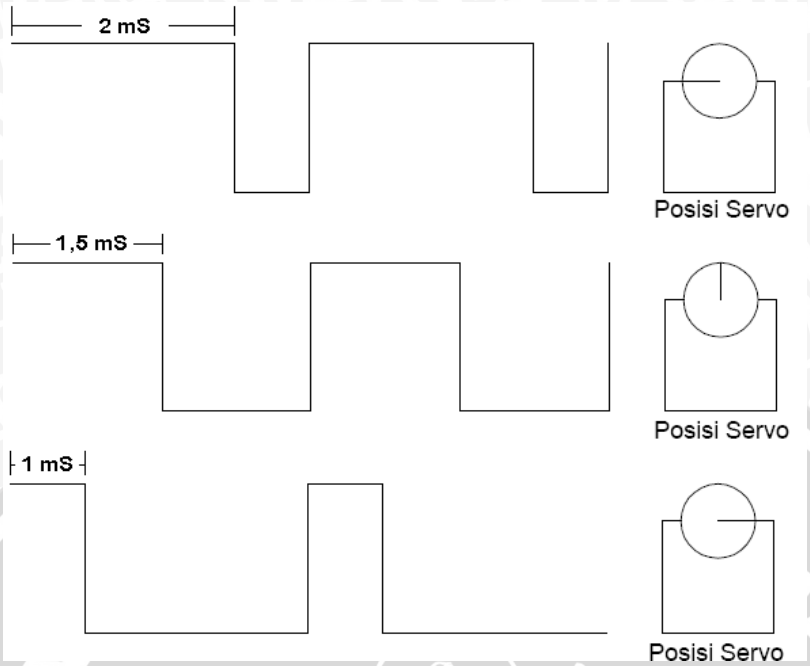
1. Motor Servo *Standar 180°*

   Motor servo jenis ini merupakan motor yang hanya mampu bergerak dua arah (*CW* dan *CCW)* dan mempunyai *defleksi* masing-masing sudut mencapai 90° sehingga total *defleksi* sudut dari kanan – tengah – kiri adalah 180°.

2. Motor Servo *Continuous*

   Motor servo jenis ini mampu bergerak dua arah *(CW* dan *CCW)* dan tanpa batasan *defleksi* sudut putar (dapat berputar secara kontinyu) sehingga motor ini berputar 360°.

Pengaturan sudut motor servo diperlukan untuk mengetahui gerakan dari motor servo dengan pulsa yang harus diberikan untuk bergerak ke kanan atau bergerak ke kiri. Gambar 2.8 menunjukkan teknik PWM *(pulse width modulation)* untuk mengatur sudut motor servo.



Gambar 2. 8 Pengaturan Sudut Motor Servo

Sumber : Parallax, Inc.

Dalam Gambar 2.8 diasumsikan bahwa saat diberikan sinyal periodik dengan lebar 1 ms maka motor servo akan bergerak dengan sudut 0^o, jika diberi sinyal 1,5 ms maka motor servo akan bergerak dengan sudut 90^o, dan jika diberi sinyal 2 ms maka motor servo akan bergerak dengan sudut 180^o. Perhitungan rumus motor servo akan ditunjukkan persamaan sebagai berikut:

$S = D\left(\frac{1000}{180}\right) + 1000\,\mu s$ atau

$S = (5{,}555)D + 1000\,\mu s$

$S = Lebar\ Pulsa\ dalam\ \mu s$

$D = Sudut\ putar\ servo\ dalam\ derajat$

(Rizal Maulana, 2011: 9).

2.5 Sensor *Gyro*

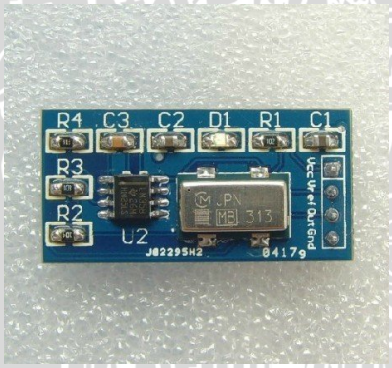
Sensor adalah jenis transduser yang digunakan untuk mengubah besaran mekanis, magnetis, panas, sinar, dan kimia menjadi tegangan dan arus listrik. Sensor merupakan salah satu komponen yang sangat penting dalam mendukung terjadinya kontrol proses yang berfungsi sebagai berikut:

- Menyediakan *input* dari proses dan dari lingkungan *eksternal.*
- Mengubah informasi fisik misalnya suhu, tekanan, laju aliran dan posisi untuk sinyal listrik.

Sensor sering digunakan untuk pendeteksian pada saat melakukan pengukuran atau pengendalian. Beberapa jenis sensor banyak digunakan dalam rangkaian elektronik antara lain sensor cahaya, sensor suhu, dan sensor tekanan.

Gyro (Sensor angular) adalah sensor yang mendeteksi perubahan kemiringan dari suatu objek yang bergerak. Sensor *Gyro* biasa digunakan pada robot atau dengan kendaraan aktuator roda dua seperti kendaraan *segway* dan dapat digunakan pada objek seperti helikopter untuk keperluan autopilot.

Jenis Sensor yang digunakan tipe ENC-03M. ENC-03M merupakan sensor kecepatan sudut yang memanfaatkan gaya coriolis yang bekerja pada benda bergetar. Sensor *gyro* ini digunakan untuk mengontrol posisi dan *balancing* dari suatu objek bergerak yang membutuhkan respon cepat dari setiap pengukurannya. Sensor *gyro* mendeteksi posisi pada *tricopter* ketika mendapat gangguan saat terbang yang berupa gerak *pitch, roll*, dan *yaw*. Gambar 2.9 menunjukkan gambar sensor *gyro* ENC-03. (seribubintang.com)



Gambar 2. 9 Sensor Gyro Enc-03M

Sumber : img.alibaba.com

2.6 Kontroler

Keberadaan kontroler dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadap perilaku sistem. Pada prinsipnya hal itu disebabkan oleh tidak dapat diubahnya komponen penyusun sistem tersebut. Artinya, karakteristik *plant* harus diterima sebagaimana adanya, sehingga perubahan perilaku sistem hanya dapat dilakukan melalui penambahan suatu subsistem, yaitu kontroler.

Salah satu fungsi komponen kontroler adalah mengurangi sinyal kesalahan, yaitu perbedaan antara nilai referensi/nilai yang diinginkan dan nilai aktual. Hal ini sesuai dengan tujuan sistem kontrol dimana mendapat nilai sinyal keluaran sama dengan nilai

yang diinginkan/referensi. Semakin kecil kesalahan yang terjadi, semakin baik kinerja sistem kontrol yang diterapkan.

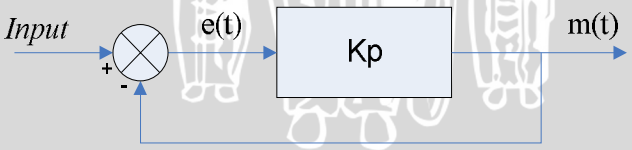
Apabila perbedaan antara nilai referensi dengan nilai keluaran relatif besar, maka kontroler yang baik seharusnya mampu mengatasi perbedaan ini untuk segera menghasilkan sinyal keluaran untuk mempengaruhi plant. Dengan demikian sistem secara cepat mengubah keluaran plant sampai diperoleh selisih dengan nilai referensi sekecil mungkin.

Prinsip kerja kontroler adalah membandingkan nilai aktual keluaran plant dengan nilai referensi, kemudian menentukan nilai kesalahan dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan kesalahan. (Ogata, 1995).

2.6.1 Kontroler Proporsional

Kontroler proporsional memiliki keluaran yang sebanding/proporsional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya). Secara lebih sederhana dapat dikatakan, bahwa keluaran kontroler proporsional merupakan perkalian antara konstanta proporsional dengan masukannya. Perubahan pada sinyal masukan menyebabkan sistem secara langsung mengubah keluarannya sebesar konstanta pengalinya.

Pada Gambar 2.10 menunjukkan diagram blok yang menggambarkan hubungan antara *input* (besaran referensi yang diinginkan), besaran aktual dengan besaran keluaran kontroler proporsional, dan besaran kesalahan (*error*). Sinyal kesalahan (*error*) merupakan selisih antara besaran *setting* dengan besaran aktualnya.



Gambar 2. 10 Blok Diagram Kontroler Proporsional

Sumber: Ogata,1995: 157

Pada pengendali proporsional hubungan antara keluaran kontroler m(t) dan sinyal kesalahan e(t) adalah

$$m(t) = Kp\, e(t) \quad \text{.....(2-1)}$$

Sumber: Ogata, 1995: 157

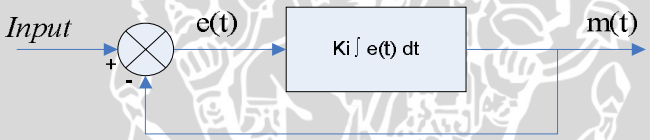
dengan Kp adalah penguatan proporsional. Keluaran m(t) hanya tergantung pada Kp dan *error*, semakin besar *error* maka semakin besar koreksi yang dilakukan.

Penambahan Kp akan menaikkan penguatan sistem sehingga dapat digunakan untuk memperbesar kecepatan respons dan mengurangi kesalahan keadaan mantap.

2.6.2 Kontroler Integral

Kontroler integral berfungsi mengurangi kesalahan keadaan mantap yang dihasilkan pada kontroler proporsional sebelumnya. Kalau sebuah *plant* tidak memiliki unsur integrator (1/s), kontroler proporsional tidak akan mampu menjamin keluaran sistem dengan kesalahan keadaan mantap nol.

Kontroler integral memiliki karakteristik seperti halnya sebuah integral. Keluaran kontroler sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan. Keluaran kontroler ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Kalau sinyal kesalahan tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan. Gambar 2.11 menunjukkan diagram blok kontroler integral.



gambar 2. 11 Diagram Blok Kontroler Integral

Sumber: Ogata, 1995: 158

Nilai keluaran kontroler *m(t)* sebanding dengan integral sinyal kesalahan e(t), Sehingga

$$\frac{dm(t)}{dt} = Ki.e(t) \quad \text{.....(2-2)}$$

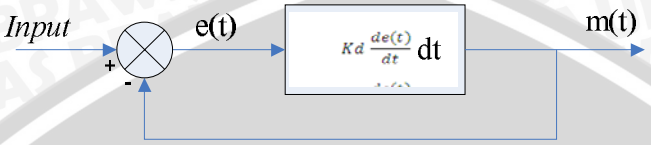
$$m(t) = Ki \int_0^t e(t)dt \quad \text{.....(2-3)}$$

Sumber: Ogata, 1995: 157

dengan Ki adalah konstanta integral. Jika sinyal kesalahan e(t)=0, maka laju perubahan sinyal kendali integral $\frac{dm(t)}{dt} = 0$ atau sinyal keluaran kendali akan tetap berada pada nilai yang dicapai sebelumnya. Aksi kontrol integral digunakan untuk menghilangkan kesalahan posisi dalam keadaan mantap (*error steady state*) tanpa memperhitungkan kecepatan respons.

2.6.3 Kontroler Diferensial

Kontroler diferensial memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan kontroler, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.12 berikut menunjukkan diagram blok pada kontroler diferensial.



Gambar 2. 12 Diagram Blok Kontroler Diferensial

Sumber: Ogata, 1995: 177

Nilai keluaran kontroler m(t) sebanding laju sinyal kesalahan $\frac{de(t)}{dt}$. Hubungan ini dapat ditulis sebagai:

$$m(t) = Kd\ \frac{de(t)}{dt} \quad \text{.....(2-4)}$$

Sumber: Ogata, 1995: 179

Kontroler diferensial akan memberikan sinyal kendali keluaran m(t)= 0, untuk sinyal kesalahan e(t) yang konstan sehingga kontroler diferensial tidak mempengaruhi keadaan mantap. Kontroler diferensial digunakan untuk memperbaiki atau mempercepat respons transien sebuah sistem serta dapat meredam osilasi.

Berdasarkan karakteristik kontroler tersebut, kontroler diferensial umumnya dipakai untuk mempercepat respons awal suatu sistem, tetapi tidak memperkecil kesalahan pada keadaan tunaknya. Kerja kontroler diferensial hanyalah efek dari lingkup yang sempit, yaitu pada periode peralihan. Oleh sebab itu kontroler diferensial tidak bisa digunakan tanpa ada kontroler lain. Dari ketiga aksi kontrol dasar di atas dapat dibuat kombinasi dari ketiganya, yaitu:

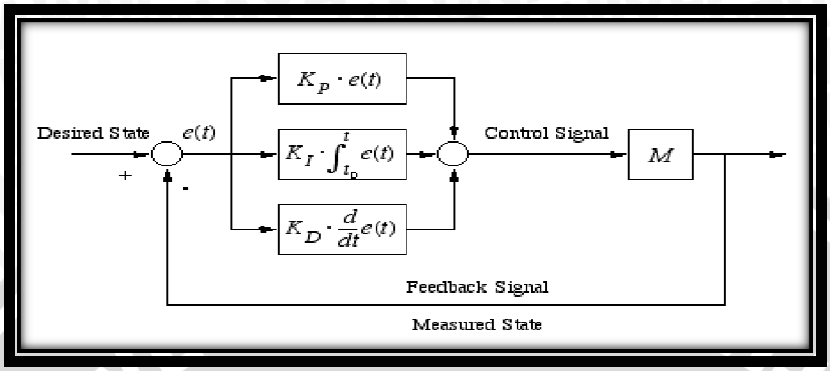
2.6.4 Kontroler Proporsional Integral Diferensial (PID)

Setiap kekurangan dan kelebihan dari masing-masing kontroler P, I dan D dapat saling menutupi dengan menggabungkan ketiganya secara paralel menjadi kontroler proporsional integral diferensial (PID). Elemen-elemen kontroler P, I dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar (Gunterus, 1994, 8-10). Kontroler PID memiliki diagram kendali seperti yang ditunjukkan dalam Gambar 2.13. Aksi kontrolnya dinyatakan sebagai:

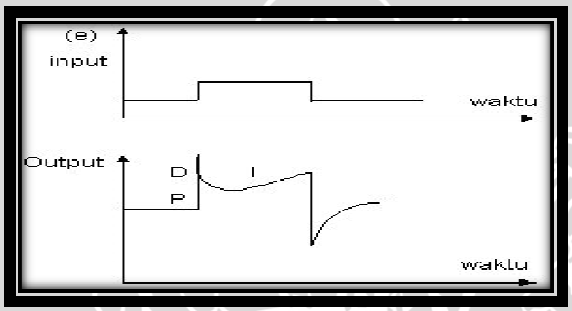
$$m(t) = K_p e(t) + \frac{Kp}{Ti}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \text{.....(2-5)}$$

Sumber: Ogata, 1995: 183

Jenis kontroler ini digunakan untuk memperbaiki kecepatan respons, mencegah terjadinya kesalahan keadaan mantap serta mempertahankan kestabilan.



Gambar 2. 13 Diagram Blok Kontroler PID



Gambar 2. 14 Fungsi Waktu antara Sinyal Keluaran dan Sinyal Masukan Kontroler PID

Sumber: Gunterus, 1994:8-11

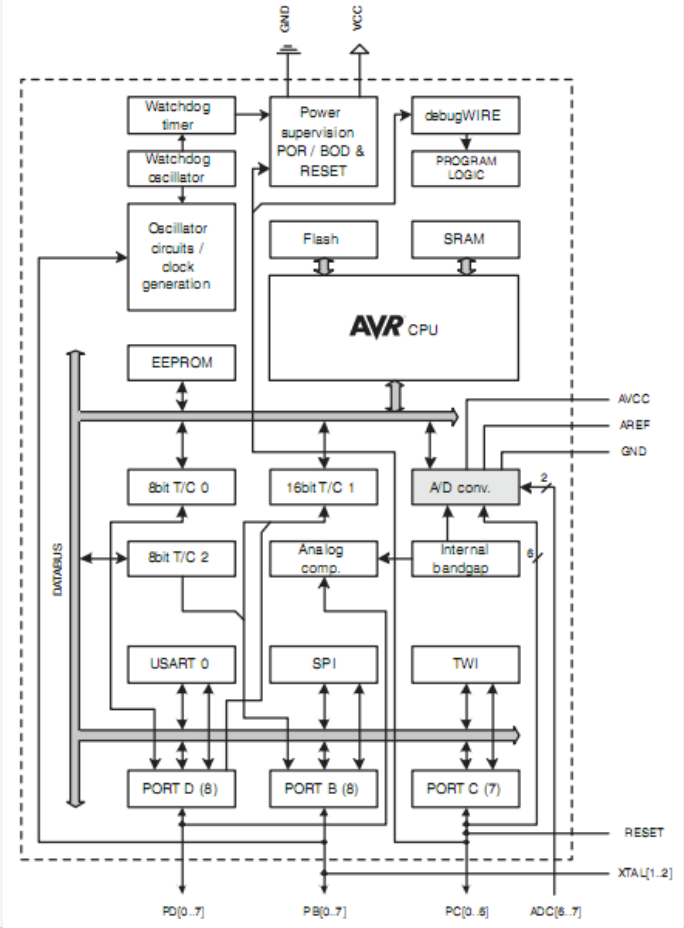
Keluaran kontroler PID merupakan penjumlahan dari keluaran kontroler proporsional, integral dan diferensial. Gambar 2.14 menunjukkan hubungan tersebut. Karakteristik kontroler PID sangat dipengaruhi oleh kontribusi besar dari ketiga parameter P,I dan D. Penyetelan konstanta Kp, Ti dan Td akan mengakibatkan penonjolan sifat dari masing-masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat disetel lebih menonjol dibanding yang lain. Konstanta yang menonjol itulah yang akan memberikan kontribusi pada respons sistem secara keseluruhan (Gunterus, 1994, 8-10).

2.7 Mikrokontroler ATMega168

Mikrokontroler ATMega168 merupakan mikrokontroler generasi AVR (*Alf and Vegards Risk processor*). Mikrokontroler AVR memiliki arsitektur RISC (*Reduced Instruction Set Computing*) sebagian besar instruksi dieksekusi dalam 1 siklus *clock*. Mikrokontroler ATMega168 mampu mengeksekusi data 1MIPS (*Million Instruction Per*

Second) per MHz(*Mega Hertz*) ini memungkinkan sebuah sistem untuk mengoptimalkan konsumsi daya dibandingkan kecepatan pengeksekusian.

