TUGAS AKHIR – TE141599

PENGENDALIAN PERGERAKAN *HOVER* QUADCOPTER MENGGUNAKAN METODE PID JARINGAN SYARAF TIRUAN

Prihatama Kunto Wicaksono
NRP 2213 106 067

Dosen Pembimbing
Eka Iskandar, ST. MT.
Ir. Rusdhianto Effendie A.K., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE141599

HOVER MOVEMENT CONTROL OF QUADCOPTER USING ARTIFICIAL NEURAL NETWORK PID

Prihatama Kunto Wicaksono
NRP 2213 106 067

Adviser
Eka Iskandar, ST. MT.
Ir. Rusdhianto Effendie A.K., MT.

DEPARTEMEN OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

PENGENDALIAN PERGERAKAN *HOVER QUADCOPTER*
MENGGUNAKAN METODE PID
JARINGAN SYARAF TIRUAN

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Eka Iskandar, ST., MT.	**Ir. Rusdhianto Effendie A.K., MT.**
NIP.198005282008121001	NIP.195704241985021001

SURABAYA
JANAURI, 2016

PENGENDALIAN PERGERAKAN *HOVER* QUADCOPTER MENGGUNAKAN METODE PID JARINGAN SYARAF TIRUAN

Nama : Prihatama Kunto Wicaksono
Pembimbing I : Eka Iskandar, ST., MT.
Pembimbing II : Ir. Rusdhianto Effendie A.K., MT.

ABSTRAK

Gerakan *hover* quadcopter dapat dilakukan apabila quadcopter dapat menjaga kestabilan saat terbang pada ketinggian tertentu, karena gerak ini terjadi dengan mempertahankan posisi sudut dan translasi pada quadcopter. Sehingga untuk melakukan gerak *hover* diperlukan suatu kontroler rotasi dan kontroler translasi. Terdapat dua jenis kontroler untuk mengatur gerak rotasi. Pertama menggunakan metode jaringan syaraf tiruan untuk menentukan nilai K_p, K_i dan K_d hasil linierisasi untuk mengatur kecepatan sudut, dan kontroler tipe PD untuk mengatur posisi sudut. Kedua digunakan kontroler tipe PD untuk kontrol translasi x dan y, sedangkan kontroler tipe PID untuk kontrol translasi Z. Kontroler ini nantinya akan dirancang agar dapat mempertahankan pergerakan *hover* walaupun mendapat gangguan dari luar**.** Hasil simulasi menunjukan bahwa metode jaringan syaraf tiruan memiliki prosentase eror posisi yang lebih kecil dibandingkan dengan kontroler PID hasil linierisasi. Prosentase eror posisi untuk metode jaringan syaraf tiruan sebesar 2,34% untuk kontrol posisi sudut *roll* dan 0,34 % untuk kontrol posisi sudut *pitch*, sedangkan kontroler PID hasil linierisasi memiliki prosentase eror sebesar 3,6% untuk kontrol posisi sudut *roll* dan 4% untuk kontrol posisi sudut *pitch.*

Kata Kunci: Quadcopter, *Hover*, Jaringan Syaraf Tiruan, PID, PD

HOVER MOVEMENT CONTROL OF* QUADCOPTER *USING ARTIFICIAL NEURAL NETWORK PID

Name ***: Prihatama Kunto Wicaksono***
Adviser I ***: Eka Iskandar, ST., MT.***
Adviser II ***: Ir. Rusdhianto Effendie A.K., MT.***

ABSTRACT

Quadcopter hover movement can be done if quadcopter can maintain stable when flying at certain altitudes, because this movement occurs by maintaining the angular position and translation in quadcopter. So to perform hover motion, it is necessary controller rotational and controller translational. There are two type controller to control rotational. First uses artificial neural network method to determine the value of K_p, K_i dan K_d *results linearized to control the angular velocity, and the type of PD controller to control the angular position.Second uses PD controller to control translational axes x and y, then PID controler to controler translational motion of axes z. Both of these controllers will be designed in order to maintain the hover movement although given the disturbance from outside.Simulation result show that neural network has position error precentage more small than PID linieritation controller. Error position precentage neural network for roll angular position control is 2,34% and 0,34 % for pitch angular position control. In the other side, PID linieritation controller has error position precentage for roll angular position control is 3,6% and 4 % for pitch angular position control.*

Key Word: *Quadcopter, Hover, Artificial Neural Network, PID, PD*

KATA PENGANTAR

Assalamualaikum wr. wb.

Puji dan syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan buku Tugas Akhir ini. Shawalat serta salam senantiasa tercurah pula kepada nabi besar baginda Rasulullah Muhammad SAW.

Buku Tugas Akhir ini disusun untuk melengkapi salah satu syarat memperoleh gelar sarjana teknik di jurusan Teknik Elektro ITS. Buku yang berjudul **"*Pengendalian Pergerakan Hover Quadcopter Menggunkan Metode PID Jaringan Syaraf Tiruan*"** dipersembahkan juga untuk kemajuan riset dan teknologi Indonesia khusunya untuk ITS, Fakultas Teknologi Industri, Jurusan Teknik Elektro, dan bidang studi Teknik Sistem Pengaturan.

Hambatan dan rintangan selalu ada dalam menyelesaikan menyelesaikan buku ini. Namun dukungan dan bantuan terus mengalir hingga penulis dapat menyelesaikan buku ini. Pada kesempatan ini tak lupa penulis menyampaikan rasa terima kasih kepada beberapa pihak, antara lain:

1. Allah SWT yang telah memlimpahkan ramhat dan hidayah-Nya serta memperlancar dalam pengerjaan Tugas Akhir ini.
2. Orang tua dan adik tercinta yang secara tidak langsung menjadi sumber semangat dalam menyelesaikan Tugas Akhir ini.
3. Dosen Pembimbing I, Bapak. Eka Iskandar, ST., MT atas segala bimbingannya kepada penulis dalam pengerjaan Tugas Akhir ini.
4. Dosen Pembimbing II, Bapak Ir. Rusdhianto Effendie A.K., MT. atas segala bimbingannya kepada penulis dalam pengerjaan Tugas Akhir ini.
5. Kepala Laboratorium, Bapak Ir. Ali Fathoni, MT. atas segala fasilitas yang dapat penulis gunakan dalam pengerjaan Tugas Akhir ini.
6. Bapak dan Ibu penguji Tugas Akhir yang telah memberi masukan kepada penulis sehingga buku ini menjadi lebih baik.
7. Teman-teman *team* TA quadcopter, Yurid, Farid, Recho, dan Temmi atas segala kerjasamanya dalam pengerjaan buku Tugas Akhir ini.
8. Rezky Mahardika atas segala bantuan dalam pengerjaan Tugas Akhir ini.

9. Teman-teman Lab B105 atas segala semangat dan kebersamaannya dalam pengerjaan Tugas Akhir ini.
10. Ferbriani Husniah yang sudah banyak memberi *suport* selama penulis mengerjakan tugas akhir.
11. Teman-teman lintas jalur angkatan 2013 ganjil yang selalu memberi warna kehidupan selama ini.
12. Teman – teman kontraan seperti mas Jainul, mas Joko, mas Hery, Adit, Ghofur, dan Yody yang sudah menjadi saudara saya selama di Surabaya.
13. Semua pihak yang turut membantu dalam pengerjaan Tugas Akhir ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari dan memohon maaf karena masih banyak kekurangan pada Tugas Akhir ini. Kritik dan saran selalu penulis nantikan agar menjadi lebih baik pada masa mendatang. Akhir kata, penulis berharap Tugas Akhir ini dapat bermanfaat dan menjadi acuan dalam penelitian selanjutnya.

Surabaya, Januari 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Tugas Akhir merupakan penelitian yang dilakukan oleh mahasiswa S1 Institut Teknologi Sepuluh Nopember Surabaya. Tugas Akhir ini merupakan salah satu syarat wajib untuk menyelesaikan studi dalam program sarjana teknik.

Pada bab ini, akan dibahas mengenai hal-hal yang mendahului pelaksanaan Tugas Akhir. Hal-hal tersebut meliputi latar belakang, perumusan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

1.1 Latar Belakang

Quadcopter merupakan pesawat tanpa awak yang memiliki empat buah motor dan baling-baling di tiap ujung-ujung kerangka utama. Bagian tengah digunakan untuk peletakan sumber daya (baterai), sistem kontrol, dan sensor dari quadcopter. Sistem kontrol digunakan untuk mengatur kecepatan dari tiap-tiap motor sesuai dengan gerakan yang diinginkan. Salah satu contohnya adalah melakukan pergerakan *hover* untuk mengambil gambar di suatu tempat yang tidak memungkinkan pesawat terbang lainnya mengambil gambar secara langsung. Kondisi ini dibutuhkan kontroler yang dapat membuat quadcopter stabil dan mampu mempertahankan posisi quadcopter pada ketinggian tertentu, walaupun terdapat gangguan *external* yang mampu membuat kecenderungan quadcopter untuk tidak stabil.

