RANCANG BANGUN *SELF BALANCING TWINROTOR* MENGGUNAKAN KONTROLER PID TUNING *PARTICLE SWARM OPTIMIZATION*

Nofrialdi

S1 Teknik Elektro, Fakultas Teknik, Universitas Negeri Surabaya, Ketintang 60231, Indonesia
e-mail : nofrialdi.18083@mhs.unesa.ac.id

Muhamad Syariffuddien Zuhrie, Bambang Suprianto, I Gusti Putu Asto

S1 Teknik Elektro, Fakultas Teknik Universitas Negeri Surabaya, Ketintang 60231, Indonesia
e-mail : zuhrie.syarif@gmail.com, bambangsuprianto@unesa.ac.id, asto@unesa.ac.id

Abstrak

Keseimbangan dari sistem yang berputar merupakan fitur yang sangat penting pada dunia sains dan industri, terutama pada sistem pesawat terbang. Salah satu sistem tersebut yaitu *twinrotor motor system* (TRMS). Tujuan penelitian ini untuk mempelajari perancangan, pembuatan dan kontrol dari TRMS. Sudut *pitch* dari sistem dikontrol menggunakan kontroler PID dengan parameter optimasi tuning menggunakan metode *particle swarm optimization* (PSO). Penelitian ini menggunakan *software* Arduino dan Matlab R2014 A yang digunakan untuk mencari nilai PID dan simulasi hasil dari respon sistem. *Hardware* dari *plant* menggunakan Arduino Mega 2560 Sebagai mikrokontroler, Mpu-6050 sebagai sensor keseimbangan, motor BLDC sebagai aktuator dan *power supply* digunakan untuk menyediakan tegangan listrik ke seluruh rangkaian. Sistem PID kontroler tuning PSO memiliki kriteria *rise time* yang sangat cepat dan memiliki *overshoot* yang kecil. Hasil nilai *overshoot* menggunakan kontroler PID PSO sebesar 33.25% dan *rise time* sebesar 0.92s dengan nilai *error steady state* sebesar 1.26. sehingga respon sistem *twinrotor* lebih baik menggunakan Kontroler PID daripada tidak menggunakan Kontroler PID.
Kata Kunci : TRMS, PID, PSO, Self Balancing

Abstract

Balancing of system rotating is one of the most important feature in engineering industry and science, especially in plane systems. One type of the systems is twinrotor motor system (TRMS). The purpose of this research is to studied design, manufacturing and kontrol of TRMS. The pitch angle of the TRMS is controlled used a PID controller with particle swarm optimization (PSO) tuning method. This research uses Arduino software and Matlab R2014 A used to find the PID value and the simulation results of the system response. Hardware uses Arduino Mega 2560 as a microcontroller, Mpu-6050 as a balance sensor, BLDC motor as an actuator and power supply is used to provide electrical voltage to the entire circuit. In this system has a fast rise time with low overshoot. The result overshoot value is 33.25% and rise time is 0.92s with a steady state error is 1.26. therefor the twinrotor system is better to use PID controller instead of using PID controller.
Keywords : TRMS, PID, PSO Self Balancing

PENDAHULUAN

Penelitian dan pengembangan yang terkait dengan *unmanned aerial vehicle* (UAV) sangat pesat dalam beberapa tahun terakhir, yang termotivasi oleh kemajuan teknologi terbaru dibidang miniaturisasi aktuator dan rangkaian elektronika. Design yang efisien, berbiaya rendah dengan kemungkinan dapat dikendalikan secara otomatis maupun manual. UAV menghadirkan terobosan baru untuk diaplikasikan warga sipil maupun militer. Misi utama pada UAV yaitu untuk menyelesaikan misi yang beresiko atau sulit diakses bagi manusia seperti penyelamatan korban bencara alam, untuk misi penyelamatan warga sipil pada perang, maupun untuk sekedar hobi. (El Gmili dkk., 2018).

Twinrotor UAV merupakan jenis helikopter yang didorong oleh dua buah *propeller*/baling baling. Baling baling berputar ke arah yang berlawanan, sehingga tidak diperlukannya rotor ekor untuk melawan momentum sudut baling baling. Dengan mengubah kecepatan motor, posisinya juga akan berubah. Keseimbangan *twinrotor* merupakan hal utama yang harus diperhatikan. (Agarwal dkk., 2013).

Kesimbangan dari sistem yang berputar merupakan fitur yang sangat penting pada dunia sains dan industri. Salah satu jenis sistem tersebut yaitu *twinrotor motor system* (TRMS). Sistem ini menggunakan dua buah motor yang ditempatkan di setiap sisi lengan. Motor dapat merubah sudut posisi dengan variasi kecepatannya (Hosseializade dkk., 2016).

Kontroler PID digunakan untuk mengatur prilaku TRMS karena kontroler PID memiliki prosedur sederhana yang dimengerti oleh *plant*. Kontroler PID memiliki tiga parameter kontrol yang memiliki efek kontrol. Kontroler proportional memberikan perubahan pada *input* yang berbanding lurus dengan kesalahan kontrol. Kontroler

integral memberi perubahan input *proporsional* dengan kesalahan terintegrasi dan tujuan utamanya adalah untuk menghilangkan nilai *error steady state*. Kontroler *derivative* digunakan dalam beberapa kasus untuk mempercepat respon atau menstabilkan sistem (Kundu dan Romio, 2018).