2.7.1 Arsitektur Mikrokontroler Atmega168



Gambar 2. 15 Diagram Blok Fungsional Atmega168

Sumber : Data sheet ATmega168

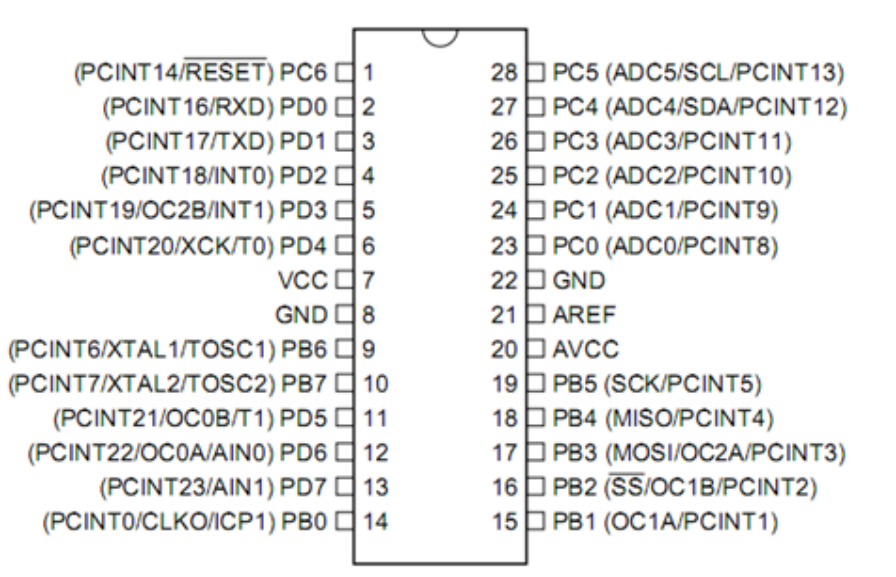
AVR (*Alf and Vegards Risk*) core memiliki banyak kombinasi instruksi dengan mengeset 32 register umum yang bekerja. Semua 32 register secara langsung terhubung ke *Aritmetic Logic Unit* (ALU), memungkinkan dua register independen untuk diakses dalam satu instruksi dieksekusi dalam satu siklus *clock*. Arsitektur kode yang dihasilkan lebih efisien sementara mencapai *throughputs* sampai dengan sepuluh kali lebih cepat daripada mikrokontroler CISC (*Complex Instruction Set Computer*) konvensional.

Gambar 2.15 memperlihatkan bahwa ATMega168 memiliki bagian sebagai berikut

1. Memori *Flash* sebesar 16K dengan kemampuan *Read While Write*.
2. EEPROM (*Electrically Erasable Programmable Read Only Memory*) sebesar 512 byte yang dapat diprogram saat operasi
3. SRAM (*Static Random Access Memory*) sebesar 1 Kb.

4. Memiliki 23 jalur umum I/O
5. ADC 10 bit sebanyak 8 saluran.
6. CPU (*central processing unit*) yang terdiri atas 32 buah register.
7. Memiliki sebuah USART (*Universal Synchronous Asynchronous serial Receiver and Transmitter*) diprogram serial, byte-2-kawat berorientasi *Serial Interface*.
8. Unit interupsi internal dan eksternal
9. Sistem mikroprosessor 8 bit berbasis RISC (*Reduced Instruction Set Computing*) dengan kecepatan maksimal 1 MIPS per MHz.
10. Port antarmuka SPI.

2.7.2 Konfigurasi Pin ATMega168



Gambar 2. 16 Konfigurasi Pin ATmega168

Sumber : Data sheet ATmega168

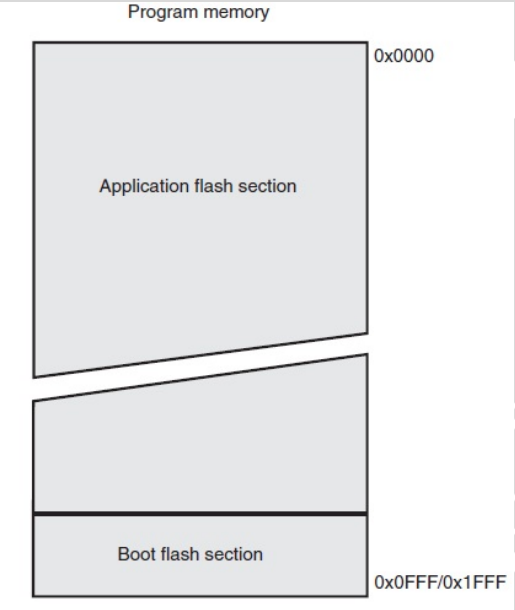
Konfigurasi pin ATMega168 dapat dilihat dalam Gambar 2.16. Secara fungsional konfigurasi pin ATMega168 sebagai berikut :

1. VCC merupakan pin yang berfungsi sebagai pin masukan catu daya.
2. GND merupakan pin ground
3. Port B ,Port C,Port D adalah *bit bi-directional port* I/O dengan resistor internal *pull-up*. Berfungsi sebagai jalur pin I/O.
4. AVcc adalah pin tegangan suplai untuk A/D konverter.
5. Aref adalah pin referensi analog untuk A/D konverter.

6. ADC7-6 berfungsi sebagai input analog untuk A/D konverter. Pin ini didukung dari catu analog dan berfungsi sebagai 10 bit saluran ADC.

2.7.3 Peta Memori

ATmega 168 berisi 8K/16K bit di dalam memori sebagai tempat untuk menyimpan program. Untuk keamanan perangkat lunak program *flash* memori dibagi menjadi dua bagian yaitu bagian *boot loader* dan bagian aplikasi program ATMega168. Program *counter* ATMega168 memiliki lebar 11/12/13 bit ditunjukkan dalam Gambar 2.17.



Gambar 2. 17 Program Peta Memori Atmel Atmega168

Sumber : *Data sheet* ATmega168

2.8 *Radio Control* (R/C)

2.8.1 Pengertian *Radio Control* (R/C)

Radio control (R/C) merupakan alat yang menggunakan sinyal radio untuk mengontrol alat lain dari jauh. Sistem kendali jarah jauh (*remote control*) yang digunakan untuk mengendalikan pesawat terbang, roket, perahu maupun mobil-mobilan sebenarnya merupakan contoh yang sederhana dari sistem pengendalian *Fly by Wire* tersebut. Sistem yang saat ini banyak ditemukan di pasaran menggunakan gelombang radio sebagai sistem penyampaian informasinya ini sudah dipergunakan orang sejak tahun 70-an. Berbeda dengan sistem *remote control* untuk alarm mobil atau untuk pengatur televisi yang umumnya menggunakan tombol tekan sebagai input pengendaliannya, sistem kendali radio

atau yang selanjutnya disebut *Radio Contro*l disingkat R/C ini lebih banyak menggunakan potensiometer sebagai masukannya.

Sistem R/C sebelumnya memang ditujukan untuk keperluan militer, yakni untuk mengendalikan peluru kendali yang tidak berawak yang dilepaskan dari pesawat terbang untuk menghancurkan daerah lawan. Saat ini R/C sudah banyak digunakan orang untuk mengendalikan berbagai sistem, baik untuk keperluan riset, industri, rekreasi maupun keperluan rumah tangga. Berbagai jenis pesawat terbang model, perahu, mobil-mobilan bahkan robot mainan saat inipun sudah banyak tersedia di toko-toko dengan dilengkapi *radio control.*

Secara umum sistem R/C terdiri dari sebuah pemancar atau *transmitter* yang berupa TX (*remote control)* dan sebuah atau lebih penerima atau *receiver* yang diletakkan pada pesawat model. Baterai sebagai sumber daya diperlukan oleh bagian pemancar maupun bagian penerima. Pemancar atau *transmitter* bertugas menerima perintah kendali dari orang yang mengendalikan dan merubahnya menjadi kode-kode elektronik dan mengirimkannya ke RX (*receiver*) melalui gelombang radio ke udara.

2.8.2 Cara Modulasi

Sistem R/C menggunakan berbagai metode modulasi seperti halnya modulasi amplitudo (*amplitude modulation* disingkat AM), modulasi frekuensi (*frequency modulation* disingkat FM) dan modulasi kode pulsa (*pulse code modulation* disingkat PCM). Dari segi kualitas dan harganya, R/C dengan gelombang FM lebih baik dibandingkan yang menggunakan gelombang AM. Sedangkan R/C yang menggunakan gelombang PCM memiliki sistem perlindungan agar tidak dapat di kacaukan oleh gelombang radio asing yang frekuensinya sama, sehingga sistem ini oleh banyak pihak dinilai lebih baik dari sistem R/C bergelombang FM. Saat ini ada beberapa bentuk R/C dikenal di kalangan pemakainya yakni para penggemar model, dilihat dari bentuk transmitter-nya yakni jenis tongkat atau *stick* yang banyak dipakai untuk menerbangkan pesawat model, R/C jenis pistol yang banyak ditemui di dunia mobil model serta Boat model dan R/C bertombol tekan (*push on*) yang banyak digunakan untuk pesawat terbang mainan anak-anak.

2.8.3 Range Frekuensi

Frekuensi sistem R/C yang dipasarkan saat ini di seluruh dunia menggunakan jalur frekuensi operasi 27 Mhz, 29 Mhz, 35 Mhz, 40 Mhz, 50 Mhz dan 72 Mhz serta 75 Mhz. Di dalam setiap jalur terdapat berpuluh-puluh kanal yang dapat digunakan diantaranya ada

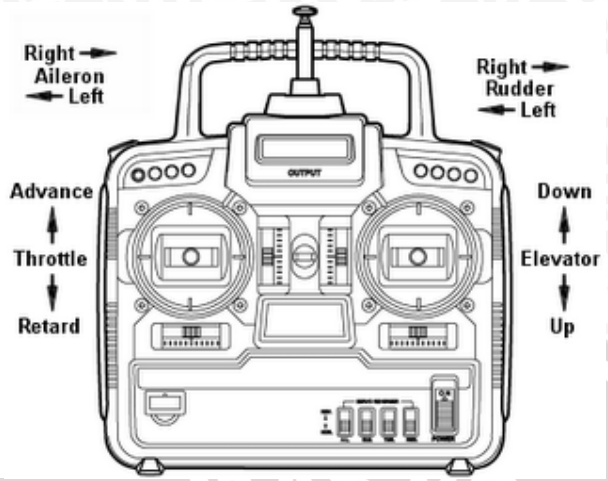
sekitar 5 kanal di frekuensi 27 Mhz, 50 kanal di frekuensi 29 Mhz dan lebih dari seratus kanal tersedia di frekuesi lainnya. Selisih frekuensi antara kanal satu dengan kanal lainnya adalah 20 kHz untuk radio tipe mutakhir dengan band width yang sempit. Jadi sebenarnya kemungkinan frekuensi R/C satu bentrok dengan frekuensi R/C lainnya adalah cukup kecil walaupun tetap ada saja kemungkinannya.

2.8.4 Daya Jangkau

Radio control yang bekerja di jalur AM (*Amplitude Modulation*) karena efeisiensi TX (*transmitter*)-nya rendah, kebanyakan di produksi dengan daya 1 watt untuk menjangkau jarak kendali radius 1 km. Untuk R/C yang beroperasi dengan sistem modulasi FM maupun PCM umumya mempunyai daya pemancar 500 mW yang dapat menjangkau jarak kendali efektif yang sama sekitar 1 km radius. Jarak ini sudah cukup jauh sebenarnya mengingat bahwa jarak pandang normal kita terhadap pesawat model umumnya hanya sekitar 300 meter. Penggunaan baterainya relatif lebih hemat dibandingkan dengan R/C yang bekerja dengan gelombang AM. Beberapa R/C yang ditujukan untuk mengendalikan mobil model dirancang untuk mempunyai daya jangkau yang lebih pendek yakni 300 meter. Dengan kenyataan ini anda sebaiknya hati-hati jika akan menggunakan R/C mobil model anda untuk mengendalikan pesawat terbang model, sebab begitu pesawat terbang model anda mengudara semenit kemudian bisa-bisa sudah berada diluar jangkauan kendali (*out of control*).

2.8.5 Pengoperasian *Radio Control*

Radio Control atau disingkat RC merupakan pemacar (*transmitter)* pengendali yang dipegang pilot di darat. pengoperasiaannya begitu vital karena pesawat bisa dikendalikan melalui alat ini. Hal ini tentunya pesawat juga harus dilengkapi dengan alat penerima (*receiver)* agar pilot dan pesawat dapat berkomunikasi. Jenis *radio control* (TX/*transmitter*) yang digunakan dalam dunia *aeromodelling* terbagi tiga yaitu model pertama yang populer di Eropa, model kedua populer di Amerika, dan model ketiga populer di Asia.



Gambar 2. 18 Transmitter yang Popular Di Asia
Sumber: aeromodelingpemula.blogspot.com

Gambar 2.18 merupakan model *transmitter* yang populer di Asia. Terdapat dua stik yaitu pada bagian kiri dan kanan. Stik kiri adalah *throttle* jika di digeser ke depan maka kecepatan pesawat akan bertambah. Sebaliknya, jika digeser kebelakang kecepatan laju pesawat akan berkurang. Jika stik digeser ke kiri atau ke kanan *aileron* maka pesawat akan berguling di udara. Stik sebelah kanan atau stik *elevator* apabila digeser ke depan maka pesawat akan turun sebaliknya apabila digeser ke belakang maka pesaat akan naik. Selanjutnya stik *rudder* apabila stik digeser ke sebalah kiri maka pesawat akan belok ke kiri dan jika digeser ke kanan maka pesawat akan belok kanan. (aeromodelingpemula.blogspot.com, 20 juni 2012).

BAB III
METODOLOGI PENELITIAN

Kajian dalam skripsi ini merupakan penelitian yang bersifat aplikatif, yaitu perencanaan dan pembuatan sistem pengontrolan gerak *hover tricopter* yang menggunakan Mikrokontroler atmega 168 yang bertujuan agar dapat menampilkan unjuk kerja sesuai dengan yang direncanakan. Berikut ini adalah metodologi yang digunakan dalam penelitian:

3.1 Studi Literatur

Perencanaan sistem dilakukan sebagai langkah awal sebelum terbentuknya suatu sistem beserta rangkaian elektronik pendukungnya, hal ini dimaksudkan agar sistem pengindentifikasi dapat berjalan sesuai dengan yang telah direncanakan.

1) *Tricopter*
2) Gerak Aerodinamika Pesawat
3) Motor DC *Brushless*
4) Motor Servo
5) Sensor *Gyro*
6) Kontroler PID
7) Mikrokontroler Atmega 168
8) *Radio Control* (RC)

3.2 Penentuan Spesifikasi Alat

Adapun spesifikasi alat yang akan direalisasikan adalah sebagai berikut:

1) *Tricopter* berbahas dasar mika dan kayu.
2) *Tricopter* memiliki struktur 1 kepala dengan 3 lengan
3) Panjang lengan *tricopter* adalah 50 cm
4) Masing-masing lengan *tricopter* memiliki 3 buah motor DC dengan *propeller* dan 1 buah motor servo untuk lengan belakang.
5) Motor brushless yang digunakan adalah tipe DT750 kV.
6) Motor servo yang digunakan tipe BMS-385DMAX
7) *Board* KKMulticopter 168PA diletakkan tepat pada *centre of gravity* dari *tricopter*
8) CV-AVR sebagai *software* pemrograman mikrokontroler ATmega168.

3.3 Perancangan Dan Realisasi Pembuatan Alat

3.3.1 Perancangan Perangkat Keras Dan Realisasi Tiap Blok

a. Pembuatan diagram blok sistem secara lengkap.
b. Penentuan dan perhitungan komponen yang akan digunakan.
c. Merakit perangkat keras (*Hardware*) untuk masing-masing blok.

3.3.2 Perancangan Perangkat Lunak

Setelah mengetahui seperti apa perangkat keras yang dirancang, maka dibutuhkan perangkat lunak untuk mengendalikan dan mengatur kerja dari alat ini. Desain dan parameter yang telah dirancang kemudian diterapkan dalam mikrokontroler dengan menggunakan software CV-AVR.

3.4 Pengujian Alat

Untuk memastikan bahwa sistem ini berjalan sesuai yang direncanakan, maka perlu dilakukan pengujian alat yang meliputi pengujian tiap blok dan keseluruhan sistem.

3.4.1 Pengujian Tiap Blok

Pengujian per blok dilakukan dengan tujuan untuk menyesuaikan nilai masukan dan nilai keluaran tiap-tiap blok sesuai dengan perancangan yang dilakukan sebelumnya. Pengujian tiap blok meliputi pengujian catu daya, pengujian sensor *gyro,* pengujian motor brushless, pengujian motor servo dan pengujian KKMulticopter 168PA.

3.4.2 Pengujian Keseluruhan Sistem

Pengujian ini dilakukan dengan cara menggabungkan semua bagian alat yang dibuat dan melihat kinerja alat. Pengujian ini bertujuan untuk mengetahui kinerja alat yang dibuat dan memberikan analisis terhadap kinerja alat.