Kontrol PID adalah salah satu jenis kontroler yang sering digunakan untuk menjadi kontroler dari quadcopter. Komponen quadcopter ini terdiri dari tiga jenis yaitu, *Proportional, Derivatif* dan *Integretif*. Ketiga jenis kontroler ini yang akan saling membantu untuk mendapatkan respon keluaran quadcopter yang stabil dan meminimalkan terjadinya osilasi pada daerah *set point.* Kontroler inilah yang nantinya akan mengatur pergerakan quadcopter dengan cara mengatur kecepatan putar dari tiap motornya, sehingga quadcopter dapat bergerak menuju *set point* dan stabil.

Kegiatan *tuning* yang dilakukan pada sistem kendali PID robot , baik robot darat maupun pesawat tanpa awak kebanyakan masih berupa *tuning* secara manual, yakni mencoba nilai koefisien PID mulai dari nilai terendah hingga ditemukan nilai PID dengan respon yang terbaik. Metode jaringan syaraf tiruan ini digunakan untuk menentukan nilai K_p,

K_i, dan K_d yang sesuai,sehingga tidak dibutuhkan *tuning* nilai PID secara manual. Proses pembelajaran jaringan syaraf tiruan diharapkan membuat sistem menjadi lebih stabil dan mendekati nilai *set point* yang diinginkan.

1.2 Permasalahan

Permasalahan dalam Tugas Akhir ini adalah bagaimana merancang dan mengimplementasikan metode jaringan syaraf tiruan untuk mendapatkan nilai K_p, K_i, dan K_d supaya tidak perlu dilakukan *tuning* nilai K_p, K_i dan K_d secara manual untuk mengontrol pergerakan *hover* quadcopter. Massa quadcopter yang ringan juga menjadi masalah tersendiri, massa yang ringan membuat quadcopter lebih mudah mendapatkan gangguan secara *external*.

1.3 Tujuan

Tujuan Tugas Akhir ini diharapkan mampu menerapkan metode kontrol jaringan saraf tiruan (JST) untuk *tuning* nilai K_p, K_i, dan K_d agar dapat menghasilkan pengendalian pergerakan *hover* quadcopter yang stabil dan mampu mempertahankan posisi dan ketinggian seperti yang diinginkan. Salah satu manfaat Tugas akhir ini diharapkan mampu membantu untuk proses pengambilan gambar yang tidak memungkinkan pesawat terbang lainnya lakukan.

1.4 Batasan Masalah

Setiap penelitian diperlukan suatu batasan agar penelitian dapat lebih spesifik. Penelitian Tugas Akhir ini digunakan model UAV jenis quadcopter dengan spesifikasi motor *brushless SunnySky V2216 KV900*, *propeller* 10x4,5, kontroler apm 2.6, sensor GY-80, ESC dengan arus 30A , *transmitter* FlySky FS-TH9XB dan *receiver* FlySky FS-R8B 8CH. Pemodelan quadcopter dilakukan dengan identifikasi parametrik untuk mendapatkan model matematika *plant*. Nilai PID yang digunakan untuk proses pembelajaran JST didapat dari hasil linierisasi kecepatan sudut dari titik 0 rad sampai 0,2 rad. Struktur JST yang digunakan satu *hidden layer* dengan 2 bobot *hidden layer*. Perhitungan nilai bobot-bobot JST dilakukan secara *offline.* Pengendalian yang dilakukan untuk metode JST hanya pada kontrol sudut. Sedangkan untuk kontrol translasi digunakan kontroler tipe PD. Perancangan kontroler dilakukan melalui *software* matlab 2014a.

1.5 Metodologi

Metodologi yang digunakan pada penelitian Tugas Akhir ini adalah sebagai berikut :

a) Studi Literatur.

Studi literatur berguna untuk mencari informasi atau data mengenai *plant*, kontroler, atau sistem secara keseluruhan. Studi literatur diperlukan sebagai landasan dalam mengerjakan Tugas Akhir agar diperoleh teori penunjang yang memadai, baik ilmu dasar, analisis, maupun metode penelitian. Hal ini dapat dilakukan dengan melihat acuan dari jurnal, buku teks, internet, dan lain-lain. Dengan adanya studi literatur, penelitian dapat dilakukan berdasarkan teori-teori yang telah ada sebelumnya.Dilakukan dengan mencari bahan *paper* mengenai pengaturan cerdas dan quadcopter yang akan dikontrol.

b) Perancangan *Hardware* dan *Software*.

Perancangan *hardware* dan *software* digunakan untuk melakukan simulasi dan implementasi. Perancangan hardware dilakukan dengan memasang komponen – komponen dalam quadcopter seperti: sensor, ESC,kontroler APM,dan lain-lain. Sedangkan perancangan *software* berupa perancangan kontroler dan model matematika *plant* didalam *software* Matlab.

c) Identifikasi dan Pemodelan *Plant*.

Identifikasi adalah tahap untuk mendapatkan nilai paramater dari quadcopter dengan cara melakukan percobaan – percoban dan pengukuran secara langsung pada quadcopter. Setelah mendapatkan nilai parameter – parameter quadcopter dilanjutkan dengan melakukan pemodelan matematika.

d) Desain Kontroler dan *Simulasi*.

Setelah didapat model matematika quadcopter langkah selanjutnya adalah mendesain kontrol PID hasil linierisasi dari model matematika quadcopter. Hasil nilai PID yang didapat digunakan sebagai proses pembelajaran JST.

e) Penulisan Buku Tugas Akhir.

Buku tugas akhir ditulis secara intensif bila proses pengujian telah selesai.

1.6 Sistematika

Pada Tugas Akhir ini sistematika penulisan dibagi menjadi lima bab, yaitu :

BAB I PENDAHULUAN

Bab ini meliputi latar belakang, permasalahan, tujuan, metodologi, sistematika penulisan dan relevansi pada tugas akhir.

BAB II TEORI DASAR

Bab ini membahas tinjauan pustaka yang membantu penelitian, diantaranya konsep dasar, kinematika dan dinamika dari quadcopter. Selain itu juga dibahas tentang kontroler untuk quadcopter, diantaranya kontroler PD, PID, JST, dan linierisasi.

BAB III PERANCANGAN SISTEM

Bab ini akan dibahas mengenai perancangan-perancangan sistem dari quadcopter. Mulai dari mendapatkan parameter – parameter model matematika quadcopter, perancangan secara *hardware* dan *software*

BAB IV HASIL DAN PEMBAHASAN

Bab ini memuat hasil simulasi tentang sistem. Mulai dari proses learning PID oleh JST, kontrol kestabilan saat hover dan saat quadcopter diberi gangguan.

BAB V PENUTUP

Bab ini berisi beberapa kesimpulan dan saran dari hasil pengujian sistem quadcopter.

1.7 Relevansi

Hasil yang didapatkan dari tugas akhir ini diharapkan dapat menjadi referensi salah satu kontroler untuk mengendalikan kestabilan *quadcopter*. Selain itu untuk mengetahui pengaruh metode kontrol dengan PID JST yang diterapkan pada quadcopter.

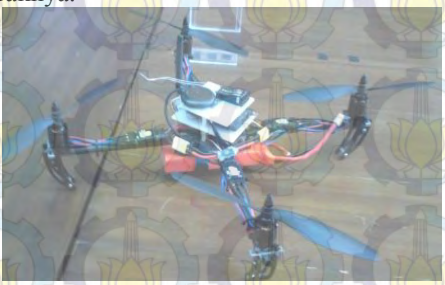
Halaman ini sengaja dikosongkan

BAB II
TEORI DASAR

Penyusunan Tugas Akhir ini terdapat beberapa teori dasar yang menjadikan pengetahuan untuk merumuskan dan menyelesaikan masalah yang diangkat. Bagian awal dicantumkan tinjauan pustaka yang menggambarkan landasan teori secara umum. Bagian selanjutnya akan dibahas tentang teori – teori pendukung, seperti konsep dasar quadcopter, kinematika, dinamika, kontroler PD, kontroler PID, jaringan syaraf tiruan dan linierisasi.

2.1 Tinjauan Pustaka[2]

Quadcopter adalah robot jelajah udara *Unmanned Aerial Vehicle* (UAV) yang termasuk kategori UAV mikro dan banyak digunakan oleh industri, instansi pendidikan, atau masyarakat umum. Robot quadcopter ini memiliki ciri yang *unique* yang mudah untuk dikenali yaitu memiliki empat buah baling –baling motor pada keempat sisinya yang digunakan sebagai penggeraknya.



Gambar 2. 1 Quadcopter dengan konfigurasi X

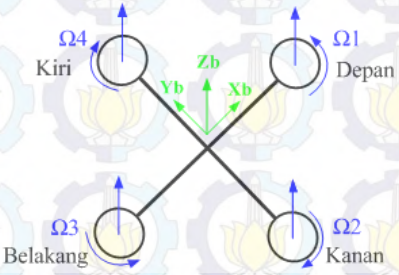
Gambar di atas adalah salah satu bentuk robot quadcopter dengan konfigurasi *plus* (+). Empat buah baling – baling yang terpasang memudahkan quadcopter untuk bermanuver sehingga dapat bergerak ke segala arah. Quadcopter dilengkapi dengan beberapa sensor, diantaranya sensor *Global Position System* (GPS) yang digunakan untuk navigasi, sensor *Inertial Measurement Unit* (IMU) yang berfungsi untuk menghitung percepatan serta orientasi arah pergerakan, sensor *ultra sonic* yang mendeteksi ketinggian quadcopter, dan sensor – sensor lainnya yang mendukung fungsi dan kinerja quadcopter.

