PERANCANGAN KESEIMBANGAN GERAK *PITCH* PADA *BICOPTER* SECARA STATIS DENGAN METODE MENGUBAH-UBAH KECEPATAN MOTOR MENGGUNAKAN KONTROLER PID

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

DANDY MUHAMMAD
NIM. 105060307111006

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2016

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN

PERANCANGAN KESEIMBANGAN *PITCH* PADA *BICOPTER* SECARA STATIS DENGAN METODE MENGUBA-UBAH KECEPATAN MOTOR MENGGUNAKAN KONTROLER PID

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan Untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun Oleh:

DANDY MUHAMMAD
NIM. 105060307111006

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 18 Maret 2016

MAJELIS PENGUJI

Dr. Ir. Erni Yudaningtyas, MT.	**Rahmadwati, ST., MT., Ph.D**
NIP. 19650913 199002 2 001	**NIP. 19771102 200604 2 003**

Goegoes Dwi Nusantoro, ST., MT.
NIP. 19711013 200604 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

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

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

JUDUL SKRIPSI:

PERANCANGAN KESEIMBANGAN GERAK *PITCH* PADA *BICOPTER* SECARA STATIS DENGAN METODE MENGUBAH-UBAH KECEPATAN MOTOR MENGGUNAKAN KONTROLER PID

Nama Mahasiswa : DANDY MUHAMMAD

NIM : 105060307111006

Program Studi : Teknik Elektro

Konsentrasi : TEKNIK KONTROL

Komisi Pembimbing :

Ketua : Ir. PURWANTO, M.T .....

Anggota : Dr. Ir. BAMBANG SISWOJO, M.T .....

TIM PENGUJI DOSEN :

Dosen Penguji 1 : GOEGOES DWI N., S.T, M.T .....

Dosen Penguji 2 : Dr. Ir. ERNI YUDANINGTYAS, M.T .....

Dosen Penguji 3 : RAHMADWATI, S.T., M.T., Ph.D .....

Tanggal Ujian : 18 MARET 2016

SK Penguji : 343/UN10.6/SK/2016

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 12 Januari 2016

Mahasiswa,

Dandy Muhammad

NIM. 105060307111006

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

RINGKASAN

Dandy Muhammad, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Januari 2016, Perancangan Keseimbangan Gerak *Pitch* Pada *Bicopter* Secara Statis Dengan Metode Mengubah-ubah Kecepatan Motor Menggunakan Kontroler PID, Dosen Pembimbing: Ir. Purwanto. MT., Dr. Ir. Bambang Siswojo. MT

Seiring dengan perkembangan teknologi modern, saat ini robot udara atau yang sering disebut *Unmaned Aerial Vehicle* (UAV) sering digunakan sebagai alat bantu pengambilan gambar dari udara dari sudut-sudut yang sulit dan berbahaya untuk dijangkau manusia. Salah satu jenis UAV yang dapat membantu manusia adalah *multicopter*. *Multicopter* memilki 4 gerak dasar yaitu *roll, pitch, yaw* dan *throttle*. *Roll* merupakan gerakan rotasi pada sumbu x. *Pitch* merupakan gerakan rotasi pada sumbu y. *Yaw* merupakan gerakan rotasi pada sumbu z. Gerakan t*hrottle* merupakan gerak translasi *bicopter* sepanjang garis sumbu z. *Multicopter* yang memiliki dua baling-baling disebut *bicopter*. Dalam perancangan *bicopter* ada beberapa hal yang mempengaruhi keseimbangan antara lain panjang frame, berat total, dan gaya dorong (*thrust*). Gaya dorong dihasilkan oleh daya hembus dari baling-baling yang berputar akibat dari putaran motor BLDC dengan kecepatan tertentu. Untuk mendesain frame *bicopter* secara lengkap diperlukan alat uji satu frame. Alat bantu desain *bicopter* merupakan frame satu absis dengan satu aktuator motor dan propeller. Sisi lainnya merupakan beban uji yang dapat diubah-ubah beratnya.

Proses perancangan PID pada penelitian ini menggunakan metode 1 Ziegler-Nichols pada setpoint 2,5 V menghasilkan Kp = 5,7; Ki = 28,5; dan Kd = 0,285 yang menunjukkan bahwa respon sistem secara keseluruhan tidak melebihi 5% dari setpoint dan mampu kembali *steady state* ketika mendapatkan gangguan perubahan beban uji. Hal ini menunjukkan bahwa kontroler PID dapat mengendalikan kecepatan putaran dengan baik.

Kata kunci : *bicopter, kecepatan putaran, PID, UAV*

UNIVERSITAS BRAWIJAYA

SUMMARY

Dandy Muhammad, Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, January 2016, Design Motion Of Pitch Balance In Immovable Bicopter With Changing Speed Of Motor Method Using PID Controller, Academic Supervisor: Ir. Purwanto. MT., Dr. Ir. Bambang Siswojo. MT

Along with the development of modern technology, today's robots air or often called *Unmaned Aerial Vehicle* (UAV) is often used as tools of aerial shots of corners that are difficult and dangerous to reach humans. One type of UAV that can help people is multicopter. Multicopter have the four basic motion that roll, pitch, yaw and throttle. Roll is rotational movement on the x axis. Pitch is a rotational movement on the y axis. Yaw is rotational movement in the z-axis. Throttle movement is a translational motion along the z-axis bicopter. Multicopter which has two propeller called bicopter. In designing bicopter there are several things that affect the balance between the other frame length, total weight, and thrust force (*thrust*). The driving force of the power generated by the blow from the propeller that rotates due to the BLDC motor rotation at a certain speed. To design a complete frame bicopter necessary test equipment one frame. Bicopter design tools is an abscission frame with a motor actuator and propeller. The other side is the test load can be varied severity.

The designing process PID in this study using the Ziegler-Nichols method 1 on setpoint 2.5 V produces Kp = 5.7; Ki = 28.5; and Kd = 0.285 which shows that the response of the system as a whole does not exceed 5% of setpoint and capable of returning steady state when getting interference test load changes. This indicates that the PID controller can control the speed of rotation well.

Keywords: bicopter, rotation speed, PID, UAV

KATA PENGANTAR

Alhamdulillâh, segala puji hanya bagi Allâh Subhanahu Wa Taála, Rabb alam semesta. Dialah Allâh, Tuhan Yang Maha Satu, Yang Maha Pengasih lagi Maha Penyayang. Dialah Sebaik baik Penolong dan Sebaik baik Pelindung. Shalawat dan salâm kepada Nabi Muhammad Rasulullâh Shallallâhu Alaihi Wa Salâm, Sang pembawa kabar gembira dan sebaik baik suri tauladan bagi yang mengharap Rahmat dan Hidayah-Nya.

Sungguh hanya melalui Pertolongan dan Perlindungan Allâh SWT semata sehingga saya dapat menyelesaikan skripsi ini. Dengan seizin Allâh SWT, di kesempatan yang baik ini saya ingin menghaturkan rasa terima kasih dan penghargaan yang sebesar besarnya atas bantuan sehingga terselesainya skripsi ini kepada:

- Keluarga tercinta, kedua orang tua Rakhmad Susatyo dan Retno Setyowati yang selalu memberikan kasih sayang dan doanya yang tiada akhir. Serta Randy Muhammad dan Pahlevy Muhammad yang selalu memberikan dukungan.
- Bapak M. Aziz Muslim, ST., MT., Ph.D selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Hadi Suyono, ST.,MT., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Bapak Ir. Purwanto, MT. selaku KKDK Teknik Kontrol dan sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, gagasan, ide, saran, motivasi yang telah diberikan, serta waktu yang diluangkan untuk bimbingan.
- Bapak Dr. Ir. Bambang Siswojo, MT. sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, gagasan, ide, saran, motivasi yang telah diberikan, serta waktu yang diluangkan untuk bimbingan.
- Bapak Ibu Dosen, karyawan, staf recording dan RBTE atas segala bantuan dan kemudahan.
- Teman - teman seperjuangan dalam mengerjakan skripsi Randy, Ariski, Bagus terima kasih telah berbagi pengalaman dan pengetahuan, serta canda dan tawa.
- Teman - teman “MaGiC”, Luthfi, Gilang, Rainer, Rangga, Mukson, Ulit, Dany, Ernanda, Genji dan Kadek terima kasih telah berbagi kesenangan, pelajaran hidup, serta canda dan tawa.

- Sahabat "NMC" dan sahabat SMAN 8 Malang, Joko, Hamdani, Jun, Udin, Ayas, Karno, Agung, Widi, Hangga, Sesa, Apep dkk yang selama ini juga selalu menghibur dan memberi support untuk menuju kesuksesan.
- Keluarga besar angkatan 2010 MAGNET'10 atas do'a, semangat, serta dukungan yang diberikan pada penulis.
- Semua pihak yang telah memberikan bantuan serta dukungan baik secara langsung maupun tidak langsung atas penyusunan skripsi ini.

Sekiranya Allâh SWT mencatat amalan ikhlas kami dan semua pihak yang turut membantu sehingga skripsi ini terselesaikan. Akhirnya, kami menyadari bahwa skripsi ini masih jauh dari sempurna namun semoga skripsi ini dapat memberikan manfaat bagi kita semua. Allâhumma Amîn.

Malang, Januari 2016

Penulis

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI iii
DAFTAR GAMBAR v
DAFTAR TABEL vii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Manfaat 3
1.6 Sistematika Penulisan 3
BAB II DASAR TEORI 5
2.1 Bicopter 5
2.2 Baling-baling (*Propeller*) 6
2.3 Motor BLDC *(Brushless Direct Current)* 7
2.4 Catu Daya. 8
2.5 ESC (*Electronic Speed Control*) 9
2.6 PWM *Digital Controller* 9
2.7 Potensiometer 11
2.8 Kontroler 11
2.8.1 Kontroler Proporsional 12
2.8.2 Kontroler Integral 13
2.8.3 Kontroler Diferensial 14
2.8.4 Kontroler Proporsional Integral (PI) 14
2.8.5 Kontroler Proporsional Derivative (PD). 15
2.8.6 Kontroler Proportional Integral Derivative (PID). 17
2.8.7 Metode Perancangan Kontroler Proporsional Integral Deferensial (PID) Menggunaakan Metode Ziegler-Nichols 17
2.9 Mikrokontroler 20
2.9.1 Mikrokontroler ATMega328 21
2.10 Arduino Mega. 21

BAB III METODE PENELITIAN 23
3.1 Penentuan Spesifikasi Alat 23
3.2 Perancangan dan Realisasi Pembuatan Alat. 23
3.2.1 Perancangan Perangkat Keras dan Realisasi Pembuatan Alat 23
3.2.2 Perancangan dan Perhitungan Komponen yang akan Digunakan 24
3.2.3 Perancangan Perangkat Lunak 24
3.3 Pengujian Alat 24
3.4 Pengambilan Kesimpulan dan Saran 25
BAB IV PERANCANGAN DAN PEMBUATAN ALAT 27
4.1 Perancangan Sistem 27
4.1.1 Perancangan Sistem Mekanik 27
4.1.2 Perancangan Sistem Kontroler 29
4.2 Diagram Blok Sistem 29
4.3 Perancangan Perangkat Keras 30
4.3.1 *Propeller* (Baling-baling). 30
4.3.2 Sensor Potensiometer. 30
4.3.3 Motor BLDC *(Brushless Direct Current)* 31
4.3.4 Modul Arduino Mega 32
4.4 Perancangan Kontrol PID. 33
4.4.1 Penentuan Nilai Penguatan Kontroler 33
4.5 Perancangan Perangkat Lunak. 35
BAB V PENGUJIAN DAN ANALISIS 37
5.1 Pengujian Sensor 37
5.2 Pengujian respon aktuator Motor DC *Brushless*. 42
5.3 Hasil Pengujian Keseluruhan Sistem 44
BAB VI PENUTUP 49
6.1 Kesimpulan 49
6.2 Saran 49
DAFTAR PUSTAKA 51
LAMPIRAN 53

DAFTAR GAMBAR

Gambar 2.1 Gerak Dasar Multicopter 6
Gambar 2.2 Motor BLDC. 6
Gambar 2.3 PWM *Digital Controller* 8
Gambar 2.4 ESC 9
Gambar 2.5 Gambar Sinyal PWM Secara Umum 10
Gambar 2.7 Potensiometer 11
Gambar 2.8 Diagram Blok Kontroler Proporsional 12
Gambar 2.9 Diagram Blok Kontroler Integral 13
Gambar 2.10 Diagram Blok Kontroler Diferensial 14
Gambar 2.11 Diagram Blok Kontroler PID 16
Gambar 2.12 Hubungan fungsi waktu antara sinyal keluaran dan masukan kontroler PID 16
Gambar 2.13 Kurva Respon Unit Step yang Menunjukkan 25% *Maximum Overshoot* 17
Gambar 2.14 Respon Plant Terhadap Masukan Berupa Unit Step 17
Gambar 2.15 Kurva Respon yang Berbentuk S 18
Gambar 2.16 Sistem Loop Tertutup dengan Kontroler Proporsional 19
Gambar 2.17 Osilasi Berkesinambungan dengan Periode *Pcr* 19
Gambar 2.18 Board Arduino Mega 22
Gambar 4.1 Skema Alat Uji Satu Frame 28
Gambar 4.2 Diagram Blok Sistem 29
Gambar 4.3 Propeller 30
Gambar 4.4 Sensor Potensiometer 31
Gambar 4.5 Struktur Potensiometer 31
Gambar 4.6 Tampak depan Arduino Mega 2560 32
Gambar 4.7 Metode 1 *Ziegler-Nichols* (hasil pengujian) 33
Gambar 4.6 *Flowchart* Perangkat Lunak 35
Gambar 5.1 Diagram Perancangan Peralatan Pengujian Sensor 38
Gambar 5.2 Perbandingan perhitungan kecepatan motor menggunakan rumus dengan menggunakan *tachometer* 39
Gambar 5.3 Perbandingan antara tegangan keluaran sensor dengan sudut kemiringan frame 40

Gambar 5.4 Grafik perbandingan antara kecepatan putaran motor BLDC terhadap sudut kemiringan *frame*..... 41
Gambar 5.5 Grafik perbandingan antara kecepatan putaran motor BLDC terhadap tegangan keluaran sensor 41
Gambar 5.6 Grafik respon aktuator saat kecepatan motor diatur maksimal seketika..... 44
Gambar 5.7 Diagram Peralatan Pengujian Motor DC *Brushless* 45
Gambar 5.8 Grafik Hasil Pengujian *Closed Loop* 47

DAFTAR TABEL

Tabel 4.1 Fungsi Pin Arduino Mega 2560 32
Tabel 4.2 Aturan Metode 1 *Ziegler-Nichols* 34
Tabel 5.1 Hasil Pengujian Sensor 38
Tabel 5.2 Respon Aktuator 43
Tabel 5.3 Hasil Pengujian Motor DC *Brushless* 46

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Robot merupakan hasil karya teknologi yang dalam hal tertentu dapat digunakan untuk menyelesaikan permasalahan-permasalahan yang dihadapi dalam kehidupan manusia. Terutama agar proses-proses yang dilakukan secara manual dapat bekerja secara otomatis; terutama pekerjaan yang memerlukan ketelitian proses yang tinggi, daerah kerja yang berbahaya, memerlukan proses yang berulang-ulang untuk mendapakan produk dengan kualitas produksi yang seragam serta alasan-alasan lainnya. Robot udara yang berkembang saat ini adalah pesawat udara yang bekerja secara *auto-pilot* atau tanpa awak. Robot pesawat udara *auto-pilot* ini biasa disebut dengan *Unmaned Aerial Vehicle* atau disingkat UAV.

Seiring dengan perkembangan teknologi modern, saat ini UAV sering digunakan sebagai alat bantu pengambilan gambar dari udara dari sudut-sudut yang sulit dan berbahaya untuk dijangkau manusia. UAV juga diaplikasikan untuk untuk kebutuhan militer, seperti pengintaian atau bahkan invasi militer. UAV sangat aplikatif untuk kebutuhan ini dimana kita dapat mengintai wilayah musuh tanpa membahayakan pilot. Atau kita dapat mengambil gambar dari wilayah yang belum diketahui. Salah satu jenis UAV yang dapat membantu manusia adalah *multicopter*. Semakin banyak baling-baling yang dimiliki, kecepatan, daya angkat dan keseimbangan *multicopter* akan meningkat, sehingga kapasitas beban yang diangkat akan semakin meningkat (Kristianto, D, 2012). *Bicopter* merupakan produk robot udara yang digerakkan oleh dua baling-baling yang dimiliki.