3.5 Pengambilan Kesimpulan

Kesimpulan diambil berdasarkan data yang diperoleh dari pengujian. Jika hasil yang didapatkan telah sesuai dengan yang direncanakan sebelumnya, maka sistem kendali tersebut telah berhasil memenuhi harapan dan tentunya memerlukan pengembangan lebih lanjut untuk penyempurnaan.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

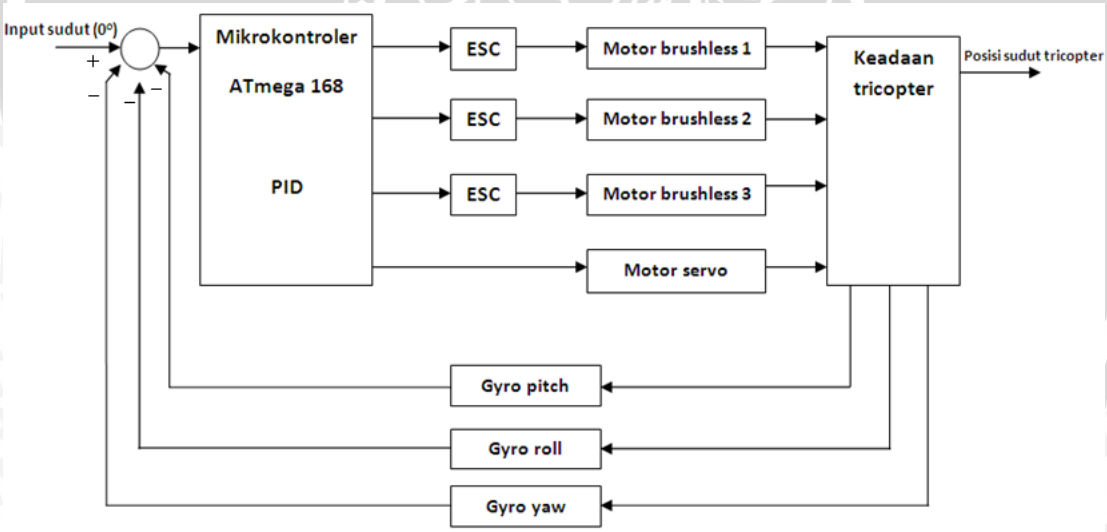
Perancangan alat ini dilakukan secara bertahap dalam bentuk blok sehingga akan memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. Perancangan ini terdiri dari:

- Perancangan diagram blok sistem.
- Perangkat keras yang terdiri dari: *power supply*, ESC (*Electronic Speed Control*), motor servo, dan KKmulticopter 168PA (sensor gyro ENC 03M dan mikrokontroler ATmega168)
- Perancangan mekanik *tricopter*.
- Perancangan sistem logika PID serta implementasinya pada mikrokontroler.

4.1 Perancangan Sistem

4.1.1 Diagram Blok Sistem

Untuk mempermudah pengerjaan alat maka perlu adanya diagram blok umum sistem yang ditunjukkan dalam Gambar 4.1 :



Gambar 4. 1 Diagram Blok Sistem

Sumber : Perancangan

4.1.2 Prinsip Kerja Alat

Tricopter diberi *input* dari *remote control* (RC) berupa *throttle* (gas) 0 – 100% dan untuk *set point tricopter* telah ditentukan pada kondisi seimbang (horizontal) yaitu *pitch* = 0^o, *roll* = 0^o, dan *yaw* = 0^o. Sedangkan umpan balik sistem berasal dari sensor *gyro* yang terdapat dalam *tricopter*. Ada 3 sensor *gyro* yang digunakan dalam *tricopter* ini, yaitu

untuk membaca posisi *pitch, roll*, dan *yaw*. Di mana sensor-sensor tersebut membaca bagaimana keadaan *tricopter* ketika *hover* di udara. Pada kondisi *hover* (melayang di udara) berarti sensor *gyro* tidak mendeteksi adanya perubahan posisi. Apabila ada gangguan yang menyebabkan posisi *tricopter* menukik ke bawah atau ke atas maka sensor *gyro picth* akan memberi masukan pada mikrokontroler Atmega168 untuk menginformasikan agar melakukan penambahan kecepatan pada motor bagian depan (motor kiri dan kanan) atau penambahan kecepatan motor bagian ekor *tricopter*. Bila posisi *tricopter* miring ke samping kiri dan kanan maka sensor *gyro roll* akan memberi masukan pada mikrokontroler Atmega168 untuk menginformasikan agar melakukan penambahan kecepatan pada motor sebelah kiri depan dan kanan depan. Sedangkan bila *tricopter* terbang memutar ke kanan dan ke kiri maka sensor *gyro ya*w akan memberi masukan pada mikrokontroler Atmega168 untuk menginformasikan pada motor servo ke mana bergerak, bila *tricopter* memutar searah jarum jam maka motor servo bergerak ke arah kanan begitu sebaliknya.

Dalam hal ini mikrokontroler Atmega168 berfungsi sebagai kontroler PID yang membaca hasil keluaran dari sensor *gyro* dan menghitung kemiringan dari *tricopter* dalam bentuk sinyal *analog*. Kemudian data tersebut dibandingkan dengan *set point* untuk kemudian dicari nilai *error* nya dan dilakukan penghitungan untuk nilai kontroler yang diperlukan. Mikrokontroler ini terhubung dengan *driver* motor yang berupa ESC yang berfungsi sebagai pengkondisi sinyal dari sinyal kontrol mikrokontroler menjadi sinyal untuk menggerakkan aktuatornya yaitu motor *brushless*. *Driver* motor ini berfungsi mengontrol kecepatan motor dengan membaca keluaran mikrokontroler yang berupa PWM (*Pulse Width Modulation*).

4.2. Perangkat keras

Perangkat keras yang terdiri dari: *power supply*, ESC (*Electronic Speed Control*), motor servo, dan KKmulticopter 168PA (sensor gyro ENC 03M dan mikrokontroler ATMega168)**.**

4.2.1 *Power Supply*

Power supply atau sumber catu daya yang digunakan dalam *tricopter* ini adalah baterai Lipo 3 cell 2200 mA 11,1 V. Baterai ini digunakan untuk mencatu *board* KKmulticopter 168 PA dan 3 ESC (*Electronic Speed Control*) untuk motor *brushless*. Tegangan yang dibutuhkan untuk mencatu board Kkmulticopter 168 PA adalah 5 volt, oleh karena itu dibutuhkan rangkaian regulator yang berupa BEC (*battery eliminator circuit*)

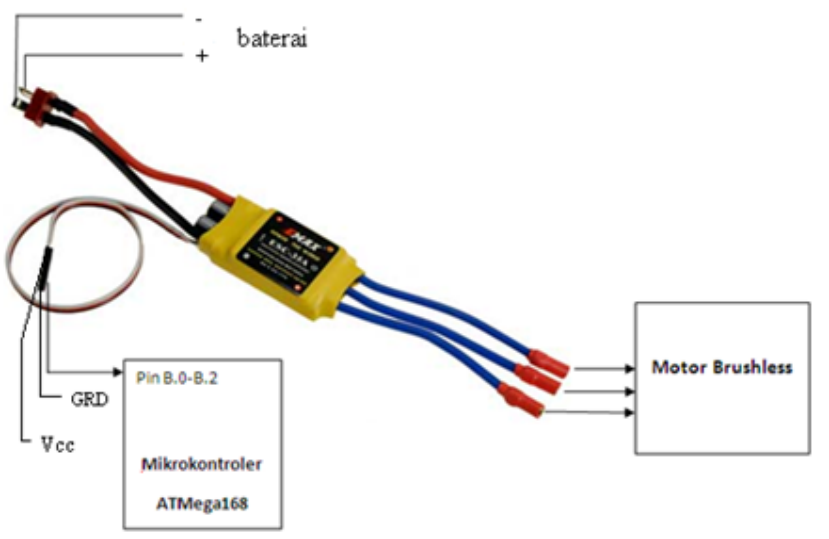
yang terdapat dalam ESC (*Electronic Speed Control*). Dengan tegangan 5 V tersebut, mikrokontroler dapat bekerja karena tegangan kerjanya antara 2,7 V hingga 5,5 V sesuai dengan *datasheet* ATmega168. Sedangkan untuk mencatu 3 motor brushless pada *tricopter* menggunakan sumber langsung dari baterai 11,1 V yang dirangkai secara paralel. Baterai *lithium polymer* 3 cell 2200 mA 11,1 V ditunjukan pada Gambar 4.2

Gambar 4. 2 Baterai *lithium polymer* 3 cell 2200 mA 11,1 V

Sumber: perancangan

4.2.2 ESC (*Electronic Speed Control*)

Jenis *driver* yang digunakan untuk *tricopter* ini adalah ESC (*Electronic Speed Control*). ESC yang digunakan adalah ESC 18 A tower pro dengan BEC (*battery eliminator circuit*) 5 V. ESC ini memiliki 5 kabel input, 2 kabel warna merah dan hitam dicatu ke baterai Lipo. Sedangkan 3 kabel *input* yang lain dalam 1 kesatuan yang terdiri dari Vcc (warna merah) yang di catu 5 V , *ground* (warna hitam/coklat) dan *control* (warna putih/kuning) yang masuk ke pin mikrokontroler. Fungsi dari ESC itu sendiri adalah sebagai pengatur kecepatan putaran motor pada *tricopter* dengan cara menterjemahkan sinyal PWM (*Pulse Width Modulation*) yang diterima dari mikrokontroler yang berupa lebar pulsa. Di dalam ESC terdapat rangkaian transistor E MOSFET kanal N yang berfungsi mengatur tegangan keluaran yang digunakan untuk masukan motor *brushless*. Gambar 4.3 menunjukkan pemasangan ESC pada *tricopter*

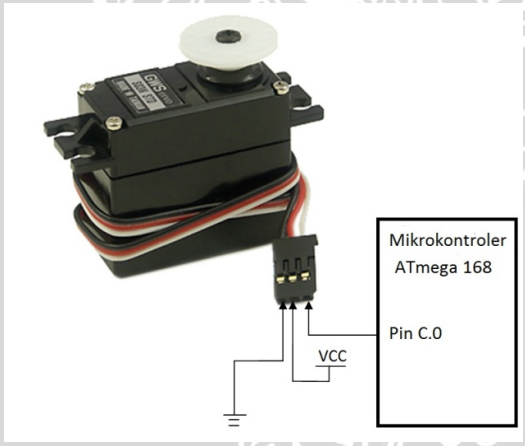


Gambar 4. 3 Pemasangan ESC pada *Tricopter*

Sumber: perancangan

4.2.3 Rangkaian Motor Servo

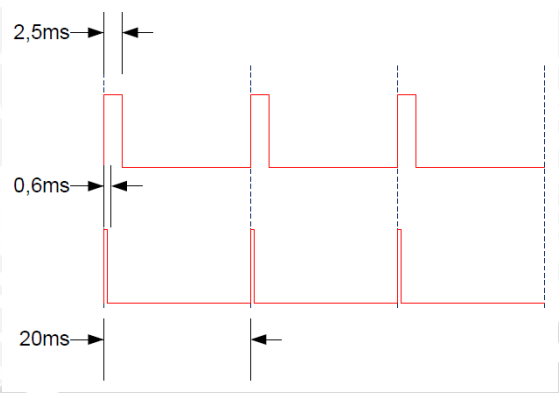
Servo memiliki tiga buah kaki atau pin yaitu pin kontrol, pin Vcc dan pin GND. Komunikasi mikrokontroler dengan motor servo ditunjukkan dalam Gambar 4.4



Gambar 4. 4 Komunikasi Mikrokontroler dengan Motor Servo

Sumber: perancangan

Motor servo memiliki periode sebesar 20ms dan motor ini hanya dapat berputar antara 0° sampai 180°. Dengan mengubah kondisi logika *high* dari *duty cycle* akan dapat mengatur besarnya sudut servo. Untuk menghasilkan sudut 0° dibutuhkan logika *high* sebesar 1ms dan sudut 180° sebesar 2ms. Pulsa minimal dan maksimal motor servo ditunjukkan dalam Gambar 4.5.



Gambar 4. 5 Pulsa Maksimal dan Minimal Motor Servo

Sumber: perancangan

Motor servo merupakan komponen utama untuk menggerakkan ekor *tricopter*. Dalam perancangan motor servo pada ekor tricoper, sudut maksimal yang dibutuhkan adalah ± 60^o. dengan ketentuan arah putar ke kiri 30^o dan ke kanan 30^o dengan pusat sudut 90^o seperti pada perancangan arah gerak motor servo dalam Gambar 4.6. Untuk mendapatkan sudut -30^o dibutuhkan logika *high* sebesar 1,666ms dan sudut 30^o dibutuhkan logika *high* 1,333 ms. Dan untuk sudut 0^o dibutuhkan logika *high* 1,5ms. Perhitungan dapat dilihat dalam persamaan (4.1), (4.2), dan (4.3)

$L(120^o) = 1000 + \frac{100}{18}$ x sudut……..(4.1)

$= 1000 + \frac{100}{18} \times 120^o$

= 1,666 ms

$L(90^o) = 1000 + \frac{100}{18}$ x sudut……..(4.2)

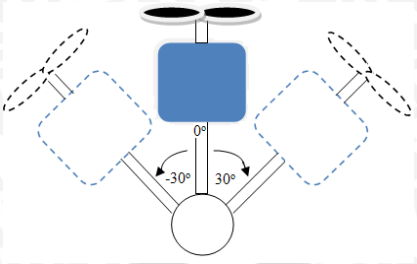
$= 1000 + \frac{100}{18} \times 90^o$

= 1,5 ms

$L(60^o) = 1000 + \frac{100}{18}$ x sudut……..(4.3)

$= 1000 + \frac{100}{18} \times 60^o$

= 1,333 ms

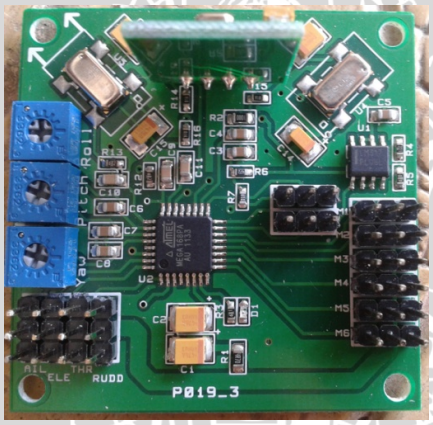


Gambar 4. 6 Arah Gerak Ekor *Tricopter*

Sumber : perancangan

4.2.4 KKmulticopter 168PA

KKmulticopter merupakan modul multicopter yang terdiri dari beberapa komponen. Komponen-komponen tersebut adalah mikrokontroler ATMega168, 3 sensor gyro ENC 03M, 3 potensiometer Bourns 3362P, LM317, resistor, capasitor, 3 pin ISP header 2 buah, 4 pin RX header 3 buah, dan 6 pin ESC header 3 buah. Modul KKmulticopter dapat dilihat dalam Gambar 4.7.

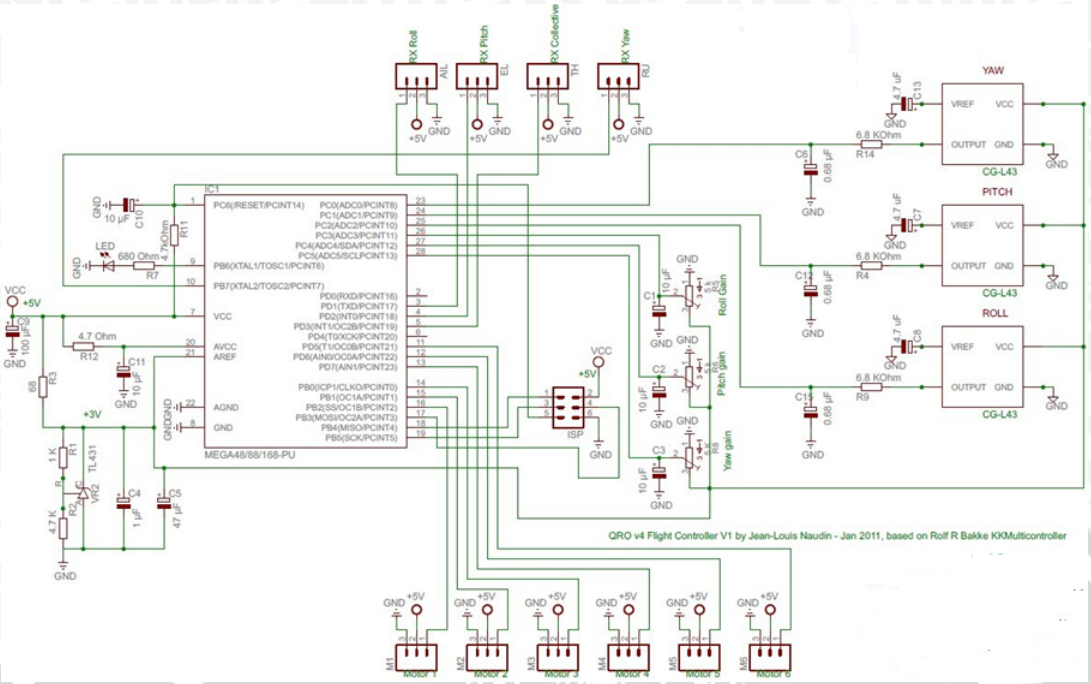


Gambar 4. 7 KKmulticopter 168 PA

Sumber: perancangan

4.2.4.1 Rangkaian *Board* Kkmulticopter

Board KKmulticopter merupakan rangkaian utama dari *tricopter*. Pada pengendalian *tricopter* digunakan mikrokontroler ATMega168. Rangkaian board KKmulticopter ditunjukkan dalam Gambar 4.8



Gambar 4. 8 Rangkaian *board* Kkmulticopter

Sumber: http://diydrones.com

Mikrokontroler ATmega 168PA memiliki 28 jalur yang dapat di program menjadi masukan atau keluaran. Penjelasan fungsi pin masukan dan keluaran mikrokontroler pada perancangan dapat dilihat dalam Tabel 4.1.