Quadcopter memiliki beberapa kelebihan yang membuatnya cocok untuk melakukan pekerjaan tertentu. Bentuknya yang kecil membuat quadcopter dapat bergerak secara leluasa ditempat yang sulit dijangkau. Quadcopter tidak memerlukan landasan pacu untuk terbang, karena quadcopter dapat terbang vertikal. Selain itu quadcopter juga dapat bergerak kedelapan arah mata angin.

Quadcopter juga memiliki beberapa kekurangan, diantaranya quadcopter hanya dapat bergerak dalam jangka waktu yang pendek. Hal ini dikarenakan sumber tenaga dari quadcopter adalah baterai yang memiliki kapasitas terbatas. Jangka waktu tersebut secara tidak langsung akan mempengaruhi kapasitas bawaan, kecepatan terbang, dan jarak tempuh dari quadcopter.

2.1.1 Konsep Dasar Quadcopter[3]

Quadcopter termasuk robot terbang yang setiap pergerakannya dipengaruhi oleh kecepatan dari keempat motornya. Sehingga jika ingin mengendalikan pergerakan quadcopter harus mengetahui beberapa gerak dasar quadcopter dan teknik pengaturan kecepatan keempat motornya.



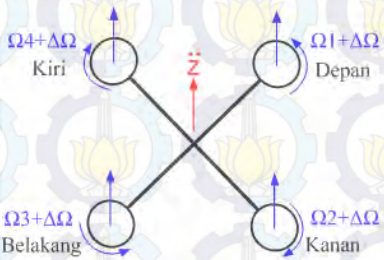
Gambar 2. 2 Quadcopter dalam posisi *hover*

Gambar 2.2 menunjukan quadcopter dalam kondisi *hover* atau gerak melayang. Kondisi ini memperlihatkan bahwa motor depan dan belakang berputar berlawanan arah jarum jam, sedangkan motor kanan dan kiri berputar searah jarum jam. Konfigurasi dari pergerakan quadcopter adalah *plus*, sedangkan Z_B, X_B, dan Y_B adalah *body frame* (*B-frame*) dari quadcopter.

Kombinasi dari perubahan kecepatan keempat motor akan menghasilkan beberapa pergerkan. Berikut adalah beberapa pergerakan yang ada dalam quadcopter:

a) Gaya *Thrust* (U1[N])

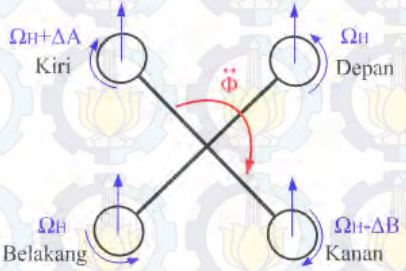
Gaya *thrust* adalah gaya yang mengakibatkan quadcopter bergerak naik atau turun searah sumbu Z *earth frame*. Gaya *thrust* terjadi jika kecepatan keempat motor sama dan secara bersamaan ditambah kecepatannya dengan nilai yang sama, maka quadcopter akan menghasilkan gaya angkat (*thrust*). Begitu juga sebaliknya, jika kecepatan dari keempat motor diperlambat secara bersamaan, quadcopter akan bergerak kebawah.



Gambar 2. 3 Gaya *thrust*

b) Torsi *Roll* (U2[Nm])

Torsi *roll* adalah torsi yang mengakibatkan quadcopter berputar disepanjang sumbu x *body frame* (X_B). Torsi ini dipengaruhi oleh motor sebelah kanan dan kiri sedangkan dua motor yang lain nilai kecepatannya tetap. Jika kecepatan motor kiri dipercepat sedangkan motor kanan diperlambat maka quadcopter akan bergerak mengguling ke kanan. Begitu pula jika sebaliknya, jika motor kanan yang dipercepat dan motor kiri yang diperlambat maka quadcopter akan bergerak mengguling ke kiri.

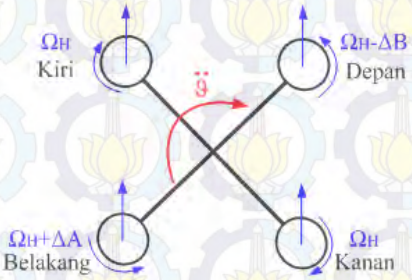


Gambar 2. 4 Torsi *roll*

ΔA dan ΔB dipilih dengan mempertahankan pergerakan *hover*. Sedangkan ΩH adalah kecepatan motor saat *hover* [rad/s].

c) Torsi *Pitch* (U3[Nm])

Torsi *Pitch* adalah torsi yang mengakibatkan quadcopter berputar disepanjang sumbu y pada *body frame* quadcopter (Y_B). Gerak ini dipengaruhi oleh perubahan kecepatan dari motor depan dan motor belakang, sedangkan 2 motor yang lain nilainya tetap. Jika kecepatan motor depan diperlambat sedangkan kecepatan motor belakang dipercepat, maka quadcopter akan bergerak mengangguk kedepan. Begitu pula jika kecepatan motor depan dipercepat dan kecepatan motor belakang diperlambat maka quadcopter akan bergerak mengangguk kebelakang.



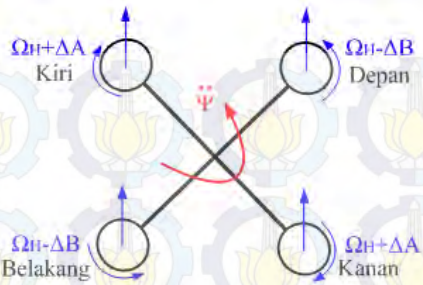
Gambar 2. 5 Torsi *pitch*

ΔA dan ΔB dipilih dengan mempertahankan pergerakan *hover*. Sedangkan ΩH adalah kecepatan motor saat *hover* [rad/s].

d) Torsi *Yaw* (U4[Nm])

Torsi *yaw* adalah torsi yang mengakibatkan quadcopter berputar disepanjang sumbu Z pada *body frame* quadcopter (Z_B). Gerak ini dipengaruhi oleh perubahan kecepatan dari keempat motor quadcopter. Jika kecepatan motor depan dan belakang diperlambat sedangkan kecepatan motor kanan dan kiri dipercepat maka quadcopter akan bergerak menyimpang ke kiri. Begitu pula sebaliknya, jika kecepatan motor depan dan belakang dipercepat sedangkan motor yang lain diperlambat , maka quadcopter akan bergerak menyimpang ke kanan.

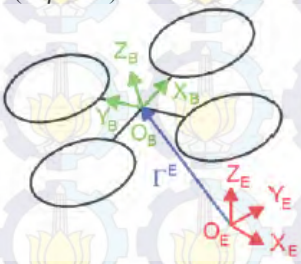
Perubahan kecepatan yang terjadi pada keempat motor memiliki nilai yang sama.



Gambar 2. 6 Torsi *yaw*

2.1.2 Kinematika[3][7]

Kinematika adalah cabang dari mekanika klasik yang membahas gerak benda dan sistem benda tanpa mempersoalkan gaya penyebab gerakan. Quadcopter juga dipelajari tentang kinematika, sehingga untuk mempermudah analisa quadcopter memiliki keluaran 6 *degree of freedom* (DOF). 6 DOF *rigid-body* di diskripsikan dalam dua buah *frame*. *Frame* yang digunakan sebagai referensi yaitu *earth frame* (*E-frame*) dan *body frame* (*B-frame*).



Gambar 2. 7 *Frame* quadcopter

Gambar 2.7 menunjukan bahwa terdapat 2 buah *frame*. *Frame* dengan warna merah adalah *earth frame* (*E-frame*) dengan masing – masing sumbu *frame* mengarah pada utara (X_E), barat (Y_E), arah keatas (Z_E),dan origin (O_E). Sedangkan *frame* dengan warna hijau adalah *body frame* (*B-frame*) dengan masing – masing sumbu *frame* mengarah pada depan (X_B), kiri (Y_B), keatas (Z_B), dan origin (O_B).

Posisi linier quadrotor (Γ_E) ditentukan dari koordinat vektor antara origin *B-frame* serta origin dari *E-frame* dengan memperhatikan *E-frame*. Posisi angular quadcopter (Θ_E) ditentukan dari orientasi *B-*

frame terhadap *E-frame*. Persamaan posisi linier dan posisi angular masing – masing ditulis pada persamaan dibawah.

