# Model Penahan Ketinggian *Quadrotor* Berbasis PID dengan Jaringan Syaraf Tiruan Propagasi Mundur

## (*PID Based Quadrotor Altitude Retaining Model with Backward Propagation Artificial Neural Network*)

Faisal Fajri Rahani[1*], Dinan Yulianto[2]

***Abstract*—A quadrotor is a type of Unmanned Aerial Vehicle (UAV) or an unmanned flying vehicle flying remotely or using automatic control. In carrying out its mission, a quadrotor requires a good control system. One of the control systems in the quadrotor system is the altitude control system. Altitude control will control the quadrotor according to the desired altitude, whether there are interference and the quadrotor load. The widely used control method is the PID control. Unfortunately, the PID control produces a poor response because the PID constant is fixed, whereas the interference when the quadrotor flies will fluctuate. Therefore, this study offers control that can make a self-adjustment when exposed to specific interference. The method offered in this study is a PID control with Artificial Neural Networks (ANN). The ANN system will tune the PID components in real-time according to the occurring interference. The use of the PID with ANN results in a faster rise time response of 0.0594 seconds, a decrease in overshoot of 7.58%, a decrease in the steady-state error of ± 0.0672, and a decrease in settling time of 1.031 seconds compared to conventional PID. It shows that the PID with ANN results in better control than the PID alone.**

***Intisari*—*Quadrotor* adalah salah satu jenis *Unmanned Aerial Vehicle* (UAV) atau wahana terbang tanpa awak yang dapat terbang dengan kendali jarak jauh maupun menggunakan kendali otomatis. Dalam melakukan misinya, *quadrotor* memerlukan sistem kendali yang baik. Salah satu sistem kendali dalam sistem *quadrotor* adalah sistem kendali ketinggian. Kendali ketinggian akan mengendalikan *quadrotor* seusai ketinggian yang diinginkan walaupun terdapat gangguan dan beban *quadrotor* itu sendiri. Metode kendali yang banyak digunakan adalah kendali PID. Kendali PID menghasilkan respons yang kurang baik karena konstanta PID yang bersifat tetap, sedangkan gangguan saat *quadrotor* terbang akan berubah-ubah. Oleh karena itu, makalah ini menawarkan kendali yang dapat menyesuaikan diri saat terkena gangguan tertentu. Metode yang ditawarkan adalah kendali PID dengan Jaringan Saraf Tiruan (JST). Sistem JST akan menala komponen PID secara *real-time* sesuai gangguan yang terjadi. Penggunaan PID dengan JST menghasilkan respons *rise time* lebih cepat 0,0594 detik, *overshoot* turun 7,58%, *steady state error* turun ±0,0672, dan *settling time* turun 1,031 detik dibandingkan dengan PID konvensional. Hal ini menunjukkan bahwa PID dengan JST menghasilkan respons kendali yang lebih baik dibandingkan dengan PID saja.**

**Kata Kunci—PID, Jaringan Saraf Tiruan, *Quadrotor*, UAV**

[1,2] *Teknik Informatika Fakultas Teknologi Industri Universitas Ahmad Dahlan, Jl Ring Road Selatan, Tamanan Banguntapan, Bantul, Yogyakarta, 55166 (tlp: 0274-563515; fax: 0274-564604; email: [1] faisal.fajri@tif.uad.ac.id, [2]dinan.yulianto@tif.uad.ac.id)*

**corresponding author*

## I. PENDAHULUAN

*Unmanned Aerial Vehicle* (UAV) diartikan sebagai wahana terbang tanpa awak yang dapat terbang dengan dikendalikan secara jarak jauh maupun menggunakan kendali otomatis. Kemudahan tersebut membuat UAV banyak digunakan dalam berbagai hal [1]. Salah satu jenis UAV yang sering digunakan saat ini adalah *quadrotor* [2]. Kalangan penggemar, pemerhati militer, serta peneliti banyak menggunakan *quadrotor* karena fleksibilitasnya yang baik dan kemampuan manuver yang andal [3]. *Quadrotor* adalah salah satu jenis UAV yang dapat lepas landas dalam landasan yang terbatas karena sifatnya yang fleksibel [4]. Sistem yang membuat fleksibel tersebut dalam istilah penerbangan disebut dengan *Vertical Takeoff and Landing* (VTOL) [5], [6]. Salah satu kebutuhan dalam misi *quadrotor* untuk terbang yaitu sistem penahan ketinggian. Sistem penahan ketinggian berfungsi untuk mempertahankan ketinggian *quadrotor* agar mudah dikendalikan dan menjalankan misinya dengan baik.

Berdasarkan tuntutan penggunaan yang kompleks, *quadrotor* memerlukan sistem kendali yang baik untuk menjalankan misinya [7]. Keadaan tersebut diperoleh dengan cara memberikan sistem kendali pada *quadrotor* yang sesuai dengan peruntukannya [8]. Salah satu sistem kendali yang penting dalam sistem *quadrotor* adalah sistem kendali ketinggian [9]. Kendali ketinggian akan mengendalikan ketinggian *quadrotor* sesuai dengan ketinggian yang diinginkan walaupun ada gangguan, seperti angin dan beban *quadrotor* itu sendiri.

Berbagai metode kendali banyak diajukan pada sistem kendali *quadrotor*. Metode kendali tersebut adalah metode Proporsional Integral Derivatif (PID) [10], *fuzzy*, *Ant Colony Optimization* (ACO) [11], dan *Full State Feedback* dengan *Linear Quadratic Regulator* (LQR) [12].

Kendali PID atau PD banyak digunakan oleh dunia industri karena kemudahan dalam penggunaannya [1]. Kendali PID (*PID controller*) adalah sebuah mekanisme kendali *loop feedback* yang banyak digunakan dalam dunia sistem kendali. Kendali PID bekerja dengan cara mengalkulasi nilai *error* sebagai perbedaan variabel yang diukur dan nilai acuan yang diinginkan. Kendali PID meminimalkan nilai *error* yang ada dengan cara menyesuaikan masukan kendali sistem tersebut.