Tabel 4. 1 Fungsi Pin Mikrokontroler

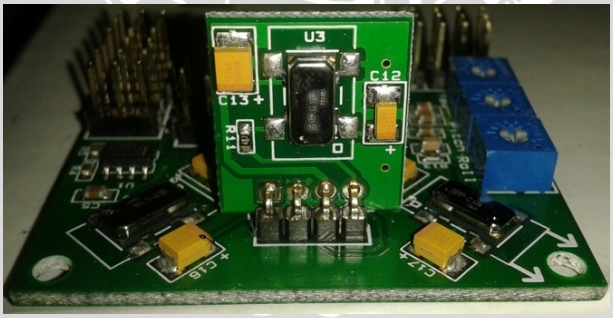
No	Pin	Fungsi
1	PC0	Jalur masukan *gyro yaw*
2	PC1	Jalur masukan *gyro pitch*
3	PC2	Jalur masukan *gyro roll*
4	PC3	Jalur masukan potensiometer *roll gain*
5	PC4	Jalur masukan potensiometer *pitch gain*
6	PC5	Jalur masukan potensiometer *yaw gain*
7	PD1	Jalur masukan Rx *roll*
8	PD2	Jalurmasukan Rx *pitch*

9	PD3	Jalurmasukan Rx *collective*
10	PB7	Jalurmasukan Rx *yaw*
11	PD7	Jalur keluaran motor servo
12	PB0	Jalur keluaran motor 3
13	PB1	Jalur keluaran motor 2
14	PB2	Jalur keluaran motor 1

Sumber :Perancangan

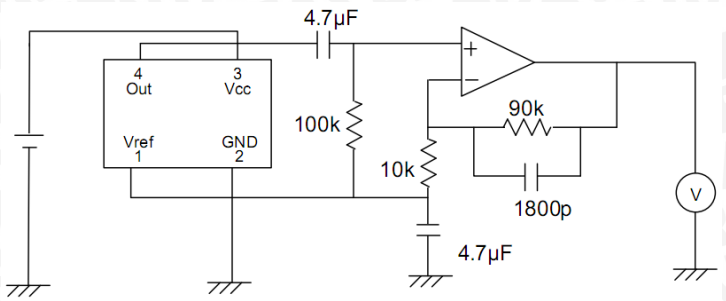
4.2.4.2 Sensor gyro ENC 03M

Sensor yang digunakan dalam *tricopter* ini adalah sensor *gyro* type *piezoelectric*. Sensor ini mendeteksi kecepatan sudut yang memanfaatkan gaya *Coriolis* pada suatu benda yang bergetar dan akan menunjukkan perubahan kecepatan sudut yang terjadi pada bidang sensor. Sensor dalam *tricopter* ini terletak dalam modul Kkmulticopter 168PA. Jenis sensor *gyro* yang digunakan adalah tipe ENC-03M produksi Murata MFG. Co.,Ltd dengan *single* axis, jadi ada 3 sensor *gyro* yang digunakan untuk *tricopter* ini agar dapat mendeteksi 3 keadaan yaitu *pitch*, *roll* dan *yaw* seperti ditunjukkan dalam Gambar 4.9.



Gambar 4. 9 Peletakan Sensor *Gyro* Enc 03m pada *Tricopter*

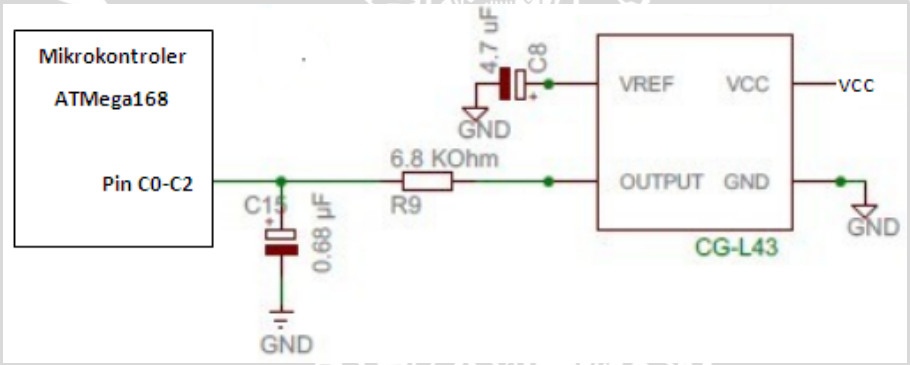
Sumber: perancangan



Gambar 4. 10 Rangkaian Sensor *Gyro* ENC 03M

Sumber: Datasheet ENC 03M

Gambar rangkaian sensor *gyro* ditunjukkan dalam Gambar 4.10. Tegangan sumber yang digunakan pada rangkaian sensor *gyro* merupakan tegangan dari *board* KKmulticopter sekitar 5 V. Berdasarkan *datasheet* ENC-03M, pin 1 merupakan tegangan referensi (Vref), pin 2 dihubungkan ke *ground*, pin 3 dihubungkan ke tegangan Vcc KKmulticopter, dan pin 4 merupakan *output* sensor yang masuk ke mikrokontroler. Gambar 4.11 menunjukkan komunikasi sensor *gyro* dengan mikrokontroler.



Gambar 4. 11 Komunikasi Sensor *Gyro* dengan Mikrokontroler

Sumber: perancangan

4.2.5 Motor *Brushless* dan *Propeller*

Kombinasi antara motor *brushless* dan *propeller* harus tepat agar mampu menghasilkan gaya angkat (*thrust*) yang sesuai dengan berat keseluruhan *tricopter*. Perhitungan mengenai *thrust* yang dihasilkan sebuah *propeller* didasarkan pada persamaan 4.1 berikut ini:

$$T = cT \frac{4\rho r^4}{\pi^2} \omega^2 \quad \text{.....4.1}$$

dimana : T = *thrust propeller* (N)

cT = koefisien *propeller* (0.1154 untuk EPP 1045)

ρ = kerapatan udara (1.225 kg/m^3)

ω = kecepatan anguler *propeller* (rad/s)

r = jari-jari *propeller*(m)

(Jorge MB Domingues, 2009)

Dalam skripsi ini menggunakan EPP 1045 dengan motor *brushless* 750kV. EPP 1045 memiliki diameter 25.4 cm. Sementara motor *brushless* 750kV mampu menghasilkan putaran hingga 7500 rpm dengan tegangan masukan 10 V. Dari persamaan 4.1 dibutuhkan koefisien ω, sedangkan data yang ada adalah nilai kecepatan (rpm) maka untuk mendapatkan ω diperlukan rumus seperti dalam persamaan 4.2. Tabel 4.2 menunjukkan *thrust* yang dihasilkan setiap kenaikan kecepatan motor *brushless*.

$$\omega = \frac{2\pi}{60} \times kecepatan \quad \text{.....4.2}$$

Tabel 4. 2 Perhitungan Thrust Motor *Brushless*

RPM	ω (rad/s)	*Thrust* (N)	*Thrust* x 3 (N)
500	52,33	0,041	0,123
1000	104,66	0,163	0,49
1500	156,99	0,368	1,104
2000	209,32	0,654	1,962
2500	261,65	1,022	3,066
3000	313,98	1,471	3,065
3500	366,31	2.002	4,413
4000	418,64	2,615	7,845
4500	470,97	3,31	9,93
5000	523,30	4,086	12,258
5500	575,68	4,944	14,832
6000	627,96	5,884	17,652

6500	680,29	6,906	20,718
7000	732,62	8,009	24,027
7500	784,95	9,193	27,579

Sumber: perancangan

Jika berat maksimal *tricopter* mencapai 1.1 Kg, maka perhitungan gaya berat *tricopter* adalah :

$F = ma$……....4.2

dengan : F = gaya berat (N)

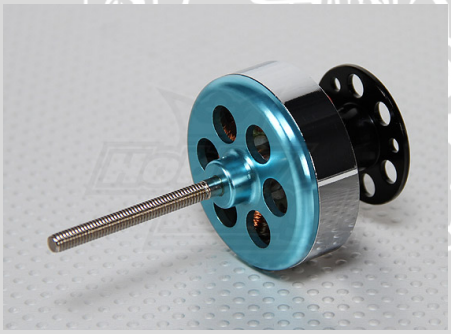
m = massa benda (Kg)

a = gravitasi bumi (9,81 m/s^2)

F = 1.1 Kg x 9.81 m/s^2

= 10,79 N

Hal ini berarti *thrust* minimal yang harus dihasilkan oleh ketiga *propeller* adalah 10,79 N. Sesuai dengan perhitungan pada Tabel 4.2, pada kecepatan 5000 RPM menghasilkan *thrust* 12,258 N. Maka dapat dipastikan bahwa pada kecepatan tersebut, *thrust* yang dihasilkan sudah mampu membuat *tricopter hover*. Motor brushless DT750kV ditunjukkan dalam Gambar 4.12 sedangkan *propeller* EPP1045 ditunjukkan dalam Gambar 4.13



Gambar 4. 12 Motor *Brushless* DT750kV

Sumber: n47-hobby.com

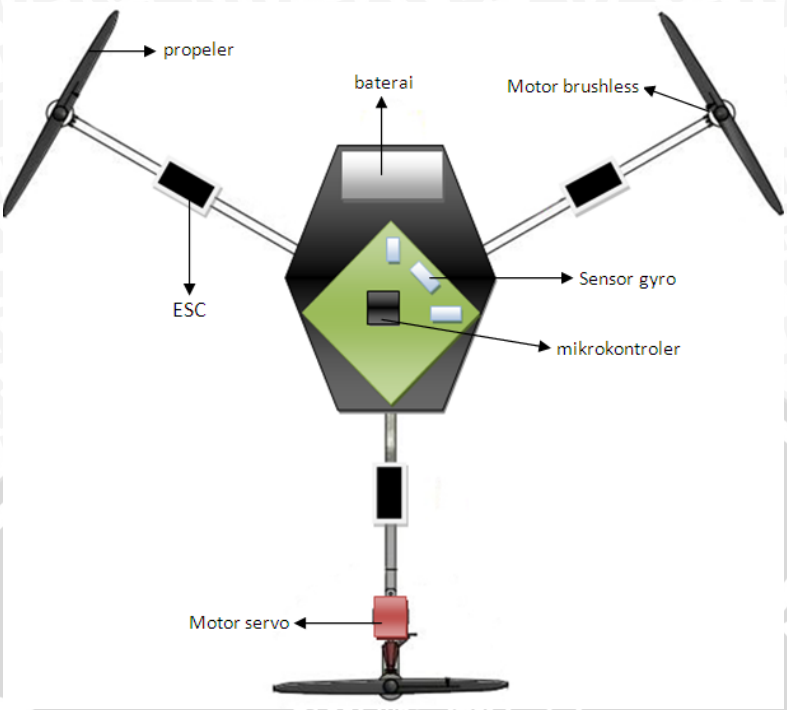
Gambar 4. 13 *Propeller* EPP1045 CW dan CCW

Sumber: quadcopters.co.uk

Pada skripsi ini menggunakan motor *brushless* dengan kV *rating* 750 dengan masukan tegangan maksimum 11.1 volt. Hal ini berarti motor *brushless* mampu menghasilkan putaran hingga 8325 RPM atau setara dengan 871,35 rad/s. Hal ini cukup untuk memenuhi nilai RPM minimal untuk melakukan gaya angkat.

4.3 Perancangan Mekanik *Tricopter*

Tricopter ini tersusun dari 3 batang kayu dengan panjang 50 cm dan dirangkaikan dengan sudut 120^{o} pada papan berdiameter 8 cm untuk letak *board* KKmulticopter 168PA. *Tricopter* memiliki 3 motor *brushless* yang diletakkan pada ujung lengan *tricopter* dengan posisi berdiri dan memiliki 1 motor servo diletakkan pada lengan *tricopter* bagian ekor yang digunakan untuk gerakan *yaw*. *Board* KKmulticopter diletakkan di *centre of gravity tricopter* (titik setimbang *tricopter*). Bentuk mekanik *tricopter* dapat dilihat dalam Gambar 4.14. Bentuk mekanik perancangan motor servo ditunjukkan dalam Gambar 4.15.



Gambar 4. 14 Bentuk Mekanik *Tricopter*

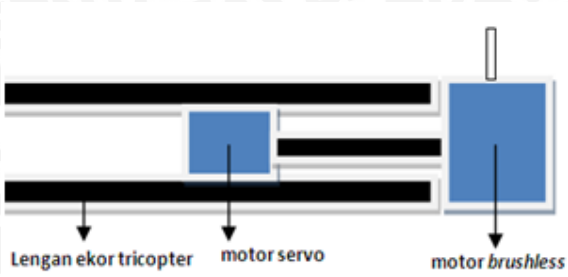
Sumber: perancangan

Spesifikasi utama mekanik *tricopter*:

1. *Tricopter* menggunakan board KKmulticopter 168 PA untuk pengontrolannya.
2. Motor *brushless* DT750 kV.
3. ESC 18A Tower pro.
4. *Propeller* 10 x 4,5
5. Motor servo Thunder Blue 16,5 gr
6. Baterai Lipo 3 C 2200mA 11,1 V

Alat pendukung:

1. *Receiver*
2. *Remote control*



Gambar 4. 15 Bentuk Mekanik Motor Servo pada Ekor *Tricopter*

Sumber : perancangan

4.4. Perancangan PID dan implementasinya pada mikrokontroler

Persamaan untuk kontrol PID adalah:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{de(t)}{dt} \tag{4.4}$$

atau

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)\,dt + T_d \frac{de(t)}{dt} \right) \tag{4.5}$$

Dengan:

u(t) = sinyal *output* pengendali PID

Kp = konstanta proporsional

Ti = waktu integral

Td = waktu derivatif

Ki = konstanta integral (Kp/Ti)

Kd = konstanta derivatif (Kp.Td)

e(t) = sinyal *error* = referensi – keluaran *plant* = *set point* – nilai sensor.

Untuk persamaan PID no.(4.4) merupakan PID bentuk *independent* dan persamaan (4.2) merupakan PID bentuk *dependent.* Istilah tersebut mengacu kepada ketergantungan setiap suku persamaan terhadap nilai Kp. Untuk persamaan (4.4), jika dilakukan perubahan nilai pada konstanta proporsional (Kp) maka tidak akan mempengaruhi konstanta parameter lainnya. Sedangkan untuk persamaan (4.5), dengan mengubah nilai Kp maka akan merubah nilai dari parameter-parameter lainnya. Disini akan digunakan persamaan PID bentuk *independent.* Jika ingin menggunakan persamaan *dependent*, maka tinggal memasukkan nilai dari Ki=Kp/Ti dan Kd=Kp.Td. Persamaan (4.4) dan (4.5) merupakan persamaan dalam kawasan waktu *continuous* (analog). Sedangkan agar persamaan-persamaan tersebut dapat direalisasikan dalam bentuk pemrograman, maka persamaan

dalam kawasan waktu *continuous* tersebut harus didiskritisasi terlebih dahulu (kawasan *digital*). Berikut adalah diskritisasi menggunakan metode numerik *rectangular* mundur (*backward rectangular*).

Penentuan besar Kp, Ki, dan Kd pada skripsi ini adalah dengan melakukan tuning menggunakan metode *try and error* sehingga memperoleh respon yang optimal. Persamaan (4.6) sampai pada persamaan (4.8) merupakan persamaan dalam kawasan waktu *continuous* (analog). Sedangkan agar persamaan- persamaan tersebut dapat direalisasikan dalam bentuk pemrograman, maka persamaan dalam kawasan waktu *continuous* tersebut harus didiskritisasi terlebih dahulu (kawasan *digital*). Berikut adalah diskritisasi menggunakan metode numerik *rectangular* mundur (*backward rectangular*).

4.4.1 Kontrol Proporsional

Jika dari persamaan (2-1) didiskritisasi maka akan menjadi:

$$u(k) = K_p e(k) \tag{4.6}$$

Berikut ini contoh *list* program dalam mikrokontroler yang menunjukkan kendali proporsional:

```
//kendali proporsional
error=set_point-nilai_sensor;
outP=Kp*error;  //nilai Kp ditentukan melalui tuning
```

4.4.2 Kontrol Integral

Jika dari persamaan (2-3) didiskritisasi maka akan menjadi:

$$u(k) = K_i \sum_{i=0}^{k} e(i)\, T_c$$

$$u(k) = K_i T_c \sum_{i=0}^{k} e(i) = K_i T_c \left[e(0) + e(1) + ... + e(k-1) + e(k) \right]$$

$$u(k) = K_i T_c \left[e(k-1) + e(k) \right] \tag{4.7}$$

Tc= waktu sampling atau waktu cuplik *(Sampling time)*

Berikut ini contoh *list* program dalam mikrokontroler yang menunjukkan kendali integral:

```
//kendali integral
errorI=error+error_sblmI;
outI=Ki*errorI*Tc;  //nilai Ki ditentukan melalui tuning
error_sblmI=errorI;
```

4.4.3 Kontrol Derivatif

Jika dari persamaan (2-4) didiskritisasi dengan menggunakan cara yang sama seperti kontrol integral maka akan menjadi:

$$u(k) = K_d \frac{e(k) - e(k-1)}{T_c} \qquad (4.8)$$

Tc= waktu sampling atau waktu cuplik *(Sampling time)*

Berikut ini contoh *list* program dalam mikrokontroler yang menunjukkan kendali derivatif:

```
//kendali Deferensial
errorD=error-error_sblmD;
outD=(Kd*errorD)/Tc;   //nilai Kd ditentukan melalui tuning
error_sblmD=error;
```

4.4.4 Kontrol Proporsional Integral Diferensial (PID)

Terakhir, dari persamaan (2-5) didiskritisasi dengan menggunakan cara yang sama maka akan menjadi:

$$u(k) = K_p e_k + K_1 T \sum_{0}^{K} ek + \frac{1}{T} K_d \, (e_k - e_{k-1})$$

Berikut ini contoh *list* program dalam mikrokontroler yang menunjukkan kendali PID:

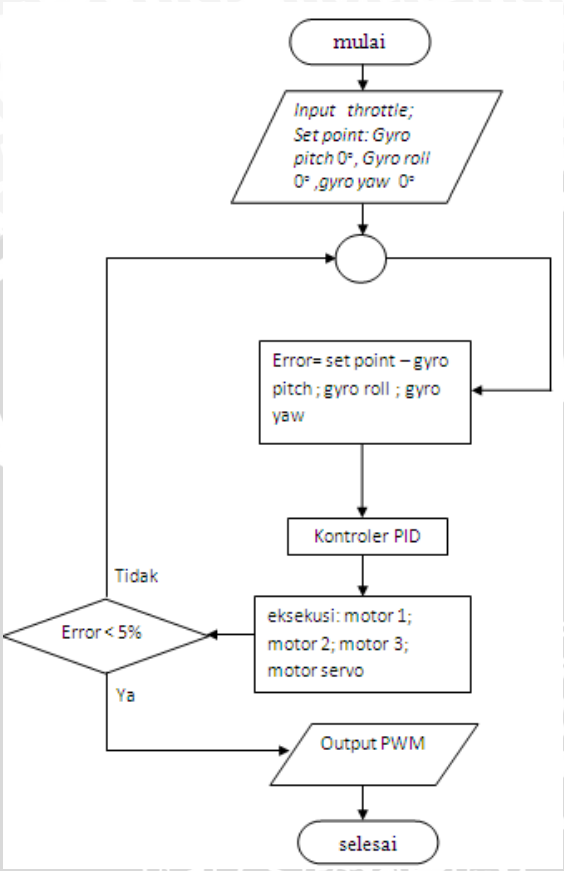
```
//kontrol PID, merupakan gabungan dari kontrol P, I dan D
outPID=outP+outI+outD;
```

Agar lebih mengerti dari persamaan (4.6), (4.7) dan (4.5) yang telah dipaparkan, berikut penjelasannya:

Derivatif (de/dt) adalah suatu operator matemamis dalam kawasan *continuous* jika didiskritisasi maka akan menjadi limit, yang merupakan operator matematis dalam kawasan diskrit. Di mana fungi dari operator limit adalah mengurangi nilai ke k dengan nilai ke k-1. Berdasarkan perhitungan di atas variabel *error* (e) yang di derivatifkan, atau dengan kata lain *error* yang sekarang dikurangi *error* yang sebelumnya.