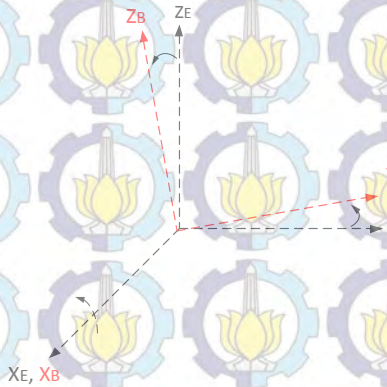
$$\Gamma^{E} = [X \quad Y \quad Z]^{T} \tag{2.1}$$

$$\Theta^{E} = [\phi \quad \theta \quad \psi]^{T} \tag{2.2}$$

Salah satu yang diperlukan untuk mentransformasi suatu nilai dari *body frame* (*B-frame*) ke *earth frame* (*E-frame*) adalah matrik rotasi. Matrik rotasi ini terdiri dari 3 buah matrik rotasi yang masing – masing berotasi terhadap sumbu x bumi (X_E), sumbu y bumi (Y_E), dan sumbu z bumi (Z_E).

a) Rotasi Sumbu X

Rotasi sepanjang sumbu x dilambangkan dengan R(ϕ,x) yang merupakan rotasi quadcopter sepanjang sumbu x menghasilkan sudut yang diberi nama sudut *roll*. Rotasi tersebut direpresentasikan dengan Gambar 2.8.



Gambar 2. 8 Rotasi sumbu X

$$\begin{cases} X_E = X_B \\ Y_E = Y_B \cos\phi - Y_B \sin\phi \\ Z_E = Y_B \sin\phi + Z_B \cos\phi \end{cases} \tag{2.3}$$

Persamaan 2.3 dapat dibuat dalam bentuk matriks menjadi Persamaan 2.4.

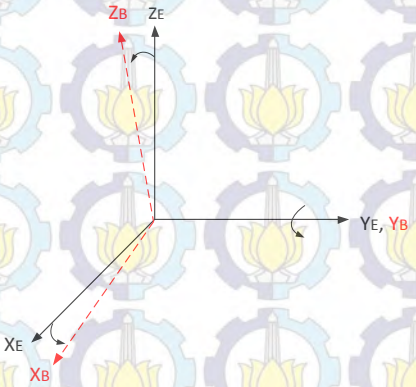
$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} \tag{2.4}$$

Adapun matrik rotasi pada sumbu x ditunjukkan pada persamaan 2.5.

$$R(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \quad (2.5)$$

b) Rotasi Sumbu Y

Rotasi sepanjang sumbu y dilambangkan dengan R(ɵ,y) yang merupakan rotasi quadcopter sepanjang sumbu y menghasilkan sudut yang diberi nama sudut *pitch*. Rotasi tersebut direpresentasikan dengan Gambar 2.9



Gambar 2. 9 Rotasi sumbu Y

$$\begin{cases} X_E = X_B\cos\theta + Z_B\sin\theta \\ Y_E = Y_B \\ Z_E = -X_B\sin\theta + Z_B\cos\theta \end{cases} \quad (2.6)$$

Persamaan 2.6 dapat dibuat dalam bentuk matriks menjadi Persamaan 2.7.

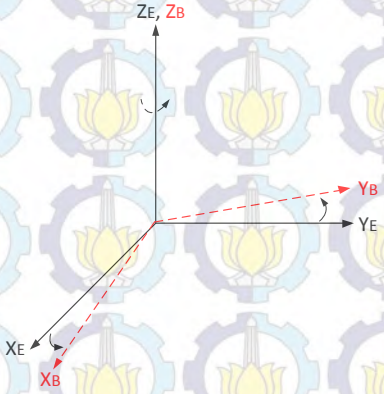
$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} \quad (2.7)$$

Adapun matriks rotasi pada sumbu y ditunjukkan pada persamaan 2.8.

$$R(\theta, y) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (2.8)$$

c) Rotasi Sumbu Z

Rotasi sepanjang sumbu z dilambangkan dengan R(ψ,z) yang merupakan rotasi quadcopter sepanjang sumbu z yang menghasilkan sudut yang diberi nama sudut *yaw*. Rotasi tersebut direpresentasikan dengan Gambar 2.10.



Gambar 2. 10 Rotasi sumbu Z

$$\begin{cases} X_E = X_B\cos\psi - Y_B\sin\psi \\ Y_E = X_B\sin\psi + Y_B\cos\psi \\ Z_E = Z_B \end{cases} \quad (2.9)$$

Persamaan 2.9 dapat dibuat dalam bentuk matriks menjadi persamaan 2.10

$$\begin{bmatrix} X_E \\ Y_E \\ Z_E \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} \quad (2.10)$$

Adapun matriks rotasi pada sumbu z ditunjukkan pada persamaan 2.11

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.11)$$

Persamaan rotasi dari tiap – tiap sumbu putar didapat satu persamaan matrik rotasi.

$$R_\Theta = R(\psi)R(\theta)R(\phi) \quad (2.12)$$

$$R_\Theta = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi s_\theta s_\phi \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \quad (2.13)$$

Sedangkan untuk kecepatan dalam quadcopter diekspresikan terhadap *body frame* (*B-frame*). Kecepatan quadcopter terdiri dari kecepatan linier V^B dan kecepatan angular ω^B. Komposisi vektornya disajikan dalam persamaan 2.14 dan 2.15.

$$V^B = [u \quad v \quad w]^T \quad (2.14)$$

$$\omega^B = [p \quad q \quad r]^T \quad (2.15)$$

Diperlukan kombinasi nilai linear dan angular untuk memberikan representasi yang lengkap dalam *space*. ξ merupakan komposisi dari vektor posisi linier Γ^E (m) dan vektor posisi sudut Θ^E (rad) terhadap *earth frame* (*E-frame*) seperti terlihat pada persamaan 2.16.

$$\xi = [\Gamma^E \quad \Theta^E]^T = [X \quad Y \quad Z \quad \phi \quad \theta \quad \psi]^T \quad (2.16)$$

V merupakan generalisasi dari vektor kecepatan linier quadcopter V^B (m s^{-1}) dan kecepatan angular quadcopter $\boldsymbol{\omega}^B$ (rad s^{-1}) pada *body frame* (*B-frame*) seperti terlihat pada persamaan dibawah ini:

$$V = [V^B \quad \omega^B]^T = [u \quad v \quad w \quad p \quad q \quad r]^T \quad (2.17)$$

Hubungan antara kecepatan linear pada *body frame* (*B-frame*) dan salah satu factor pada *earth frame* (*E-frame*) V^E (atau $\dot{\Gamma^E}$) [m/s] dapat dilihat pada Persamaan 2.18.

$$V^E = \dot{\Gamma^E} = R_\Theta V^B \quad (2.18)$$

Dimana R_Θ adalah matrik rotasi dari *body frame* (*B-frame*) ke *earth frame* (*E-frame*). Seperti pada kecepatan linear, hal tersebut juga berlaku untuk menghubungkan kecepatan angular pada *E-frame* (atau kecepatan Euler) $\dot{\Theta^E}$ [rad/s] ke *B-frame* $\boldsymbol{\omega}^B$ atau sebaliknya. Hubungan itu dapat dilihat pada persamaan 2.19.

$$\omega^B = T_\Theta^{-1} \dot{\Theta^E} \quad (2.19)$$

$$\dot{\Theta^E} = T_\Theta \omega^B \quad (2.20)$$

dimana T_Θ adalah matriks transformasi. Matriks transformasi T_Θ dapat ditetapkan dengan menggunakan kecepatan Euler dalam *B-frame*, dengan membalik pola perputaran sudut dari *roll*, *pitch* dan *yaw* seperti pada Persamaan 2.21.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi)^{-1} R(\theta)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \quad (2.21)$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = T_{\Theta}{}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (2.22)$$

Persamaan 2.13 dan 2.19 maka diperoleh matriks transformasi dari *body frame* (*B-frame*) menuju *earth frame* (*E-frame*).

$$T_{\Theta}{}^{-1} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \quad (2.23)$$

$$T_{\Theta} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \quad (2.24)$$

Persamaan yang sudah didapat, maka dibentuk suatu hubungan antara kecepatan terhadap *earth frame* (*E-frame*) dan *body frame* (*B-frame*).

$$\dot{\xi} = J_\Theta v \quad (2.25)$$

Dimana $\dot{\xi}$ adalah vektor kecepatan yang mengacu pada *earth frame* (*E-frame*) , ν adalah vektor kecepatan mengacu *body frame* (*B-frame*) dan J_Θ adalah matrik *jacobian*. Matrik *jacobian* terdiri dari 4 *sub*-matrik sebagaimana Persamaan 2.26

$$J_\Theta = \begin{bmatrix} R_\Theta & 0_{3x3} \\ 0_{3x3} & T_\Theta \end{bmatrix} \quad (2.26)$$

2.1.3 Dinamika[3][7]

Dinamika adalah cabang dari ilmu fisika (terutama mekanika klasik) yang mempelajari gaya , torsi dan efeknya pada gerak. Dinamika merupakan kebalikan dari kinematika, yang mempelajari gerak suatu objek tanpa memperhatikan penyebabnya. Isaac Newton menciptakan hukum-hukum fisika yang menjadi panduan dalam fisika dinamika. Secara umum, dinamika sangat berkaitan erat dengan hukum kedua newton tentang gerak.