Penentuan parameter kontrol PID pada penerapannya masih dilakukan secara *trial and error* atau menggunakan metode Ziegler-Nichols berdasarkan kurva reaksi hasil respon model orde satu atau menggunakan metode *sustain oscillation*, akan tetapi parameter yang diperoleh belum bisa memberikan solusi yang tepat bagi parameter kontrol PID karena masih terdapat banyak noise yang dihasilkan. (Alrijadjis dan Katjuk, 2013).

Metode-metode ini menggunakan asumsi proses yang dikendalikan memiliki dinamika minimum, linear, tidak ada *noise* dan sebagainya, akan tetapi pada kenyataanya banyak proses pengendalian yang sangat kompleks. Metode tuning lain yang selama ini digunakan selain Ziegler-Nichols yaitu Stepestdescent, Cohen-Coon dan Newtons Method's yang merupakan metode *Gradient-based* bersifat meminimumkan *costfunction* (Nugroho dkk., 2014).

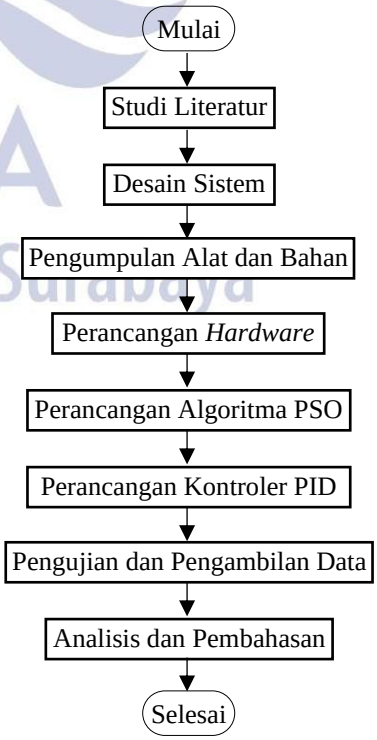
Saat ini moetode *particle swarm optimization* (PSO) digunakan secara luas karena kesederhanaannya, biaya komputasi yang rendah, dan hasilnya yang dapat dihandalkan. PSO digunakan sebagai tuning parameter kontrol pid untuk meningkatkan respon dari sistem dengan mempercepat *rise time* dan meminimalkan *overshoot* sehingga respon sistem dapat berjalan sesuai dengan kriteria yang diinginkan. PSO merupakan salah satu teknik untuk keperluan optimasi yang digunakan baru baru ini (J. Marie dkk., 2015).

Particle Swarm Optimization (PSO) memiliki 2 konsep komponen utama yang berkaitan dengan kecerdasan buatan. Secara umum, PSO ditiru berdasarkan tingkah laku dari kelompok burung dan ikan dalam berkawanan. Setiap individu memliki kecepatan dan posisi masing masing yang akan berubah setiap waktu. Prilaku tersebut ditiru menggunakan algoritma pemograman pada komputer sehingga PSO dapat diimplementasikan ke dalam penyelesaian masalah komputasi. (Kennedy dan Russell, 1995).

Pada penelitian ini akan dilakukan pengaturan sudut terhadap keseimbangan *twin rotor motor system* (TRMS) menggunakan kontroler PID dengan tuning *particle swarm optimization*. Kontroler PID digunakan untuk mengatur kecepatan putar motor BLDC. Kecepatan diatur dengan cara merubah arus yang diberikan kepada motor BLDC. Dengan mengatur kecepatan putar motor BLDC maka sudut *pitch* dari lengan dapat diatur sesuai dengan *setpoint* yang diberikan. Penggunaan metode PID PSO digunakan untuk meningkatkan kinerja yang lebih efisien dalam meningkatkan karakteristik *step respons* seperti mengurangi *error steady state*, *rise time*, *settling time* dan *overshoot* dalam kontrol TRMS. Khususnya PSO akan meningkatkan *rise time* dan mengurangi *overshoot* pada sistem dengan sangat cepat.

METODE

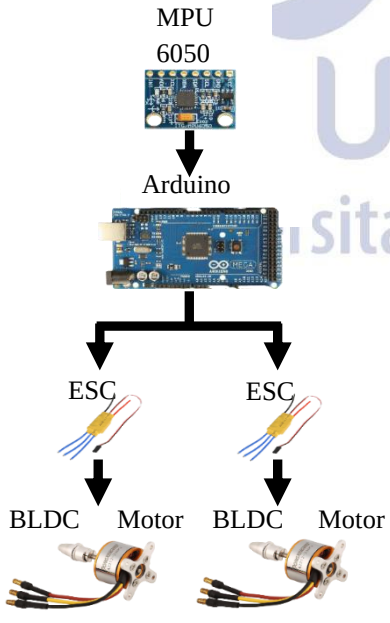
Pendekatan yang dilakukan pada penelitian ini menggunakan pendekatan kuantitatif. Menurut (Machali, 2016) Pada pemaparannya penelitian kuantitatif lebih banyak menampilkan dan memaknai angka-angka disertai dengan tabel, grafik, gambar atau tampilan lainnya. Rancangan dari penelitian "Rancang Bangun *Self Balancing Twinrotor* Menggunakan Kontroler PID Tuning *Particle Swarm Optimization*" terdiri dari berbagai tahapan yang dimulai dari studi literatur yaitu mempelajari tentang *self balancing twinrotor* dan kontroler PID dengan tuning *particle swarm optimization* secara menyeluruh, desain sistem yaitu merancang desain *self balancing* menggunakan *software* desain grafis, pengumpulan alat dan bahan yaitu mengumpulkan alat dan bahan yang diperlukan untuk membuat *self balancing twinrotor*, perancangan *hardware* yaitu merancang *hardware twinrotor* sesuai dengan yang diharapkan, perancangan algoritma PSO yaitu merancang parameter parameter PID menggunakan algoritma dengan cara menentukan parameter PSO dengan menggunakan *software* Matlab, perancangan kontroler PID yaitu memasukkan pemograman PID kedalam mikrokontroler arduino Mega 2560, pengujian dan pengambilan data hingga analisis dan pembahasan yaitu menguji *self balancing* dengan cara memasukkan *setpoint* dan melihat respon sistem. Rancangan penelitian dapat dilihat pada gambar 1.