Waktu *sampling* adalah lamanya waktu yang digunakan untuk mencuplik atau men *sampling* nilai dari sensor. Nilai dari sensor ini berguna untuk mendapatkan sinyal *error* (*error*(e)=*set point*-nilai sensor. Di mana waktu *sampling* ini sangat berpengaruh pada kesensitifan sistem yang akan dikontrol.

Parameter PID dalam perancangan *tricopter* ini menggunakan metode *trial and error* dengan cara memutar potensiometer pada *board* KKmulticopter. *Flowchart* perancangan pengontrolan *tricopter* ditunjukkan dalam Gambar 4.16.



Gambar 4. 16 Flowchart Pengontrolan *Tricopter*

Sumber: perancangan

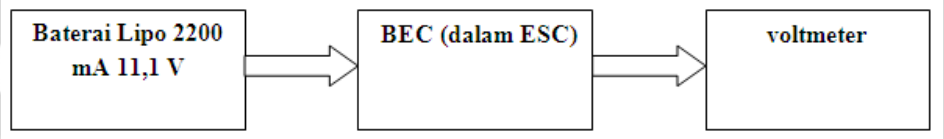
BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan yang telah dilakukan. Pengujian dilakukan per blok sistem kemudian secara keseluruhan. Pengujian per blok dilakukan untuk mempermudah analisis apabila alat tidak bekerja sesuai dengan perencanaan. Adapun pengujian yang dilakukan sebagai berikut:

- Pengujian catu daya sistem.
- Pengujian sensor *gyro ENC 03M* .
- Pengujian gaya dorong motor *brushless*
- Pengujian motor servo
- Pengujian keluaran sinyal *board* KKmulticopter.
- Pengujian keseluruhan *tricopter*

5.1 Pengujian catu daya sistem

Pengujian rangkaian catu daya sistem bertujuan untuk mengetahui apakah catu daya dari sumber sudah sesuai dengan *board* mikrokontroler. Rangkaian catu daya ini berupa BEC (*battery eliminator circuit*) dalam ESC (*electronic speed control*). Pengujian dilakukan dengan menghubungkan masukan rangkaian catu daya dengan *power supply* dan keluarannya diukur dengan multimeter yang difungsikan sebagai *voltmeter* untuk mengetahui besar tegangan keluaran rangkaian. Diagram blok pengujian ditunjukkan dalam Gambar 5.1.



Gambar 5. 1 Diagram Blok Pengujian Catu Daya Sistem

5.1.1 Peralatan pengujian

1. ESC 18 A Tower Pro.
2. Voltmeter digital.
3. Baterai Lipo 3 *Cell* 2200 mA.

5.1.2 Prosedur pengujian

1. Merangkai ESC dengan baterai Lipo 3 *Cell* 2200 mA.
2. Mengukur tegangan keluaran BEC (*battery eliminator circuit*) dengan voltmeter digital seperti dalam Gambar 5.2.

5.1.3 Hasil pengujian

Gambar 5. 2 Pengujian BEC pada ESC 18 A Tower Pro.

Power supply yang digunakan dalam *tricopter* ini berupa baterai Lipo 3 *Cell* 2200 mA 11,1 V. Dari hasil pengujian, diperoleh besar tegangan keluaran rangkaian catu daya adalah 5,07 V, nilai tegangan ini digunakan untuk mengaktifkan KKmulticopter 168PA yang di gunakan untuk mencatu mikrokontroler ATMega168, sensor *gyro* ENC 03M, dan *recevier remote control*.

5.2 Pengujian Sensor *Gyro ENC 03M*

Pengujian ini bertujuan untuk mengetahui berfungsi atau tidaknya sensor *gyro ENC 03M* dalam mendeteksi perubahan kecepatan sudut. Untuk pengujian respon rangkaian sensor *gyro* digunakan fasilitas *Hyperterminal* pada Windows XP untuk melihat data keluaran sensor.

5.2.1 Peralatan pengujian

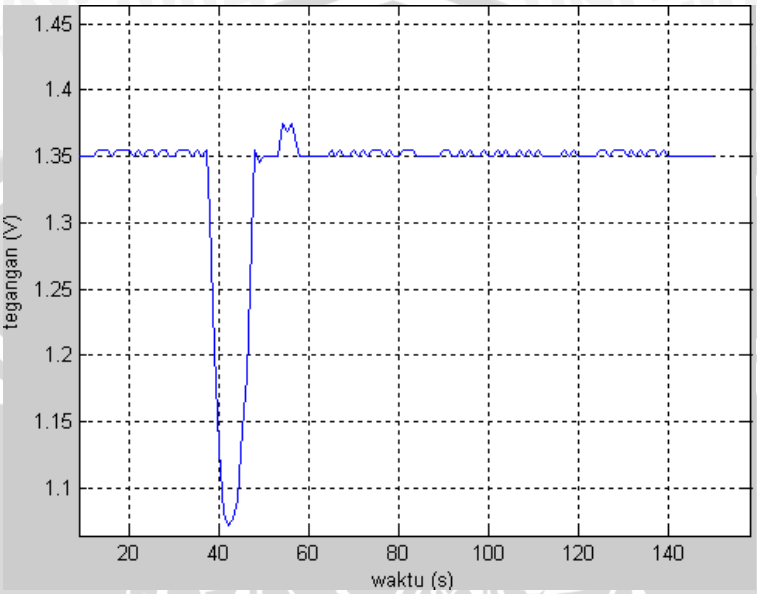
1. Sensor *gyro* ENC 03M pada board Kkmulticopter168PA.
2. *Converter* USB *to* RS232 PL2303
3. Laptop dengan OS (*Operating System*) Windows XP
4. Beberapa buah kabel penghubung.
5. Matlab

5.2.2 Prosedur pengujian

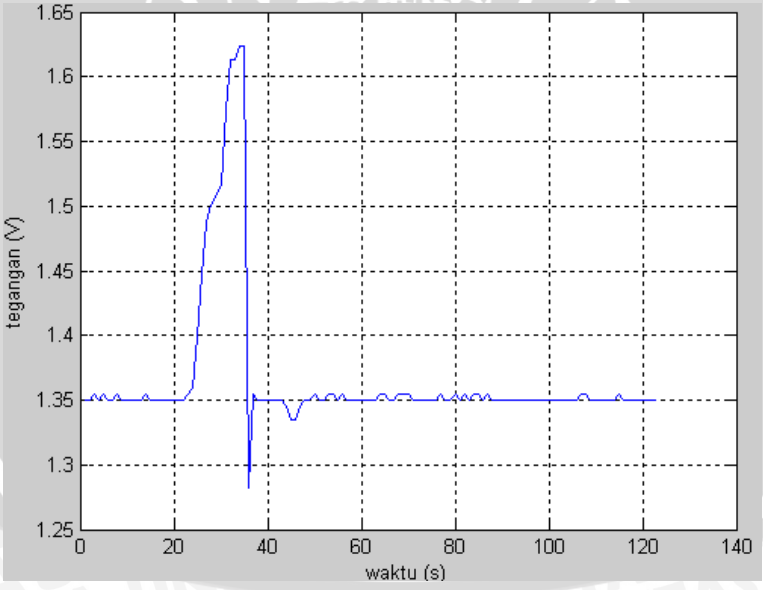
1. Menghubungkan 4 pin keluaran motor dengan pin Rx pada board Kkmultikopter168.
2. Menghubungkan pin sinyal M6 dengan pin Rx pada RS232 dan menghubungkan pin ground M6 dengan pin ground RS323.
3. Menghubungkan USB ke port USB pada laptop.

4. Menggerakkan sensor *gyro* ENC 03M dengan gerakan *roll* ke kiri dan ke kanan.
5. Menampilkan data keluaran.
6. Menge-*plot* data keluaran menggunakan matlab .

5.2.3 Hasil pengujian



Gambar 5. 3 Grafik Keluaran Sensor Gyro ENC 03M saat rotasi CW



Gambar 5. 4 Grafik Keluaran Sensor Gyro ENC 03M saat rotasi CCW

Gambar 5.5 menunjukkan grafik keluaran sensor gyro ENC 03M saat rotasi CW (*clockwise*). Gambar 5.6 menunjukkan grafik keluaran sensor gyro ENC 03M saat rotasi

CCW (*counter clockwise*). Dari kedua grafik tersebut, saat tidak melakukan rotasi tegangan keluaran rata-rata sekitar 1,35 V.

5.3 Pengujian Motor *Brushless* DT750

Pengujian motor Brushless ini bertujuan untuk mengetahui gaya angkat yang dihasilkan motor *brushless* DT750 terhadap pergerakan tuas *throttle*. Dengan mengetahui gaya angkat, dapat dipastikan kapan *tricopter* mampu untuk melakukan gerakan hover. Pengujian ini menggunakan hanya 1 buah motor *brushless* dalam menguji gaya angkat. Pengujian gaya angkat motor *brushless* ditunjukan pada gambar 5.7.

Gambar 5. 5 Foto Pengujian Motor *Brushless* DT750.

5.3.1 Peralatan pengujian

1. ESC 18 A Tower Pro.
2. Baterai Lipo 3 *Cell* 2200 mA 11,1 V.
3. 1 buah Motor *brushless* DT750 KV.
4. *Propeller* 10x3,5.
5. TX (*transmitter / remote control*) dan RX (*receiver*).
6. Timbangan gantung digital.
7. Tachometer

5.3.2 Prosedur pengujian ESC 18 A Tower Pro

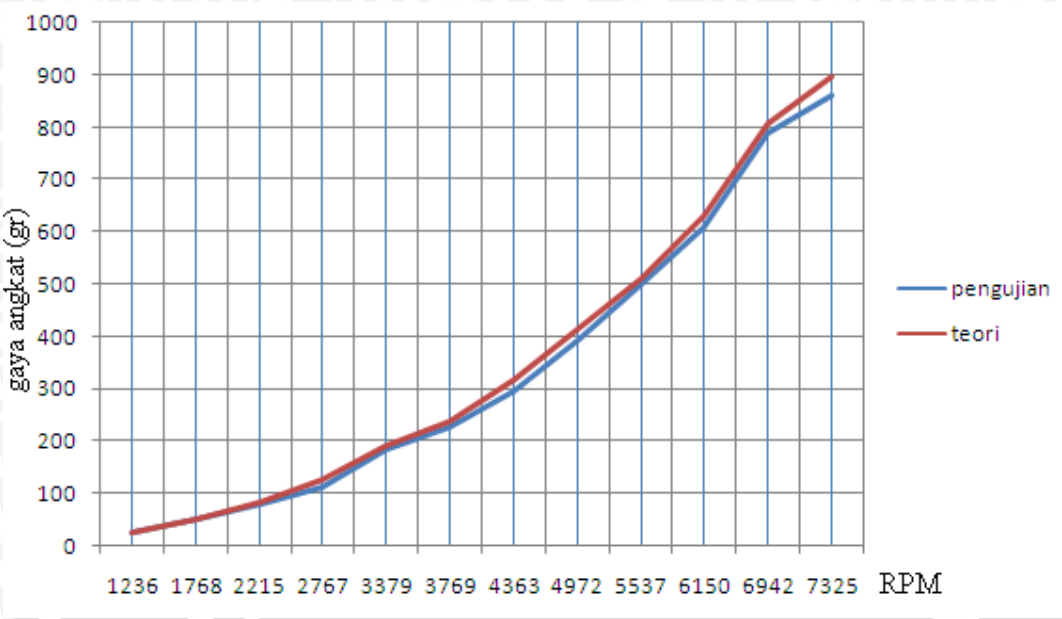
1. Menghubungkan ESC 18 A Tower Pro dengan baterai Lipo 3 *Cell* 2200 mA.
2. Menghubungkan 3 kabel (Vcc, *ground*, dan *control*) ESC pada *receiver* (RX).

3. Menghubungkan 3 kabel motor *brushless* ke kabel *output* ESC.
4. Memasang *propeller* 10x3,5 pada motor *brushless*.
5. Merangkai motor *brushless* pada lengan *tricopter* kemudian mengkaitkan pada timbangan digital tepat di bawah motor *brushless.*
6. Menggerakkan tuas *throttle* pada *remote control* (TX).
7. Membaca kecepatan motor dengan tachometer dan gaya angkat dengan menggunakan timbangan gantung digital.
8. Memasukkan data hasil pengukuran ke dalam Tabel 5.1.

5.3.3 Tabel hasil pengujian

Tabel 5. 1 Hasil Pengujian Gaya Angkat Motor *Brushless* terhadap Penambahan *Throttle*

No	RPM	Gaya angkat (gr)	gaya angkat teori (gr)
1	1236	24,1	25,4
2	1768	51	52,1
3	2215	79,2	81,8
4	2767	111	127,6
5	3379	183,5	190,4
6	3769	225,3	236,9
7	4363	295	317,4
8	4972	391,9	412,2
9	5537	501,8	511,3
10	6150	608,3	630,8
11	6942	787,5	803,7
12	7325	861,2	894,98



Gambar 5. 6 Grafik Perbandingan untuk Pengujian dan Teori

Berdasarkan grafik yang ditunjukkan dalam Gambar 5.8, dapat dilihat bahwa perbandingan antara pengujian dan teori terdapat selisih yang kecil. Hal ini berarti gaya angkat yang dihasilkan motor dc *brushless* telah sesuai dengan yang direncanakan sehingga dapat bekerja pada sistem.

5.4 Pengujian Motor Servo

Pengujian motor servo ini bertujuan untuk mengetahui sudut yang dihasilkan motor servo pada ekor *tricopter*. Motor servo ini digunakan untuk mengontrol gerakan *yaw* (menggeleng) pada *tricopter.*

5.4.1 Peralatan pengujian

1. ESC 18 A Tower Pro.
2. Baterai Lipo 3 *cell* 2200 mA 11,1 V.
3. Servo *Thunder Blue* 15 gr.
4. Busur derajat.

5.4.2 Prosedur pengujian

1. Menghubungkan baterai Lipo 3 *cell* dengan ESC 18 A Tower Pro.
2. Menghubungkan 3 kabel kontrol ESC *receiver*.
3. Menghubungkan 3 kabel control motor servo ke *receiver.*
4. Menggerakkan tuas *rudder* pada *remote control* (TX) sesuai dengan Tabel 5.2.
5. Membaca besarnya sudut servo dengan busur derajat.

6. Memasukkan data hasil pengukuran ke dalam Tabel 5.2.

5.4.3 Tabel pengujian

Tabel 5. 2 Hasil Pengujian Motor Servo dengan *Output* Sudut terhadap Penambahan Tuas *Rudder*

no	Tuas *rudder* (%)	Besar sudut servo
1	-100	32
2	-80	26
3	-60	18
4	-40	12
5	-20	6
6	0	0
7	20	7
8	40	11
9	60	19
10	80	27
11	100	33

Dari Tabel 5.2 nilai tuas *rudder* negatif menggerakkan tuas ke arah kiri, sudut yang dihasilkan 0-32^o dengan acuan sumbu lurus (vertikal) ke kanan. Sedangkan nilai tuas *rudder* positif merupakan *tuning* ke arah kanan sudut yang dihasilkan 0-33^o dengan acuan sumbu lurus (vertikal) ke kiri. Besar sudut maksimal yang mampu diputar oleh servo adalah 66^o, sudut ini merupakan hasil dari sinyal keluaran maksimal dari *radio control*. *Error* masing-masing adalah 2^o saat ke kanan dan 3^o saat gerakan ke kiri dengan hasil tersebut servo dapat bekerja dengan baik. Pengujian motor servo dapat dilihat dalam Gambar 5.7.

Gambar 5. 7 Pengujian Motor Servo dengan Tuas Rudder 100% ke Kanan Didapatkan Sudut 32^{o}.

5.5 Pengujian *Board* Kkmulticopter 168PA

Pengujian ini bertujuan untuk mengetahui sinyal keluaran dari *board* KKmulticopter 168PA dalam pengontrolan gerakan *hover tricopter* . Sinyal keluaran *board* ini ada 4 channel yaitu untuk mengontrol motor dan motor servo.

5.5.1 Peralatan pengujian

1. Baterai Lipo 3 *cell* 2200 mA 11,1 V.
2. ESC 18 A Tower Pro.
3. E-lab.
4. Komputer
5. Kabel Konektor

5.5.2 Prosedur pengujian

1. Menghubungkan kabel ESC 18 A ke baterai Lipo 3 cell 2200 mA.
2. Menghubungkan 3 kawat ESC ke board KKmulticopter 168PA *channel* ke 5 (1-4 merupakan *channel* yang akan diuji)
3. Menghubungkan 4 pin *output* KKmulticopter yang sinyal control dan *ground* ke E-lab.
4. Menggerakkan *board* KKmulticopter dengan gerakan *pitch, roll,* dan *yaw*.
5. Menampilkan sinyal keluaran dari masing-masing ke komputer dengan *software* e-lab 080.