Berdasarkan aksioma pertama Euler dari Hukum Kedua Newton, turunan dari komponen linear dari gerakan suatu benda dapat dilihat pada Persamaan 2.27.

$$\begin{cases} m\,\ddot{\Gamma}^{\mathrm{E}} = F^{E} \\ m\,\dot{\overbrace{\mathrm{R}_{\Theta}\mathrm{V}^{\mathrm{B}}}} = R_{\Theta}\,F^{B} \\ m\left(\mathrm{R}_{\Theta}\dot{\mathrm{V}}^{\mathrm{B}} + \dot{\mathrm{R}}_{\Theta}\mathrm{V}^{\mathrm{B}}\right) = R_{\Theta}\,F^{B} \\ m\,\mathrm{R}_{\Theta}\left(\dot{\mathrm{V}}^{\mathrm{B}} + \omega^{\mathrm{B}}\,\mathrm{x}\,\mathrm{V}^{\mathrm{B}}\right) = \mathrm{R}_{\Theta}F^{B} \\ m\left(\dot{\mathrm{V}}^{\mathrm{B}} + \omega^{\mathrm{B}}\,\mathrm{x}\,\mathrm{V}^{\mathrm{B}}\right) = F^{B} \end{cases} \tag{2.27}$$

Dimana m (kg) adalah massa quadrotor, $\ddot{\Gamma}^{\mathrm{E}}$ (m s^{-2}) adalah vektor percepatan linear quadrotor yang mengacu pada E-*frame*, F^{E} (N) adalah vektor gaya quadrotor terhadap E-*frame*, $\dot{\mathrm{V}}^{\mathrm{B}}$ (m s^{-2}) adalah percepatan linear quadcopter terhadap B-*frame*, dan $\dot{\mathrm{R}}_{\Theta}$ adalah turunan pertama matriks rotasi. Dan simbol × merupakan perkalian produk suatu vektor.

Berdasarkan aksioma kedua Euler dari Hukum Kedua Newton, dengan cara yang sama, turunan dari gerakan angular komponen dari suatu benda dapat dilihat pada Persamaan (2.28).

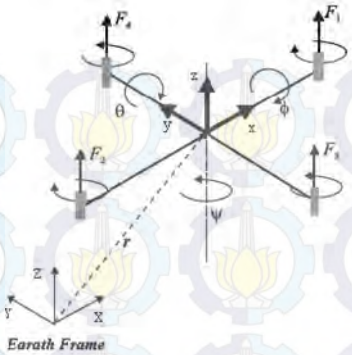
$$\begin{cases} I\ddot{\Theta}^{\mathrm{E}} = \tau^{\mathrm{E}} \\ I\,\dot{\overbrace{\mathrm{T}_{\Theta}\omega^{\mathrm{B}}}} = \mathrm{T}_{\Theta}\tau^{\mathrm{B}} \\ I\dot{\omega}^{\mathrm{B}} + \omega^{\mathrm{B}}\,x\,(I\omega^{\mathrm{B}}) = \tau^{\mathrm{B}} \end{cases} \tag{2.28}$$

Persamaan 2.28, I (N m s^{2}) adalah matriks inersia quadcopter (pada B-*frame*), $\ddot{\Theta}^{\mathrm{E}}$ (rad s^{-2}) adalah vektor percepatan sudut quadcopter terhadap E-*frame*, $\dot{\omega}^{\mathrm{B}}$ (rad s^{-2}) adalah vektor percepatan sudut quadcopter terhadap B-*frame*, dan $\boldsymbol{\tau}^{E}$ (N m) adalah torsi quadcopter terhadap E-*frame*.

Persamaan 2.27 dan 2.28, dapat dibentuk suatu persamaan baru dengan 6 derajat kebebasan (DOF). Persamaan 2.29 menunjukkan formulasi matriks dari dinamika sistem.

$$\begin{bmatrix} m\,I_{3\,x\,3} & 0_{3x3} \\ 0_{3x3} & I \end{bmatrix}\begin{bmatrix} \dot{\mathrm{V}}^{\mathrm{B}} \\ \dot{\omega}^{\mathrm{B}} \end{bmatrix} + \begin{bmatrix} \omega^{\mathrm{B}}\,\mathrm{x}\,(m\mathrm{V}^{\mathrm{B}}) \\ \omega^{\mathrm{B}}\,\mathrm{x}\,(I\omega^{\mathrm{B}}) \end{bmatrix} = \begin{bmatrix} F^{B} \\ \tau^{B} \end{bmatrix} \tag{2.29}$$

di mana notasi $\mathrm{I}_{3\mathrm{x}3}$ berarti matriks identitas 3x3. Sedangkan $0_{3\mathrm{x}3}$ adalah matriks nol 3x3. Quadcopter memiliki empat buah motor yang menghasilkan gaya dorong seperti terlihat pada gambar 2.11.



Gambar 2. 11 *E-frame* dan *B-frame* quadcopter beserta empat gaya disetiap *propeller*

Vektor gaya yang terjadi pada quadcopter dapat didefinisikan melalui persamaan 2.30.

$$\Lambda = [F^B \ \tau^B]^T = [F_x \quad F_y \quad F_z \quad \tau_x \quad \tau_y \quad \tau_z]^T \tag{2.30}$$

Persamaan 2.30 dapat juga ditulis dalam bentuk formulasi matriks seperti pada persamaan 2.31.

$$M_B\dot{v} + C_B(v)v = \Lambda \tag{2.31}$$

di mana $\dot{v}$ adalah vektor percepatan quadcopter terhadap *B-frame*. M_B adalah matriks inersia sistem dan C_B adalah adalah matriks sentripetal Coriolis. Persamaan 2.32 menunjukkan matriks M_B**.**

$$M_B = \begin{bmatrix} m\,I_{3x3} & 0_{3x3} \\ 0_{3x3} & I \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{XX} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{YY} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{ZZ} \end{bmatrix} \tag{2.32}$$

Dapat dilihat bahwa matriks M_B adalah matriks diagonal, sedangkan matriks C_B ditunjukkan oleh Persamaan 2.33.

$$C_B = \begin{bmatrix} 0_{3x3} & -m\,S(\mathrm{V_B}) \\ 0_{3x3} & -S(I\ \omega_\mathrm{B}) \end{bmatrix}$$

$$C_B = \begin{bmatrix} 0 & 0 & 0 & 0 & m\,w & -m\,v \\ 0 & 0 & 0 & -m\,w & 0 & m\,u \\ 0 & 0 & 0 & m\,v & -m\,u & 0 \\ 0 & 0 & 0 & 0 & I_{ZZ}\,r & -I_{YY}\,q \\ 0 & 0 & 0 & -I_{ZZ}\,r & 0 & I_{XX}\,p \\ 0 & 0 & 0 & I_{YY}\,q & -I_{XX}\,p & 0 \end{bmatrix} \quad (2.33)$$

Pada persamaan ini, diadopsi operator *skew-symmetric* $S(.)$ untuk vektor 3 dimensi k, matriks *skew-symmetric* dari k (S(k)) ditunjukkan pada Persamaan 2.34.

$$S(k) = -S^T(k) = \begin{bmatrix} 0 & -k3 & k1 \\ k3 & 0 & -k1 \\ -k2 & k1 & 0 \end{bmatrix}, k = \begin{bmatrix} k1 \\ k2 \\ k3 \end{bmatrix} \quad (2.34)$$

Matriks Λ dapat dibagi menjadi tiga komponen menurut kontribusi sifat dasar quadcopter. Pertama adalah vektor gravitasi $G_B(\dot{\xi})$ yang diperoleh dari percepatan gravitasi g (m s^{-2}). Percepatan gravitasi hanya berlaku pada persamaan linear quadcopter. Persamaan matriks $G_B(\xi)$ dituliskan pada Persamaan 2.35.

$$G_B(\xi) = \begin{bmatrix} F_{GB} \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} {R_\Theta}^{-1} F_{GE} \\ 0_{3x1} \end{bmatrix}$$

$$= \begin{bmatrix} {R_\Theta}^{T} \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} mg\,sin\,\theta \\ -mg\,cos\,\theta\,sin\,\phi \\ -\,mg\,cos\,\theta\,sin\,\phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.35)$$

dimana F_{GB} (N) adalah vektor gaya gravitasi terhadap B-*frame* dan F_{GE} (N) terhadap E-*frame*. R_Θ adalah matriks ortogonal, sehingga matriks inversnya sama dengan matriks transposnya.

Kontribusi yang kedua mengenai efek giroskopis yang dihasilkan oleh perputaran propeler. Selama dua propeler yang berhadapan berputar searah jarum jam dan dua propeler lainnya berputar berlawanan arah jarum jam, terjadi ketidak-seimbangan saat jumlah aljabar dari kecepatan motor tidak sama dengan nol. Jika *roll* dan *pitch* juga tidak nol, quadcopter akan mengalami torsi efek giroskopis yang ditunjukkan pada Persamaan 2.36.

$$O_B(v)\Omega = \begin{bmatrix} 0_{3x1} \\ -\sum_{k=1}^{4} J_{TP}\left(\omega^B \text{ x } \begin{bmatrix}0\\0\\1\end{bmatrix}\right)(-1)^k\Omega_k \end{bmatrix} = \begin{bmatrix} 0_{3x1} \\ J_{TP}\begin{bmatrix}-q\\p\\0\end{bmatrix}\Omega \end{bmatrix}$$

$$= J_{TP}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -q & -q & -q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix}\Omega \qquad (2.36)$$

O_B adalah matriks giroskopis dari propeller dan J_{TP} (N m s^2) adalah momen inersia total di sekitar sumbu *propeller*. Efek giroskopis yang dihasilkan oleh putaran *propeller* hanya berhubungan dengan persamaan angular, dan tidak mempengaruhi persamaan linear. Ω (rad s^{-1}) merupakan vektor kecepatan putar *propeller*, yang ditunjukkan oleh Persamaan 2.37.