Dalam pemanfaatan *bicopter* untuk berbagai tujuan unjuk kerja, kestabilan pada saat keadaan melayang (*hover*) pada *bicopter* sangatlah penting dan harus dimiliki *bicopter* agar pemanfaatannya dapat secara maksimal. Dalam perancangan *bicopter* ada beberapa hal yang mempengaruhi keseimbangan antara lain adalah panjang *frame*, berat total, dan gaya dorong motor (*thrust*). Semakin pendek panjang *frame* artinya semakin ringan beban pada sisi tersebut. Begitu pula

sebaliknya, semakin panjang *frame* maka semakin berat pula beban pada sisi tersebut. Motor menghasilkan gaya dorong sesuai spesifikasi. Motor digabungkan bersama dengan propeller dengan spesifikasi tertentu.

Untuk mendesain *frame bicopter* secara lengkap diperlukan alat uji satu *frame*. Alat bantu desain *frame bicopter* merupakan model satu *frame* dengan satu aktuator motor dan propeller. Sisi lainnya merupakan beban uji yang dapat diubah-ubah beratnya. Dengan menggabungkan kontroler PID, dapat diuji kestabilan yang dihasilkan dengan mengubah-ubah parameter: panjang *frame*, spesifikasi motor dan propeller dengan konstanta PID tertentu. Parameter yang diperoleh dari alat uji untuk membantu desain *bicopter* akan dapat digunakan sebagai dasar desain *bicopter* secara keseluruhan.

Dalam skripsi ini penulis merancang sebuah alat uji coba statis satu lengan yaitu sebuah alat yang berada dalam keadaan diam dan terdapat dua sisi, depan dan belakang, yang terdiri dari satu rotor *bicopter* dan sebuah beban uji di sisi lainnya yang bisa diubah-ubah beratnya . Alat uji coba ini adalah sebuah alat yang mampu mengendalikan kestabilan horizontal dengan menggunakan metode Kontrol Proporsional Integral Differensial.

1.2 Rumusan Masalah

1. Bagaimana mendesain model mekanik satu *frame bicopter*?
2. Bagaimana merancang *hardware* dan *software* sistem pengendalian kecepatan putaran motor pada propeller menggunakan metode Proporsional Integral Differensial?

1.3 Batasan Masalah

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

1. Pembahasan ditekankan pada aplikasi Kontrol PID pada sistem pengontrolan.
2. Sensor yang digunakan adalah sensor *potensiometer*.
3. Arduino Mega sebagai pusat pengendali sistem.
4. Gangguan diberikan dengan mengubah-ubah berat beban uji.
5. Kinerja rangkaian elektrik tidak dibahas mendalam.

1.4 Tujuan

Merancang dan membuat alat bantu desain *frame bicopter* secara statis satu *frame* dengan Kontroler Proporsional Integral Differensial.

1.5 Manfaat

Memberikan referensi dan kontribusi penelitian untuk pengembangan UAV *RC Airplane, tricopter, quadcopter,* atau *multicopter*.

1.6 Sistematika Penulisan

Agar penyusunan laporan skripsi ini dapat mencapai sasaran dan tidak menyimpang dari judul yang telah ditentukan, maka diperlukan sistematika pembahasan yang jelas. Pembahasan dalam skripsi ini secara garis besar adalah sebagai berikut:

BAB I Pendahuluan

Membahas latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan alat.

BAB III Metode Penelitian

Menjelaskan tentang metode penelitian yang terdiri dari studi literatur, perancangan alat, pembuatan alat, pengujian alat, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan dan Pembuatan Alat

Menjelaskan tentang perancangan dan pembuatan alat yang meliputi prinsip kerja alat, perancangan perangkat keras, dan perangkat kontroler PID dan menerapkannya ke dalam *software*,sehingga sistem dapat bekerja dengan baik.

BAB V Pengujian dan Analisis

Menjelaskan tentang pengujian alat dan analisa yang meliputi pengujian bagian blok sistem dan pengujian sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

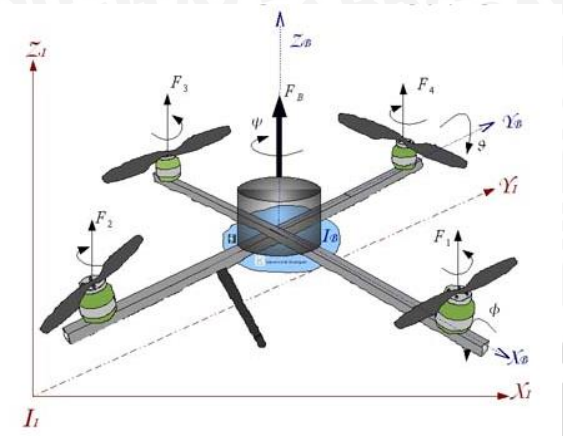
Menjelaskan tentang pengambilan kesimpulan sesuai dengan hasil masalah, serta pemberian saran untuk pengembangan.

BAB II
DASAR TEORI

2.1 *Bicopter*

Bicopter merupakan salah satu jenis *multicopter* yang tersusun dari dua buah rotor yang diletakan simetris pada tepi-tepi ujungnya. Dalam implementasinya, gerak dasar *bicopter* mirip dengan gerak dasar quadcopter yaitu difungsikan dalam suatu bidang x, y dan z sebagaimana diungkapkan pada penelitian terdahulu (Dharmawan, A. & Putera, C.A.L., 2012:97) dengan titik pusat berada pada bagian tengah *bicopter*. Sumbu x merupakan garis sejajar dengan bidang lengan depan *bicopter*, sumbu y merupakan garis bidang lengan depan *bicopter*, sumbu y merupakan garis bidang sejajar dengan bidang lengan samping kanan *bicopter*, dan sumbu z merupakan garis tegak lurus 90 derajat ke arah bawah *bicopter*.

Ketiga sumbu tersebut digunakan sebagai acuan gerak *bicopter*. Gerakan dan kecepatan birotor ditentukan oleh kecepatan masing-masing rotor. Terdapat 4 gerakan dasar pada *bicopter* yakni gerakan akselerasi (*throttle*), gerakan sudut *roll*, gerakan sudut *pitch*, dan gerakan sudut *yaw*. Gerakan t*hrottle* merupakan gerak translasi *bicopter* sepanjang garis sumbu-z. Gerakan ini dipengaruhi dengan perubahan kecepatan keempat rotor dengan nilai yang sama. Dengan pergerakan *throttle bicopter* akan naik atau turun sesuai dengan kecepatan pada setiap rotor *bicopter*. Gerakan *roll* merupakan gerakan rotasi pada sumbu x. Gerakan ini dipengaruhi oleh perubahan kecepatan rotor depan dan belakang. Namun pada *bicopter* rotor depan dan belakang digantikan dengan sebuah sirip. Gerakannya dipengaruhi oleh perubahan sudut sirip tersebut. Gerakan *pitch* merupakan gerakan rotasi pada sumbu y. Gerakan ini dipengaruhi oleh perubahan kecepatan pada motor kanan dan kiri. Gerakan sudut *yaw* merupakan gerakan rotasi pada sumbu z.

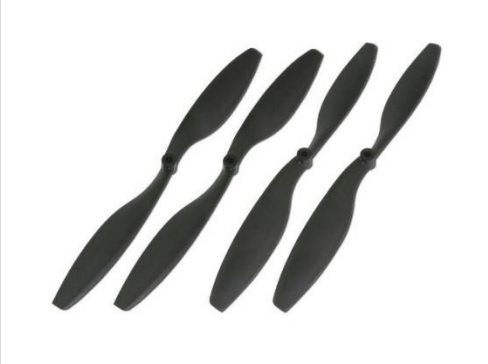


Gambar 2.1 Gerak Dasar Multicopter

Sumber: ex-sheffield.org

2.2 Baling-baling (*Propeller*)

Baling-baling adalah alat yang mengubah gerak putar menjadi daya dorong. Pembahasan baling-baling pada tugas akhir ini dibatasi hanya pada paramater baling-baling yang digunakan dalam RC (*Radio Control*) *aeromodelling*. Bentuk propeller ditunjukkan dalam Gambar 2.2 berikut ini.



Gambar 2.2 Propeller

Sumber: http://helitech-jp.com/

Ada beberapa parameter penting yang dimiliki baling-baling pada RC *aeromodelling* sebagaimana diungkapkan pada penelitian terdahulu (Kristianto, D. 2012). Parameter-parameter ini bisa dijadikan pedoman untuk memilih baling-baling sesuai dengan kebutuhan:

1. Diameter dan *pitch*

Semua baling-baling RC yang tersedia memiliki 2 buah ukuran, yaitu diameter dan *pitch*. Diameter dihitung berdasarkan diameter lingkaran yang dibentuk saat baling-baling berputar. Jika baling dianalogikan sebagai sebuah sekrup, *pitch* merupakan jarak yang ditempuh oleh baling-baling jika diputar 1

putaran penuh. Semakin panjang diameter dan *pitch* baling-baling semakin banyak pula udara disapu dan semakin besar pula daya dorong yang dihasilkan. Tapi diameter dan *pitch* dari baling-baling ini harus disesuaikan degan motor dan sumber daya yang digunakan. Biasanya produsen motor sudah memberikan spesifikasi baling-baling untuk motornya.

2. Jumlah bilah

Umumnya, jumlah bilah pada baling-baling RC *aeromodelling* adalah 2 bilah, Tetapi ada beberapa yang menggunakan 3 bilah dan 4 bilah. Semakin banyak bilah pada baling-baling menyebabkan banyak udara yang disapu sehingga menghasilkan daya dorong yang lebih besar. Biasanya penambahan jumlah bilah bertujuan untuk memperkecil diameter baling-baling, tentunya untuk menghasilkan performa yang sama (dengan motor yang sama) *pitch*-nya harus dikurangi.

3. Arah putar

Dengan arah gaya dorong yang sama, baling-baling RC *aeormodelling* memiliki dua jenis arah putaran: searah jarum jam (CW, *clockwise*) dan berkebalikan arah jarum jam (CCW, *counter clockwise*). Arah putar ini menentukan *yawing moment* yang dihasilkan dari baling-baling.

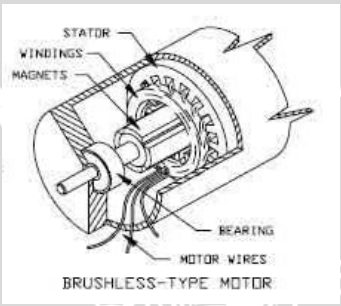
2.3 Motor BLDC *(Brushless Direct Current)*

Bicopter membutuhkan penggerak berupa baling-baling yang diputar oleh motor. Spesifikasi yang harus dipenuhi oleh sistem gerak ini adalah torsi, efisiensi dan getaran yang ditimbulkan oleh berputarnya motor dan baling-baling. Motor dengan getaran yang terlalu besar dapat mengganggu sensor-sensor yang digunakan pada *Attitude Heading Reference System* (AHRS). AHRS merupakan integrasi dari beberapa sensor dan menggunakan perhitungan tertentu untuk memadukan data dari sensor-sensor tersebut. Efisiensi motor berkaitan dengan durabilitas terbang dari pesawat. Mengingat sumber daya (*battery*) yang digunakan terbatas.

Brushless Direct Current Motor atau biasa disebut BLDC adalah motor DC yang proses komutasinya tidak menggunakan sikat seperti motor DC pada umumnya. Dibandingkan dengan motor DC denga sikat, BLDC memiliki beberapa kelebihan yaitu: efisiensi tinggi, kecepatan dan torsi yang tinggi, respon dinamis yang tinggi, masa operasi yang panjang dan operasi tanpa *noise*. Sehingga dengan

kelebihan-kelebihan tersebut, BLDC banyak digunakan pada aplikasi *aeromodelling* dan termasuk pada *bicopter*.

Motor BLDC adalah tipe motor sinkron. Artinya medan magnet yang dihasilkan oleh stator dan medan magnet yang dihasilkan oleh rotor mempunyai frekuensi yang sama. Rotor (bagian motor yang berputar) pada BLDC terdiri dari magnet permanen, sedangkan stator terdiri dari kumparan. Berbeda dengan motor DC dengan sikat, dimana rotor berupa lilitan dan stator berupa magnet tetap. Motor BLDC ditunjukkan dalam Gambar 2.3 di bawah ini:



Gambar 2.3 Motor BLDC

Sumber: http://www.rcspark.com

2.4 Catu Daya

Sumber tegangan atau catu daya memegang peranan yang sangat penting dalam hal perancangan sebuah *payload*. Tanpa bagian ini *payload* tidak akan berfungsi. Begitu pula dengan pemilihan sumber tegangan yang tidak tepat, maka *payload* tidak akan bekerja dengan baik.

Penentuan sistem catu daya yang akan digunakan ditentukan oleh banyak faktor sebagaimana diungkapkan pada penelitian terdahulu (Rohiman, S., 2011), diantaranya:

1. Tegangan

   Setiap aktuator atau motor tidak memiliki tegangan yang sama. Hal ini akan berpengaruh terhadap desain catu daya. Tegangan tertinggi dari salah satu aktuator akan menetukan nilai tegangan catu daya.

2. Arus

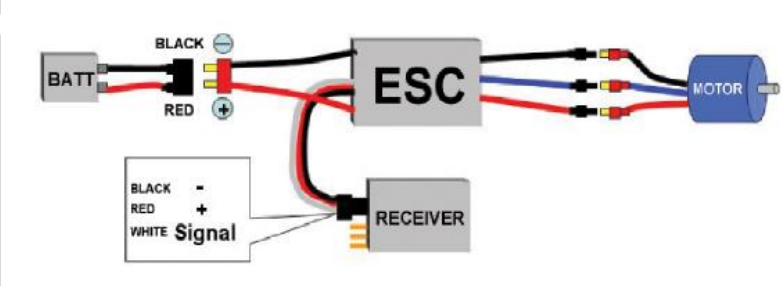
   Arus memiliki satuan Ah (*Ampere-hour*). Semakin besar Ah, semakin lama daya tahan baterai bila digunakan pada beban yang sama.

3. Teknologi Baterai

Baterai isi ulang ada yang dapat diisi kapan saja, dan ada pula yang harus diisi ulang sebelum batas tegangan minimum.

2.5 ESC (*Electronic Speed Control*)

Pada umumnya, satu unit ESC digunakan untuk men-drive satu unit aktuator. Dalam hal ini yang sebagai aktuator adalah motor BLDC. ESC adalah sebuah sirkuit elektronik berfungsi sebagai pengatur kecepatan motor, selain itu juga berfungsi untuk menaikkan jumlah arus dengan cara menaikkan pulsa dari mikrokontroler sehingga menghasilkan tegangan yang diperlukan oleh motor (Chmelai, 2011).