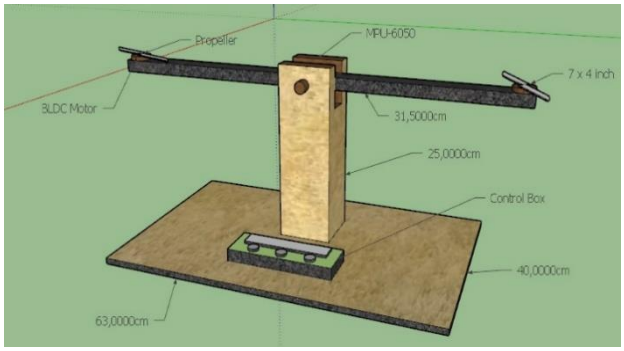
Gambar 1 *Flowchart* Tahapan Penelitian

Perancangan dan Pembuatan Alat

Dalam perancangan dan pembuatan alat terdapat tahapan tahapan yang dibagi menjadi dua tahap, yaitu tahap perancangan *hardware* dan tahap perancangan *software.* Tahap perancangan *hardware* meliputi spesifikasi alat dan wiring pengkabelan. Sedangkan tahap perancangan *software* meliputi pemodelan sistem, perancangan pemograman alat pada *software* Arduino IDE dan percancangan kontroler PID dengan tuning PSO pada *software* Matlab. Diagram *input output* dapat dilihat pada gambar 2. Pada gambar 2 diagram *input output* terdiri dari MPU 6050, Arduino Mega 2560, *electronic speed controller* (ESC) 20 ampere dan BLDC motor A2212 930 KV. MPU 6050 digunakan sebagai sensor kemiringan yang akan menerima pembacaan posisi sudut dari kemiringan lengan *twinrotor*, hasil pembacaan sensor MPU 6050 akan dikirim ke mikrokontroler Arduino Mega 2560. Arduino Mega 2560 sebagai piranti akuisisi data antara *input* dan *output.* ESC 20 A digunakan sebagai driver dari BLDC motor. BLDC motor digunakan sebagai penggerak lengan *twinrotor* sesuai *setpoint* yang diinginkan. Pada penelitian ini bahan utama pembuatan *twinrotor* menggunakan bahan dasar kayu pada bagian sasis dan menggunakan bahan dasar besi alumunium pada bagian lengan. Lengan pada *twinrotor* didesain ringan yang bertujuan untuk menambah gaya angkat dari motor sehingga respon *twinrotor* dapat menyesuaikan dengan *setpoint* yang diberikan dengan cepat. Desain *self balancing twinrotor* dan daftar bahan komponen penyusun dan spesifikasi alat yang dapat dilihat pada gambar 3, tabel 1 dan tabel 2.



Gambar 2 Diagram Input Output



Gambar 3 Desain *Self Balancing* Twinrotor

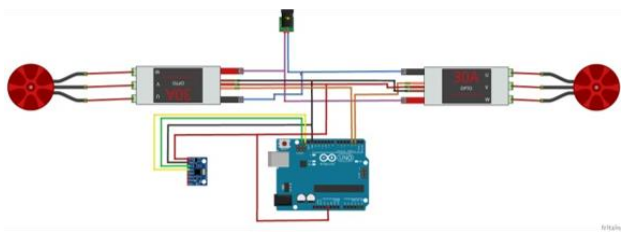
Tabel 1 Komponen Penyusun

No.	Daftar Bahan	Jumlah
1	*Propeller*	2
2	Arduino mega 2560	1
3	ESC	2
4	BLDC motor A2212	2
5	MPU 6050	1
6	*Switch* / sakelar	1
7	LCD 16x2	1
8	*Power supply*	1

Tabel 2 Spesifikasi alat

No.	Spesifikasi	Nilai	Satuan
1	Diameter *propeller*	7x4	Inch
2	Massa sasis	2	Kg
3	Massa lengan	0,5	Kg
4	Jarak tengah lengan ke ujung lengan	31,5	Cm
5	Lebar sasis	40	Cm
6	Tinggi sasis	25	Cm
7	Koefisien redaman	1	-
8	Memen inersia motor	0,33	$Kg.m^2$

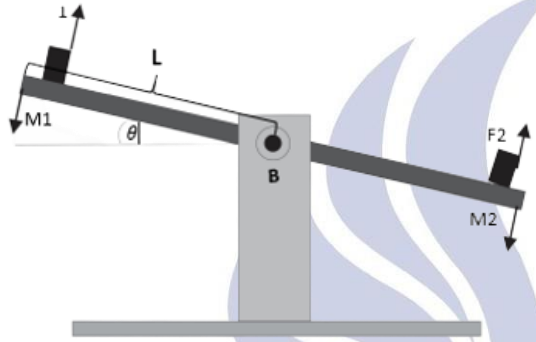
Arduino Mega 2560 yang digunakan sebagai mikrokontroler dihubungkan dengan semua komponen yang digunakan yaitu komponen sensor MPU-6050 menggunakan komunikasi I2C PIN SDA dan SCL, *electronic speed controller* (ESC) menggunakan PIN PWM, dan motor BLDC lansung dihubungkan dengan ESC dan *power supply* 12 volt. *Wiring* Arduino dengan modul yang digunakan dapat dilihat pada gambar 4 dan tabel 3.