Pada sebuah *quadrotor*, kendali PID digunakan untuk menjaga *quadrotor* tetap stabil saat mengudara. Kendali PID mengumpulkan data dari sensor-sensor yang ada, kemudian membandingkannya dengan nilai yang diinginkan, lalu menyesuaikan kecepatan dari masing-masing motor agar *quadrotor* tetap stabil.

Pengendalian terbang *quadrotor* menggunakan metode kendali klasik PID memiliki kekurangan. Sistem kendali klasik PID didasarkan pada sistem yang linier atau sistem yang dimodelkan secara linier [5]. Sistem kendali yang menggunakan model sistem seperti kendali PID akan menghasilkan keluaran yang kurang akurat akibat linierisasi [13]. Kekurangan tersebut disebabkan komponen konstanta P, I, dan D bersifat tetap, sedangkan pada sistem ini dibutuhkan kendali yang dapat menyesuaikan diri saat terkena gangguan tertentu.

Beberapa metode dapat digunakan untuk menala komponen PID untuk penahan ketinggian. Beberapa metode dapat digunakan untuk menala komponen PID agar dapat berubah menyesuikan kebutuhan sistem, seperti *fuzzy* dan JST. JST digunakan pada sistem ini karena jaringan ini merupakan representasi tiruan dari otak manusia yang selalu mencoba untuk menyimulasikan proses pembelajaran seperti pada otak manusia [14]. Jaringan saraf digunakan untuk menyesuaikan nilai konstanta PID penahan ketinggian [13]. Metode JST digunakan sebagai sistem untuk menala konstanta PID pada sistem penahan ketinggian agar sesuai dengan kebutuhan sistem serta dapat menyesuaikan terhadap gangguan yang ada.

Pengembangan sistem PID dengan JST yang digunakan untuk mempertahankan ketinggian *quadrotor* diharapkan menjadi salah satu solusi untuk mempermudah pengguna dalam menyelesaikan misi *quadrotor* pada berbagai kebutuhan yang semakin kompleks.

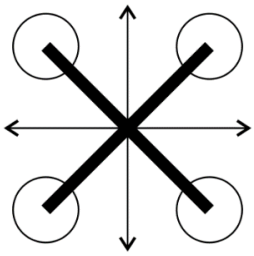
## II. METODOLOGI

### A. Quadrotor

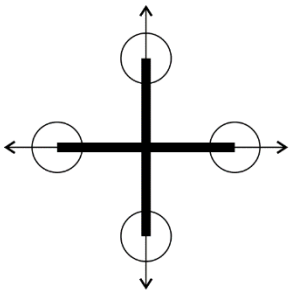
*Quadrotor* adalah salah satu jenis UAV yang memiliki ciri menggunakan empat buah rotor pada setiap lengannya. Masing- masing rotor pada *quadrotor* diberikan baling-baling yang berfungsi untuk mengatur gerakan *quadrotor* dengan menghasilkan gaya angkat (*thrust)*. Masing-masing baling-baling pada *quadrotor* dipasang dengan posisi horizontal [15].

Konfigurasi *quadrotor* secara umum ada dua jenis, yaitu konfigurasi X dan konfigurasi +. Perbedaan penting dari kedua konfigurasi tersebut didasari pada bagian yang menjadi sumbu referensi. Konfigurasi X diperlihatkan pada Gbr. 1, sedangkan konfigurasi + diperlihatkan pada Gbr. 2 [16].

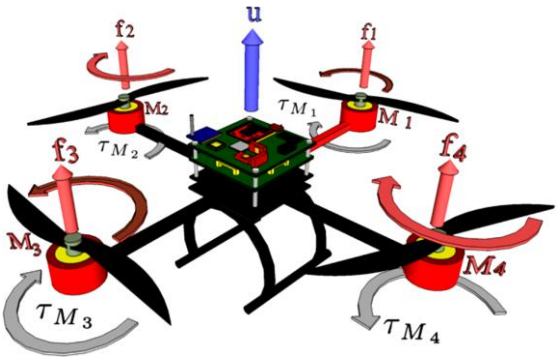
*Quadrotor* dengan konfigurasi kerangka X menggunakan empat buah rotor, terdiri atas dua rotor depan dan dua rotor belakang. Setiap rotor dengan baling-baling dapat menghasilkan gaya angkat dan torsi. Kombinasi dari gaya tersebut akan mengkasilkan gaya angkat ke atas, torsi *pitch*, *roll*, dan *yaw* [17]. Gaya angkat *quadrotor* dapat diperoleh dengan cara mengatur kecepatan keempat rotor dengan kecepatan tertentu sesuai dengan gerakan yang diinginkan. Pada pergerakan sumbu *pitch*, *roll*, dan *yaw*, *thrust* keempat motor saling berpengaruh [17].



Gbr. 1 Konfigurasi X



Gbr. 2 Konfigurasi +.



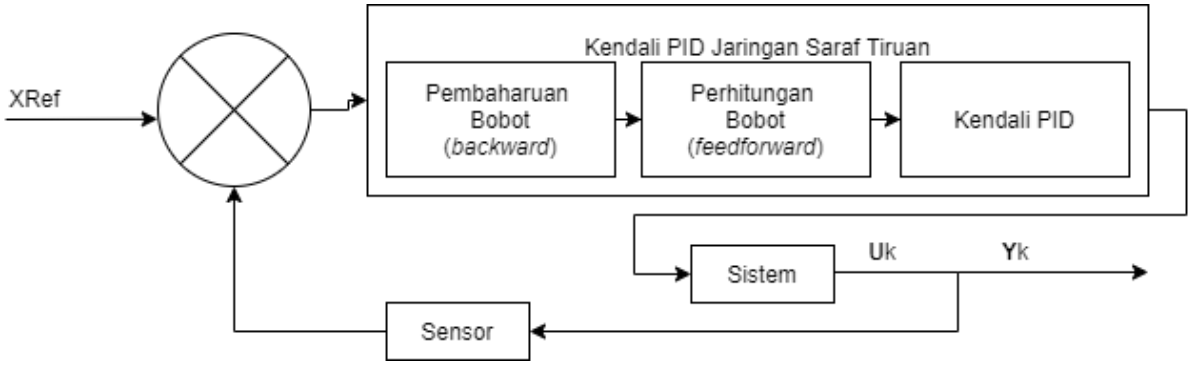

Gbr. 3 Masukan kendali *quadrotor*.