Gambar 2.4 *Electronic Speed Control*

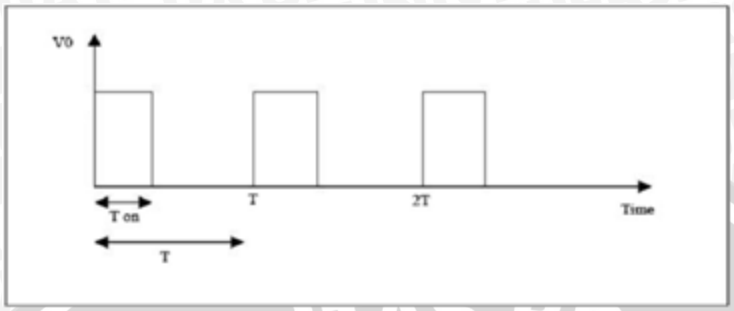
Sumber: hobbyking.com/esc

2.6 PWM *Digital Controller*

PWM (*Pulse Width Modulation*) adalah suatu logika *on off* atau 1 dan 0 berdasarkan kecepatan *dutycycle*. PWM *Digital Controller* digunakan untuk mengatur kecepatan dari motor DC, dimana kecepatan motor DC tergantung pada besarnya *dutycycle* yang diberikan pada motor DC tersebut. Bentuk PWM *Digital Controller* dapat dilihat dalam Gambar 2.4 berikut:

Gambar 2.5 PWM *Digital Controller*

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *dutycycle* bervariasi dari 0%-100%. Dengan mengatur *dutycycle* akan diperoleh keluaran yang diinginkan. Sinyal PWM secara umum dapat dilihat dalam Gambar 2.5 berikut:



Gambar 2.6 Gambar Sinyal PWM Secara Umum
Sumber: electronics-scheme.com

$$Dutycycle = \frac{Ton}{T} x\ 100\% \ldots (\%) \quad (2\text{-}1)$$

dengan:

Ton = Periode logika tinggi

T = Periode keseluruhan

$$Vdc = Dutycycle\ x\ Vcc \ldots (V) \quad (2\text{-}2)$$

dengan:

Vdc = Tegangan keluaran

Vcc = Tegangan masukan/*supply*

Sedangkan frekuensi sinyal dapat ditentukan dengan rumus berikut:

$$fon = \frac{fclk\ I/0}{N.256} \ldots (Hz) \quad (2\text{-}3)$$

dengan:

f_{on} = Frekuensi sinyal keluaran

f_{clk} = Frekuensi *clock* yang dibutuhkan

N = Skala *clock* (mempunyai nilai 1, 8, 64, 256, dan 1024)

2.7 Potensiometer

Potensiometer memiliki 3 terminal, 2 terminal terhubung ke kedua ujung elemen resistif, dan terminal ketiga terhubung ke kontak geser yang disebut wiper. Potensiometer pada dasarnya berfungsi sebagai pembagi tegangan variabel. Unsur resistif dapat dilihat sebagai dua resistor seri, dimana posisi wiper menentukan rasio resistensi dari resistor pertama ke resistor kedua. Potensiometer juga dikenal sebagai potmeter atau pot.

Bentuk paling umum dari potmeter adalah potmeter putar. Jenis pot sering digunakan dalam kontrol volume suara audio dan berbagai aplikasi lainnya. Unsur resistif pada potensiometer biasanya terbuat dari bahan seperti karbon, keramik logam, gulungan kawat (*wirewound*), plastik konduktif, atau film logam.

Salah satu jenis potensiometer adalah potensiometer linier. Potensiometer linier adalah potensiometer yang perubahan tahanannya sangat halus dengan jumlah putaran sampai sepuluh kali putaran (*multi turn*). Untuk keperluan sensor posisi potensiometer linier memanfaatkan perubahan resistansi. Bentuk potensiometer secara umum dapat dilihat dalam Gambar 2.7 berikut:

Gambar 2.7 Potensiometer
Sumber: http://www.resistorguide.com/

2.8 Kontroler

Keberadaan kontroler dalam sebuah sistem kontrol adalah sebagai pengolah *error*. Salah satu fungsi komponen kontroler adalah mengurangi sinyal *error*. *Error* adalah perbedaan antara nilai referensi/nilai yang diinginkan (*setpoint*) dan nilai aktual. Hal ini sesuai dengan tujuan sistem kontrol dimana mendapat nilai sinyal keluaran sama dengan nilai *setpoint*. Semakin kecil kesalahan yang terjadi, semakin baik kinerja sistem kontrol yang diterapkan.

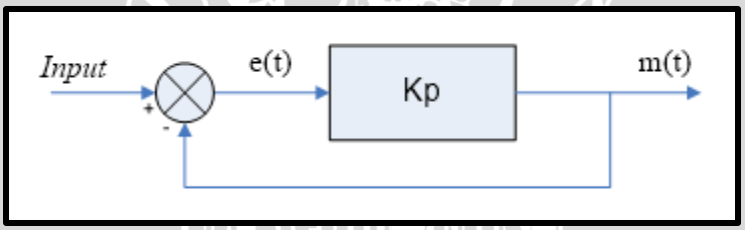
Apabila perbedaan antara nilai referensi dengan nilai keluaran relatif besar, maka kontroler yang baik seharusnya mampu mengamati perbedaan ini untuk segera menghasilkan sinyal keluaran untuk mempengaruhi plant. Dengan demikian sistem secara cepat mengubah keluaran plant sampai diperoleh selisih dengan 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. Jenis-jenis kontroler (Ogata, K., 2002).

2.8.1 Kontroler Proporsional

Kontroler proporsional memiliki keluaran yang sebanding/proporsional dengan besarnya sinyal kesalahan/*error*. Dapat dikatakan bahwa keluaran kontroler proporsional merupakan perkalian antara konstanta proporsional dengan masukanya. Perubahan sinyal masukan akan segera menyebabkan sistem secara langsung mengubah keluaranya sebesar konstanta pengalinya.

Untuk lebih jelasnya dapat dilihat pada blok diagram pada Gambar 2.8



Gambar 2.8 Diagram blok kontroler proporsional

Sumber: Ogata, K., 2002

Pada Gambar 2.8 menunjukan blok diagram yang menggambarkan hubungan antara *input* (besaran yang diinginkan),besaran aktual dengan besaran keluaran kontroler proporsional,dan besaran kesalahan *(error)*. Sinyal kesalahan *(error)* merupakan selisih antara besaran *setting* dengan besaran aktualnya.

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

$$m(t) = Kp\ e(t) \qquad (2\text{-}4)$$

Dengan Kp adalah penguatan proporsional, keluaran m(t) hanya bergantung pada Kp dan *error*,semakin besar *error* maka semakin besar koreksi yang dilakukan. Penambahan Kp akan menaikan penguatan sistem sehingga dapat digunakan untuk memperbesar kecepatan respon dan mengurangi kesalahan keadaan mantap.

2.8.2 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 mantabnya nol.

Kontroler integral memiliki karakteristik seperti halnya sebuah integral. Keluaran kontroler sangat dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal kesalahan (Rusli, 1997:18). 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 2.9 menunjukkan blok diagram kontroler integral.

Gambar 2.9 Diagram Blok Kontroler Integral

Sumber: Ogata, K., 2002

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

$$\frac{dm(t)}{dt} = Ki.e(t) \qquad (2\text{-}5)$$

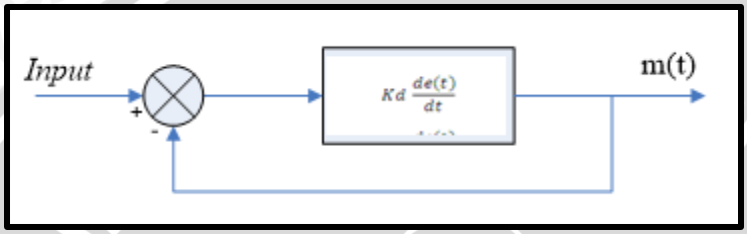
$$m(i) = Ki \int_0^t e(t)dt \qquad (2\text{-}6)$$

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 respon.

2.8.3 Kontroler Diferensial

Kontroler differensial memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan kontroler, akan mengakibatkan perubahan yang sangat besar dan cepat. Gambar 2.10 berikut menunjukkan blok diagram pada kontroler differensial.



Gambar 2.10 Blok diagram kontroler differensial

Sumber: Ogata, K., 2002

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

$$m(t) = Kd\ \frac{de(t)}{dt} \qquad (2\text{-}7)$$

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 respon transien sebuah sistem serta dapat meredam osilasi.

Berdasarkan karakteristik kontroler tersebut, kontroler diferensial umumnya dipakai untuk mempercepat respon 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 differensial tidak bisa digunakan tanpa ada kontroler lain.

2.8.4 Kontroler Proporsional Integral (PI)

Aksi kontrolnya dinyatakan dalam persamaan:

$$\mathrm{m(t)} = Kp.e(t) + \frac{Kp}{Ti}\int_0^t e(t)dt \qquad (2\text{-}8)$$

kontroler ini menghasilkan sinyal kesalahan $\int e(t)dt$ kemudian ditambahkan dengan sinyal kesalahan *e(t)*.

2.8.5 Kontroler Proporsional Derivative (PD)

Aksi kontrolnya dinyatakan dalam persamaan:

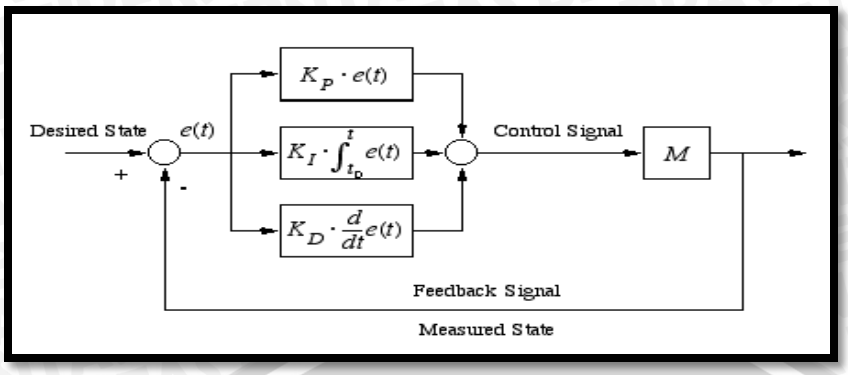
$$\mathrm{m(t)} = \mathrm{Kp.e(t)} + Kp.Td\ \frac{de(t)}{dt} \qquad (2\text{-}9)$$

Kontroler PD selalu mengukur kemiringan (*slope*) sinyal kesalahan $\frac{de(t)}{dt}$ dan memperkirakan akan besar *overshoot* yang akan terjadi serta memberikan koreksi sebelum terjadi lewatan sebenarnya sehingga diperoleh *maximum overshoot* yang kecil.

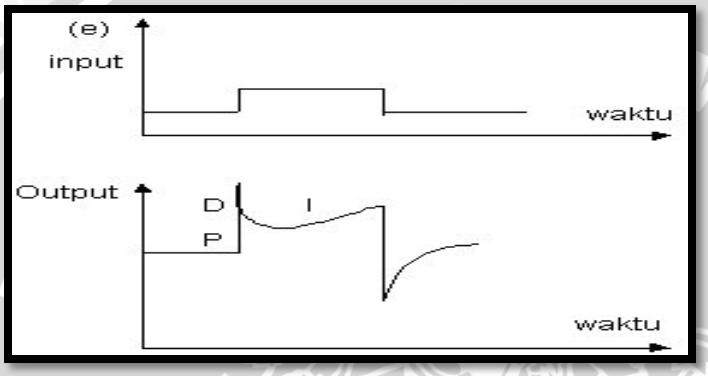
Jika kesalahan keadaan mantap tidak berubah terhadap waktu maka turunanya terhadap waktu sama dengan nol, sehingga kontroler PD tidak mempunyai pengaruh terhadap kesalahan keadaan mantap,tetapi jika terdapat perubahan kesalahan,kontroler PD akan mengurangi besar kesalahan keadaan mantap. Jadi kontrol PD digunakan untuk memperbaiki suatu sistem pengendalian yang tanggapan peralihanya mempunyai *maximum overshoot* yang berlebihan tanpa memperhitungkan kecepatan responnya.

2.8.6 Kontroler Proportional Integral Derivative (PID)

Setiap kekurangan dan kelebihan masing-masing kontroler P, I dan D dapat saling menutupi dengan menggabungkan ketiganya secara paralel menjadi kontroler proporsional integral derivative (PID). Elemen-elemen 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 ditunjukan dalam Gambar 2.11.



Gambar 2.11 Diagram blok kontroler PID



Gambar 2.12 Hubungan fungsi waktu antara sinyal keluaran dan masukan kontroler PID

Sumber: Gunterus, 1994:8-11

Aksi kontrolnya dapat dinyatakan sebagai berikut:

$$m(t) = K_p e(t) + \frac{Kp}{Ti} \int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \qquad (2\text{-}10)$$

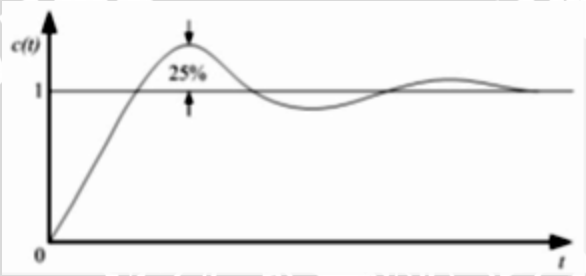
Jenis kontroler ini digunakan untuk memperbaiki kecepatan respon dan mencegah terjadinya kesalahan keadaan mantap serta mempertahankan kestabilan.

Keluaran kontroler PID merupakan penjumlahan dari keluaran kontroler proporsional,integral dan derivative. Gambar 2.6 menunjukan hubungan tersebut. Karakteristik kontroler PID sangat dipengaruhi oleh konstribusi besar dari ketiga marameter P,I dan D. Penyetelan konstanta Kp, Ki dan Kd akan mengakibatkan penonjolan sifat dari masing-masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat disetel lebih menonjol dibandingkan yang lain. Konstanta yang menonjol itulah yang akan memberikan kontribusi pengaruh pada respon sistem secara keseluruhan (Gunterus, 1994, 8-10).

2.8.7 Metode Perancangan Kontroler Proporsional Integral Deferensial (PID) Menggunakan Metode Ziegler-Nichols

Ziegler dan Nichols mengemukakan aturan-aturan untuk menentukan nilai dari gain proporsional Kp, waktu integral Ti, dan waktu derivatif Td berdasarkan karakteristik respon transien dari *plant* yang diberikan. Penentuan parameter kontroler PID atau penalaan kontroler PID tersebut dapat dilakukan dengan bereksperimen dengan plan.(Ogata, K., 1997)

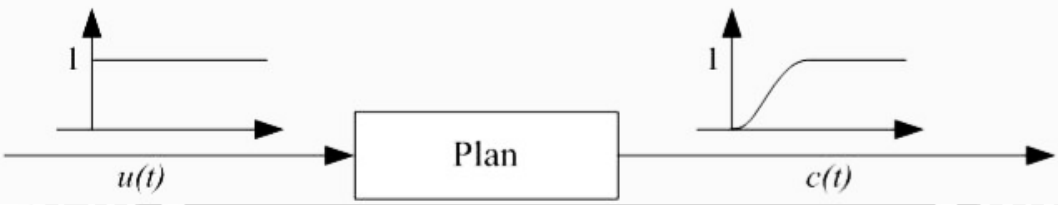
Terdapat dua metode yang disebut dengan aturan penalaan Ziegler-Nichols, pada kedua metode tersebut memiliki tujuan yang sama yaitu untuk mencapai 25% *maximum overshoot* pada respon unit step, seperti ditunjukkan dalam Gambar 2.13



Gambar 2.13 Kurva Respon Unit Step yang Menunjukkan 25% *Maximum Overshoot*
Sumber: Ogata, K., 1997

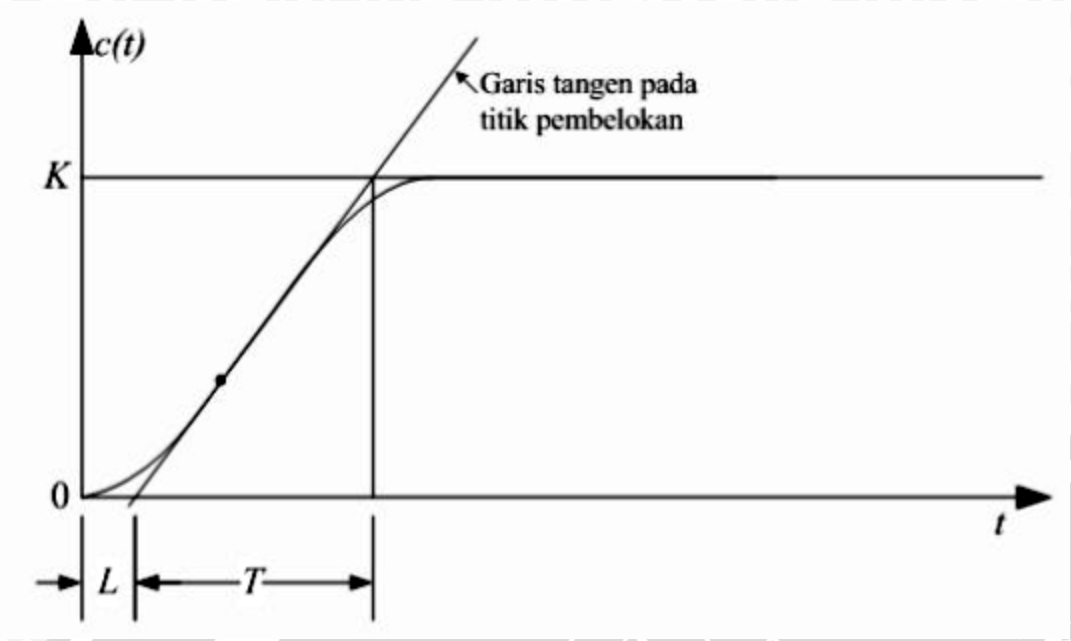
a). Metode Pertama

Metode pertama atau sering disebut metode kurva reaksi, respon dari plan dapat dapat diperoleh secara eksperimental dengan masukan berupa unit step, seperti yang ditunjukkan dalam Gambar 2.14.