Gambar 4 *Wiring* Arduino dengan Modul

Tabel 3 *Wiring* Arduino dengan Modul

No.	Pin Arduino	Pin Modul	Modul
1	SDA	SDA	MPU 6050
2	SCL	SCL	
3	5V	VCC	
4	GND	GND	
5	PWM 3	Data	ESC (1)
6	5V	VCC	
7	GND	GND	
8	PWM 2	Data	ESC (2)
9	5V	VCC	
10	GND	GND	



Gambar 5 Skematik Gaya *Self Balancing Twinrotor* (Sumber : Hossealizade,2016)

Pada gambar 5, skematik gaya yang digunakan pada *self balancing twinrotor* menggunakan *Newton method*. Dalam metode *Newton*, terdapat persamaan gaya angkat, grafitasi dan torsi motor yang diasumsikan. *plant* diasumsikan sebagai benda tegar dengan massa/berat di kedua sisi adalah sama. penulis berasumsi massa maksimal beban terdapat pada masing-masing sisi beban. M2 dan M1 adalah jumlah massa motor dan *propeller* di ujung lengan. L, B dan j adalah, setengah panjang balok, koefisien redaman dan momen inersia (Hosseializade dkk., 2016). Dari gambar 5, didapatkan persamaan 1 sebagai berikut.

$$j\ddot{\theta} = F_1L - F_2L - M_1Lgcos\theta + M_2Lgcos\theta - B\dot{\theta} \quad (1)$$

Keterangan:

j = Momen inersia turunan kecepatan
F_1 = Gaya angkat (*thrust*) pada motor 1
F_2 = Gaya angkat (*thrust*) pada motor 2
L = Jarak titik pusat lengan ke ujung lengan
θ = sudut yang terbentuk dari kemiringan lengan
g = gaya gravitasi bumi
B = koefisien redaman
M_1 = Massa motor dan *propeller* lengan 1
M_2 = Massa motor dan *propeller* lengan 2

Untuk persamaan sistem linieritas, penulis menggunakan 2 buah variabel

$$\theta = x_1 \quad (2)$$

$$\dot{\theta} = w = x_2 \quad (3)$$

Sekarang persamaan linearnya menjadi

$$\ddot{x_2} = \ddot{\theta} = \frac{1}{j}\left(F_1L - F_2L - M_1Lgcos\theta + M_2Lgcos\theta - B\dot{\theta}\right) = f_2(t) \quad (4)$$

Dengan menggunakan metode jacobian, dapat menjadikan persamaan (4) menjadi $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$$\begin{bmatrix} \frac{\partial f_1}{\partial f} \\ \frac{\partial f_2}{\partial f} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{L}{j} \end{bmatrix} \quad (5)$$

Sekarang persamaan linear state space menjadi

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ \frac{Lgsin(0)(M_1-M_2)}{j} & \frac{-B}{j} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{L}{j} \end{bmatrix} F \quad (6)$$

$$y = \theta = x_1 = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (7)$$

Transfer function sistem G(s) didapatkan

$$G(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s & -1 \\ \frac{-Lgsin(0)(M_1-M_2)}{j} & s + \frac{Bs}{j} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \frac{L}{j} \end{bmatrix} \quad (8)$$

$$G(s) = \frac{\frac{L}{j}}{s^2 + \frac{Bs}{j}} \quad (8)$$

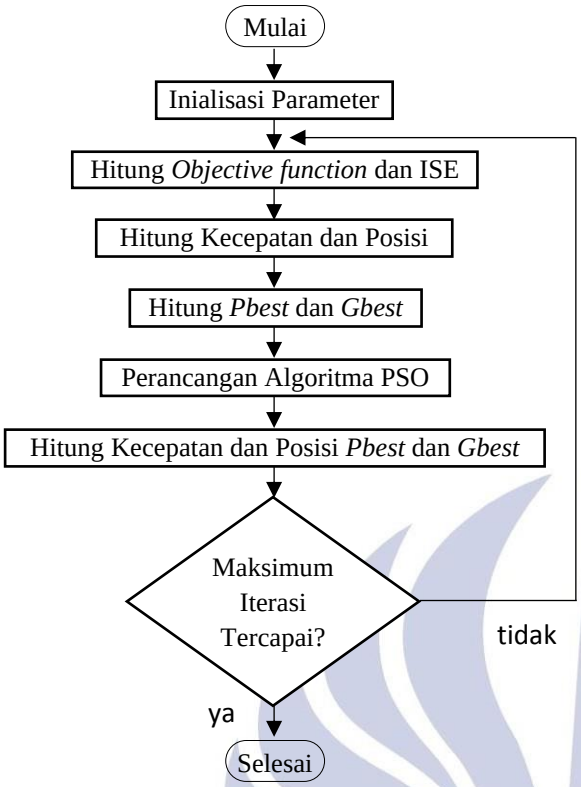
Dengan memasukkan nilai $B = 1, j = 0{,}33\ Kg.m^2$, $L = 0{,}315\ m$ berdasarkan spesifikasi *hardware* didapatkan

$$G(s) = \frac{0{,}954}{s^2 + 3{,}0303s} \quad (9)$$

Perancangan kontroler PID dengan tuning *particle swarm optimization* (PSO) dilakukan dengan menggunakan *software* Matlab. Algoritma pemograman PSO akan mengoptimasi nilai *error* dari *objective function*. terdapat beberapa parameter yang digunakan pada algoritma PSO. berikut ini beberapa parameter pada ruang dimensi PSO :