Terdapat perbedaan arah putaran masing-masing rotor pada *quadrotor*. Dua rotor berputar searah putaran jarum jam dan dua rotor lainnya berputar melawan arah putaran jarum jam. Dua rotor yang memiliki arah putar yang sama diletakkan berseberangan dengan rotor yang lain. Putaran masing-masing rotor pada *quadrotor* terlihat pada Gbr. 3 [17]. Pada konfigurasi *quadrotor* X, rotor 1 ($M_1$) dan rotor 3 ($M_3$) berputar searah dengan arah jarum jam. $M_1$ diletakkan pada bagian depan kanan, sedangkan $M_3$ diletakkan pada bagian belakang kiri. Rotor $M_2$ dan $M_4$ masing-masing berputar berlawanan dengan arah jarum jam. $M_2$ diletakkan pada bagian depan kanan dan $M_4$ diletakkan pada bagian belakang kiri dari *quadrotor*. Masing-masing rotor *Mi* (dengan $i$ = 1,2,3,4) akan menghasilkan gaya $F_i$ yang sebanding dengan kuadrat kecepatan motor. Gaya angkat pada *quadrotor* didapatkan dari penjumlahan gaya dorong ke atas yang dihasilkan oleh masing-masing rotor.

Sistem pada *quadrotor* memiliki gaya angkat yang diperoleh melalui (1) [18].

$$Fi = b\omega_i^2, \quad i = 1,2,3,4 \tag{1}$$

dengan $F$ merupakan gaya angkat pada setiap rotor, $i$ merupakan indeks *rotor* ke-*i*, $b$ merupakan nilai konstanta gaya




Gbr. 4 Diagram Sistem Kendali PID-JST.

angkat dari baling-baling yang digunakan, dan $\omega$ merupakan kecepatan putar masing-masing baling-baling. Gerakan translasi *quadrotor* ditunjukkan pada (2) [19].

$$\boldsymbol{m\dot{v}} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - {}^{o}R_B \begin{bmatrix} 0 \\ 0 \\ F_T \end{bmatrix} \quad (2)$$

dengan *m* merupakan massa *quadrotor*, $\dot{v}$ merupakan turunan pertama dari kecepatan $v$, $F_T$ merupakan gaya angkat total, ${}^{\circ}R_B$ adalah koordinat *frame* bumi, dan $g$ merupakan percepatan gravitasi bumi.

Gerak vertikal *quadrotor* dipengaruhi oleh gaya dorong total *T* dan berat *w* sepanjang sumbu z seperti yang ditunjukkan pada Gbr. 3. Kecepatan putar dari empat rotor yang berputar diubah secara bersamaan untuk mencapai gerakan vertikal [20].

Besar nilai torsi yang terjadi pada setiap baling-baling oleh rotor akan melawan gaya gesek udara. Besaran nilai torsi untuk rotor yang mengelilingi sumbu z diperoleh menggunakan (3) [19].

$$\tau_i = k\omega_i^2. \quad (3)$$

Pemodelan pada sistem *quadrotor* dilakukan menggunakan pendekatan persamaan Newton-Euler. Hukum Newton II digunakan untuk mencari hubungan antara gaya dorong **F** dan percepatan **a** yang dialami sebuah pusat massa. Hal tersebut ditunjukkan pada (4). Gaya angkat pada *quadrotor* merupakan hasil dari total gaya angkat pada masing-masing rotor pada *quadrotor*. Gaya total didapatkan menggunakan (5) [21].

$$\mathbf{F} = m\mathbf{a} \quad (4)$$

$$F_T = u_1 = \sum_{i=1}^{4} F_i. \quad (5)$$

Pergerakan *quadrotor* pada sumbu z bumi dapat dimodelkan dengan matriks rotasi pada (4) dan hukum Newton II pada (5), sehingga dapat diturunkan menjadi tiga persamaan sesuai dengan gaya yang terjadi pada masing-masing sumbu translasi.

Gerakan translasi pada sumbu z dapat digambarkan dengan (6) [22]. Gaya angkat atau $F_{total}$ didapat dari (5), sedangkan nilai $R_{B_z}^{E_y}$ didapatkan dari (7). Gerakan translasi pada sumbu z bumi adalah nilai gaya *F* dikurangi berat *quadrotor*. Pada saat pergerakan translasi pada sumbu y bumi, sudut yang berubah hanya sudut $\psi$ (*yaw*). Sudut $\theta$ (*pitch*), dan $\phi$ (*roll*) dianggap tidak mengalami perubahan sehingga dianggap bernilai 0. Hal tersebut membuat (7) dapat disederhanakan menjadi (8). Persamaan (9) menujukkan gerakan translasi yang bekerja pada sumbu z [9].

$$F_z = -F_T . R_{B_z}^{E_z} + m.g \quad (6)$$

$$m\ddot{z} = -(cos\,\phi\, cos\,\theta)F_T + mg \quad (7)$$

$$m.\ddot{z} = -F_T + m.g \quad (8)$$

$$\ddot{z} = -\frac{1}{m}F_T + g. \quad (9)$$

### *B. Rancangan Sistem Kendali*

Makalah ini menggunakan sistem PID-JST dalam kendali *quadrotor* untuk ketinggian. JST berfungsi untuk penalaan mandiri komponen-komponen konstanta PID yang digunakan untuk menjalankan kendali ketinggian pada *quadrotor*. Diagram blok pengendalian PID pada JST ditunjukkan pada Gbr. 4. Masukan kendali PID berupa selisih referensi masukan dan keluaran sistem. Hasil dari nilai tersebut kemudian diproses dalam JST untuk menala masing-masing komponen PID. Hasil penalaan tersebut kemudian menjadi masukan kendali PID. Selanjutnya, hasil keluaran dari kendali tersebut menjadi masukan kendali kembali.