Gambar 2.14 Respon Plant Terhadap Masukan Berupa Unit Step
Sumber: Ogata, K. 1997

Jika dalam plan tersebut terdapat integrator atau *dominan complex-conjugate poles*, maka kurva respon unit step berbentuk seperti huruf S, seperti dalam Gambar 2.15 jika respon tidak memberikan bentuk kurva S, maka metode ini tidak berlaku.(Ogata, K., 1997).



Gambar 2.15 Kurva Respon yang Berbentuk S
Sumber: Ogata, K. 1997

Kurva berbentuk S tersebut dapat dikarakteristikkan menjadi dua konstanta yaitu waktu tunda *L* dan konstanta waktu *T*. Waktu tunda dan konstanta waktu ditentukan dengan menggambar sebuah garis tangen pada titik pembelokan dari kurva S, dan menentukan perpotongan antara garis tangen dengan sumbu waktu *t* dan sumbu *c(t) =K*, seperti yang telah ditunjukkan dalam Gambar 2.15 Fungsi alih *C(s)/U(s)* dapat dilakukan pendekatan dengan sistem orde satu dengan persamaan sebagai berikut:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1}$$

Sumber: Ogata, K. 1997

Ziegler dan Nichols menyarankan untuk menentukan nilai-nilai dari Kp, Ti dan Td berdasarkan pada formula yang ditunjukkan dalam Tabel 2.1. (Ogata, K., 1997).

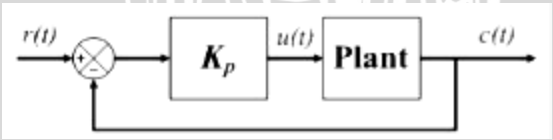
Tabel 2.1. Aturan Penalaan Ziegler-Nichols Berdasarkan Respon Unit Step Dari Plan

Tipe Kontroler	K_p	T_i	Td
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\ \frac{T}{L}$	$2L$	$0{,}5\ L$

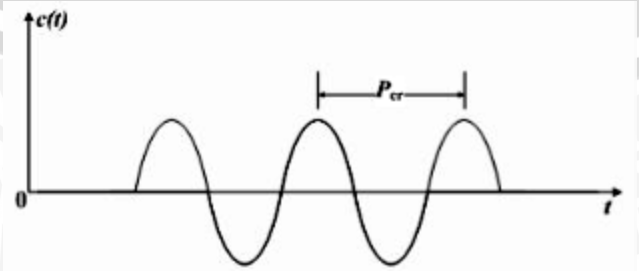
Sumber: Ogata, K. 1997

b). Metode Kedua

Dalam metode kedua ziegler-nichols, mula-mula yang dilakukan adalah membuat Ti = ∞ dan Td = 0. Kemudian hanya dengan menggunakan tindakan kontrol proporsional, harga ditingkatkan dari nol ke suatu nilai kritis *K*cr, disini mula-mula keluaran memiliki osilasi yang berkesinambungan (Jika keluaran tidak memiliki osilasi berkesinambungan untuk nilai Kp manapun yang telah diambil, maka metode ini tidak berlaku). Dari keluaran yang berosilasi secara berkesinambungan, penguatan kritis *K*cr dan periode *P*cr dapat ditentukan. Diagram blok sistem loop tertutup dengan kontroler proporsional dapat dilihat dalam Gambar 2.16. dan untuk osilasi berkesinambungan dengan periode Pcr dapat dilihat dalam gambar 2.17. Ziegler dan Nichols menyarankan penyetelan nilai parameter Kp, Ti, Td dan berdasarkan rumus yang diperlihatkan dalam Tabel 2.2 (Ogata, K., 1997).



Gambar 2.16 Sistem Loop Tertutup dengan Kontroler Proporsional
Sumber: Ogata, K., 1997



Gambar 2.17 Osilasi Berkesinambungan dengan Periode *P*cr
Sumber: Ogata, K., 1997

Tabel 2.2 Aturan Dasar Ziegler-Nichols Berdasarkan Critical Gain *Kcr* dan Critical Period *Pcr*

Tipe Kontroler	K_p	T_i	Td
P	0.5 Kcr	∞	0
PI	0.45 Kcr	$\frac{1}{1,2}$ Pcr	0
PID	0.60 Kcr	0.5 Pcr	0.125 Pcr

Sumber: Ogata, K., 1997

2.9 Mikrokontroler

Mikrokontroler populer yang pertama dibuat oleh Intel pada tahun 1976, yaitu mikrokontroler 8-*bit* Intel 8748.Mikrokontroler tersebut adalah bagian dari keluarga mikrokontroler MCS-48. Sebelumnya, Texas *instruments* telah memasarkan mikrokontroler 4-*bit* pertama yaitu TMS 1000 pada tahun 1974. TMS 1000 yang mulai dibuat sejak 1971 adalah mikrokomputer dalam sebuah *chip*, lengkap dengan RAM dan ROM.

Pengendali mikro (*microcontroller*) adalah sistem mikroprosesor lengkap yang terkandung di dalam sebuah *chip*.Mikrokontroler berbeda dari mikroprosesor serba guna yang digunakan dalam sebuah PC, karena sebuah mikrokontroler umumnya telah berisi komponen pendukung sistem minimal mikroprosesor, yakni memori dan antarmuka I/O.

Berbeda dengan CPU serba-guna, mikrokontroler tidak selalu memerlukan memori eksternal, sehingga mikrokontroler dapat dibuat lebih murah dalam kemasan yang lebih kecil dengan jumlah *pin* yang lebih sedikit.

Sebuah *chip* mikrokontroler umumnya memiliki fitur:

a) *central processing unit* - mulai dari prosesor 4-*bit* yang sederhana hingga prosesor kinerja tinggi 64-*bit*.
b) *input/output* antarmuka jaringan seperti *port serial* (UART)
c) antarmuka komunikasi serial lain seperti I²C, *Serial Peripheral Interface* and *Controller Area Network* untuk sambungan sistem
d) periferal seperti *timer* dan *watchdog*
e) RAM untuk penyimpanan data

f) ROM, EPROM, EEPROM atau *Flash memory* untuk menyimpan program komputer
g) pembangkit *clock* - biasanya berupa resonator rangkaian RC
h) pengubah *analog* ke *digital*

2.9.1 Mikrokontroler ATMega328

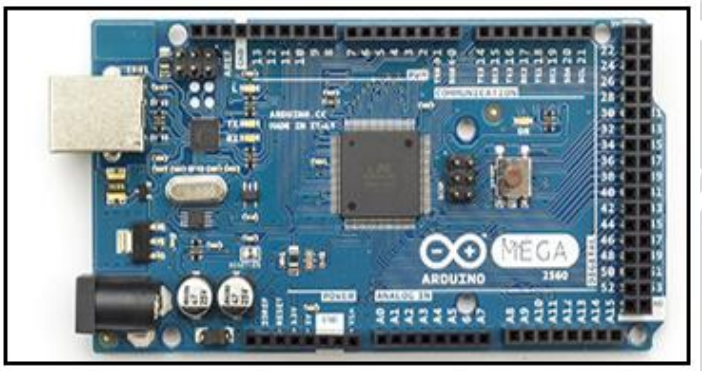
Atmel ATmega328 adalah mikrokontroler CMOS 8-bit berdaya rendah berbasis AVR yang arsitektur RISCnya telah ditingkatkan. Hampir semua instruksi dieksekusi dalam satu siklus clock, mempunyai throughput mendekati 1 MIPS per MHz membuat desainer sistem untuk mengoptimasi konsumsi daya versus kecepatan proses.

Atmel ATMega328 memiliki beberapa fitur antara lain 8Kbytes In-system Programmable Flash with Read-While-Write, 1K bytes EEPROM, 2K bytes SRAM, 23 jalur I/O untuk tujuan umum, 32 working registers untuk tujuan umum, tiga timer/counter yang fleksibel dengan compare mode, internal dan external interrupt, sebuah serial programmable USART, sebuah byte-oriented 2-wireSerial Interface, sebuah port SPI serial, sebuah 6-channel 10-bit ADC, sebuah Watchdog Timer yang programmable dengan internal osilator.

2.10 Arduino Mega

Arduino Mega 2560 adalah *board* mikrokontroler berbasis ATmega 168/328, dapat dilihat dalam Gambar 2.9. Memiliki 53 pin input dari output digital dimana 15 pin input tersebut dapat digunakan sebagai output *Pulse Width Modulation* (PWM) dan 16 pin input analog, 16 MHz osilator kristal, koneksi USB, jack power, ICSP header, dan tombol reset.

Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan *Board* Arduino Mega ke komputer dengan menggunakan kabel USB atau listrik dengan AC yang ke Arduino Mega berbeda dengan semua *board* sebelumnya dalam hal koneksi USB-to-serial yaitu menggunakan fitur Atmega8U2 yang diprogram sebagai konverter USB-to-serial berbeda dengan *board* sebelumnya yang menggunakan chip FTDI *driver* USB-to-serial.



Gambar 2.18 Board Arduino Mega

Sumber: electroschematic.com

BAB III
METODE PENELITIAN

Kajian dalam skripsi ini merupakan penelitin yang bersifat aplikatif, yaitu merancang suatu sistem pengendalian menggunakan PID yang bertujuan agar dapat menampilkan informasi sistem yang sesuai dengan yang direncanakan.

Langkah-langkah yang perlu dilakukan untuk merealsisasikan alat yang akan dibuat adalah sebagai berikut:

3.1 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditentukan terlebuh dahulu sebagai acuan dalam perancangan selanjutnya. Dengan perencanaan sebagai berikut:

1. Baling-baling adalah alat yang mengubah gerak putar menjadi daya dorong.
2. Menggunakan sensor *potensiometer*.
3. Motor BLDC sebagai pemutar baling-baling.
4. Motor brushless memiliki sebuah ESC (*Electronic Speed Control*) yang berfungsi sebagai pengatur kecepatan motor.
5. Kontrol yang digunakan adalah Kontrol PID
6. Menggunakan Arduino Mega sebagai tempat pemrograman Kontrol PID.

3.2 Perancangan dan Realisasi Pembuatan Alat

3.2.1 Perancangan Perangkat Keras dan Realisasi Pembuatan Alat

a. Pembuatan diagram blok
b. Penentuan dan Perhitungan komponen yang akan digunakan dalam perancangan alat
c. Merakit perangkat keras (*hardware*) untuk masing-masing blok.

3.2.2 Perancangan dan Perhitungan Komponen yang akan Digunakan

Setelah merancang perangkat keras, maka langkah selanjutnya adalah merancang perangkat lunak guna mengendalikan dan mengatur kerja daripada alat. Desain dan parameter yang telah dirancang kemudian diterapkan pada Arduino Mega 2560 dengan menggunakan *software* Arduino ERW 1.0.5.

3.2.3 Perancangan Perangkat Lunak

Perancangan perangkat lunak dimulai dari pembuatan *flowchart*, kemudian penulisan *listing code* program PID pada software Arduino ERW 1.0.5.

3.3 Pengujian Alat

Setelah semua komponen pada alat sudah terealisasikan sesuai dengan diagram blok yang telah dirancang dan perangkat lunak pendukung sistem sudah dibuat, maka diadakan pengujian dan analisa alat. Metode pengujiannya adalah sebagai berikut :

1. Pengujian Sensor

   Pengujian sensor dilakukan dengan cara mensimulasikan rangkaian sensor dan hasil pemodelan rangkaian sensor. Pengujian ini bertujuan untuk memastikan sensor dan hasil pemodelan sensor dapat bekerja sesuai dengan perancangan dan memberikan analisis terhadap hasil pengujian. Terdapat rangkaian sensor utama yang akan diuji, yaitu sensor *potensiometer* sebagai pengendali keseimbangan *frame*.
2. Pengujian Sinyal Kontrol Motor BLDC

   Pengujian sinyal kontrol motor BLDC ini bertujuan untuk melihat bagaimana bentuk sinyal saat berada pada posisi sudut yang telah ditentukan serta melihat tegangan yang dikeluarkan untuk setiap perubahan berat beban uji.
3. Pengujian Motor BLDC terhadap *propeller*

Pengujian ini dilakukan untuk mengetahui pengaruh perubahan berat beban uji terhadap kecepatan putaran motor BLDC, sehingga kita dapat menentukan besarnya perubahan kecepatan putaran tiap perubahan berat beban ujinya.

4. Pengujian tanpa Kontroler

   Pengujian ini bertujuan untuk mengetahui bagaimana kinerja sistem secara keseluruhan dan mengamati respons terhadap *setpoint* ketika tanpa kontroler PID. Dengan begitu kita dapat menentukan bahwa sistem ini perlu diberikan pengontrolan atau tidak.

5. Pengujian Keseluruhan Sistem

   Pengujian ini bertujuan untuk mengetahui bagaimana kinerja sistem secara keseluruhan dan mengamati respons kontroler PID terhadap *setpoint* dan ketika mendapatkan gangguan. Dengan adanya gangguan maka kita dapat melihat sistem pengendalian telah berjalan dengan baik atau tidak.

3.4 Pengambilan Kesimpulan dan Saran

Kesimpulan diambil berdasarkan data yang didapat dari hasil pengujian sistem secara keseluruhan. Apabila hasil yang didapatkan sesuai dengan yang direncanakan sebelumnya, maka sistem kendali tersebut telah berhasil memenuhi harapan dan dapat dikembangkan untuk penelitian selanjutnya untuk disempurnakan.

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Dalam skripsi ini penulis merancang sebuah alat uji coba statis satu lengan yaitu sebuah alat yang berada dalam keadaan diam dan terdapat dua sisi, depan dan belakang, yang terdiri dari satu rotor bicopter dan sebuah beban uji di sisi lainnya yang bisa diubah-ubah beratnya . Alat uji coba ini adalah sebuah alat yang mampu mengendalikan kestabilan horizontal

Dalam bab ini membahas mengenai perancangan dan pembuatan sistem pengendali kecepatan motor *propeller* pada *bicopter* menggunakan Kontrol Proporsional Integral Differensial (PID). Perancangan perangkat tersebut meliputi perangkat keras maupun perancangan perangkat lunak. Sedangkat pembuatan bertujuan untuk menghasilkan semua pendukung maupun alat secara keseluruhan.

4.1 Perancangan Sistem

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

1. Perancangan perangkat keras (sensor *Potensiometer*, *propeller* (baling-baling), dan modul Arduino Mega)
2. Perancangan perangkat lunak (perancangan algoritma PID pada software Arduino Mega ERW 1.0.5).