Tabel 4 Parameter pada ruang dimensi PSO

No	Parameter pada ruang dimensi PSO
1	Jumlah populasi
2	Maksimum iterasi
3	Bobot inersia
4	Kecepatan partikel
5	Konstanta akselerasi partikel
6	Konstanta akselerasi global
7	Solusi yang dicari
8	Posisi awal partikel



Gambar 6 *Flowchart* Pemograman Algoritma PSO

Partikel terbaik (*pbest*) diantara semua partikel dalam sebuah (*swarm*) kawanan ditulis dengan $gbest_d$. Kecepatan setiap patikel ke-i ditulis dengan $v_i = (v_{i,1}, v_{i,2}, \dots, v_{i,d})$. Modifikasi dari Posisi dan kecepatan setiap partikel dapat dihitung menggunakan kecepatan saat ini dan jarak dari $pbest_{i,d}$ ke $gbest_d$ seperti ditunjukkan persamaan 10 dan 11.

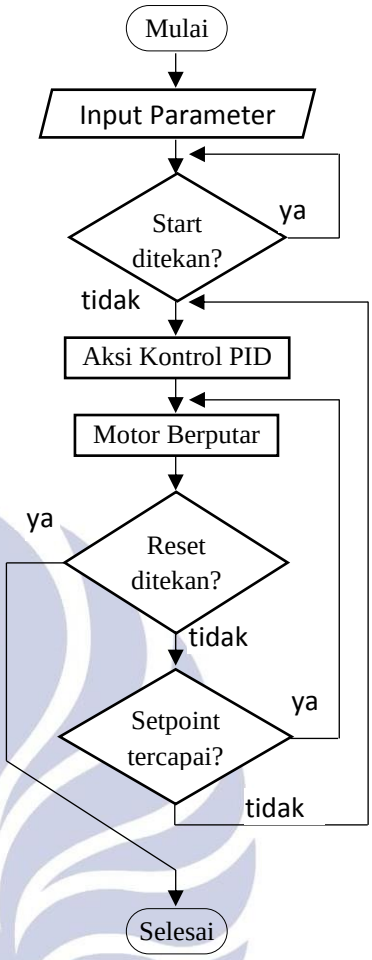
$$v_{i,m}^{(t+1)} = wv_{i,m}^{(t)} + c_1 * R * \left(pbest_{i,m} - x_{i,m}^{(t)}\right) + c_2 * R * \left(gbest_m - x_{i,m}^{(t)}\right) \quad (10)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (11)$$

Keterangan :

n	= Jumlah partikel dalam sebuah kawanan
d	= Dimensi
$v_{i,m}^{(t)}$	= Kecepatan saat ini dari partikel ke-i iterasi ke-t
w	= Faktor bobot inersia
c_1, c_2	= Konstanta akselerasi
R	= Bilangan random
$x_{i,m}^{(t)}$	= posisi saat ini dari partikel ke-i iterasi ke-t
$pbest_i$	= Posisi terbaik sebelumnya dari partikel ke-i
$gbest_m$	= partikel terbaik dalam sebuah kawanan

Pemograman alat pada software Arduino IDE yang ditunjukkan pada gambar 7.



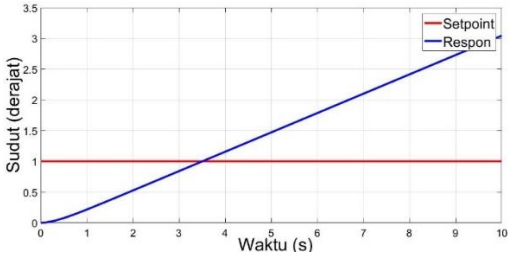
Gambar 7 *Flowchart* Pemograman Alat

HASIL DAN PEMBAHASAN

Pada pengujian ini akan dilakukan simulasi *plant* tanpa diberikan kontroler PID, tuning kontroler PID, simulasi *plant* dengan kontroler PID, pengujian *real plant* tanpa beban dan pengujian *real plant* dengan beban. Pengujian simulasi *plant* tanpa kontroler dilakukan dengan menggunakan software matlab untuk melihat respon sistem yang diberikan input sinyal step. Pada pengujian ini menggunakan model matematika dari persamaan 9, yaitu

$$G(s) = \frac{0{,}954}{s^2 + 3{,}0303s}$$

Model matematika dari persamaan 9 dimasukan pada diagram blok simulink pada matlab. Hasil respon sistem dapat dilihat pada gambar 8. Respon simulasi sistem *twinrotor* tidak terarah semakin tinggi nilainya mencapai tak hingga terhadap waktu. Hal ini tidak memungkinkan jika diterapkan pada *real plant* karena akan terjadi *osilasi* yang mengakibatkan kerusakan pada komponen elektronika yang digunakan. Olehkarana itu perlu dirancang sebuah kontroler yang dapat mengatur kecepatan putar motor bldc sehingga twinrotor dapat mengatur posisi sudut sesuai dengan setpoint yang diberikan. Kontroler PID digunakan pada sistem untuk mengatur putaran motor BLDC.