### *C. Representasi Ruang Keadaan*

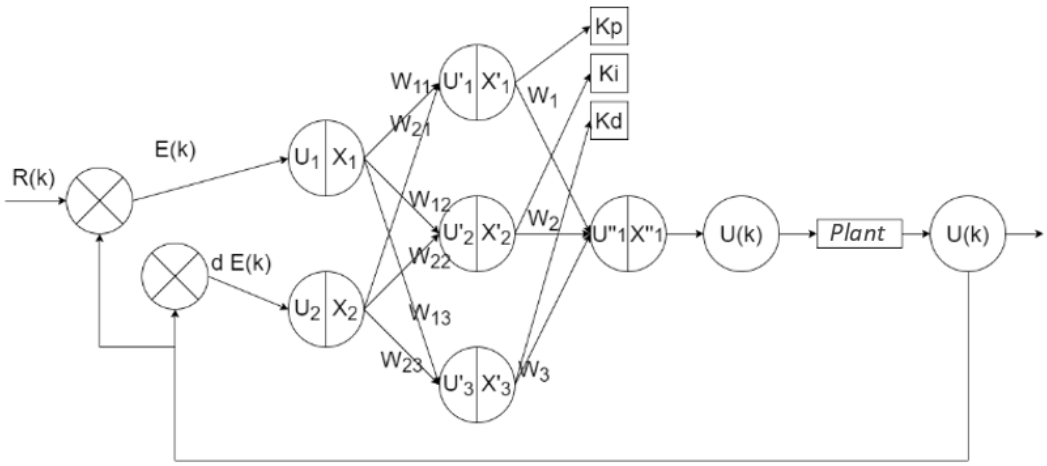
Representasi ruang keadaan (*state space*) adalah model matematis dari suatu sistem fisik sebagai himpunan variabel masukan, keluaran, dan keadaan yang terkait dengan persamaan diferensial orde pertama [20]. Ruang keadaan adalah ruang yang memiliki variabel keadaan sebagai sumbunya. Vektor digunakan di dalam ruang keadaan untuk menunjukkan keadaan (*state*) sistem. Secara umum, representasi ruang keadaan dari sistem linier dengan masukan *p*, keluaran *q*, dan variabel *n* keadaan ditulis dalam bentuk (10) dan (11) [21].

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (10)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \quad (11)$$

dengan

$\dot{\mathbf{x}}(t)$ = vektor keadaan (*state vector*)
$\mathbf{y}(t)$ = vektor keluaran (*output vector*)
$\mathbf{u}(t)$ = vektor masukan/kendali (*input/control vector*)
**A** = matriks sistem (*system matrix*)
**B** = matriks masukan (*input matrix*)




Gbr. 5 Diagram jaringan saraf tiruan yang digunakan.

**C** = matriks keluaran (*output matrix*)
**D** = matriks umpan maju (*feed forward matrix*).

Proses dimulai dengan memilih keadaan sistem. Variabel $u_1$ adalah total gaya ke atas pada *quadrotor* sepanjang sumbu z yang didapatkan dari $(F_T - mg)$, sehingga dari (5) dan (9) didapatkan (12) dan (13).

$$\dot{z} = \dot{z} \tag{12}$$

$$\ddot{z} = -u_1/m \tag{13}$$

Dua keadaan berikut secara praktis cocok untuk *quadrotor* di satu DOF untuk kendali ketinggian yang didapatkan dari (10) hingga (13) sehingga didapatkan (14) dan (15).

$z$ = posisi sepanjang sumbu z (tinggi)
$v_z$ = kecepatan sepanjang sumbu z.

Keluaran *y* pada sistem ini terdiri atas posisi z vertikal $(z)$.

$$\underset{\dot{\mathbf{x}}}{\begin{bmatrix}\dot{z}\\ \dot{v}_z\end{bmatrix}} = \underset{\mathbf{A}}{\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}} \underset{\mathbf{x}}{\begin{bmatrix}z\\ v_z\end{bmatrix}} + \underset{\mathbf{B}}{\begin{bmatrix}0\\ 1/m\end{bmatrix}} \underset{\mathbf{u}}{[u_1]} \tag{14}$$

$$\underset{\mathbf{y}}{[z]} = \underset{\mathbf{C}}{[1 \quad 0]} \underset{\mathbf{D}}{[0]} \underset{\mathbf{x}}{\begin{bmatrix}z\\ v_z\end{bmatrix}} \tag{15}$$

*D. Rancangan Sistem Jaringan Saraf Tiruan*

Struktur diagram kendali PID-JST pada Gbr. 5 merupakan kendali dengan menggunakan prinsip JST. JST yang digunakan pada sistem ini menggunakan sistem *online learning.* JST menggunakan tiga struktur bagian, yaitu:

- lapisan masukan (*input layer*),
- lapisan tersembunyi (*hidden layer*), dan
- lapisan keluaran (*output layer*).

Jaringan yang digunakan memiliki dua *node* pada jaringan masukan, tiga *node* pada jaringan tersembunyi, dan satu *node* untuk jaringan keluaran. Jaringan yang sederhana ini digunakan karena kebutuhan keluaran yang harus *real-time* dan agar sistem ini dapat langsung diimplementasikan pada mikrokontroler.

Pada JST ini terdapat aktivitas untuk mencari nilai bobot baru pada setiap jaringan. Pencarian nilai bobot baru ini dilakukan dengan prinsip mendapatkan nilai bobot sekecil mungkin dengan nilai *error* dan nilai *delta error* sekecil mungkin. Pencarian nilai bobot pada jaringan dilakukan secara terus-menerus berdasarkan masukan *error* pada sistem JST. Hal ini dilakukan dengan menggunakan fungsi turunan pada masing-masing perhitungan. Perhitungan bobot baru ini dilakukan setiap iterasi sebagai bagian dari pembelajaran propagasi mundur.