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4, \qquad \Omega = \begin{bmatrix}\Omega_1\\\Omega_2\\\Omega_3\\\Omega_4\end{bmatrix} \qquad (2.37)$$

Kontribusi yang ketiga adalah perhitungan gaya dan torsi yang dihasilkan oleh pergerakan input. Pergerakan matriks E_B dikali dengan Ω^2 untuk memperoleh vektor perpindahan $U_B(\Omega)$. Efek aerodinamis (*factor thrust b* (N s^2) dan *drag d* (N m s^2)) berpengaruh pada gaya dan torsi yang dihasilkan. Persamaan vektor perpindahan pada dinamika quadcopter ditunjukkan pada Persamaan 2.38.

$$U_B(\Omega) = E_B\Omega^2 = \begin{bmatrix}0\\0\\U_1\\U_2\\U_3\\U_4\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \\ bl(-{\Omega_2}^2 + {\Omega_4}^2) \\ bl(-{\Omega_1}^2 + {\Omega_3}^2) \\ d(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2) \end{bmatrix} \qquad (2.38)$$

dimana l (m) adalah jarak antara pusat quadcopter dengan pusat *propeller*. U_1, U_2, U_3, dan U_4 adalah komponen vektor gerak. Hubungan komponen vektor tersebut dengan *propeller* dapat diperoleh dari perhitungan aerodinamis.

Dari persamaan sebelumnya, dapat diketahui konstanta matriks E_B yang dikalikan dengan kuadrat dari kecepatan propeler Ω^2 sehingga menghasilkan $U_B(\Omega)$. Persamaan 2.39 menunjukkan matriks E_B.

$$E_B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b & b & b & b \\ 0 & -bl & 0 & bl \\ -bl & 0 & bl & 0 \\ -d & d & -d & d \end{bmatrix} \tag{2.39}$$

Persamaan 2.31, dinamika quadcopter dapat dibuat dengan mempertimbangkan ketiga kontribusi menurut Persamaan 2.40.

$$M_B \dot{v} + C_B(v)v = G_B(\xi) + O_B(v)\,\Omega + E_B\Omega^2 \tag{2.40}$$

Dengan mengatur ulang persamaan 2.40 diatas, turunan dari vektor kecepatan terhadap B-*frame* dapat dirumuskan sebagai berikut:

$$\dot{v} = {M_B}^{-1}\,(-C_B(v)v + G_B(\xi) + O_B(v)\,\Omega + E_B\Omega^2) \tag{2.41}$$

Persamaan 2.42 menunjukkan persamaan-persamaan sebelumnya bukan dalam bentuk matriks, melainkan dalam bentuk sistem persamaan.

$$\begin{cases} \dot{u} = (v\,r - w\,q) + g\,s_\theta \\ \dot{v} = (w\,p - u\,r) - g\,c_\theta s_\phi \\ \dot{w} = (u\,q - v\,p) - g\,c_\theta s_\phi + \frac{U_1}{m} \\ \dot{p} = \frac{I_{YY} - I_{ZZ}}{I_{XX}} q\,r - \frac{J_{TP}}{I_{XX}} q\,\Omega + \frac{U_2}{I_{XX}} \\ \dot{q} = \frac{I_{ZZ} - I_{XX}}{I_{YY}} p\,r - \frac{J_{TP}}{I_{YY}} p\,\Omega + \frac{U_3}{I_{YY}} \\ \dot{r} = \frac{I_{XX} - I_{YY}}{I_{ZZ}} p\,q - \frac{U_4}{I_{ZZ}} \end{cases} \tag{2.42}$$

Persamaan input kecepatan *propeller* (baling-baling) dapat dilihat dalam Persamaan 2.43.

$$\begin{cases} U_1 = b({\Omega_1}^2 + {\Omega_2}^2 + {\Omega_3}^2 + {\Omega_4}^2) \\ U_2 = bl(-{\Omega_2}^2 + {\Omega_4}^2) \\ U_3 = bl(-{\Omega_1}^2 + {\Omega_3}^2) \\ U_4 = d(-{\Omega_1}^2 + {\Omega_2}^2 - {\Omega_3}^2 + {\Omega_4}^2) \\ \Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{cases} \tag{2.43}$$

Persamaan sistem dinamik quadcopter 2.42 diperoleh dari B-*frame*. Dalam pemodelannya, dibutuhkan persamaan dinamik sistem yang mencakup sistem *hybrid* yang disusun oleh persamaan linear dari *E-frame* dan persamaan angular dari B-*frame*. Hal ini dilakukan untuk mempermudah pengaturan. Persamaan-persamaan berikut akan disajikan dalam kerangka "*hybrid*", atau H-*frame*. Persamaan 2.44 menunjukkan vektor kecepatan quadcopter yang digeneralisasi pada *H-frame*.

$$\zeta = [\dot{\Gamma}^{E} \quad \omega^{B}]^{T} = [\dot{X} \quad \dot{Y} \quad \dot{Z} \quad p \quad q \quad r]^{T} \tag{2.44}$$

Dinamika sistem pada H-*frame* dapat dituliskan dalam bentuk martriks menurut Persamaan (2.45).

$$M_H\dot{\zeta} + C_H(\zeta)\zeta = G_H + O_H(\zeta)\,\Omega + E_H(\zeta)\Omega^2 \tag{2.45}$$

Berikut ini akan ditentukan semua matriks dan vektor yang ditunjukkan oleh persamaan di atas. Matriks inersia sistem terhadap *H-frame* M_H sama dengan matriks inersia sistem terhadap B-*frame*, dan ditentukan seperti pada Persamaan 2.46.

$$M_H = M_B = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{XX} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{YY} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{ZZ} \end{bmatrix} \tag{2.46}$$

Namun, matriks sentripetal Coriolis terhadap *H-frame* $C_H(\zeta)$ tidak sama dengan matriks sentripetal Coriolis terhadap *B-frame* dan ditentukan seperti pada Persamaan 2.47.

$$C_H(\xi) = \begin{bmatrix} 0_{3x3} & 0_{3x3} \\ 0_{3x3} & -S(I\ \omega_B) \end{bmatrix} \tag{2.47}$$

Vektor gravitasi terhadap H-*frame* G_H dituliskan pada Persamaan 2.48. Persamaan tersebut menunjukan bahwa gravitasi mempengaruhi ketiga persamaan linear, namun lebih berpengaruh terhadap ketinggian quadcopter.

$$G_H = \begin{bmatrix} F_G^E \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.48}$$

Efek giroskopis yang dihasilkan oleh putaran *propeller* tidak berubah karena hanya mempengaruhi persamaan angular yang mengacu kepada *B-frame*. Oleh karena itu, matriks giroskopis terhadap *H-frame* $O_H(\boldsymbol{\zeta})$ dibuat sama dengan Persamaan 2.49.

$$O_H(\zeta)\Omega = O_B(v)\,\Omega = \begin{bmatrix} 0_{3x1} \\ J_{TP}\begin{bmatrix} -q \\ p \\ 0 \end{bmatrix}\Omega \end{bmatrix} = J_{TP}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix}\Omega \tag{2.49}$$

Matriks perpindahan terhadap *H-frame* $E_H(\boldsymbol{\xi})$ berbeda dengan perpindahan terhadap *B-frame* karena input U_1 mempengaruhi semua persamaan linear dengan matriks rotasi $R_{\boldsymbol{\Theta}}$. Perkalian produk antara matriks perpindahan dengan kuadrat kecepatan propeler ditunjukkan pada Persamaan (2.50).

$$E_H(\xi)\Omega^2 = \begin{bmatrix} R_\Theta & 0_{3x3} \\ 0_{3x3} & I_{3x3} \end{bmatrix} E_B\Omega^2 = \begin{bmatrix} (s_\psi s_\phi + c_\psi s_\theta c_\phi)U_1 \\ (-c_\psi s_\phi + s_\psi s_\theta c_\phi)U_1 \\ (c_\theta c_\phi)\,U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{2.50}$$

Dengan menyusun kembali Persamaan (2.45), dapat ditemukan rumus untuk turunan vektor kecepatan yang digeneralisasi terhadap *H-frame* yang dapat dilihat pada persamaan (2.51).

$$\dot{\zeta} = M_H{}^{-1}\left(-C_H(\zeta)\zeta + G_H + O_H(\zeta)\,\Omega + E_H(\zeta)\Omega^2\right) \quad (2.51)$$

Persamaan 2.52 adalah model matematika dari quadcopter terhadap *earth frame (E-Frame)*

$$\begin{cases} \ddot{X} = \frac{U_1}{m}(\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi) \\ \ddot{Y} = \frac{U_1}{m}(-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi) \\ \ddot{Z} = -g + \frac{U_1}{m}(\cos\theta\cos\phi) \\ \dot{p} = \frac{I_{YY}-I_{ZZ}}{I_{XX}} q\,r - \frac{J_{TP}}{I_{XX}} q\,\Omega + \frac{U_2}{I_{XX}} \\ \dot{q} = \frac{I_{ZZ}-I_{XX}}{I_{YY}} p\,r - \frac{J_{TP}}{I_{YY}} p\,\Omega + \frac{U_3}{I_{YY}} \\ \dot{r} = \frac{I_{XX}-I_{YY}}{I_{ZZ}} p\,q - \frac{U_4}{I_{ZZ}} \end{cases} \quad (2.52)$$

2.2 Kontroler Quadcopter

Bermacam - macam kontroler dapat digunakan dalam mengendalikan quadcopter. Mulai dari kontroler cerdas, kontrol optimal, kontrol adaptif dan kontroler konvensional (P,I,D). Dibawah ini akan dijelaskan kontroler yang nantinya akan digunakan untuk mengendalikan quadcopter pada Tugas Akhir ini.