Gambar 8 Respon Simulasi *Plant* Tanpa Kontroler PID

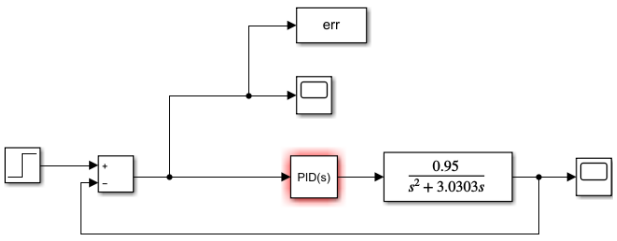
Tuning kontroler PID dilakukan untuk mencari nilai Kp, Ki dan Kd. Tuning kontroler PID dengan metode *particle swarm optimization* (PSO). Dalam penentuan parameter tuning kontrol PID, PSO harus memiliki fungsi objektif (*objective function*) yang digunakan untuk mengevaluasi nilai error melalui proses iterasi partikel dalam ruang pencarian. Metode PSO yang digunakan sebagai optimasi tuning memiliki beberapa partikel dalam populasi, dimana setiap partikel berisi tiga anggota P, I dan D, itu artinya setiap partikel memiliki tiga dimensi ruang pencarian dan partikel bergerak dalam tiga ruang dimensi tersebut. Dalam pencarian solusi, partikel memiliki posisi baru yang digunakan sebagai dasar update posisi partikel yang dihubungkan dengan posisi partikel terbaik sebelumnya. Persamaan umum dari kontroler PID dapat dilihat pada persamaan 12.

$$U(t) = K_p e(t) + \frac{1}{\tau_i}\int e(t)dt + \tau_d \frac{de(t)}{dt} \quad (12)$$

Nilai error merupakan selisih nilai antara *setpoint* dan nilai aktual keluaran dari proses. Besar sinyal kontrol yang dikeluarkan berdasarkan algoritma kontrol PID yaitu mengalikan error dengan nilai Kp, Ki dan Kd. Kriteria performansi yang digunakan untuk memperkecil nilai error yaitu ISE (integral square error). ISE merupakan index performansi yang mengintegralkan nilai error kuadrat dari waktu sekarang ke waktu sebelumnya. Nilai error yang relatif kecil akan menimbulkan perubahan besar pada nilai ISE, dengan meminimalkan nilai ISE, maka akan meminimalkan nilai error yang besar dengan cepat. Persamaan ISE dapat dilihat pada persamaan 13.

$$ISE = \int_0^{\infty} \{e(t)\}^2 dt \quad (13)$$

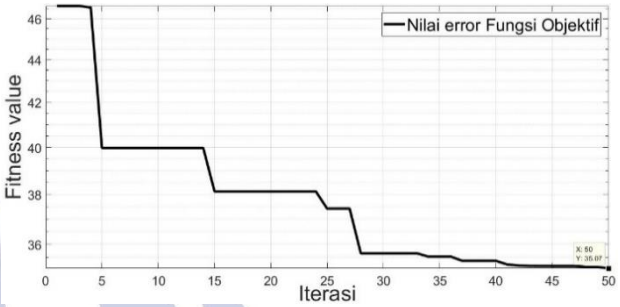
Nilai error diambil dari respon sistem digram blok yang ditunjukkan pada gambar 9, parameter algoritma PSO dapat dilihat pada tabel 4 dan grafik dari *fitness value* nilai error dari setiap iterasi dapat dilihat pada gambar 10.



Gambar 9 *Objective Function*

Tabel 5 Parameter PSO

Parameter PSO	Nilai
Jumlah partikel	100
Max iterasi	50
Variabel	3
Konstanta akselerasi partikel	2
Konstanta akselerasi global	2
Bobot inersia	0.2

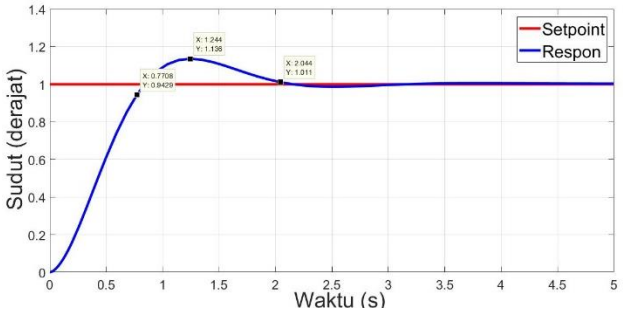


Gambar 10 *Fitness Value Objective Function*

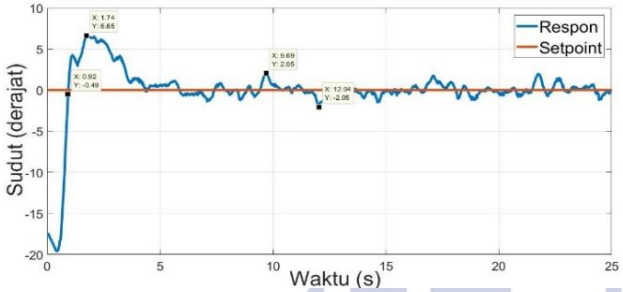
Tabel 6 Parameter PID

	Metode Tuning PSO	
Parameter PID	Kp	3.514
	Ki	0.268
	Kd	1.472
Indeks Performansi	ISE	35.07

Gambar 9 merupakan diagram blok dari fungsi objektif yang akan dioptimisasi oleh algoritma PSO. Digram blok dari fungsi objektif dilakukan dengan simulai menggunakan *software Matlab*. Nilai error yang dihasilkan dari perbedaan setpoint dan nilai terukur akan dimasukkan kedalam algoritma PSO. Algoritma PSO akan memperkecil nilai error berdasarkan kriteria ISE. PSO meminimalisasi nilai *error* dari perbedaan *setpoint* dan aktual keluaran diagram blok *objective function* dari setiap iterasi. Nilai *error* akan terus berukurang setiap iterasi yang disebabkan oleh aksi kontroler PID yang mempercepat *rise time* dan mengurasi *overshoot* dari sistem. Berdasarkan gambar 10 dapat dilihat bahwa algoritma PSO meminimalisasi nilai *error integral square error* (ISE) sampai dengan 35.07 pada iterasi ke 50. *Fitness value* diminimalisasi pada setiap iterasi secara bertahap. Nilai grafik yang ditampilkan merupakan grafik dari *swarn global best* pada setiap iterasi. Nilai parameter PID yang dihasilkan dapat dilihat pada tabel 5. Setelah mendapatkan nilai parameter PID yaitu $Kp = 3.514$, $Ki = 0.268$ dan Kd 1.472, nilai parameter PID tersebut dimasukkan kedalam diagram blok program *simulink* seperti yang ditunjukkan pada gambar 9. Hasil simulasi *plant* dapat dilihat pada gambar 11, gambar 12 dan gambar 13.