*1) Lapisan Masukan:* Lapisan ini merupakan lapisan masukan pada JST. Dua buah *node* masukan terdiri atas $U_1$ dan $U_2$. Masukan $U_1$ berupa nilai *error*. Nilai *error* didapatkan dari selisih nilai keluaran aktual sistem dengan nilai referensi yang digunakan. Masukan dari nilai $U_2$ adalah nilai perubahan *error* yang didapatkan dari selisih nilai *error* dengan nilai *error* sebelumnya. Nilai $U_1$ dan $U_2$ diberikan fungsi aktivasi *node* dengan fungsi *sigmoid* sehingga menjadi nilai $X_1$ dan $X_2$.

*2) Lapisan Tersembunyi:* Pada jaringan lapisan tersembunyi JST terdapat tiga *node*. *Node* lapisan tersembunyi didapatkan dari nilai $X_1$ dan $X_2$ yang dikalikan dengan bobot pada jaringan masukan menuju lapisan tersembunyi sehingga menghasilkan nilai $U'_1$, $U'_2$, dan $U'_3$. Masukan dari lapisan tersembunyi menggunakan (16) [9].

$$u'_j(k) = \sum_{j=1}^{2} w_{ij} \cdot x_i(k). \tag{16}$$

Berdasarkan penjelasan sesuai dengan Gbr. 5, keluaran dari lapisan tersembunyi menjadi nilai yang memengaruhi konstanta PID.

Aktivasi $U'_1$ dilakukan dengan memberikan fungsi *sigmoid* pada nilai $U'_1$ sehingga didapatkan nilai $X'_1$. Nilai $X'_1$ akan memengaruhi perubahan nilai Kp. Aktivasi $U'_2$ dilakukan dengan memberikan fungsi *sigmoid* pada nilai penjumlahan $U'_2$ dengan nilai $X'_2$ sebelumnya sehingga didapatkan nilai $X'_2$. Nilai node $X'_2$ akan memengaruhi perubahan nilai Ki. Adapun Aktivasi $U'_3$ dilakukan dengan memberikan fungsi *sigmoid* pada nilai selisih $U'_3$ dengan nilai $U'_3$ sebelumnya sehingga

didapatkan nilai $X'_3$. Nilai node $X'_3$ akan memengaruhi perubahan nilai Kd.

*3) Lapisan Keluaran:* Pada jaringan lapisan keluaran JST terdapat satu *node*. *Node* lapisan keluaran didapatkan dari nilai $X'_1$ , $X'_2$ dan $X'_3$ yang dikalikan dengan bobot pada jaringan lapisan tersembunyi menuju lapisan keluaran, menghasilkan nilai $U"_1$. Aktivasi $U"_1$ dilakukan dengan memberikan fungsi *sigmoid* pada nilai $U"_1$ sehingga didapatkan nilai $X"_1$.

Propagasi mundur pada sistem JST ini akan mengubah nilai bobot JST berdasarkan nilai bobot sebelumnya dan nilai *error* yang terjadi selama jaringan ini digunakan. Hal tersebut terlihat pada (17) [9].

$$j = E_p = \sum_n \frac{1}{2}\left(y_{(n)} - X_{ref(n)}\right)^2 \quad (17)$$

Hasil akhir dari JST pada sistem ini digunakan untuk meminimalkan nilai rerata kesalahan sistem yang dikuadratkan ($j$). Nilai $y$ merupakan keluaran sistem berupa bacaan sensor, sedangkan *Xref* merupakan nilai acuan sistem yang telah ditentukan. Selisih dari kedua nilai tersebut merupakan sebuah nilai *error*. Nilai $n$ merupakan banyaknya keadaan yang diamati pada sistem ini. Perubahan nilai bobot diatur dengan menggunakan fungsi *learning rate* ($\eta$) dengan fungsi pembaharuan bobot sebesar 0,01.

## III. Hasil dan Pembahasan

### A. Penggunaan Metode Ziegler-Nichols

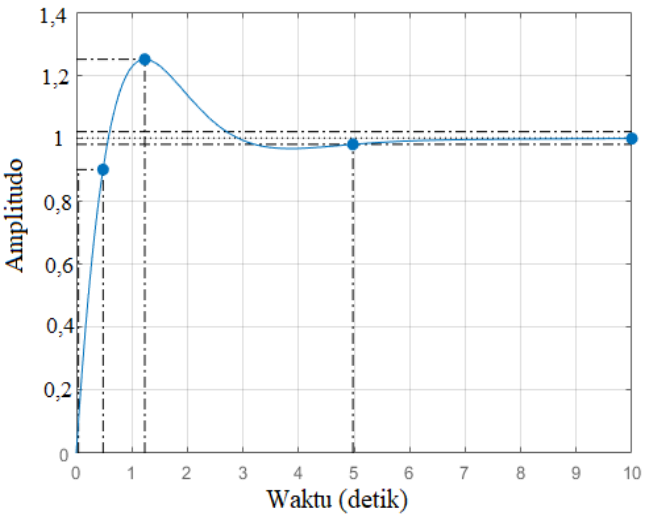
Pengujian sistem diawali dengan mencari nilai konstanta PID menggunakan metode Ziegler-Nichols. Dari penalaan konstanta ini didapatkan nilai konstanta proporsional sebesar 4,11, konstanta integral 1,31, serta konstanta derivatif sebesar 3,23.

Hasil dari penalaan tersebut kemudian diujikan dengan menggunakan respons undak. Respons ini ditunjukkan pada Gbr. 6. Pengujian tersebut mendapatkan nilai respons *rise time* sebesar 0,4432 detik, *overshoot* sebesar 24,97% dari nilai gangguan, *steady state error* ±0,2497, dan *settling time* sebesar 4,979 detik.