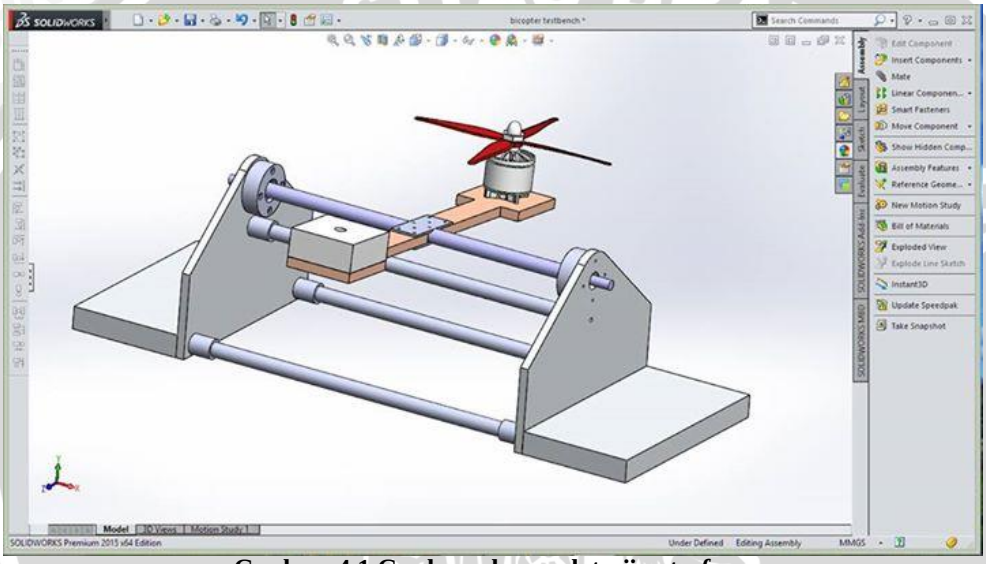
4.1.1 Perancangan Sistem Mekanik

Untuk mendesain *frame bicopter* secara lengkap diperlukan alat uji satu *frame*. Alat bantu desain *frame bicopter* merupakan model satu *frame* dengan satu aktuator motor dan propeller. Sisi lainnya merupakan beban uji yang dapat diubah-ubah beratnya. Dengan menggabungkan kontroler PID, dapat diuji kestabilan yang dihasilkan dengan mengubah-ubah parameter: panjang *frame*, spesifikasi motor dan propeller dengan konstanta PID tertentu. Parameter yang diperoleh dari alat uji

untuk membantu desain *bicopter* akan dapat digunakan sebagai dasar desain *bicopter* secara keseluruhan. Spesifikasi mekanik *frame* adalah sebagai berikut:

1. Panjang 95 cm
2. Lebar 40 cm
3. Tinggi 31 cm

Ukuran dimensi alat dipilih dengan pertimbangan agar alat uji menghasilkan berat yang cukup sehingga alat uji tidak bergeser atau bergerak disebabkan pengaruh dari daya hembus baling-baling pada saat melakukan pengujian. Konstruksi alat yang dirancang dapat dilihat dalam dalam Gambar 4.1.



Gambar 4.1 Gambar skema alat uji satu *frame*

Frame adalah sebuah kerangka keseimbangan menyerupai jungkat-jungkit dengan kedua sisi sama panjang, satu sisi terdapat *propeller* dan sisi lainnya merupakan beban uji yang dapat diubah beratnya. Selanjutnya kecepatan putaran motor diatur dengan kontroler PID agar memiliki kecepatan yang diinginkan. Umpan balik yang digunakan berupa sensor *potensiometer* yang diletakkan pada tepat ditengah antara kedua sisi.

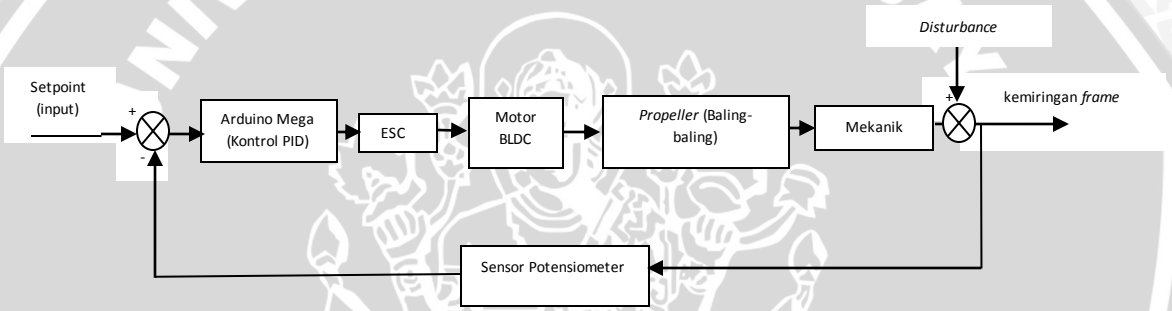
4.1.2 Perancangan Sistem Kontroler

Spesifikasi alat yang di rancang adalah sebagai berikut:

1. Arduino Mega 2560 dan software Arduino ERW 1.0.5 untuk pengembangan kontroler Proporsional Integral Differensial (PID).
2. Sensor yang digunakan berupa sensor *potensiometer* berfungsi sebagai feedback pengontrolan posisi.

4.2 Diagram Blok Sistem

Dalam skripsi ini dibuat diagram blok agar dalam pengerjaan dapat dilakukan sesuai dengan rangcangan sistem. Adapun diagram blok tersebut dapat dilihat pada Gambar 4.2:



Gambar 4. 2 Diagram Blok Sistem

Keterangan dari diagram blok dalam Gambar 4.2:

- *Input* sistem berupa kemiringan *frame* yang diinginkan yaitu *frame* pada kondisi seimbang.
- Kontroler berfungsi sebagai penerima sinyal *error* dan memberi sinyal kontrol pada aktuator.
- *Output* dari mikrokontroler ini berupa sinyal pulsa yang dirubah menjadi tegangan oleh ESC.
- Kemudian ESC melakukan proses sesuai instruksi dari kontroler yaitu mengatur jumlah arus yang diperlukan oleh motor yang dicatu oleh power supply.
- Motor menghasilkan putaran pada baling-baling yang menghasilkan daya angkat/daya dorong pada *frame*.

- *Output* dari mekanik berupa kemiringan *frame* alat uji kemudian ditambahkan dengan gangguan dari luar sistem dan dibaca oleh sensor potensiometer. Keluaran dari sensor yang berupa analog kemudian diubah menjadi digital oleh converter ADC. Keluaran dari sensor disebut sinyal *feedback*.
- Hasil akhir sinyal *feedback* kemudian dikurangkan dengan *input/setpoint* sehingga mikrokontroler mampu mengkompensasi *error* yang terjadi.

4.3 Perancangan Perangkat Keras

4.3.1 *Propeller* (Baling-baling)

Propeller yang digunakan pada ujung motor BLDC adalah *propeller* tipe 10 inch. Pemilihan ini didasarkan pada rekomendasi pabrik jika digunakan pada dalam RC (*Radio Control*) aeromodelling. *Propeller* digunakan untuk mengubah sudut (*pitch*) pada alat uji satu *frame* ini sehingga mempengaruhi daya angkat (*thrust*) pada *frame*. Bentuk *propeller* ditunjukkan dalam Gambar 4.3 berikut ini.

Gambar 4.3 *Propeller*

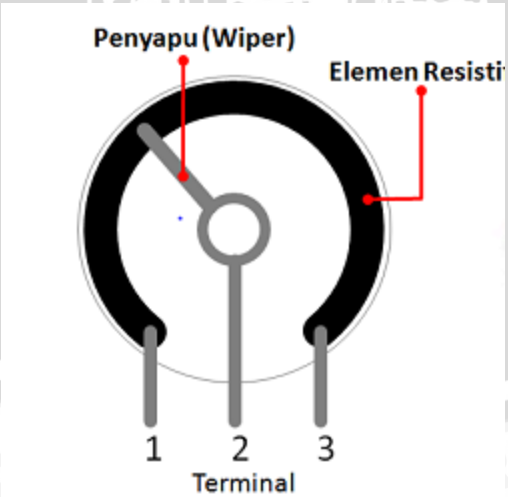
4.3.2 Sensor Potensiometer

Sensor yang di gunakan pada perancangan alat ini adalah sensor potensiometer tipe *linear wire-wound* 10 kΩ yang bekerja sebagai pengontrolan posisi. Bentuk sensor potensiometer ditunjukkan dalam Gambar 4.4 berikut ini.

Gambar 4.4 Sensor Potensiometer

Sebuah Potensiometer terdiri dari sebuah elemen resistif yang membentuk jalur (*track*) dengan terminal di kedua ujungnya. Sedangkan terminal lainnya (biasanya berada di tengah) adalah Penyapu (*Wiper*) yang dipergunakan untuk menentukan pergerakan pada jalur elemen resistif (*Resistive*). Pergerakan Penyapu (*Wiper*) pada Jalur Elemen Resistif inilah yang mengatur naik-turunnya Nilai Resistansi sebuah Potensiometer.

Elemen Resistif pada Potensiometer umumnya terbuat dari bahan campuran Metal (logam) dan Keramik ataupun Bahan Karbon (*Carbon*). Struktur potensiometer ditunjukkan dalam Gambar 4.5



Gambar 4.5 Struktur Potennsiometer

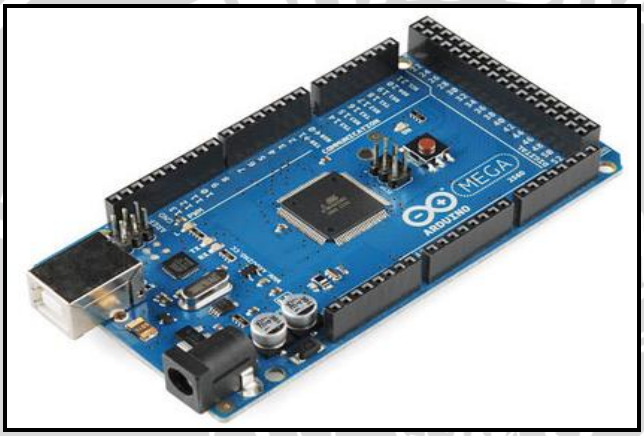
Sumber: teknikelektronika.com

4.3.3 Motor BLDC *(Brushless Direct Current)*

Motor *BLDC* yang digunakan dalam perancangan kali ini berguna sebagai penggerak baling–baling (*propeller*). Motor BLDC ini juga dapat langsung terhubung ke Arduino Mega tanpa menggunakan driver karena bekerja pada maksimum tegangan masukan 4,8 V.

4.3.4 Modul Arduino Mega

Pada perancangan alat ini Arduino Mega berbasis mikrokontroler ATmega328 digunakan sebagai pusat pengolah utama dalam melakukan proses pengendalian. . Tampak depan Arduino mega 2560 ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Tampak depan Arduino Mega 2560

Arduino Mega 2560 adalah merupakan *board* mikrokontroler berbasis ATMega 2560. Modul ini memiliki 54 digital *input/output* di mana 14 digunakan untuk *output* PWM dan 16 digunakan sebagai analog *input*, 4 untuk UART, 16 MHz osilator kristal, koneksi USB, *power jack*, *ICSP Header*, dan tombol reset.

Modul ini memiliki segalanya yang dibutuhkan untuk memprogram mikrokontroler seperti kabel USB dan sumber daya melalui Adaptor ataupun baterai. Pin masukan dan keluaran Arduino Mega 2560 pada perancangan ini akan difungsikan sesuai Tabel 4.1

Tabel 4.1 Fungsi Pin Arduino Mega 2560

No	Pin	Fungsi
1	A0	Masukan *ESC*
2	8	Masukan motor
3	GND	Jalur masukan GND seluruh sistem
4	Vin	Jalur masukan 5V seluruh sistem

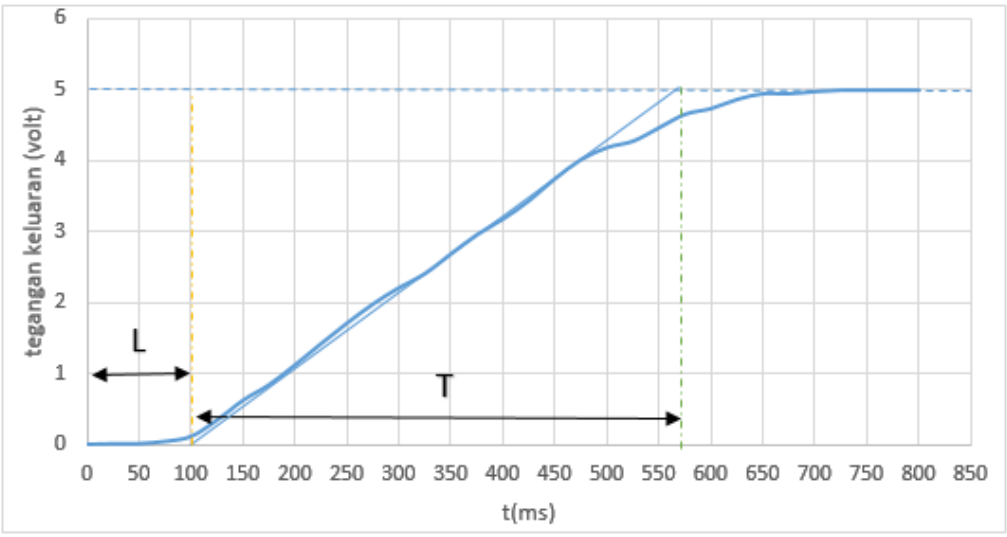
4.4 Perancangan Kontrol PID

4.4.1 Penentuan Nilai Penguatan Kontroler

Untuk menentukan penguatan kontroler (Kp, Ki, Kd) digunakan metode *tuning eksperimen* untuk mendapatkan hasil respon sesuai dengan yang diinginkan. *Tuning eksperimen* adalah proses yang dilakukan untuk mendapatkan hasil kontroler yang optimal dengan cara suatu percobaan. Inti dari *tuning* eksperimen adalah menentukan nilai dari tiga buah parameter yang terdapat pada kontroler PID yaitu konstanta proporsional (Kp), konstanta integral (Ki) dan konstanta diferensial (Kd) menggunakan metode 1 Ziegler-Nichols.

Dari hasil pengujian *open loop* yang diperlihatkan dalam Gambar 5.2 pada bab selanjutnya berupa kurva S akan digunakan menentukan parameter *tuning* PID dengan metode 1 Ziegler-Nichols. Adapun langkah-langkah yang dilakukan sebagai berikut:

1. Menarik garis tangent pada titik infleksi grafik karakteristik *open loop* seperti yang diperlihatkan dalam Gambar 4.7
2. Menentukan perpotongan garis tangent terhadap sumbu waktu t untuk mendapatkan L
3. Menentukan perpotongan garis tangent terhadap sumbu *steady* untuk mendapatkan nilai T
4. Nilai L dan T digunakan untuk menentukan nilai Kp, Ti dan Td sesuai dengan Tabel 4.2



Gambar 4.7 Metode 1 *Ziegler-Nichols* (hasil pengujian)

Tabel 4.2 Aturan Metode 1 Ziegler-Nichols (Ogata K., 1997)

Tipe Kontrol	Kp	τ_i	τ_d
P	$\frac{\tau}{L}$	∞	0
PI	$0,9\ \frac{\tau}{L}$	$\frac{L}{0,3}$	0
PID	$1,2\ \frac{\tau}{L}$	2L	0,5L

Dari Gambar 4.7 diperoleh besarnya L = 100ms dan T = 575-100 = 475 ms.

Dengan demikian parameter kontroler diperoleh sebagai berikut:

$$K_p = 1,2\ \frac{\tau}{L} = 1,2 \times \frac{0,475\ s}{0,1\ s} = 5,7$$

$$\tau_i = 2L = 2 \times 0,1s = 0,2$$

$$\tau_d = 0,5L = 0,5 \times 0,1s = 0,05$$

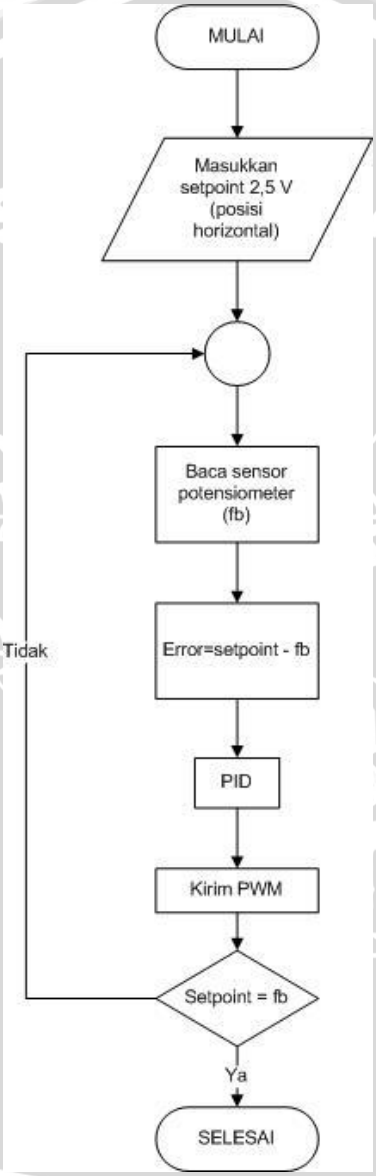
Selanjutnya akan diperoleh Ki dan Kd sebagai berikut:

$$K_i = \frac{Kp}{\tau_i} = \frac{5,7}{0,2} = 28,5$$

$$K_d = Kp \times \tau_d = 5,7 \times 0,05 = 0,285$$

4.5 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada pengendalian ini dengan menggunakan *template* PID pada *software* Arduino ERW 1.0.5. *Tuning* kontroler PID adalah dengan memasukkan nilai Kp, Ki, dan Kd dari hasil perhitungan metode Ziegler-Nichols. *Flowchart* perancangan perangkat lunak dapat dilihat pada Gambar 4.6.