2.2.1 Kontroler PD[5]

Salah satu kontroler yang digunakan dalam quadcopter ini adalah kontroler PD. Kontroler PD adalah penggabungan antara kontroler tipe proporsional ditambah dengan kontroler tipe differensial. Sinyal kealahan $e(t)$merupakan masukan dalam kontroler sedangkan keluaran berupa sinyal kontrol $u(t)$. Hubungan antara masukan dan keluaran dalam kontroler ini adalah:

$$u(t) = K_p\left\{e(t) + \tau d\frac{de(t)}{dt}\right\} \quad (2.53)$$

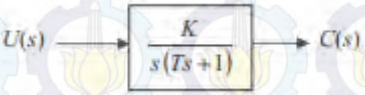
Jika dalam transformasi *laplace*

$$U(s) = K_p(1 + \tau_d s)E(s) \quad (2.54)$$

Dimana K_p adalah penguat proporsional dan τ_d adalah waktu differensial.

Kontroler proporsional ditambah differensial sangat cocok diterapkan pada plant orde 2 tanpa *delay* yang salah satu kutup *loop* tertutupnya terletak di pusat koordinat. Suatu plant orde 2 yang salah

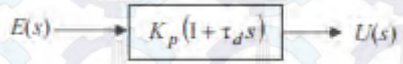
satu kutup *loop* tertutupnya terletak pada kutup dapat direpresentasikan pada diagram blok berikut ini:



Gambar 2. 12 Diagram blok plant orde 2 tanpa delay

dimana K: Gain overall ; T: konstanta.

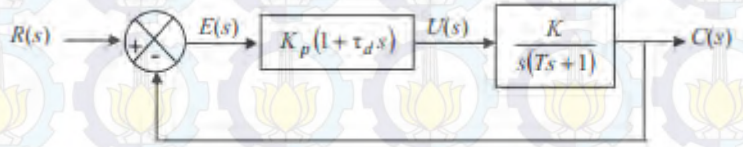
Sedangkan diagram blok dari kontroler proporsional ditambah differensial adalah sebagai berikut:



Gambar 2. 13 Kontroler tipe PD

Dimana Kp: penguatan proporsional, τ_d : waktu differensial.

Jika kontroler proporsional ditambah differensial diterapkan pada plant orde kedua tanpa *delay* yang salah satu kutub *loop* tertutupnya terletak di pusat koordinat dalam suatu sistem pengaturan maka diagram bloknya adalah sebagai berikut :



Gambar 2. 14 Diagram blok plant dan kontroler tipe PD

Sistem pengaturan di atas memiliki fungsi alih *loop* tertutup seperti persamaan 2.55:

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K_p(1+\tau_d s)\left(\frac{K}{s(Ts+1)}\right)}{1+K_p(1+\tau_d s)\left(\frac{K}{s(Ts+1)}\right)} \qquad (2.55)$$

Jika dipilih τ_d = T maka persamaan 2.55 akan menjadi persamaan 2.56:

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K_p.{}^{K}/_{s}}{1+K_p.{}^{K}/_{s}} = \frac{1}{\frac{1}{K.K_p}s+1} \qquad (2.56)$$

Tampak bahwa suatu plant orde kedua tanpa delay yang salah satu kutub loop tertutupnya terletak di pusat koordinat dengan kontroler

proporsional ditambah differensial menghasilkan sistem orde pertama (model yang diinginkan) dengan fungsi alih sebagai berikut

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K^*}{\tau^* s+1} \tag{2.57}$$

Dimana τ* dan K* masing-masing adalah konstanta waktu dan gain overall dari sistem hasil (model yang diinginkan) Dengan membandingkan persamaan (2.56) dan persamaan (2.57), diperoleh :

$$\tau^* = \frac{1}{K.K_p} \tag{2.58}$$

Dimana $K^* = 1$

Masukan unit step, r(t) = u(t) → R(s) = 1/s maka besarnya error steady state dari sistem hasil adalah :

$$e_{ss} = R_{ss} - C_{ss} \tag{2.59}$$

dimana *Css* adalah keluaran sistem hasil pada keadaan tunak dan *Rss* adalah masukan sistem hasil pada keadaan tunak yang besarnya adalah

$$C_{ss} = \lim_{s\to 0} sC_{(s)} = \lim_{s\to 0} s\left(\frac{1}{s(\tau^* s+1)}\right) = 1 \tag{2.60}$$

$$R_{ss} = \lim_{s\to 0} sR_{(s)} = \lim_{s\to 0} s\left(\frac{1}{s}\right) = 1 \tag{2.61}$$

$$e_{ss} = 1 - 1 = 0 \tag{2.62}$$

2.2.2 Kontroler PID[5]

PID adalah penggabungan antara kontroler tipe proporsional, ditambah dengan kontroler tipe differensial dan ditambah dengan kontroler tipe integral. Sinyal kealahan $e(t)$merupakan masukan dalam kontroler sedangkan keluaran berupa sinyal kontrol $u(t)$. Hubungan antara masukan dan keluaran dalam kontroler ini adalah:

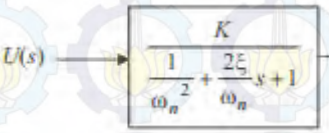
$$u(t) = K_p\left\{e(t) + \frac{1}{\tau_i}\int_0^t e(t)dt + \tau d\frac{de(t)}{dt}\right\} \tag{2.63}$$

Jika dalam transformasi *laplace*

$$U(s) = K_p\left(1 + \frac{1}{\tau_i} + \tau_d s\right)E(s) \tag{2.64}$$

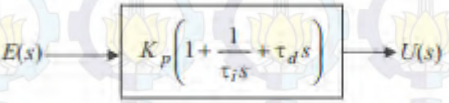
Dimana K_p adalah penguat proporsional, τ_i adalah waktu integral dan τ_d adalah waktu differensial.

Suatu plant orde kedua tanpa delay dapat direpresentasikan dalam bentuk diagram blok seperti berikut ini :



Gambar 2. 15 Diagram blok plant orde 2 tanpa delay

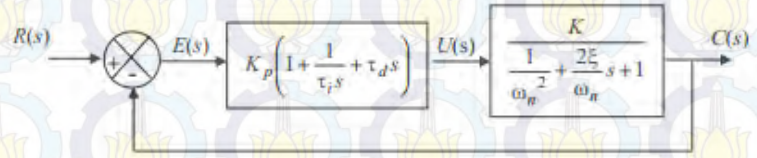
Dimana K adalah Gain overall, ωn adalah frekuensi alami tak teredam dan ξ adalah rasio peredaman Sedangkan diagram blok dari kontroler proporsional ditambah integral ditambah differensial adalah sebagai berikut :



Gambar 2. 16 Kontroler PID

K_p: penguatan proporsional ; τ_i : waktu integral ; τ_d : waktu differensial

Jika kontroler proporsional ditambah integral ditambah differensial diterapkan pada plant orde kedua tanpa delay dalam suatu sistem pengaturan maka diagram bloknya adalah sebagai berikut :



Gambar 2. 17 *Plant* orde 2 dengan kontroler PID

Sistem pengaturan diatas memiliki fungsi alih *loop* tertutup seperti persamaan 2.65

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K_p\left(\frac{\tau_i\tau_d s^2+\tau_i s+1}{\tau_i s}\right)\left(\frac{K}{1/\omega_n{}^2+2\zeta/\omega_n s+1}\right)}{1+K_p\left(\frac{\tau_i\tau_d s^2+\tau_i s+1}{\tau_i s}\right)\left(\frac{K}{1/\omega_n{}^2+2\zeta/\omega_n s+1}\right)} \quad (2.65)$$

Jika dipilih $\tau_i = \tau_d = {}^{1}/_{\omega_n{}^2}$ dan $\tau_i = {}^{2\zeta}/_{\omega_n}$ maka persamaan 2.65 akan menjadi persamaan 2.66

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K_p.{}^{K}/_{\tau_i s}}{1+K_p.{}^{K}/_{\tau_i s}} = \frac{1}{\frac{\tau_i s}{K.K_p}+1} \tag{2.66}$$

Tampak bahwa suatu plant orde kedua tanpa *delay* dengan kontroler proporsional ditambah integral ditambah differensial menghasilkan sistem orde pertama (model yang diinginkan) dengan fungsi alih seperti persamaan 2.67.