Gambar 11 Respon Simulasi dengan Kontroler PID



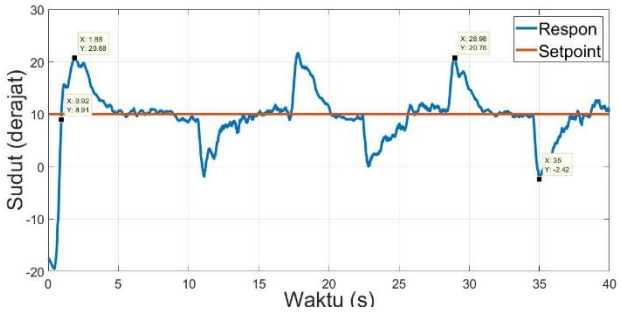
Gambar 12 Respon *Real Plant* dengan *Setpoint* 0 derajat

Gambar 13 Respon *Real Plant* dengan *Setpoint* 15 derajat

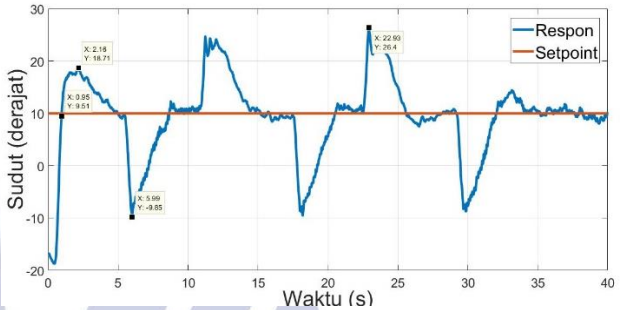
Berdasarkan gambar 11 respon sistem menggunakan kontroller PID tuning PSO, dapat dilihat bahwa respon mengalami *overshoot* sebesar 23% dengan *maximum overshoot* (M_p) sebesar 1.23, rise time selama 0,835 s, *settling time* diwaktu 2.93s dan *steady state error* = 0. kontroler PID mampu menghasilkan respon yang sangat cepat, namun diikuti dengan *overshoot* yang besar sesuai dengan kriteria kontroler PID yaitu menghasilkan *risetime* yang cepat dan menghilangkan nilai *error steady state*..

Hasil respon *real plant* tanpa beban dapat dilihat pada gambar 12 dan gambar 13. Dari percobaan dua kondisi pada gambar 12 dan gambar 13. Dapat dilihat bahwa respon *real plant twinrotor* tidak jauh beda dengan simulasi. Respon sistem memiliki *overshoot* sebesar 33.25%, namun dengan *rise time* yang sangat cepat yaitu 0.92 detik. *Twinrotor* sangat sulit untuk berada pada titik *steady state* karena sensor kemiringan yang digunakan masih terdapat banyak *error* sehingga tidak dapat mempertahankan nilai sudut dikemiringan yang sama, nilai sensor selalu berubah ubah.

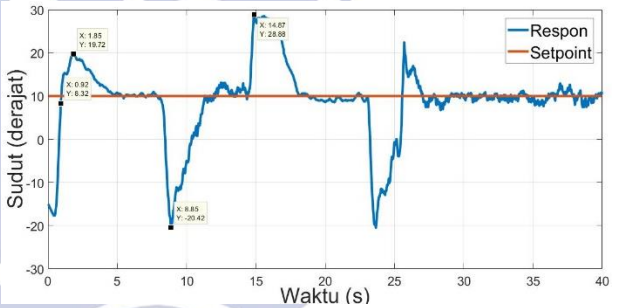
Hasil respon *real plant* dengan memberikan beban yang berbeda dapat dilihat pada gambar 14, gambar 15 dan gambar 16.



Gambar 14 Respon *Real Plant* dengan Beban 75 gr



Gambar 15 Respon *Real Plant* dengan Beban 125 gr



Gambar 16 Respon *Real* Plant dengan Beban 175 gr

Tabel 7 Hasil Pengujian Respon *Real Plant*

Pengujian Respon	M_p	t_p	t_r	e_{ss}	t_s
Tanpa gangguan	17.46	1.71s	0.92s	1.26	4.69s
Beban 75 gr	20.76	1.88s	0.92s	2.21	4.91s
Beban 125 gr	26.4	2.16s	0.95s	4.52	5.03s
Beban 175 gr	28.8	1.85s	0.92s	3.76	5.23s

Berdasarkan percobaan respon *real plant* dengan memberikan beban yang berbeda, dapat diketahui bahwa *twinrotor* melakukan *self balancing* dengan baik. Hasil pengujian respon *real plant* dapat dilihat pada tabel 6.