Gambar 4.6 *Flowchart* Perangkat Lunak

Bila dijelaskan lebih detail, dengan menggunakan simbol dan keterangan flowchart dapat dijabarkan langkah-langkah yang bisa dilakukan oleh user sebagai berikut:

a) *User* pilih "mulai" yang diwakili oleh simbol *terminator* yang menggambarkan kegiatan awal atau akhir dari suatu proses. Pada langkah ini simbol *terminator* menjabarkan kegiatan awal program memberi masukan sebesar 2,5 volt.
b) pada saat *user* memilih memberi masukan 2,5 volt, *user* masih berada pada halaman yang sama (simbol *on-page reference*).
c) Kemudian membaca nilai sensor potensiometer (fb) dimana diwakili oleh simbol proses yang berfungsi menggambarkan suatu proses.
d) Setelah mendapatkan nilai sensor potensiometer, kemudian melakukan proses masukan 2,5 volt dikurangkan dengan nilai potensiometer yang terbaca (fb) menghasilkan nilai *error*.
e) nilai *error* sebagai masukan kontroler PID untuk menentukan gain Kp, Ki, dan Kd.
f) Kontroler PID melakukan proses mengirim sinyal pulsa PWM.
g) Setelah mengirim sinyal pulsa PWM, *user* disediakan dua kondisi yang harus dia pilih. Kondisi yang ada diwakili oleh simbol *decision*. *Decision* adalah simbol untuk menunjukkan sebuah langkah pengambilan keputusan. Umumnya, menggunakan bentuk pertanyaan, dan biasanya jawabannya terdiri dari 'yes' dan 'no' atau 'ya' dan 'tidak' yang menentukan bagaimana alur dalam flowchart berjalan selanjutnya berdasarkan kriteria atau pertanyaan tersebut. Jika kondisi sesuai dengan kriteria maka *user* memilih "selesai" yang menggambarkan akhir dari proses ini. Jika kondisi tidak sesuai, maka *user* kembali ke proses membaca sensor potensiometer (fb). Proses terus berulang sampai kondisi sesuai dengan yang diinginkan.

BAB V
PENGUJIAN DAN ANALISIS

Tujuan pengujian sistem ini adalah untuk menentukan apakah alat yang telah dibuat berfungsi dengan baik dan sesuai dengan perancangan. Pengujian pada sistem ini meliputi pengujian setiap blok maupun pengujian secara keseluruhan. Pengujian setiap blok ini dilakukan untuk menemukan letak kesalahan dan mempermudah analisis pada sistem apabila alat tidak bekerja sesuai dengan perancangan. Adapun langkah – langkah pengujian yang dilakukan adalah:

1. Pengujian sensor *potensiometer*
2. Pengujian Respon Aktuator Motor DC *brushless*
3. Pengujian keseluruhan sistem

5.1 Pengujian Sensor

a. Tujuan

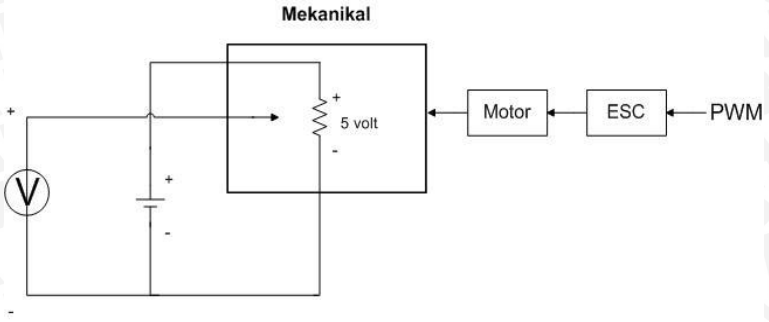
Menguji respon statis keluaran sensor potensiometer berdasarkan variasi kecepatan motor BLDC.

b. Peralatan yang digunakan

- Sensor *potensiometer*
- *Voltmeter*
- Mekanikal alat uji satu *frame*
- Baterai 12 V DC ESC catu daya 5 VDC

c. Langkah pengujian

1. Merangkai peralatan pengujian seperti pada Gambar 5.1
2. Beban yang diberikan pada pengujian ini adalah 300 gram pada sisi lengan lainnya
3. Sambungkan catu daya
4. Mengatur kecepatan putaran motor mulai minimal sampai maksimal dengan menggunakan pembangkit PWM
5. Ukur tegangan keluaran sensor setiap kecepatan yang diberikan.



Gambar 5. 1 Diagram Perancangan Peralatan Pengujian Sensor

d. Hasil Pengujian

Hasil pengujian didapatkan dengan melihat tegangan keluaran yang terbaca oleh sensor potensiometer diperlihatkan dalam tabel 5.1.

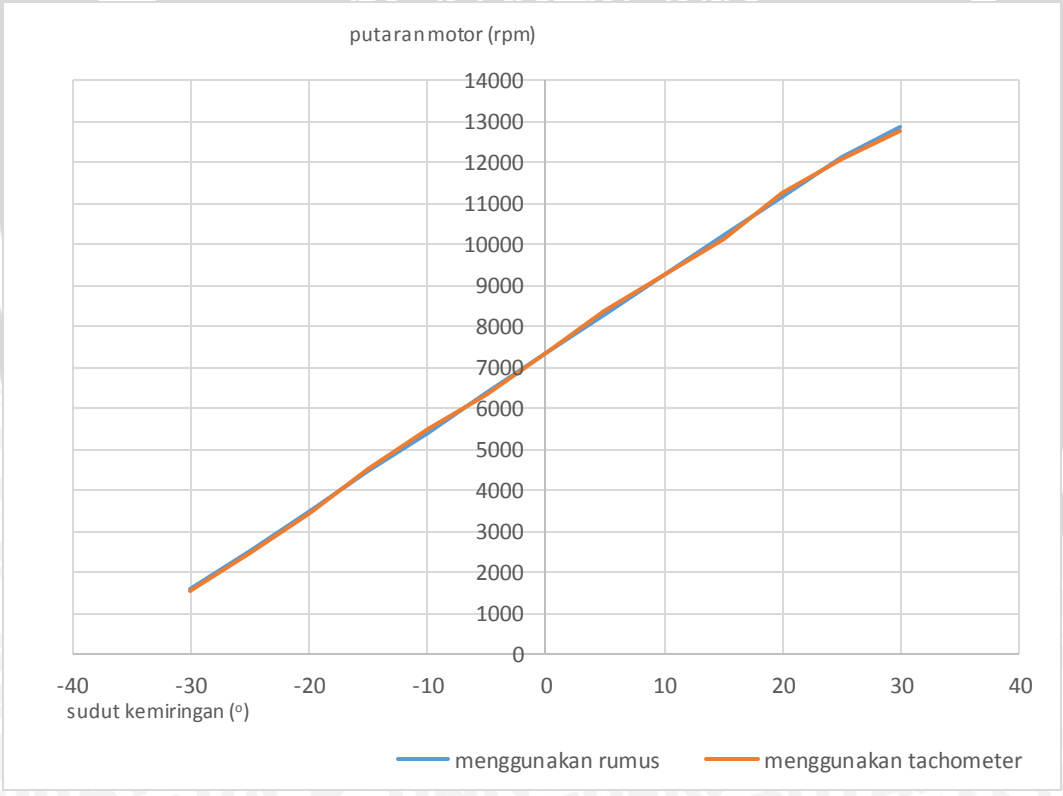
Tabel 5. 1 Hasil Pengujian Sensor

PWM	Kecepatan motor (rpm)	Sudut kemiringan frame (°)	Tegangan Keluaran (Volt)
28	1581,176	-30	0
45	2541,176	-25	0,42
62	3501,176	-20	0,83
79	4461,176	-15	1,25
96	5421,176	-10	1,67
113	6381,176	-5	2,08
130	7341,176	0	2,50
147	8301,176	5	2,92
164	9261,176	10	3,33
181	10221,18	15	3,75
198	11181,18	20	4,17
215	12141,18	25	4,58
228	12875,29	30	5,00

Spesifikasi motor BLDC adalah 1200 KV yaitu jika diberikan tegangan 1 V akan menghasilkan kecepatan motor sebesar 1200 rpm. Untuk mengatur tegangan dengan cara memberi sinyal pulsa menggunakan PWM *Generator* dengan nilai 0 sampai 255. Baterai *lippo* yang digunakan sebesar 12 V. Jadi PWM 0 sampai 255 ekuivalen dengan tegangan 0 sampai 12 V. Sebagai contoh untuk PWM 130 maka:

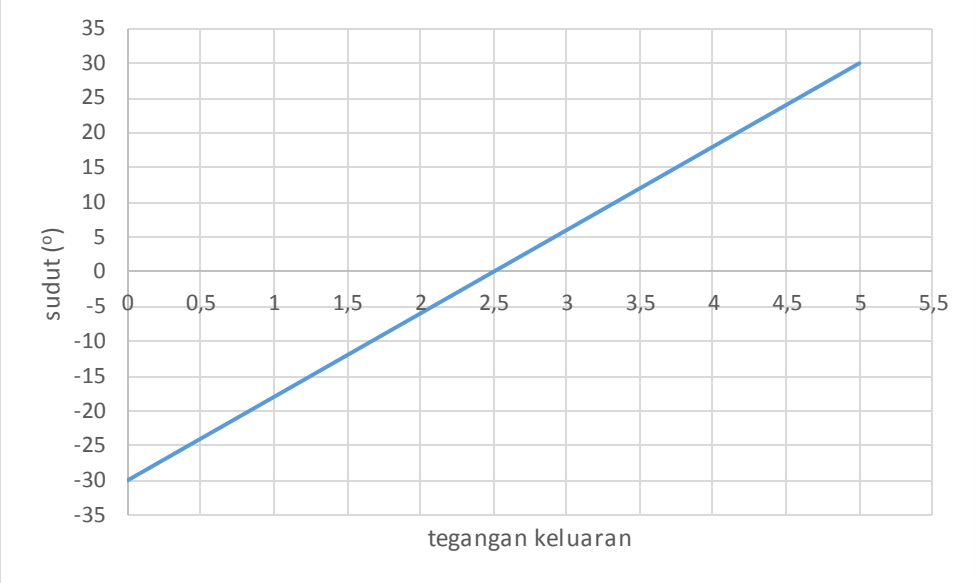
$$rpm = \frac{130}{255}\ x\ 14.400\ rpm = 7341{,}17\ rpm$$

Berikut adalah grafik dari kecepatan motor terhadap sudut dari perhitungan diatas. Kemudian dilihat pula hasil dengan pengukuran menggunakan *tachometer* untuk membuktikan kebenaran hasil perhitungan dari rumus diatas.



Gambar 5.2 Perbandingan perhitungan kecepatan motor menggunakan rumus dengan menggunakan *tachometer*

Perbandingan antara tegangan keluaran sensor dengan sudut kemiringan frame ditunjukkan pada Gambar 5.3 berikut.



Gambar 5.3 Perbandingan antara tegangan keluaran sensor dengan sudut kemiringan frame

Dari grafik di atas dengan menggunakan persamaan gradien :

$$\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1} \qquad (5\text{-}1)$$

Sehingga jika garis melewati titik (2,5;0) dan (5,30), maka persamaan garis pada grafik diatas adalah :

$$\frac{y-y1}{y2-y1} = \frac{x-x1}{x2-x1}$$

$$\frac{y-0}{30-0} = \frac{x-2,5}{5-2,5}$$

$$\frac{y}{30} = \frac{x-2,5}{2,5}$$

$$2,5\,y = 30x - 75$$

$$y = 12x - 30$$

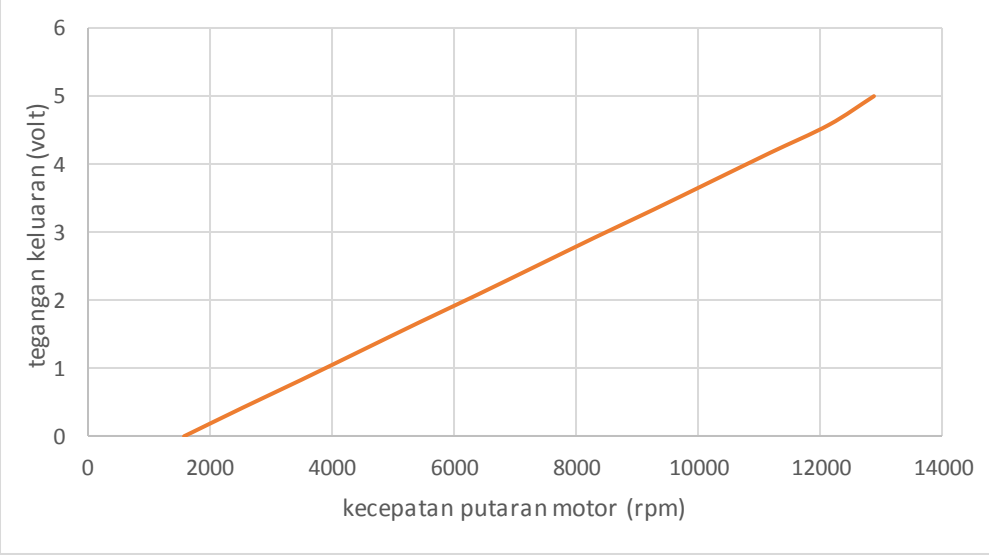
Jadi, misal tegangan keluaran sensor terbaca 3 volt maka sudut kemiringan framenya adalah:

$$y = 12(3) - 30$$

$$y = 6^o$$

Hasil dari pengujian sensor potensiometer ditunjukkan Gambar 5.4 dan Gambar 5.5 berikut.

Gambar 5.4 Grafik perbandingan antara kecepatan putaran motor BLDC terhadap sudut kemiringan *frame*



Gambar 5.5 Grafik perbandingan antara kecepatan putaran motor BLDC terhadap tegangan keluaran sensor

Dari hasil pengujian yang dilakukan terlihat hubungan antara kecepatan putaran motor dengan kemiringan frame, baik secara fisik yaitu dengan pengamatan sudut maupun dengan pembacaan tegangan pada potensiometer menggunakan multimeter. Jadi sensor dapat bekerja dengan maksimal dan terlihat kelinieran yang baik sehingga ideal untuk digunakan untuk membaca sudut kemiringan pada *frame*.

5.2 Pengujian Respon Aktuator Motor DC *Brushless*

a. Tujuan

Menguji respon aktuator secara *open loop* untuk mendapatkan kurva S

b. Peralatan yang digunakan

- Sensor *potensiometer*
- Mekanikal alat uji satu *frame*
- Baterai 12 V DC ESC catu daya 5 VDC
- *Data logger*

c. Langkah pengujian

1. Merangkai peralatan seperti Gambar 5.1 dan mengganti voltmeter dengan *data logger*
2. Beban yang diberikan pada pengujian ini adalah 300 gram pada sisi lengan lainnya
3. Sambungkan catu daya
4. Kecepatan motor BLDC diatur maksimal seketika dengan menggunakan PWM *Generator* dengan sinyal pulsa sebesar 255
5. Respon yang dihasilkan direkam menggunakan *data logger*.