5.5.3 Hasil pengujian

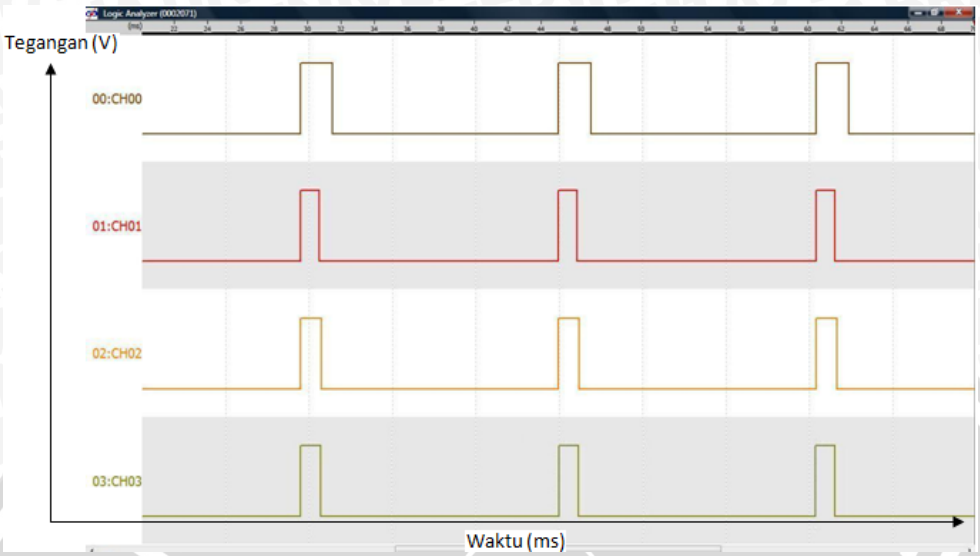
CH00 = untuk mengontrol motor kiri depan

CH01 = untuk mengontrol motor kanan depan

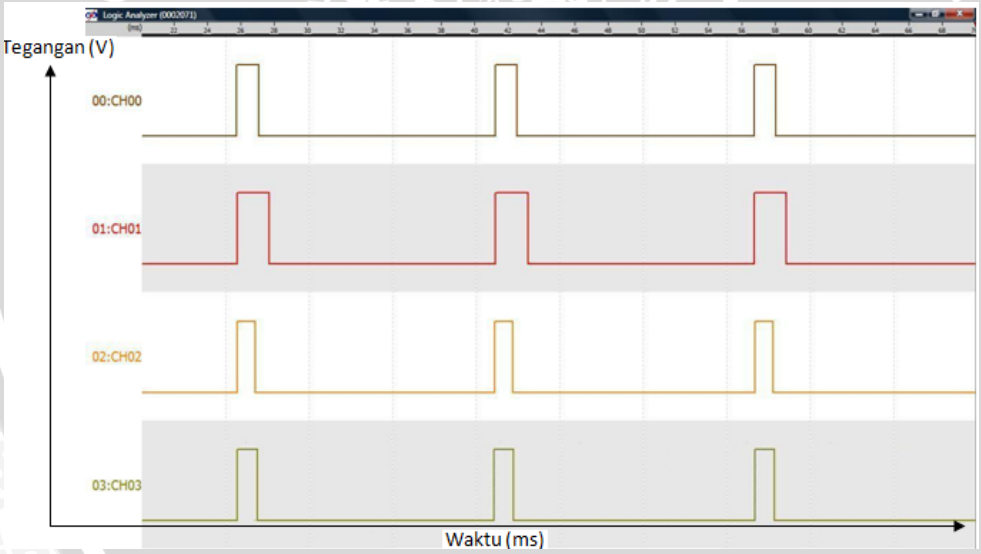
CH02 = untuk mengontrol motor bagian ekor

CH03 = untuk mengontrol motor servo

1. Pengujian *board* KKmulticopter 168 PA dengan gerakan *roll*



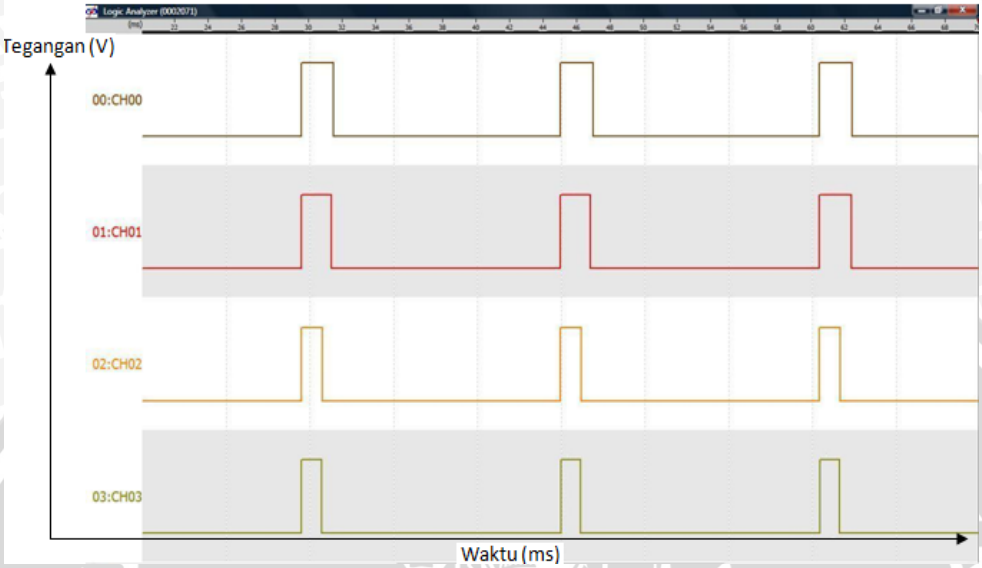
Gambar 5. 8 Bentuk Sinyal Keluaran Saat *Roll* Ke Kiri



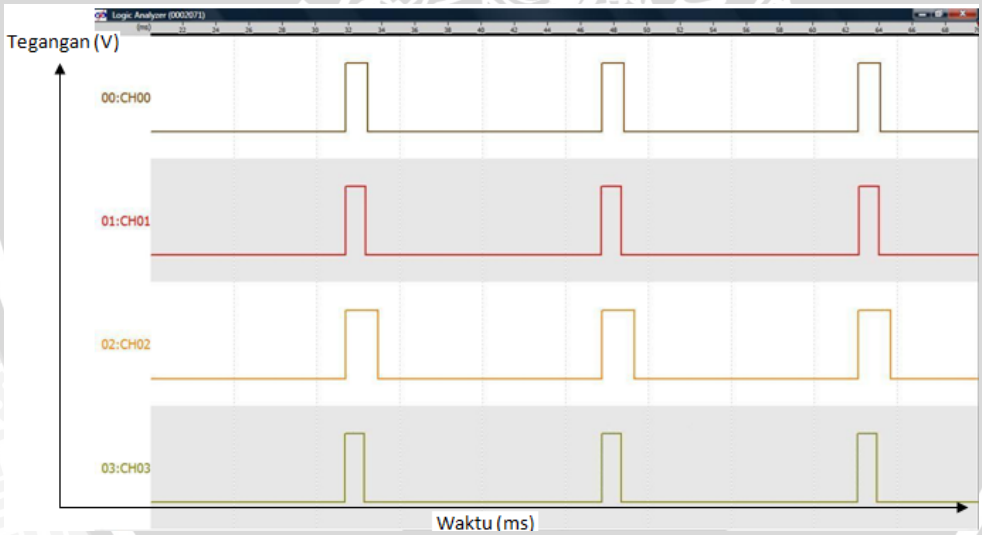
Gambar 5. 9 Bentuk Sinyal Keluaran Saat *Roll* Ke Kanan

Gambar 5.8 menunjukkan sinyal keluaran saat *board* diberi gangguan gerakan *roll* 45^{o}/s ke kiri. Dari gambar 5.8 tersebut lebar pulsa *high* pada CH00 lebih lebar dari CH01dan CH02, artinya sinyal tersebut akan mengontrol motor kiri depan menambah kecepatan. Sedangkan gambar 5.9 menunjukkan sinyal keluaran *board* saat diberi gangguan gerakan *roll* 45^{o}/s ke kanan. Gambar 5.9 lebar pulsa *high* CH01 lebih lebar dari CH00 dan CH02, artinya sinyal tersebut akan mengontrol motor kanan depan menambah kecepatan.

2. Pengujian *board* KKmulticopter 168 PA dengan gerakan *pitch*



Gambar 5. 10 Bentuk Sinyal Keluaran Saat *Pitch* Ke Depan

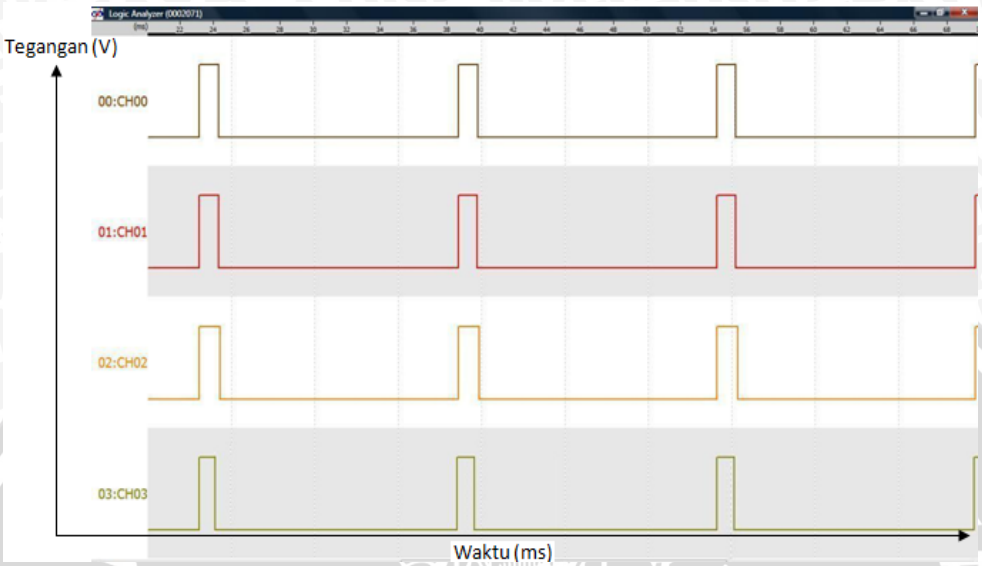


Gambar 5. 11 Bentuk Sinyal Keluaran Saat *Pitch* Ke Belakang

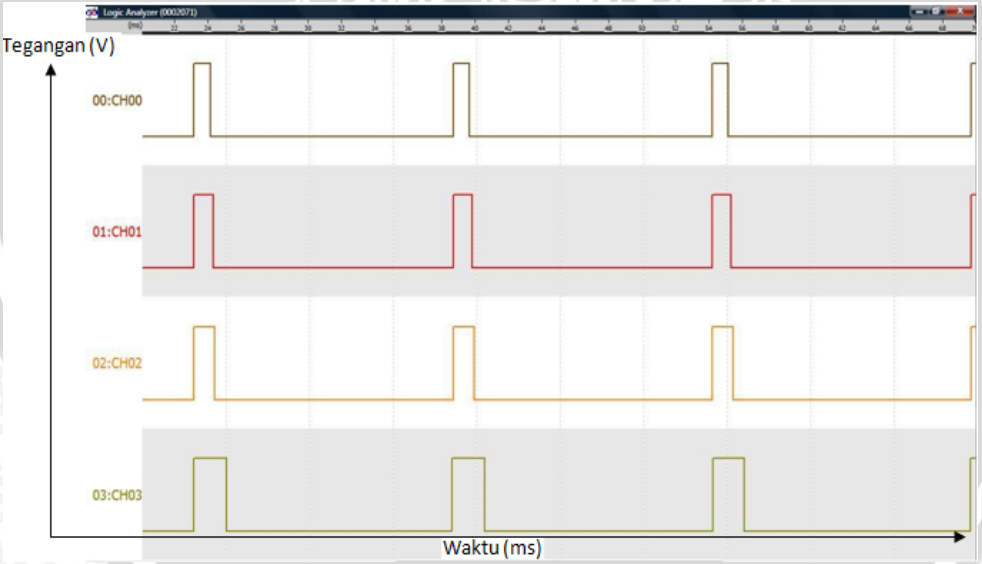
Gambar 5.10 menunjukkan sinyal keluaran saat *board* diberi gangguan gerakan *pitch* 45^{o}/s ke depan. Dari gambar 5.10 tersebut lebar pulsa *high* pada CH00 dan CH01 lebih lebar dari CH02 dan CH03, artinya sinyal tersebut akan mengontrol motor kiri dan kanan depan menambah kecepatan. Sedangkan gambar 5.11 menunjukkan sinyal keluaran *board* saat diberi gangguan gerakan *roll* 45^{o}/s

ke kanan. Gambar 5.11 lebar pulsa *high* CH02 lebih lebar dari CH00 dan CH01, artinya sinyal tersebut akan mengontrol motor belakang menambah kecepatan.

3. Pengujian board kkmulticopter 168 PA dengan gerakan yaw



Gambar 5. 12 Bentuk Sinyal Keluaran Saat *Yaw* searah jarum jam



Gambar 5. 13 Bentuk Sinyal Keluaran Saat *Yaw* berlawanan jarum jam

Gambar 5.12 menunjukkan sinyal keluaran saat *board* diberi gangguan gerakan *yaw* 45^{o}/s searah jarum jam. Dari gambar 5.12 tersebut lebar pulsa *high* pada CH03 lebih lebar, artinya sinyal tersebut akan mengontrol motor servo bergerak ke kanan. Sedangkan gambar 5.13 menunjukkan sinyal keluaran *board* saat diberi gangguan gerakan *yaw* 45^{o}/s berlawanan jarum jam. Gambar 5.13 lebar

pulsa *high* CH03 lebih kecil, artinya sinyal tersebut akan mengontrol motor servo bergerak ke kiri.

5.6 Pengujian keseluruhan sistem

Pengujian ini bertujuan untuk melihat respon kesetimbangan sistem *tricopter* secara keseluruhan yang mencakup gerak *pitch, roll,* dan *yaw*

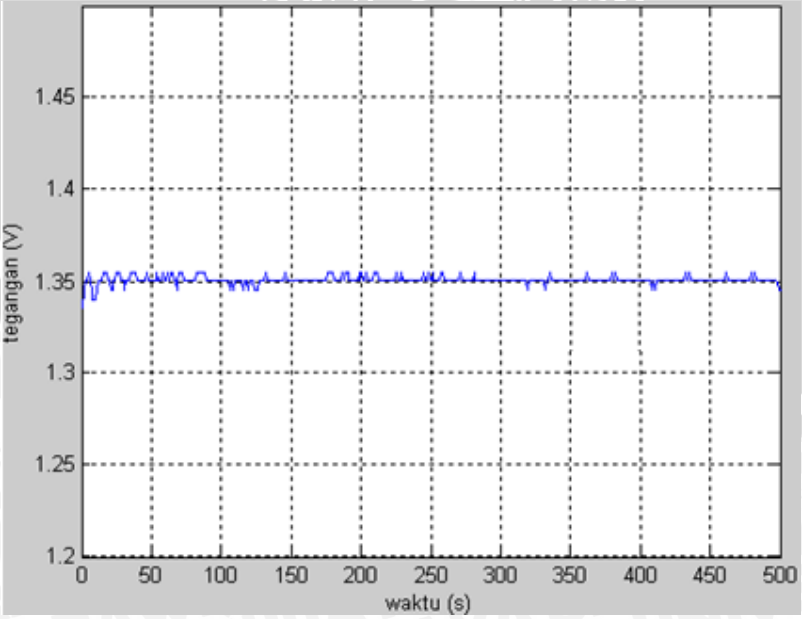
5.6.1 Peralatan Pengujian

- *Converter* USB *to* RS232 PL2303
- Laptop dengan CVAVR
- Beberapa buah kabel

5.6.2 Prosedur Pengujian

Tricopter digantungkan dengan tali sehingga dapat melihat respon pengujian secara per-*axis.* Untuk dapat melihat respon per-*axis*, *pin* Tx pada mikrokontroler dihubungkan dengan *pin* Rx pada *converter* RS232 dan *pin ground* pada mikrokontroler juga dihubungkan dengan *pin ground* pada RS232. *Converter* kemudian dihubungkan pada *port* USB laptop. Parameter kontroler PID dalam *tricopter* ini menggunakan tuning potensiometer pada *board* KKmuticopter *tricopter* dengan Kp= 50%, Ki= 60% dan Kd= 50%.