Pada pengujian *real plant twinrotor* dilakukan sebanyak 5 kali yaitu berupa pengujian respon *real plant twinrotor* tanpa diberikan gangguan dengan *setpoint* 0 derajat dan 15 derajat serta diberikan 3 gangguan dengan ukuran berat yang berbeda yaitu beban 75 g, 125 g, 175 g. Gangguan yang terdapat pada *twinrotor* yaitu memberikan

beban dengan berat yang berbeda pada saat *real plant* sudah dalam keadaan *steady state*. Pengujian ini dilakukan dengan menggunakan kontroler PID tuning PSO dan nilai $Kp = 3.514$, $Ki = 0.268$ dan Kd 1.472.

Hasil pengujian *real plant twinrotor* ditunjukkan pada tabel 6. Diketahui bahwa nilai *eror steady state* terkecil terjadi pada saat *twinrotor* tidak diberikan gangguan yaitu *error steady state* sebesar 1.26. Nilai *error steady state* paling besar terjadi pada saat *twinrotor* diberikan gangguan 125 gram yaitu *error steady state* sebesar 4.52. nilai *maksimum peak* paling kecil terjadi pada saat *twinrotor* tidak diberikan gangguan yaitu *maksimum peak* sebesar 17.46. Kemudian nilai *maksimum peak* paling besar terjadi pada saat *twinrotor* diberikan gangguan 175 gram yaitu *maksimum peak* sebesar 28.8. nilai *peak time* paling kecil terjadi pada saat *twinrotor* tidak diberikan gangguan yaitu *peak time* sebesar 1.71 s. Kemudian nilai *peak time* paling besar terjadi pada saat *twinrotor* diberikan gangguan 125 gram yaitu *peak time* sebesar 2.16 s. nilai *rise time* semua respon sama yaitu 0.92 s kecuali pada gangguan 125 gram, yaitu 0.95 s. nilai *settling time* paling kecil terjadi pada saat *twinrotor* tidak diberikan gangguan yaitu *settling time* sebesar 4.69 s. Kemudian nilai *settling time* paling besar terjadi pada saat *twinrotor* diberikan gangguan 175 gram yaitu *settling time* sebesar 5.23 s.

PENUTUP

Simpulan

Berdasarkan data dari hasil pengujian, *twinrotor* dapat melakukan *self balancing* dengan nilai *error setady state* dan *overshoot* yang relatif kecil, yaitu sebesar 1.26 dan 33.25%. hal ini dikarenakan pengaruh kontroler PID dengan metode tuning *particle swarm optimization* (PSO) yang menghasilkan sistem dengan respon yang sangat baik dibandingkan dengan beberapa metode sebelumnya. *Error* yang terjadi juga disebabkan oleh beberapa faktor, yaitu buruknya kualitas komponen sensor IMU yang digunakan, sehingga nilai pembacaan sensor tidak bisa stabil pada keadaan yang sama dan buruknya kualitas mikrokontroler sehingga nilai pembacaan tidak akurat.

Respon dari sistem menunjukkan karakteristik dari kontroler PID yang memliki *rise time* yang cepat namun diikuti dengan *overshoot*.

Berdasarkan tabel 6, nilai respon dari sistem semakin buruk karna bertambahnya beban gangguan semakin besar beban gangguan maka semakin besar juga *error* yang terjadi. Hal ini dikarenakan proses tuning nilai PID dilakukan pada saat sistem *twinrotor* tanpa beban gangguan.

Saran

Menggunakan jenis sensor IMU yang lebih baik agar hasil data percobaan lebih baik. Menggunakan metode kontrol *adaptive PID controller* agar respon *twinrotor* menghasilkan nilai yang baik dalam keadaan dengan beban maupun tanpa beban.

DAFTAR PUSTAKA

Agarwal, Shlok, Apoorva Mohan dan Kamlesh Kumar. 2013. *Design And Fabrication Of Twinrotor UAV*. India : Department of Mechatronics Manipal University.

Alrijadjis dan katjuk Astrowulan. 2013. Optimasi Kontroler PID Berbasis Particle Swarm Optimization (PSO) untuk Sistem dengan Waktu Tunda. Surabaya : Jurusan Teknik Elektro ITS.

El Gmili, Nada, Mostafa Mjahed dan Hassan Ayad. 2018. *Platform design and Experimental Regulation of Twinrotor UAV*. Morroco : Department of Applied Physics Cadi Ayyad University.

Hosseializade, T., S.M.J Hosseini dan H. Khaloozadeh. 2016. Design *and Implementation Classical, State Feedback and Fuzzy Controllers on Twin Rotor System.* Iran : Department of Control Engineering K.N. Toosi University of Technology.

J. Marie, Mehdi, Ghaida A. AL-Suhil dan Wisam A. Latif. 2015. *PSO-based Optimal PID Controler for Twin Rotor MIMO System*. Iraq : Department of Electrical & Computer Engineering University of Basrah.

Kennedy, James dan Russell Eberhart. 1995. *Particle Swarm Optimization.* USA : Purdue School of Engineering and Technology Indianapolis.

Kundu, Suvodip dan Romio Atha. 2018. *Pid Tuning By Particle* Swarm *Optimization Techniq and Comparison With Classical Methods.* India : Electrical Engineering Techno India College of Technology.

Machali, Imam. 2016. Metode Penelitian Kuantitatif Panduan Praktis Merencanakan dan Analisis Kuantitatif. Yogyakarta : UIN Sunan Kalijaga.

Nugroho, Anung, Bambang Dwi Argo dan Yusuf Hendrawan. 2014. *Pemodelan Dan Optimasi Sistem Kontrol Pada Multiple Effect Evaporator Dengan Menggunakaan Particle Swarm Optimization*. Malang : Fakultas Teknologi Pertanian Universitas Brawijaya.