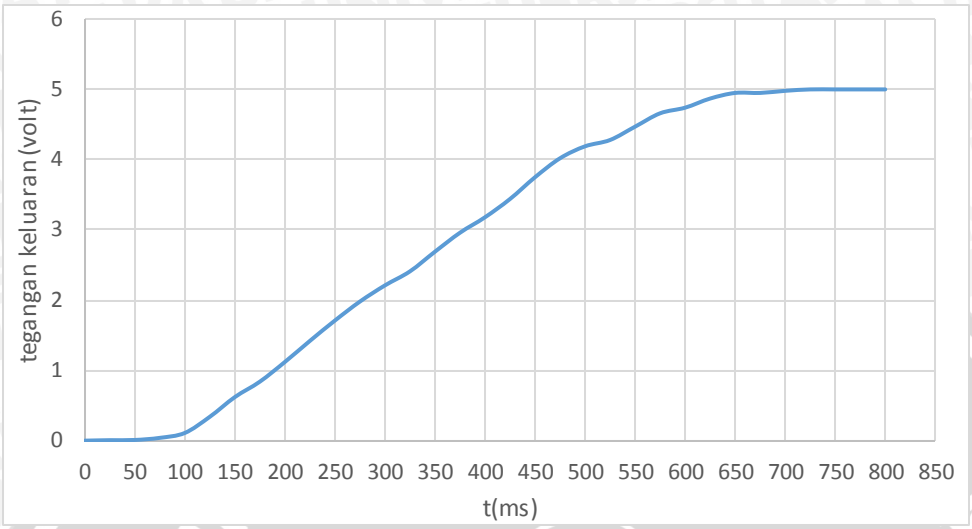
d. Hasil Pengujian

Setelah melakukan pengujian sesuai dengan langkah diatas, didapatkan respon aktuator saat kecepatan motor BLDC diatur maksimal seketika ditunjukkan dalam Tabel 5.2 berikut:

Tabel 5.2 Respon aktuator

t(ms)	Tegangan keluaran (volt)
25	0,007
50	0,01
75	0,04
100	0,11
125	0,34
150	0,62
175	0,84
200	1,12
225	1,42
250	1,71
275	1,98
300	2,21
325	2,41
350	2,69
375	2,96
400	3,18
425	3,44
450	3,75
475	4,02
500	4,19
525	4,28
550	4,47
575	4,66
600	4,74
625	4,87
650	4,95
675	4,95
700	4,98
725	5,00
750	5,00
775	5,00
800	5,00

Hasil pengujian respon aktuator secara grafik ditunjukkan dalam grafik 5.6 berikut.



Gambar 5.6 Grafik respon aktuator saat kecepatan motor diatur maksimal seketika

Dari hasil pengujian yang dilakukan, dalam grafik dapat dilihat bahwa saat kecepatan motor diatur maksimal seketika, sistem mampu mencapai keadaan *steady state* pada saat 650 ms.

5.3. Pengujian sistem keseluruhan

a. Tujuan

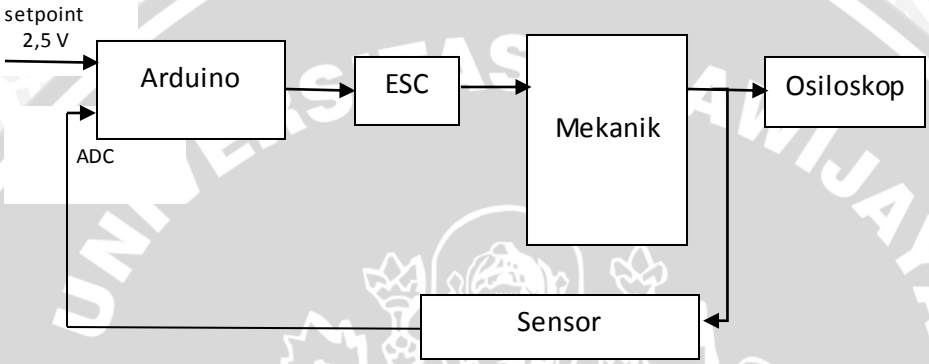
Pengujian ini bertujuan untuk mengetahui respon sistem secara *closed loop* dengan memberikan gangguan sementara.

b. Peralatan yang digunakan

- Catu Daya 5 V DC.
- Mikrokontroler Arduino Mega
- Komputer
- *Digital Oscilloscope* Vellemen PCSU1000 dan software PC Lab 2000SE.
- Program dan software Arduino.

c. Langkah pengujian

1. Merangkai peralatan seperti Gambar 5.7.
2. Mengunduh program pengatur kecepatan motor pada software Arduino ERW 1.0.5.
3. Menjalankan software PC Lab 2000SE
4. Memilih mode osiloskop lalu pilih *Run* untuk menjalankan osiloskop.
5. Mengamati sinyal kontrol dan parameter motor pada osiloskop.



Gambar 5.7 Diagram Peralatan Pengujian Motor DC *brushless*

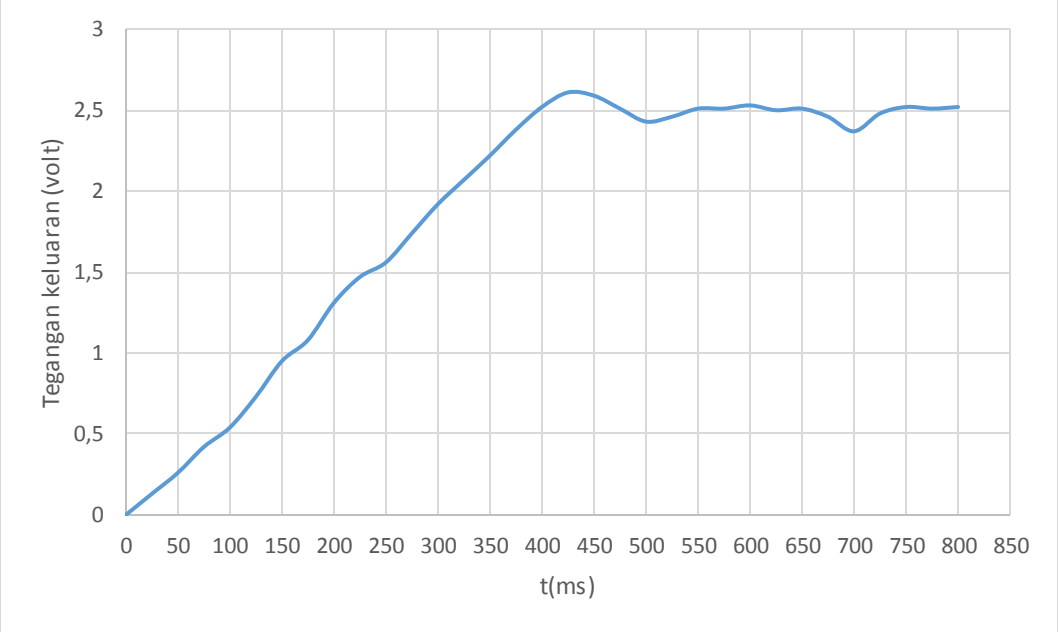
d. Hasil pengujian

Setelah melakukan pengujian sesuai dengan langkah diatas, didapatkan ditunjukkan dalam Tabel 5.4 berikut:

Tabel 5.3 Hasil Pengujian Sistem Keseluruhan

t (ms)	Tegangan keluaran (volt)
0	0
25	0,13
50	0,26
75	0,42
100	0,54
125	0,73
150	0,95
175	1,08
200	1,31
225	1,47
250	1,56
275	1,74
300	1,92
325	2,07
350	2,22
375	2,38
400	2,52
425	2,61
450	2,59
475	2,51
500	2,43
525	2,46
550	2,51
575	2,51
600	2,53
625	2,50
650	2,51
675	2,46
700	2,37
725	2,48
750	2,52
775	2,51
800	2,52

Secara grafik hasil pengujian di perlihatkan dalam gambar.



Gambar 5.8 Grafik hasil pengujian *closed loop* dengan memberikan gangguan sementara

Dari hasil pengujian secara *closed loop* seperti yang diperlihatkan dalam Gambar 5.8 dapat diperoleh parameter unjuk kerja sistem sebagai berikut:

1. *Peak Time* (t_p) adalah waktu yang diperlukan untuk respon untuk mencapai puncak pertama *overshoot*. t_p berdasarkan pengujian adalah 425 ms
2. Besarnya *Maximum Overshoot* sebesar $\frac{2,61-2,5}{2,5} \times 100\% = 4,4\%$
3. *Steady State* terjadi pada waktu 550 ms
4. *Error steady state* adalah nilai kesalahan saat respon telah mencapai pada keadaan tunak / *steady*. Persentase kesalahan dapat dicari dengan menggunakan persamaan (5-2) (Mahesta C., 2014).

$$\%E = \frac{1}{N}\sum_{i=1}^{N} \frac{|\text{PV} - \text{SP}|}{\text{SP}} \; x \; 100\ \% \qquad (5\text{-}2)$$

Dimana : E = Error

N = Jumlah data saat mencapai *setpoint*

PV = *Present value*

SP = *Setpoint*

$$\%E = \frac{1}{23}\ x\ 0{,}00291\ x\ 100\%$$

$$= 0{,}1265\%$$

5. *Recovery Time* adalah waktu yang dibutuhkan sistem untuk mengembalikan ke kondisi *steady-state* setelah terjadinya *error*. Berdasarkan pengujian *recovery time* dengan gangguan secara acak adalah 75 ms.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang dapat diambil dalam penelitian ini adalah sebagai berikut:

1. Mekanik yang digunakan adalah sebuah model menyerupai kerangka *bicopter*. Model memiliki dua sisi, yaitu di satu sisi terdapat motor dan *propeller* sebagai daya dorong dan berguna untuk mengontrol keseimbangan kerangka, dan di sisi lainnya terdapat variabel bebas berupa beban uji yang dapat diubah-ubah beratnya.
2. Hardware yang digunakan adalah model *frame bicopter* rancangan sendiri, Arduino Mega 2560, motor BLDC, sensor potensiometer, *Electric Speed Control* (ESC). Software yang digunakan adalah Arduino 1.0.6 yang berfungsi memberikan perintah kepada hardware yang ada agar alat berfungsi sesuai dengan yang diinginkan. Berdasarkan data respons sistem yang diperoleh dari pengujian dengan menggunakan metode 1 Ziegler-Nichols, maka parameter kontroler PID dapat ditentukan dengan gain Kp = 5,7; Ki = 28,5 dan Kd = 0,285.

6.2 Saran

Sebagai pengembangan selanjutnya terdapat beberapa saran-saran sebagai berikut:

1. Sistem dapat dikembangkan untuk aksis yang lain dengan metode kontroler yang sama.

   Dapat dikembangkan menggunakan kontroler selain PID.

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

DAFTAR PUSTAKA

Chmelai, Pavel. 2011. *Building And Controlling The Quadrocopter*:Online: *http://pernerscontacts.upce.cz/24_2011/Chmelar.pdf*.

Dharmawan A. & Putera, C.A.L. 2012. Purwarupa Sistem Integrasi *Quadcopter* dan *Mobile Robot*. IJEIS (2):97-108

Ferdiansyah, F. 2014. Teori Kontrol PID (Proportional–Integral–Derivative). *https://www.academia.edu/9928544/Teori_Kontrol_PID_Proportional_Integral_Derivative* (diakses pada 12 Maret 2016)

Gunterus, Frans. 1994. *Falsafah Dasar : Sistem Pengendalian Proses*. Jakarta: Elex Media Komputindo.

Kristianto, D. 2012. *Rancang Bangun Pesawat Mandiri Tanpa Awak Dengan Empat Baling-Baling Penggerak (Autonomous Quadcopter)*. Salatiga: Teknik Elektronika dan Informatika Universitas Kristen Satya Wacana.

Ogata, K. 2002. *Modern Control Engineering*. *New Jersey : Prentice-Hall, Inc.*

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

Rohiman, S. 2011. Perancangan Autonomous Payload Berbasis GPS Dan Mikrokontroler Picaxe-40X2 (studi kasus korindo 2010). *Skripsi*. Bandung: Universitas Komputer Indonesia.

Santoso, Ahmad Dzikri. 2014. *Analisis Pengaruh Kecepatan Motor Pada Quadcoper Terhadap Kestabilan Multicopter Aerial Cam*. Teknik Elektro Universitas Pancasila.

Setiawan, Iwan. Kontrol PID untuk Proses Industri. Elex Media Komputindo, Jakarta, 2008.

Yusuf, Sam dan Fatoni ,Ali. 2012. *Perancangan Dan Implementasi Kontroler PID Untuk Autonomus Moving Forward Manuever Pada Quadcopter*. Surabaya: Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS).

Welander, Peter. "*Understanding Derivative in PID Control*", Control Engineering, 2, 24-27. 2010.

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

LAMPIRAN 1

FOTO ALAT

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

LAMPIRAN 2

DATA SHEET

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

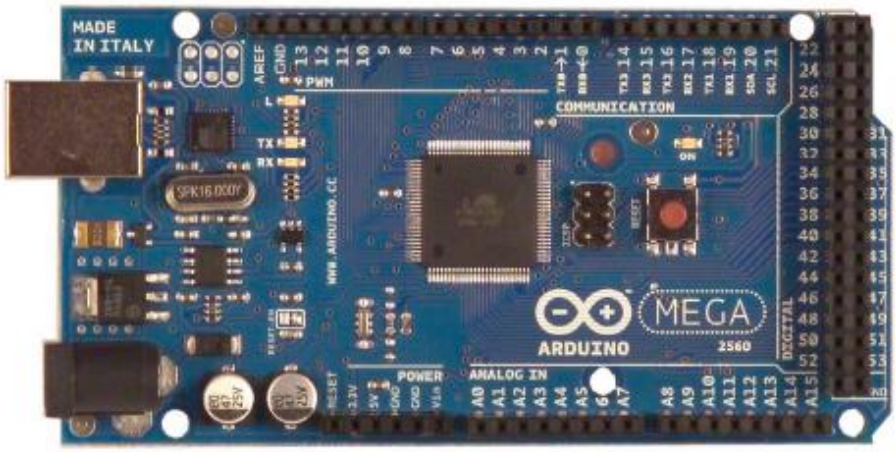
The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

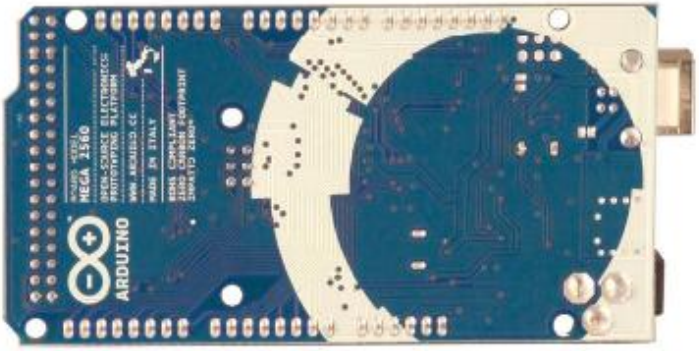
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:





Overview

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

+ **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

+ **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

+ **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

+ **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

+ **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

+ **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

+ **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.

+ **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.

+ **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

+ **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with **analogReference**().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A **SoftwareSerial library** allows for serial communication on any of the Mega2560's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the **documentation on the Wiring website** for details. For SPI communication, use the **SPI library**.

Programming

The Arduino Mega can be programmed with the Arduino software (**download**). For details, see the **reference** and **tutorials**.

The ATmega2560 on the Arduino Mega comes preburned with a **bootloader** that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (**reference**, **C header files**).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see **these instructions** for details.