5.6.3 Hasil Pengujian

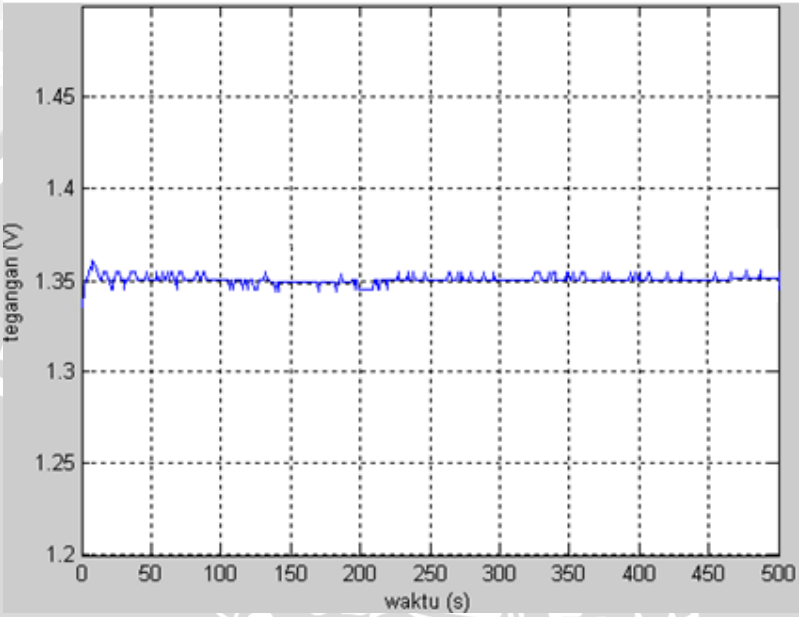


Gambar 5. 14 Respon Sensor *Pitch Tricopter*

Sumber:Pengujian

Gambar 5.14 menunjukkan respon dari sensor *pitch* pada *tricopter*. Dari respon sensor tersebut didapatkan *Error steady state*, **E**ss adalah selisih antara nilai keluaran dengan nilai masukan pada saat kondisi *staedy state*. **E**ss yang didapatkan dari pengujian ini adalah

$$Ess = \frac{1{,}36 - 1{,}35}{1{,}35} \times 100\% = 0{,}74\%$$

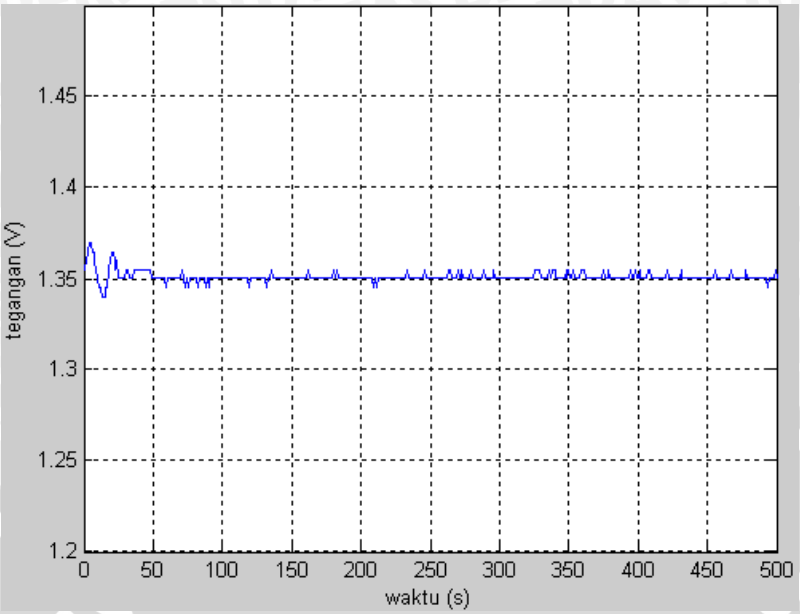


Gambar 5. 15 Respon Sensor *Roll Tricopter*

Sumber:Pengujian

Gambar 5.15 menunjukkan respon dari sensor *roll* pada *tricopter*. Dari respon sensor tersebut didapatkan *Error steady state*, **E**ss adalah selisih antara nilai keluaran dengan nilai masukan pada saat kondisi *staedy state*. **E**ss yang didapatkan dari pengujian ini adalah

$$Ess = \frac{1{,}37 - 1{,}35}{1{,}35} \times 100\% = 1{,}48\%$$



Gambar 5. 16 Respon Sensor *Yaw Tricopter*

Sumber:Pengujian

Gambar 5.12 menunjukkan respon dari sensor *yaw* pada *tricopter*. Dari respon sensor tersebut didapatkan *Error steady state*, **E**ss adalah selisih antara nilai keluaran dengan nilai masukan pada saat kondisi *staedy state*. **E**ss yang didapatkan dari pengujian ini adalah

$$Ess = \frac{1{,}38 - 1{,}35}{1{,}35} \times 100\% = 2{,}22\%$$

Dari hasil yang didapatkan dari ketiga respon sensor *pitch*, *roll* dan *yaw error steady state* yang didapatkan < 5% dari nilai sensor saat 0^o yang ditunjukan dalam persamaan 5.1.

Vo = 1.35 + 0.67 mV/deg/s x 0 deg/s……..….(5.1)

= 1.35 V

Dapat disimpulkan pengontrolan *tricopter* mempunyai respon yang baik.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil pengujian tiap blok dan pengujian sistem secara keseluruhan yang telah dilakukan dalam Bab V, dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem elektronika yang telah dirancang dapat berfungsi dengan baik, hal tersebut ditunjukkan oleh:
   a) Sensor *gyro ENC 03M* menghasilkan respons yang sesuai *datasheet.*
   b) Gaya angkat maksimal yang mampu dihasilkan 1 motor brushless sebesar 861,2 gr. Dengan gaya angkat 3 motor, *tricopter* dengan massa 1100 gr dapat melakukan gerak *hover*.
   c) Sudut servo maksimal pada ekor *tricopter* yang didapatkan dari pengujian adalah $65^{o,}$ dengan sudut masing-masing 32^{o} ke kanan dan 33^{o} ke kiri sudah sesuai dengan perancangan pada bab IV.
   d) Pada pengujian KKmulticopter dengan gerakan *pitch, roll* , dan *yaw* , terjadi kesesuaian antar*a input* yang berupa gerakan *pitch, roll* , dan *yaw* yang diberikan deng*an output* yang dihasilkan. Hal tersebut menunjukkan bahwa KKmulticopter dapat berfungsi dengan baik untuk pengontrolan gerak *hover*.
   e) Pada saat pengujian keseluruhan *tricopter*, didapatkan 3 respon sensor yaitu respon sensor pitch dengan Ess sebesar 0,74%, respon sensor roll dengan Ess sebesar 1,48% dan respon sensor roll dengan Ess sebesar 2,22%.

   Maka sistem elektronika yang telah dirancang dapat berfungsi dengan baik dan menunjang sistem *tricopter* secara keseluruhan. Hal ini ditunjukkan dengan adanya kesesuaian antara tujuan dengan sistem yang telah dibuat dengan error <5%.
2. Mikrokontroler Atmega168 dengan menggunakan kontroler PID yang telah dirancang, sistem *tricopter* mampu mengontrol ketiga motor *brushless* dan motor servo dalam melakukan eksekusi. Parameter PID ditentukan dengan metode *trial and error* dan didapatkan nilai Kp = 50%, Ki = 60%, dan Kd = 50% dari tuas potensiometer *board* KKmulticopter *tricopter*, dengan parameter tersebut sistem mampu melakukan gerak *hover* dengan baik.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan lebih lanjut adalah:

1. Penggunaan sensor yang digunakan selain sensor *gyro* yaitu sensor *accelerometer* agar pembacaan sudut kemiringan lebih tepat sehingga respon pengontrolan *tricopter* semakin baik.
2. Pembuatan mekanik yang baik dan lebih presisi akan membuat *tricopter* semakin stabil dalam pergerakan ketika terbang.

DAFTAR PUSTAKA

Atmel Corporation. 2011. *ATMEGA 168 Series.*

Bakke, Rolf. *KKMulticopter Black Board*.

Brata, Lucas Yuda. 1997. *Perencanaan dan Pembuatan Kontrol Kecepatan Brushless DC Motor 24 V 0.11A*, Fakultas Teknik Universitas Kristen Petra.

Brown, Ward. 2002. *Brushless DC Motor Control Made Easy*. Microchip Technology. Inc.

Domingues, Jorge MB. 2009. *Quadrotor prototype*. Unpublished PhD. Portugal: mechanica engineering, technical university Lisbon.

ENC-03M Sensor. http://www.datasheetarchive.com/ENC-03M-datasheet.html Diakses tanggal 5 Desember 2011.

Gresko, Jim. 2011. *The Rise of the DIY Micro Copter*. http://www.fuelyourproductdesign.com/the-rise-of-the-diy-micro-copter/. Diakses tanggal 25 Januari 2012.

Louis Naudin, jean. 2011. *Quad Rotor Observer v5 flights with the new KKmulticopter blue board* . http://diydrones.com/profiles/blog/show?id=705844:BlogPost:249942&commentId=705844:Comment:249917&xg_source=activity. Diakses tanggal 25 Desember 2011

Murata Manufacturing.,Co.Ltd. *Piezoelectric Vibrating Gyroscope ENC Series.*

Michael. 2005. *brushless conversion*. http://mikeltigabelas.webs.com/brushless.html. Diakses tanggal 2 Februari 2012.

Ogata, Katsuhiko. 1997. *Teknik Kontrol Automatik Jilid 1*. Jakarta. Penerbit Erlangga.

Ogata, K.1995. *Discrete-Time Control Systems 2nd edition*.

Panjicero. 2011, *4 GAYA PESAWAT TERBANG*. http://panjicero.wordpress.com/2011/12/28/4-gaya-pesawat-terbang/. Diakses tanggal 27 Desember 2011.

Paul Escallier.2010. *Make Your Own Gadget: From Idea To Reality.* http://www.tomsguide.com/us/tricopter-diy-gadget,review-1552-6.html. Diakses tanggal 20 Januari 2012.

Pemula, aeromodelling.blogspot, 2012. http://aeromodellingpemula.blogspot.com/20/2008/12/sistem-rc.html. Diakses tanggal 10 Februari 2012.

Thoha, Muhamad. 2009. *Gyroscope dan Accelerometer 2 in 1*. http://seribubintang.com/?p=503. Diakses tanggal 9 Januari 2012

LAMPIRAN

LAMPIRAN I

FOTO ALAT

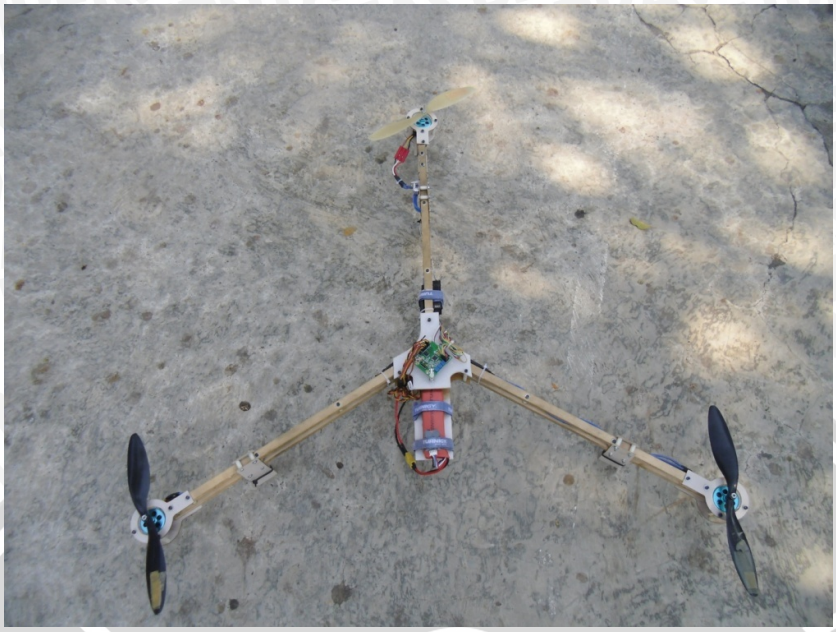


Foto *Tricopter* keseluruhan

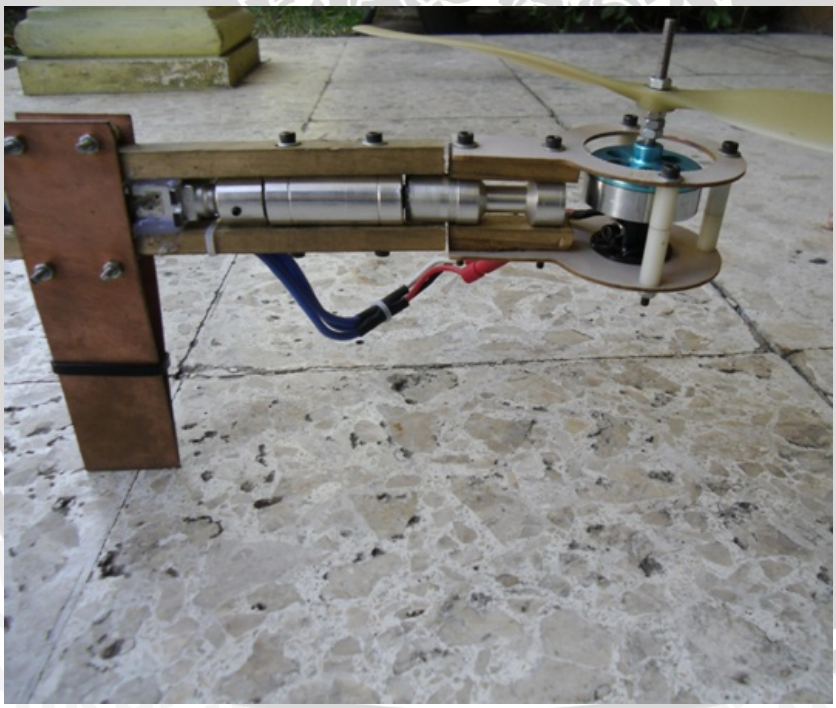


Foto pemasangan motor servo pada lengan ekor *tricopter*

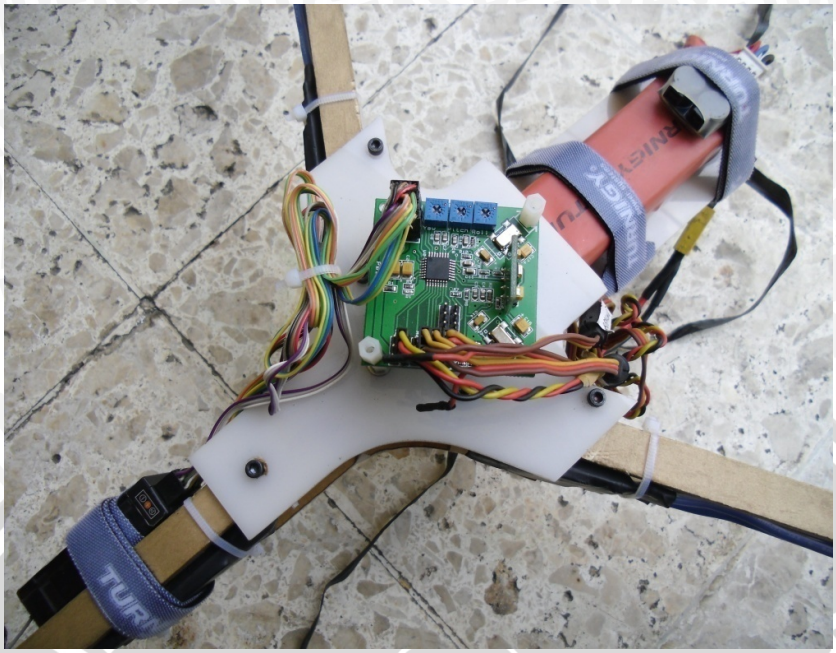


Foto Receiver, Board KKmulticopter168PA, dan baterai pada *tricopter*

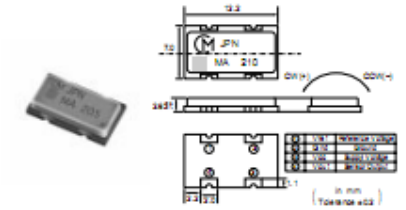
LAMPIRAN II

DATASHEET

muRata

Piezoelectric Vibrating Gyroscopes (GYROSTAR®)

This product is an angular velocity sensor that uses the phenomenon of Coriolis force, which is generated when a rotational angular velocity is applied to the vibrator.
Murata's original small ceramic bimorph vibrator and simple Cap-Base structure achieve an ultra-small size of about 0.2cc. Their small and lightweight shape increase flexibility of installment and help your apparatuses to be downsized.
These surface-mountable devices can be mounted by automatic surface mounters.



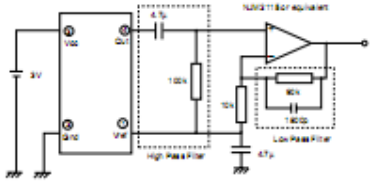
■ Features
1. Ultra-small and ultra-lightweight
2. Quick response
3. Low driving voltage, low current consumption
4. Lead type : SMD
5. Reflow soldering (standard peak temp. 245 degree C)

■ Applications
1. Detecting hand movement involved in video and still cameras
2. Detecting vibrations in various vibration-free tables and isolators
3. Detecting various movements comprising rotation

Part Number	Supply Voltage (Vdc)	Maximum Angular Velocity (deg./sec.)	Output (at Angular Velocity=0) (Vdc)	Scale Factor (mV/deg./sec.)	Linearity (%FS)	Response (Hz)	Weight (g)
ENC-03M	2.7 - 5.25	+/-300	1.35	0.67	+/-5	50 max.	0.4

Operating Temperature Range : -5°C to 75°C Storage Temp. Range: -30°C to 85°C

■ Sample Amplifier Circuit

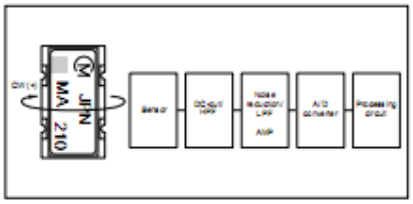


The High-pass Filter's Cut-off frequency in this circuit is approx. 0.3Hz.
The Low-pass Filter's Cut-off frequency in this circuit is approx. 1kHz.

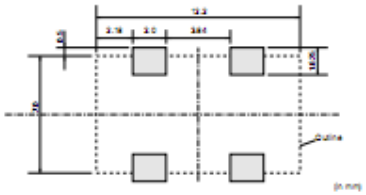
Application/Packaging

■ Application

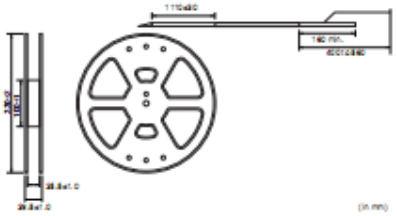
1. One sensor detects rotation on one axis. If two axes are to be detected in the same equipment, two different types of sensors (EMC-03MA and EMC-03MB) should be used.
2. To reduce the effect of temperature drift (due to change of ambient temperature), a high pass filter must be connected to sensor output to eliminate DC component.
3. To suppress output noise component around 22-25kHz (resonant frequency of sensor element), a low pass filter which has higher cut-off frequency than required response frequency must be connected to sensor output.