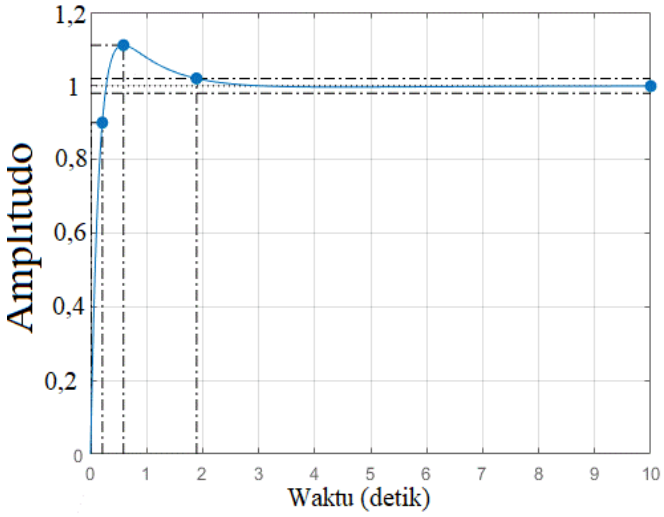
Dengan melihat hasil respons dari metode Ziegler-Nichols tersebut, sistem kemudian diperbaiki agar diperoleh nilai respons yang lebih baik. Parameter nilai yang baik di sini adalah nilai *rise time* kurang dari 0,2 detik. Pada penalaan ini diperoleh nilai konstanta proporsional sebesar 12,96, konstanta integral 3,44, dan konstanta derivatif sebesar 9,80. Tampak nilai konstanta yang dihasilkan lebih besar. Hasil dari perubahan nilai konstanta yang lebih besar ini adalah nilai respons yang kebih cepat. Hal ini terlihat dari uji undak yang dilakukan, yang mendapatkan nilai respons *rise time* sebesar 0,1965 detik, *overshoot* sebesar 11,04 % dari nilai gangguan, *steady state error* ±0,1105, dan *settling time* sebesar 1,892 detik, seperti ditunjukkan pada Gbr. 7.

### B. Penggunaan Metode PID dengan JST

Hasil dari penalaan konstanta PID dengan Ziegler-Nichols kemudian menjadi nilai acuan pada sistem penalaan konstanta menggunakan JST. Nilai tersebut akan dijadikan nilai awal pada pembelajaran JST yang digunakan pada sistem ini.




Gbr. 6 Hasil penalaan sumbu z dengan Ziegler-Nichols.

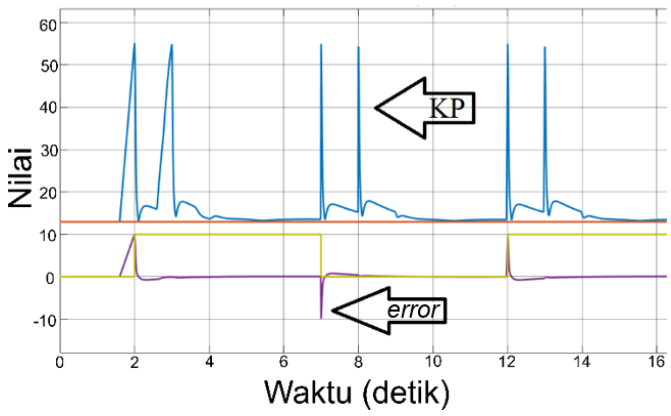



Gbr. 7 Hasil penalaan sumbu z dengan Ziegler-Nichols perbaikan.

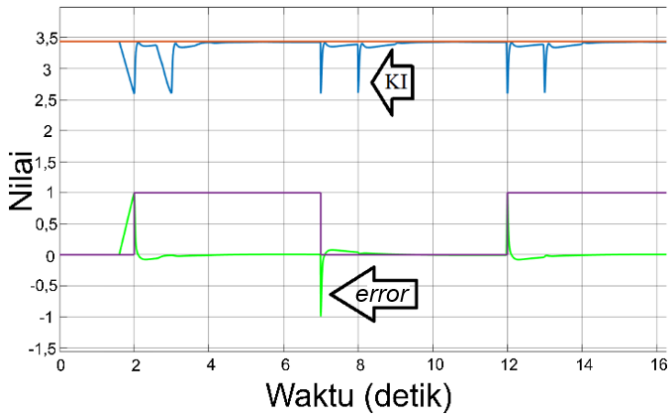
Hasil dari penalaan konstanta PID dengan JST menghasilkan nilai konstanta yang berubah sesuai dengan perubahan nilai *error* yang terjadi saat uji undak. Diperoleh nilai konstanta proporsional 13,46 hingga 54,65, nilai konstanta integral 3,45 hingga 4,27, dan nilai konstanta derivatif 10,29 hingga 51,48.

Perubahan nilai masing-masing konstanta tersebut terlihat pada Gbr. 8 sampai Gbr. 10. Pada Gbr. 8 terlihat nilai konstanta proposional (garis biru) berubah menjadi jauh lebih besar saat terjadi *error* yang besar (garis ungu). Perubahan nilai konstanta proporsional ini hanya terjadi saat *error* berubah menyesuaikan referensi yang berubah. Hal ini sesuai dengan tujuan konstanta proporsional yaitu untuk memberikan respons yang cepat jika memiliki nilai yang besar [9].

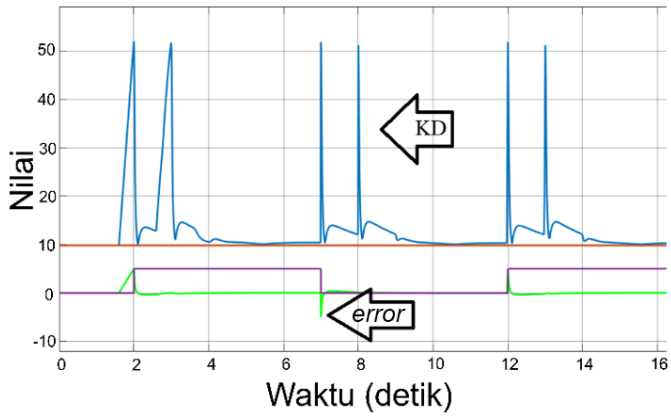
Pada Gbr. 9 terlihat grafik perubahan nilai konstanta integral (garis biru) yang menyesuaikan dengan perubahan *error* yang ada (garis ungu). Nilai konstanta integral semakin mengecil jika terjadi *error*. Hal ini bertujuan membuat nilai respons menjadi lebih cepat saat nilai *error* besar [23].




Gbr. 8 Perubahan nilai KP saat pengujian dengan perubahan referensi.