The ATmega8U2 firmware source code is available **in the Arduino repository**. The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use **Atmel's FLIP software** (Windows) or the **DFU programmer** (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See **this user-contributed tutorial** for more information.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can

have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

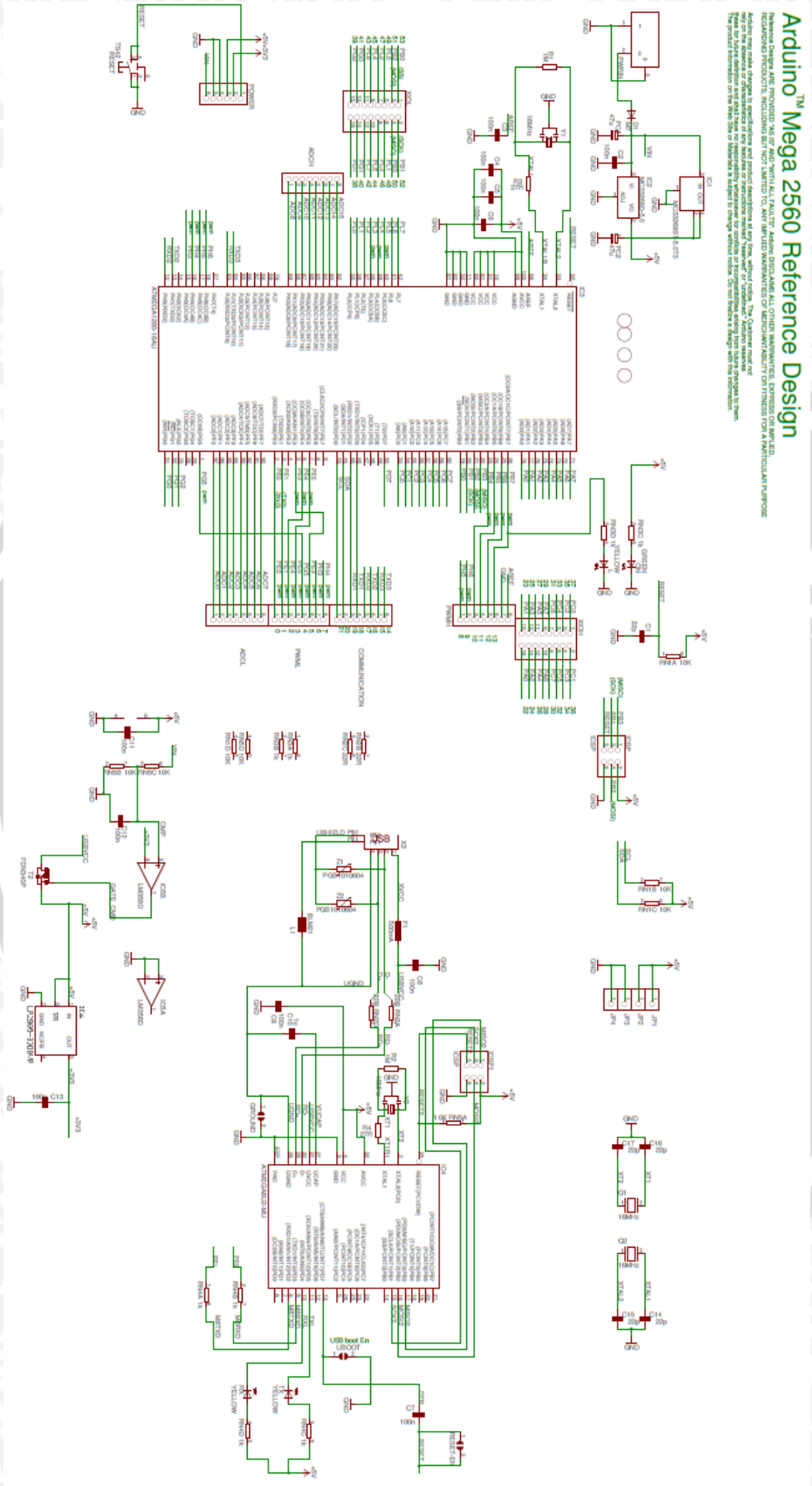
USB Overcurrent Protection

The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I²C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*


Arduino™ Mega 2560 Reference Design

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

LAMPIRAN 3

LISTING PROGRAM

```
/*******************************************************************
 * TUGAS AKHIR
 * JUDUL:
PERANCANGAN KESEIMBANGAN GERAK PITCH PADA
BICOPTER SECARA STATIS DENGAN METODE MENGUBAH-UBAH
KECEPATAN MOTOR MENGGUNAKAN KONTROLER PID
* NAMA: DANDY MUHAMMAD
 *NIM : 105060307111006
 *******************************************************************/
/*******************************************************************
 * Arduino PID Library - Version 1.1.1
 * by Brett Beauregard <br3ttb@gmail.com> brettbeauregard.com
 *
 * This Library is licensed under a GPLv3 License
 *******************************************************************/

#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

#include <PID_v1.h>

/*Constructor (...)*********************************************************
 *    The parameters specified here are those for for which we can't set up
 *    reliable defaults, so we need to have the user set them.
 *******************************************************************/
```

```
PID::PID(double* Input, double* Output, double* Setpoint,
      double Kp, double Ki, double Kd, int ControllerDirection)
{

   myOutput = Output;
   myInput = Input;
   mySetpoint = Setpoint;
         inAuto = false;

  PID::SetOutputLimits(0, 255);              //default output limit corresponds to

                                             //the arduino pwm limits

   SampleTime = 100;                         //default Controller Sample Time is 0.1 seconds

    PID::SetControllerDirection(ControllerDirection);
    PID::SetTunings(Kp, Ki, Kd);

    lastTime = millis()-SampleTime;
}


/* Compute()
**********************************************************************
 *    This, as they say, is where the magic happens.  this function should be called
 *   every time "void loop()" executes.  the function will decide for itself whether a new
 *   pid Output needs to be computed.  returns true when the output is computed,
 *   false when nothing has been done.

**********************************************************************
**********/
bool PID::Compute()
```

```
{
  if(!inAuto) return false;
  unsigned long now = millis();
  unsigned long timeChange = (now - lastTime);
  if(timeChange>=SampleTime)
  {
    /*Compute all the working error variables*/
          double input = *myInput;
    double error = *mySetpoint - input;
    ITerm+= (ki * error);
    if(ITerm > outMax) ITerm= outMax;
    else if(ITerm < outMin) ITerm= outMin;
    double dInput = (input - lastInput);

    /*Compute PID Output*/
    double output = kp * error + ITerm- kd * dInput;

          if(output > outMax) output = outMax;
    else if(output < outMin) output = outMin;
          *myOutput = output;

    /*Remember some variables for next time*/
    lastInput = input;
    lastTime = now;
          return true;
  }
  else return false;
}
```

```
/*
SetTunings(...)*************************************************************
*
 * This function allows the controller's dynamic performance to be adjusted.
 * it's called automatically from the constructor, but tunings can also
 * be adjusted on the fly during normal operation

******************************************************************************
******/
void PID::SetTunings(double Kp, double Ki, double Kd)
{
  if (Kp<0 || Ki<0 || Kd<0) return;

  dispKp = Kp; dispKi = Ki; dispKd = Kd;

  double SampleTimeInSec = ((double)SampleTime)/1000;
  kp = Kp;
  ki = Ki * SampleTimeInSec;
  kd = Kd / SampleTimeInSec;

 if(controllerDirection ==REVERSE)
  {
    kp = (0 - kp);
    ki = (0 - ki);
    kd = (0 - kd);
  }
}

/* SetSampleTime(...)
*********************************************************
 * sets the period, in Milliseconds, at which the calculation is performed

******************************************************************************
******/
```

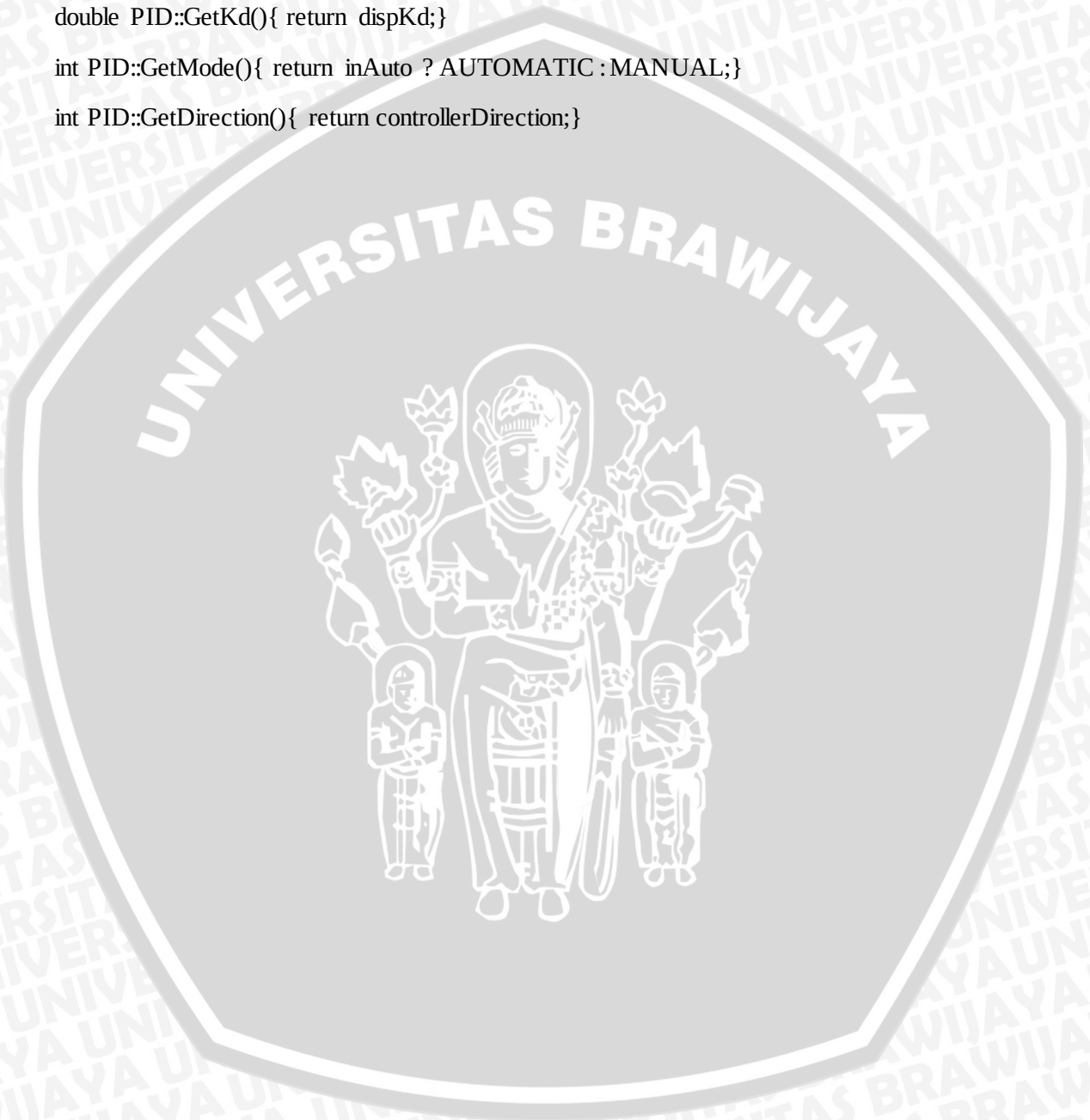
repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

```
void PID::SetSampleTime(int NewSampleTime)
{
  if (NewSampleTime > 0)
  {
    double ratio  = (double)NewSampleTime
                    / (double)SampleTime;
    ki *= ratio;
    kd /= ratio;
    SampleTime = (unsigned long)NewSampleTime;
  }
}

/* SetOutputLimits(...)****************************************************
 *    This function will be used far more often than SetInputLimits.  while
 *  the input to the controller will generally be in the 0-1023 range (which is
 *  the default already,)  the output will be a little different.  maybe they'll
 *  be doing a time window and will need 0-8000 or something.  or maybe they'll
 *  want to clamp it from 0-125.  who knows.  at any rate, that can all be done
 *  here.
**************************************************************************
**/
void PID::SetOutputLimits(double Min, double Max)
{
  if(Min >= Max) return;
  outMin = Min;
  outMax = Max;

  if(inAuto)
  {
          if(*myOutput > outMax) *myOutput = outMax;
          else if(*myOutput < outMin) *myOutput = outMin;
```

```
            if(ITerm > outMax) ITerm= outMax;
            else if(ITerm < outMin) ITerm= outMin;
    }
}

/*
SetMode(...)*********************************************************
**
 * Allows the controller Mode to be set to manual (0) or Automatic (non-zero)
 * when the transition from manual to auto occurs, the controller is
 * automatically initialized

*********************************************************************
******/
void PID::SetMode(int Mode)
{
    bool newAuto = (Mode == AUTOMATIC);
    if(newAuto == !inAuto)
    {  /*we just went from manual to auto*/
        PID::Initialize();
    }
    inAuto = newAuto;
}

/*
Initialize()*********************************************************
*
 *       does all the things that need to happen to ensure a bumpless transfer
 * from manual to automatic mode.

*********************************************************************
******/
void PID::Initialize()
```

```
{
  ITerm = *myOutput;
  lastInput = *myInput;
  if(ITerm > outMax) ITerm = outMax;
  else if(ITerm < outMin) ITerm = outMin;
}

/*
SetControllerDirection(...)*************************************************
 * The PID will either be connected to a DIRECT acting process (+Output leads
 * to +Input) or a REVERSE acting process(+Output leads to -Input.)  we need to
 * know which one, because otherwise we may increase the output when we should
 * be decreasing.  This is called from the constructor.
******************************************************************************
******/
void PID::SetControllerDirection(int  Direction)
{
  if(inAuto && Direction !=controllerDirection)
  {
          kp = (0 - kp);
    ki = (0 - ki);
    kd = (0 - kd);
  }
  controllerDirection  = Direction;
}

/* Status
Funcions*******************************************************************
 * Just because you set the Kp=-1 doesn't mean it actually happened.  these
 * functions query the internal state of the PID.  they're here for display
 * purposes.  this are the functions the PID Front-end uses for example
```

```
**********************************************************************
******/
double PID::GetKp(){ return dispKp; }
double PID::GetKi(){ return dispKi;}
double PID::GetKd(){ return dispKd;}
int PID::GetMode(){ return inAuto ? AUTOMATIC : MANUAL;}
int PID::GetDirection(){ return controllerDirection;}
```



```
#include <PID_v1.h>

// Mendefinisikan kanal adc setpoint Input dan Output
double Setpoint, Input, Output;
double Kp, Ki, Kd

// setting parameter PID

Kp = 5.7 ;
Ki = 28.5;
Kd = 0.285;
PID myPID(&Input, &Output, &Setpoint,Kp,Ki,Kd, DIRECT);

void setup()
{
 // inisialisasi
 // baca feedback kanal adc 0
 Input = analogRead(0);
 // baca setpoint kanal adc 1
 Setpoint = analogRead(1);

 //turn the PID on
 myPID.SetMode(AUTOMATIC);
}

void loop()
{
 // baca feedback
 Input = analogRead(0);
 // baca setpoint
 Setpoint = analogRead(1);
```

```
 myPID.Compute();
 // keluarkan hasilnya
 analogWrite(3,Output);
}

// deklarasi sensor //
int sensorpin = A1; //pin sensor = A1
int sensorvalue; // nilai sensor
int banyakData;
float v_sen; // tegangan sensor

void loop()
{
 {
  sensorvalue = analogRead(sensorpin);
  v_sen = ((sensorvalue/(sensorvalue+100))*5.0;


  banyakData++;

  Serial.println(rpm);
  banyakData=0;
 }
  delay(20);

//perhitungan error//
 now=millis();
 if(lTime!=0){dTime =(double)(now-lTime);}

 sensors.requestPotentiometer();
 error = setpoint-potentiometer;
```

```
  sError += Error;
   dError -= lError;

/*perhitungan*/

//------- kontrol PID -------//
//Rumus pid

  output = (Kp*Error) + (((Ki/10)*sError)*(dTime/1000))
+((Kd*dError)/(dTime/1000));
   if(PID<=0){PID=0;}
   else{PID+=50;}
   PID=(int)PID;

   lError = Error;
   lTime = now;
}
//sinyal PID dikonversi menjadi sinyal PWM

  if(PID>=127)
   PID = 127;
  if(PID<=-126)
   PID = -126;

//PWM output harus antara 1 sampai 255
  PWMOutput = PID + 127;

// comment out to speed up PID loop
//  Serial.print("PWMOutput= ");
//  Serial.println(PWMOutput,DEC);
}
```