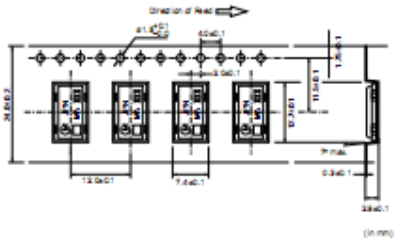
■ Dimensions of Land Pattern



■ Dimensions of Reel



■ Dimensions of Plastic Tape



LAMPIRAN III

LISTING PROGRAM

```
#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>

#include "typedefs.h"
#include "io_cfg.h"

#define EEPROM_DATA_START_POS 0

// Max ADC
#define UC_ADC_MAX 1023

enum GyroDirection { GYRO_NORMAL = 0, GYRO_REVERSED };
enum GyroArrayIndex { ROLL = 0, PITCH, YAW };
typedef struct Config_Struct CONFIG_STRUCT;
struct Config_Struct
{
        uint8_t setup;
        uint8_t RollGyroDirection;
        uint8_t PitchGyroDirection;
        uint8_t YawGyroDirection;
        // allows setting to zero
        uint16_t RxChannel1ZeroOffset;
        uint16_t RxChannel2ZeroOffset;
        uint16_t RxChannel3ZeroOffset;
        uint16_t RxChannel4ZeroOffset;
};

CONFIG_STRUCT Config;
bool GyroCalibrated;
volatile BOOL RxChannelsUpdatingFlag;
uint16_t GainInADC[3];
uint16_t GainIn[3];

volatile uint16_t RxChannel1;
volatile uint16_t RxChannel2;
volatile uint16_t RxChannel3;
volatile uint16_t RxChannel4;

uint16_t RxChannel1Start;
uint16_t RxChannel2Start;
uint16_t RxChannel3Start;
uint16_t RxChannel4Start;

int16_t RxInRoll;
int16_t RxInPitch;
```

```
int16_t RxInCollective;
int16_t RxInYaw;
#ifdef TWIN_COPTER
int16_t RxInOrgPitch;
#endif

int16_t gyroADC[3];
int16_t gyroZero[3] = {0,0,0};

uint16_t ModeDelayCounter;

bool output_motor_high = false;
uint16_t PWM_Low_Pulse_Interval = PWM_LOW_PULSE_INTERVAL;

int16_t MotorOut1;
int16_t MotorOut2;
int16_t MotorOut3;
int16_t MotorOut4;
int16_t StickRollGain;
int16_t StickPitchGain;
int16_t StickYawGain;

void setup(void);
void loop(void);

void Init_ADC(void);
void ReadGyros(bool calibrate);
void CalibrateGyros(void);

void ReadGainPots(void);
void RxGetChannels(void);
void read_adc(uint8_t channel);

void output_motor_ppm(void);

ISR(PCINT2_vect)
{
        if ( RX_ROLL )                          // rising
        {
                RxChannel1Start = TCNT1;

        } else {                                // falling
                RxChannelsUpdatingFlag = 1;
                RxChannel1 = TCNT1 - RxChannel1Start;
                RxChannelsUpdatingFlag = 0;
        }
}

// RX_PITCH
ISR(INT0_vect)
```

```
{
        if (RX_PITCH)
        {
                RxChannel2Start = TCNT1;

        } else {                                // falling
                RxChannelsUpdatingFlag = 1;
                RxChannel2 = TCNT1 - RxChannel2Start;
                RxChannelsUpdatingFlag = 0;
        }
}
// RX_COLL
ISR(INT1_vect)
{
        if (RX_COLL)
        {
                RxChannel3Start = TCNT1;

        } else {                                // falling
                RxChannelsUpdatingFlag = 1;
                RxChannel3 = TCNT1 - RxChannel3Start;
                RxChannelsUpdatingFlag = 0;
        }
}
// RX_YAW
ISR(PCINT0_vect)
{
        if ( RX_YAW )                           // rising
        {
                RxChannel4Start = TCNT1;

        } else {                                // falling
                RxChannelsUpdatingFlag = 1;
                RxChannel4 = TCNT1 - RxChannel4Start;
                RxChannelsUpdatingFlag = 0;
        }
}
int main(void)
{
        setup();

        while (1)
        {
                loop();
        }

        return 1;
}
void setup(void)
{
```

```
    uint16_t i;       // nb was uint8_t, must be uint16_t for TRI
    uint16_t RxChannel1ZeroOffset, RxChannel2ZeroOffset, RxChannel4ZeroOffset;

    MCUCR |= (1<<PUD);        // Pull-up Disable
    RX_ROLL_DIR               = INPUT;
    RX_PITCH_DIR              = INPUT;
    RX_COLL_DIR               = INPUT;
    RX_YAW_DIR                = INPUT;

    GYRO_YAW_DIR              = INPUT;
    GYRO_PITCH_DIR            = INPUT;
    GYRO_ROLL_DIR             = INPUT;
    GAIN_YAW_DIR              = INPUT;
    GAIN_PITCH_DIR            = INPUT;
    GAIN_ROLL_DIR             = INPUT;

    M1_DIR                              = OUTPUT;
    M2_DIR                              = OUTPUT;
    M3_DIR                              = OUTPUT;
    M4_DIR                              = OUTPUT;

    LED                 = 0;
    RX_ROLL     = 0;
    RX_PITCH    = 0;
    RX_COLL     = 0;
    RX_YAW      = 0;
    // pin change interrupt enables
    PCICR |= (1 << PCIE0);                      // PCINT0..7
    PCICR |= (1 << PCIE2);                      // PCINT16..23

    // pin change masks
    PCMSK0 |= (1 << PCINT7);          // PB7
    PCMSK2 |= (1 << PCINT17);                   // PD1
    // external interrupts
    EICRA  = (1 << ISC00) | (1 << ISC10);       // Any change INT0, INT1
    EIMSK  = (1 << INT0) | (1 << INT1);                 // External Interrupt Mask
Register
    EIFR |= (1 << INTF0) | (1 << INTF1);
    TCCR0A = 0;                                         // normal operation
    TCCR0B = (1 << CS00);                       // clk/0
    TIMSK0 = 0;                                 // no interrupts

    TCCR1A = 0;
    TCCR1B = (1 << CS11);

    // timer2 8bit - run @ 8MHz / 1024 = 7812.5KHz
    // and Stick-Arming
    TCCR2A = 0;
    TCCR2B = (1 << CS22) | (1 << CS21) | (1 << CS20);       // /1024
    TIMSK2 = 0;
```

```
        TIFR2  = 0;
        TCNT2 = 0;              // reset counter

        ServoPPMRateDivider = 0;
        do {
                ServoPPMRateDivider++;
                i = ESC_RATE / ServoPPMRateDivider;
        } while (i>50);

        Init_ADC();

        GyroCalibrated = false;
        Armed = false;
        RxChannelsUpdatingFlag = 0;

        RxChannel1 = Config.RxChannel1ZeroOffset;       // prime the channels
1520;
        RxChannel2 = Config.RxChannel2ZeroOffset;       // 1520;
        RxChannel3 = Config.RxChannel3ZeroOffset;       // 1120;
        RxChannel4 = Config.RxChannel4ZeroOffset;       // 1520;

        sei();                                                          //
Global Interrupts

        delay_ms(1500);
        ReadGainPots();
        ReadGainPots();
                while (1)       // loop forever
                {
                        RxGetChannels();
                        MotorOut1 = RxInCollective;
                        MotorOut2 = RxInCollective;
                        MotorOut3 = RxInCollective;
                        MotorOut4 = 50;                 // Center: 50
                        MotorOut5 = 50;                 // Center: 50, Reverse

                        output_motor_ppm(); // this regulates rate at which we output
signals
                }
        }
}
void loop(void)
{
        static uint8_t i;
        static uint16_t Change_Arming=0;
        static uint8_t Arming_TCNT2=0;

        RxGetChannels();

        if (RxInCollective < 0) {
```

```
        Change_Arming += (uint8_t) (TCNT2 - Arming_TCNT2);
        Arming_TCNT2 = TCNT2;

        if (!Armed) {
                if (RxInYaw<STICK_ARMING || abs(RxInPitch) > 30)
Change_Arming = 0;
        } else {
                if (RxInYaw>-STICK_ARMING || abs(RxInPitch) > 30)
Change_Arming = 0;
        }

        if (Change_Arming>0x0F42)
        {
                Armed = ! Armed;
                LED = 0;
                ModeDelayCounter = 0;

                if (Armed) {
                        CalibrateGyros();
                        output_motor_high = false;    // re-set 1st time flag
                        LED = 1;

                        // Normal
                        StickRollGain = NORMAL_STICK_ROLL_GAIN;
                        StickPitchGain = NORMAL_STICK_PITCH_GAIN;
                        StickYawGain = NORMAL_STICK_YAW_GAIN;
                } else if (output_motor_high) {
                        output_motor_ppm();                    // turn off
                }
                return;
        }

        if (Armed && RxInYaw < -STICK_ARMING && RxInPitch >
STICK_ARMING)
        {
                if (ModeDelayCounter==0)
                {
                        //ModeDelayCounter = 0xFB4F;
                        CalibrateGyros();
                        output_motor_high = false;    // re-set 1st time flag
                        delay_ms(150);

                        // Normal
                        StickRollGain = NORMAL_STICK_ROLL_GAIN;
                        StickPitchGain = NORMAL_STICK_PITCH_GAIN;
                        StickYawGain = NORMAL_STICK_YAW_GAIN;

                        // flash LED 1 time
                        for (i=0;i<1;i++)
                        {
```

```
                                LED = 0;
                                delay_ms(25);
                                LED = 1;
                                delay_ms(25);
                        }
                }
                ModeDelayCounter++;
        }
        ReadGyros(false);
        MotorOut1 = RxInCollective;
        MotorOut2 = RxInCollective;
        MotorOut3 = RxInCollective;
        MotorOut4 = 50;
        MotorOut5 = 50;                 // Reverse

        gyroADC[ROLL] = gyroADC[ROLL] * GainIn[ROLL] *
ROLL_GAIN_MULTIPLIER;
        RxInRoll = (RxInRoll * StickRollGain / 100);

        if (Config.RollGyroDirection == GYRO_NORMAL) {
                RxInRoll += gyroADC[ROLL];
        } else {
                RxInRoll -= gyroADC[ROLL];
        }

        RxInRoll   = (RxInRoll * 20)/23;
        MotorOut1 += RxInRoll;
        MotorOut2 -= RxInRoll;

        //--- Calculate pitch gyro output ---
        // nb IF YOU CHANGE THIS CODE, YOU MUST REMOVE PROPS BEFORE
TESTING !!!
        gyroADC[PITCH] = gyroADC[PITCH] * GainIn[PITCH] *
PITCH_GAIN_MULTIPLIER;
        gyroADC[PITCH] /= ADC_GAIN_DIVIDER;
        RxInPitch = (RxInPitch * StickPitchGain / 100);      // Stick Controll %


        if (Config.PitchGyroDirection == GYRO_NORMAL) {
                RxInPitch += gyroADC[PITCH];
        } else {
                RxInPitch -= gyroADC[PITCH];
        }

        MotorOut3 -= RxInPitch;
        RxInPitch = (RxInPitch >> 1);
        MotorOut1 += RxInPitch;
        MotorOut2 += RxInPitch;
        gyroADC[YAW] = (gyroADC[YAW] * GainIn[YAW] *
YAW_GAIN_MULTIPLIER);
```

```
        gyroADC[YAW] /= ADC_GAIN_DIVIDER;
        RxInYaw = (RxInYaw * StickYawGain / 100);

        RxInYaw -= LowpassOutYaw;
        RxInYaw = (RxInYaw >> 3);
        LowpassOutYaw += RxInYaw;

        MotorOut4 += LowpassOutYaw;

        if ( MotorOut1 < 10 ) MotorOut1 = 10;
        if ( MotorOut2 < 10 ) MotorOut2 = 10;
        if ( MotorOut3 < 10 ) MotorOut3 = 10;
                MotorOut1 = 0;
                MotorOut2 = 0;
                MotorOut3 = 0;
                MotorOut4 = 50;
                MotorOut5 = 50;
}

void Init_ADC(void)
{
        DIDR0       = 0b00111111;
        ADCSRB      = 0b00000000;
}

void ReadGainPots(void)
{
        read_adc( 3 );
        GainInADC[ROLL] = ADCL;
        GainInADC[ROLL] += ((uint16_t) ADCH <<8);
        GainInADC[ROLL] = 1024 - GainInADC[ROLL];
        GainIn[ROLL] = GainInADC[ROLL] / 10;

        read_adc( 4 );
        GainInADC[PITCH] = ADCL;
        GainInADC[PITCH] += ((uint16_t) ADCH <<8);
        GainInADC[PITCH] = 1024 - GainInADC[PITCH];
        GainIn[PITCH] = GainInADC[PITCH] / 10;

        read_adc( 5 );
        GainInADC[YAW] = ADCL;
        GainInADC[YAW] += ((uint16_t) ADCH <<8);
        GainInADC[YAW] = 1024 - GainInADC[YAW];
        GainIn[YAW] = GainInADC[YAW] / 10;
}

void read_adc(uint8_t channel)
{
        ADMUX       = channel;
        ADCSRA      = 0b11000110;
```

```
        while (ADCSRA & (1 << ADSC));   // wait to complete
}

void ReadGyros(bool calibrate)
{
        read_adc( 2 );                  // read roll gyro ADC2
        gyroADC[ROLL] = ADCL;
        gyroADC[ROLL] += ((uint16_t) ADCH <<8);
        if (!calibrate) gyroADC[ROLL]       -= gyroZero[ROLL];
        read_adc( 1 );                  // read pitch gyro ADC1
        gyroADC[PITCH] = ADCL;
        gyroADC[PITCH] += ((uint16_t) ADCH <<8);
        if (!calibrate) gyroADC[PITCH] -= gyroZero[PITCH];
}

void CalibrateGyros(void)
{
        uint8_t i;

        ReadGainPots();
        gyroZero[ROLL]      = 0;
        gyroZero[PITCH] = 0;
        gyroZero[YAW]       = 0;

        for (i=0;i<32;i++)
        {
                ReadGyros(true);

                gyroZero[ROLL]      += gyroADC[ROLL];

                gyroZero[PITCH] += gyroADC[PITCH];
                gyroZero[YAW]       += gyroADC[YAW];
        }

        gyroZero[ROLL]      = (gyroZero[ROLL] >> 5);

        gyroZero[PITCH] = (gyroZero[PITCH] >> 5);
        gyroZero[YAW]       = (gyroZero[YAW] >> 5);

        GyroCalibrated = true;
        LowpassOutYaw                   = 50;                                   // Center


void RxGetChannels(void)
{
        static int16_t RxChannel;

        while ( RxChannelsUpdatingFlag );
```

```
        RxChannel = RxChannel1;
        RxChannel -= Config.RxChannel1ZeroOffset;                    //
normalise
        RxInRoll = (RxChannel >> 2);                //    "

        while ( RxChannelsUpdatingFlag );

        RxChannel = RxChannel2;
        RxChannel -= Config.RxChannel2ZeroOffset;                    //
normalise
        RxInPitch = (RxChannel >> 2);               //    "

        while ( RxChannelsUpdatingFlag );

        RxChannel = RxChannel3;
        RxChannel -= Config.RxChannel3ZeroOffset;                    // scale
0->100
        RxInCollective = (RxChannel >> 3);          //

        while ( RxChannelsUpdatingFlag );

        RxChannel = RxChannel4;
        RxChannel -= Config.RxChannel4ZeroOffset;                    //
normalise
        RxInYaw = (RxChannel >> 2);                 //    "
}

void output_motor_ppm(void)
{
        static uint8_t i;
        static uint16_t MotorStartTCNT1, ElapsedTCNT1;
        static int16_t MotorAdjust;
        static uint16_t PWM_Low_Count;
        static int8_t num_of_10uS;
        if (output_motor_high)
        {
                ElapsedTCNT1 = (TCNT1 - MotorStartTCNT1);

                // start output timer
                TIFR0 &= ~(1 << TOV0);                  // clr overflow
                TCNT0 = 0;                                      // reset counter

                // convert into 10uS intervals
                num_of_10uS = (ElapsedTCNT1 / 10) + 1;
                MotorAdjust = 100 - num_of_10uS;

        M1 = 1;
        M2 = 1;
        M3 = 1;
        if(ServoPPMRateCount==ServoPPMRateDivider)
```

```
        {
                M4 = 1;
                M5 = 1;
                ServoPPMRateCount = 1;
        } else {
                ServoPPMRateCount++;
        }
}

void eeprom_write_byte_changed( uint8_t * addr, uint8_t value )
{
        if(eeprom_read_byte(addr) != value)
        {
                eeprom_write_byte( addr, value );
        }
}

void eeprom_write_block_changes( const uint8_t * src, void * dest, size_t size )
{
        size_t len;

        for(len=0;len<size;len++)
        {
                eeprom_write_byte_changed( dest,  *src );

                src++;
                dest++;
        }
}

void Initial_EEPROM_Config_Load(void)
{
        // load up last settings from EEPROM
        if(eeprom_read_byte((uint8_t*) EEPROM_DATA_START_POS )!=0x47)
        {
                Config.setup = 0x47;
                Set_EEPROM_Default_Config();
                // write to eeProm
                Save_Config_to_EEPROM();
        } else {
                // read eeprom
                eeprom_read_block(&Config, (void*) EEPROM_DATA_START_POS,
sizeof(CONFIG_STRUCT));
        }
}

void Set_EEPROM_Default_Config(void)
{
        Config.RollGyroDirection     = GYRO_REVERSED;
        Config.PitchGyroDirection    = GYRO_REVERSED;
```

```
	Config.YawGyroDirection		= GYRO_NORMAL;

	Config.RxChannel1ZeroOffset	= 1520;
	Config.RxChannel2ZeroOffset	= 1520;
	Config.RxChannel3ZeroOffset	= 1120;
	Config.RxChannel4ZeroOffset	= 1520;
}

void Save_Config_to_EEPROM(void)
{
	// write to eeProm
	cli();
	eeprom_write_block_changes( (const void*) &Config, (void*)
EEPROM_DATA_START_POS, sizeof(CONFIG_STRUCT));	//current_config
CONFIG_STRUCT
	sei();
}
void delay_us(uint8_t time)		/* time delay for us */
{
 while(time--)
 {
	asm volatile ("NOP"); asm volatile ("NOP");
	asm volatile ("NOP"); asm volatile ("NOP");
	asm volatile ("NOP"); asm volatile ("NOP");
	asm volatile ("NOP");
 }
}

void delay_ms(uint16_t time)
{
	uint8_t i;
	while(time--)
	{
		for(i=0;i<10;i++) delay_us(100);
	}
}
```