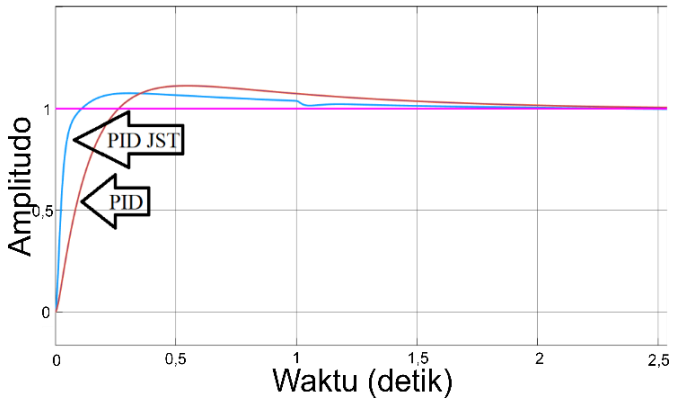

Gbr. 9 Perubahan nilai KI saat pengujian dengan perubahan referensi.



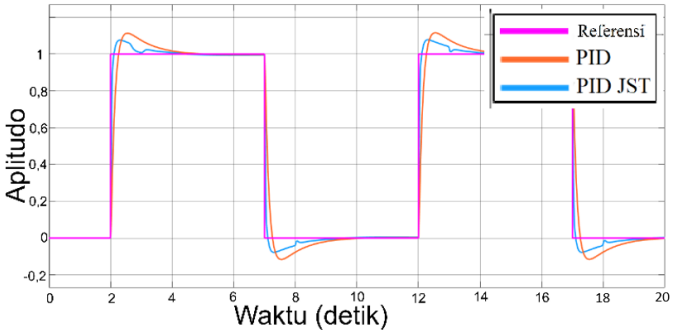

Gbr. 10 Perubahan nilai KD saat pengujian dengan perubahan referensi.

Gbr. 10 merupakan grafik perubahan nilai konstanta derivatif saat terjadi perubahan *error*. Nilai konstanta derivatif naik saat terjadi *error*. Nilai besaran perubahan konstanta tersebut bergantung dengan perubahan nilai *error* yang terjadi. Nilai konstanta derivatif membesar (garis biru) saat terjadi *error* yang besar (garis hijau). Nilai konstanta derivatif yang membesar ini memengaruhi nilai *overshoot* sistem yang akan mengecil saat terjadi *error* [24].

Pengujian dengan metode PID-JST dengan uji undak ditunjukkan pada Gbr. 11. Pada uji ini terlihat bahwa sistem JST (garis biru) menunjukkan respons yang lebih baik




Gbr. 11 Respons undak sumbu z.




Gbr. 12 Respons sistem sumbu z.

TABEL I
NILAI KONSTANTA PID PENGUJIAN

| | PID ZN | PID ZN *Fine Tune* | PID NN |
|---|---|---|---|
| KP | 4,11 | 12,96 | 13,46-54,65 |
| KI | 1,31 | 3,44 | 3,45-4,27 |
| KD | 3,23 | 9,80 | 10,29-51,48 |

TABELII
HASIL RESPON UJI UNDAK

| Respons | PID ZN | PID ZN *Fine Tune* | PID NN | Selisih |
|---|---|---|---|---|
| *Rise Time* | 0,4432 detik | 0,1965 detik | 0,0594 detik | 0,3838 detik |
| *Overshoot* | 24,97 % | 11,04 % | 7,58% | 17,39% |
| *Steady state error* | ±0,2497 | ±0,1105 | ±0,0672 | ±0,1825 |
| *Settling time* | 4,979 detik | 1,892 detik | 1,031 detik | 3,948 detik |

dibandingkan PID saja (garis coklat). Dari respons sistem PID-JST tersebut diperoleh *rise time* sebesar 0,0594 detik, *overshoot* sebesar 7,58% dari nilai gangguan, *steady state error* ±0,0672, dan *settling time* sebesar 1,031 detik pada uji undak. Sistem JST ini kemudian diberikan respons dengan perubahan nilai acuan secara berulang, yang ditunjukkan pada Gbr. 12. Pada gambar tersebut ditunjukkan respons PID dan PID-JST. PID-JST menunjukkan respons yang lebih cepat dan *overshoot* lebih rendah serta konsisten pada setiap perubahan nilai referensi.

Perbandingan nilai konstanta PID dari masing-masing metode ditunjukkan pada Tabel I. Pada data hasil pengujian dalam Tabel II terlihat bahwa penggunaan metode JST untuk

menala komponen PID secara *real-time* saat terjadi gangguan menghasilkan respons yang lebih baik pada sistem ini dibandingkan dengan metode PID konvensional. Hal ini disebabkan oleh nilai konstanta P, I, dan D yang dapat berubah sesuai dengan gangguan yang masuk pada sistem secara *real-time*.

Hasil dari pengujian ini menunjukkan bahwa PID dengan JST secara konsisten dapat menghasilkan respons yang lebih cepat, nilai *overshoot* yang lebih rendah, nilai *steady state error* yang lebih rendah, dan *settling time* yang lebih cepat.

## IV. KESIMPULAN

Dari hasil pengujian respons sistem pada simulasi, hasil respons undak pada sistem PID-JST lebih baik dibandingkan dengan PID Ziegler-Nichols. *Rise time* mengalami perbaikan sebesar 0,3838 detik, ada selisih *overshoot* 17,39%, *steady state error* mengalami perbaikan ±0,1825, dan *settling time* mengalami perbaikan 3,948 detik. Hal tersebut menunjukkan bahwa penggunaan sistem PID-JST dapat memperbaiki respons sistem sesuai dengan perubahan gangguan yang terjadi dibandingkan dengan penggunaan metode kendali PID Ziegler-Nichols.

## UCAPAN TERIMA KASIH

Terima kasih diucapkan kepada Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) Universitas Ahmad Dahlan yang telah memfasilitasi kegiatan penelitian ini.

## REFERENSI

[1] J. Gómez-Avila, C. López-Franco, A.Y. Alanis, dan N. Arana-Daniel, "Control of Quadrotor using a Neural Network based PID," *2018 IEEE Lat. Am. Conf. on Comput. Intel. (LA-CCI)*, Nov. 2018, hal. 1–6.

[2] H. Mansoor, I.U.H. Shaikh, dan S. Habib, "Genetic Algorithm Based LQR Control of Hovercraft," *2016 Int. Conf. on Intel. Sys. Eng. (ICISE)*, 2016, hal. 335–339.

[3] X. Wang, C. Sun, X. Lin, dan Y. Yu, "Adaptive Neural Network Control of a Quadrotor with Input Delay," *Proc. 2018 Chinese Autom. Congr. (CAC 2018)*, 2018, hal. 4095–4100.

[4] Z. Qing, M. Zhu, dan Z. Wu, "Adaptive Neural Network Control for a Quadrotor Landing on a Moving Vehicle," *Proc. 30th Chinese Control Decis. Conf. CCDC 2018*, 2018, hal. 28–33.

[5] Y.F. Teng, B. Hu, Z.W. Liu, J. Huang, dan Z.H. Guan, "Adaptive Neural Network Control for Quadrotor Unmanned Aerial Vehicles," *2017 Asian Control Conf. ASCC 2017*, 2018, hal. 988–992.

[6] T. Bodrumlu, A. Bek, M.A. Akbulut, dan F. Caliskan, "Design and Control of Artificial Neural Network Based Low-Cost Autonomous Quadrotor System," *2018 6th Int. Conf. Control Eng. Inf. Technol. CEIT 2018*, 2018, hal. 25–27.

[7] S. Liao, X. Lei, dan Y. Xiao, "The Compound Control Method for Pesticide Spraying Quadrotor UAVs," *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. (ITNEC 2019)*, 2019, hal. 1022–1027.

[8] Y.M. Chen, Y.L. He, dan M.F. Zhou, "Decentralized PID Neural Network Control for a Quadrotor Helicopter Subjected to Wind Disturbance," *J. Cent. South Univ.*, Vol. 22, No. 1, hal. 168–179, 2015.

[9] F.F. Rahani dan T.K. Priyambodo, "Implementasi Full State Feedback LQR dengan JST pada Kendali Ketinggian Quadrotor," *J. Nas. Tek. Elektro dan Teknol. Inf.*, Vol. 8, No. 4, hal. 357-363, Nov. 2019.

[10] T.K. Priyambodo, A.E. Putra, dan A. Dharmawan, "Optimizing Control Based on Ant Colony Logic for Quadrotor Stabilization," *2015 IEEE Int. Conf. on Aero. Electron. and Remote Sens. Technol. (ICARES)*, 2015, hal. 1–4.

[11] T.K. Priyambodo, A. Dharmawan, dan A.E. Putra, "PID Self Tuning Control Based on Mamdani Fuzzy Logic Control for Quadrotor Stabilization," *AIP Conf. Proc.*, 2016, Vol. 1705, No. 1, hal. 020013.

[12] F.F. Rahani dan T.K. Priyambodo, "Penalaan Mandiri Full State Feedback dengan LQR dan JST pada Kendali Quadrotor," *Indonesian J. Electron. Instrum. Syst. (IJEIS)*, Vol. 9, No. 1, hal. 21-32, Apr. 2019.

[13] C. Sun, T. Lu, dan K. Yuan, "Balance Control of Two-Wheeled Self-Balancing Robot Based on Linear Quadratic Regulator and Neural Network," *2013 Fourth Int. Conf. Intell. Control Inf. Process.(ICICIP)*, 2013, hal. 862–867.

[14] P. Gautam, "Optimal Control of Inverted Pendulum System Using ADALINE Artificial Neural Network with LQR," *2016 Int. Conf. Recent Adv. Innov. Eng.*, 2016, hal. 1–6.

[15] A.Z. Azfar dan D. Hazry, "A Simple Approach on Implementing IMU Sensor Fusion in PID Controller for Stabilizing Quadrotor Flight Control," *Proc. 2011 IEEE 7th Int. Colloq. on Signal Proces. and Its Appl, (CSPA 2011)*, 2011, hal. 28–32.

[16] S. Gupte dan J.M. Conrad, "A Survey of Quadrotor Unmanned Aerial Vehicles," *2012 Proc. IEEE Southeastcon*, 2012, hal. 1–6.

[17] L.R. García Carrillo, A.E. Dzul López, R. Lozano, dan C. Pégard, *Quad Rotorcraft Control*, Vol. 1, London, UK: Springer London, 2013.

[18] P. Pounds, R. Mahony, dan P. Corke, "Modelling and Control of a Quad-Rotor Robot," *Proc. of the 2006 Australas. Conf. on Robot. and Automat.*, 2006, pp. 1-10.

[19] A. Ollero, "Aerial Robotic Manipulators," dalam *Encyclopedia of Robotics*, M.H. Ang, O. Khatib, dan B. Siciliano, Eds. Heidelberg, Germany: Springer Berlin Heidelberg, 2019, hal. 1–8.

[20] Z. Tahir, "State Space System Modeling of a Quad Copter UAV," *Indian J. Sci. Technol.*, Vol. 8, No. 1, hal. 1–5, Jan. 2015.

[21] K. Ogata, *Modern Control Engineering*, New York, USA: Prentice Hall, 2010.

[22] A.B. Zakaria dan A. Dharmawan, "Sistem Kendali Penghindar Rintangan pada Quadrotor Menggunakan Konsep Linear Quadratic," *Indones. J. Electron. Instrum. Syst.*, Vol. 7, No. 2, hal. 219–230, 2017.

[23] A. O'Dwyer, "PI and PID Controller Tuning Rules: An Overview and Personal Perspective," *Proc. IET Irish Signals Syst. Conf.*, 2006, hal. 161–166.

[24] T.K. Priyambodo, A. Dharmawan, O.A. Dhewa, dan N.A.S. Putro, "Optimizing Control Based on Fine Tune PID Using Ant Colony Logic for Vertical Moving Control of UAV System," *AIP Conf. Proc.*, Vol. 1755, No. 1, hal. 170011.1-7, 2016.