$$\frac{C_{(s)}}{R_{(s)}} = \frac{K^*}{\tau^* s+1} \tag{2.67}$$

Dimana τ^* dan K^* masing-masing adalah konstanta waktu dan gain *overall* dari sistem hasil (model yang diinginkan).
Dengan membandingkan persamaan 2.66 dan persamaan 2.67, diperoleh nilai τ^* dan K^* :

$$\tau^* = \frac{\tau_i}{K.K_p} \tag{2.68}$$

$$K^* = 1 \tag{2.69}$$

Masukan unit step, r(t) = u(t) → R(s) = 1/s maka besarnya error steady state dari sistem hasil adalah :

$$e_{ss} = R_{ss} - C_{ss} \tag{2.70}$$

dimana *Css* adalah keluaran sistem hasil pada keadaan tunak dan *Rss* adalah masukan sistem hasil pada keadaan tunak yang besarnya adalah

$$C_{ss} = \lim_{s\to 0} sC_{(s)} = \lim_{s\to 0} s\left(\frac{1}{s(\tau^* s+1)}\right) = 1 \tag{2.71}$$

$$R_{ss} = \lim_{s\to 0} sR_{(s)} = \lim_{s\to 0} s\left(\frac{1}{s}\right) = 1 \tag{2.72}$$

$$e_{ss} = 1 - 1 = 0 \tag{2.73}$$

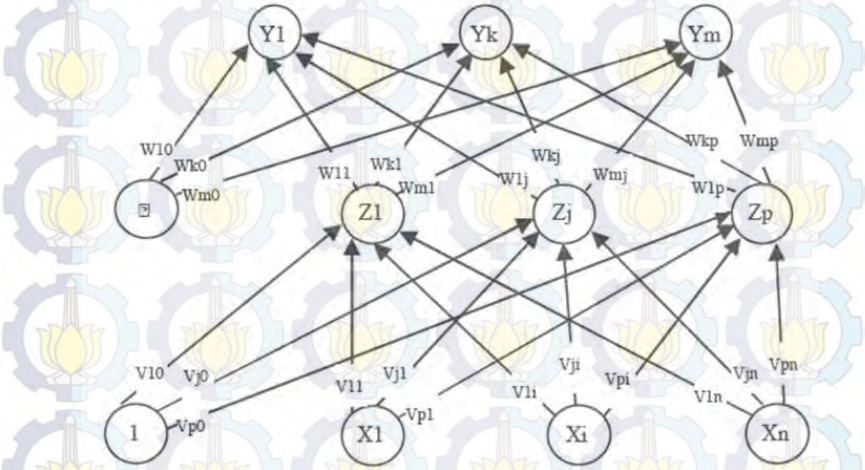
2.2.3 Jaringan Syaraf Tiruan[4]

Jaringan syaraf tiruan (JST) atau pada umumnya disebut *neural network* (NN) adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan syaraf manusia. JST merupakan sistem adaptif yang dapat merubah strukturnya untuk memecahkan masalah berdasarkan informasi ekternal maupun internal yang mengalir melalui jaringan tersebut.

Secara sederhana, JST adalah sebuah pemodelan data statistik non-linier. JST dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data.

Salah satu model Jaringan Syaraf Tiruan adalah *Back-propagation* .Seperti halnya model JST yang lain, *Backpropagation* melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respon yang benar terhadap pola masukan yang serupa walaupun tidak sama dengan pola yang dipakai selama pelatihan.

Arsitektur backpropagation dengan n buah masukan (ditambah satu bias), dan sebuah layar tersembunyi yang terdiri dari p unit (ditambah satu bias), dan juga m buah unit keluaran, terdapat pada gambar 2.18



Gambar 2. 18 Arsitektur JST

Gambar 2.18 dapat dijelaskan bahwa V_{ji} adalah bobot garis dari unit masukan X_i ke unit layar tersembunyi Z_j (V_{j0} adalah bobot garis yang menghubungkan bias di unit masukan ke unit layar tersembunyi Z_j). Sedangkan W_{kj} adalah bobot dari unit layer tersembunyi Z_j ke unit keluaran Y_k (W_{k0} adalah bobot dari bias dilayer tersembunyi ke unit keluaran Y_k).

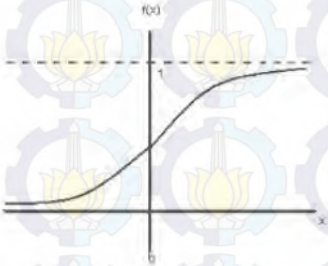
Fungsi aktivasi dalam *backpropagation* yang dipakai harus memenuhi beberapa syarat yaitu: kontinyu, terdiferensial dengan mudah dan merupakan fungsi yang tidak turun. Salah satu fungsi aktivasi adalah fungsi sigmoid biner yang memiliki range (0,1).

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.74}$$

dengan turunan

$$f'(x) = f(x)\big(1 - f(x)\big) \tag{2.75}$$

Grafik fungsinya tampak pada gambar 2.19.



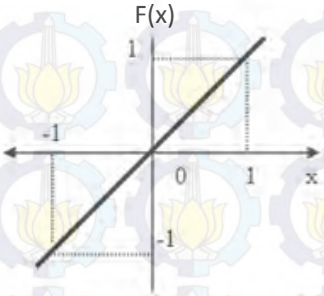
Gambar 2. 19 Fungsi aktifasi sigmoid biner

Fungsi aktivasi yang sering digunakan selain fungsi aktivasi sigmoid adalah fungsi aktivasi linier. Fungsi linier akan membawa nilai input sebanding dengan nilai output. fungsi ini digambarkan sebagai berikut:

$$f(x) = \lambda x \tag{2.76}$$

Dengan fungsi turunan

$$f'(x) = \lambda \tag{2.77}$$



Gambar 2. 20 Fungsi aktivasi linier

Pelatihan *backpropagation* meliputi 3 fase. Fase pertama adalah fase maju atau perhitungan maju. Fase kedua adalah propagasi mundur. Fase ketiga adalah perubahan nilai bobot.

Berikut adalah langkah – langkah melakukan perhitungan JST.

a) Langkah 0: inisialisasi semua bobot dengan bilangan acak kecil.

b) Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2-9.

c) Langkah 2: Setiap pasang data pelatihan, lakukan langkah 3-8.

Fase 1: Perhitungan Maju.

d) Langkah 3: Tiap unit masukan menerima sinyal dan meneruskannya ke unit tersembunyi diatasnya.

e) Langkah 4: Hitung semua keluaran di unit tersembunyi $Z_j (j = 1,2, \dots . p)$

$$z\_net_j = v_{j0} + \sum_{i=1}^{n} x_i \, v_{ji} \quad (2.78)$$

$$z_j = \text{f}(z\_net_j) = \frac{1}{1+e^{-z\_net_j}} \quad (2.79)$$

f) Langkah 5: Hitung semua keluaran jaringan di unit $y_k (k = 1,2, \dots . m)$

$$y\_net_k = w_{k0} + \sum_{j=1}^{p} z_j \, w_{kj} \quad (2.80)$$

$$y_k = \text{f}(k\_net_j) = \frac{1}{1+e^{-y\_net_k}} \quad (2.81)$$

Fase 2 : Perhitungan Mundur.

g) Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran $y_k(k = 1,2, \dots . m)$

$$\delta_k = (t_k - y_k) f'(y_{net_k}) = (t_k - y_k) y_k (1 - y_k) \quad (2.82)$$

δ_k merupakan unit kesalahan yang akan dipakai dalam perubahan bobot layar pada langkah 7.

Menghitung suku perubahan bobot w_{kj} (yang akan dipakai nanti untuk merubah bobot w_{kj}) dengan laju percepatan α.

$$\Delta w_{kj} = \alpha \, \delta_k \, z_j \; ; \; k = 1,2, \dots , m \; ; j = 0,1, \dots . p \quad (2.83)$$

h) Langkah 7: Hitung δ unit tersembunyi berdasarkan kesalahan disetiap unit tersembunyi $Z_j (j = 1,2, \dots . p)$

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k \, w_{kj} \quad (2.84)$$

Faktor δ unit tersembunyi :

$$\delta_j = \delta\_net_j \, f'(z_{net_j}) = \delta\_net_j z_j (1 - z_j) \quad (2.85)$$

Hitung suku perubahan bobot v_{ji} (yang akan dipakai nanti untuk merubah bobot v_{ji}.

$$\Delta v_{ji} = \alpha \, \delta_j \, x_i \; ; \; j = 1,2, \dots , p \; ; i = 0,1, \dots . n \quad (2.86)$$

i) Langkah 8: Hitung semua perubahan bobot. Perubahan bobot baris yang menunju ke unit keluaran:

$$w_{kj}(baru) = w_{kj}(lama) + \Delta w_{kj}$$

$$(k = 1,2, \dots . m \; ; j = 0,1, \dots p) \quad (2.87)$$

Perubahan bobot baris yang menuju ke unit tersembunyi:

$$v_{ji}(baru) = v_{ji}(lama) + \Delta v_{ji}$$

$$(j = 1,2, \dots , p \; ; i = 0,1, \dots . n) \quad (2.88)$$

Setelahpelatihan selesai dilakukan, jaringan dapat dipakai untuk pengenalan pola. Dalam hal ini, hanya propagasi maju (langkah 4 dan 5) saja yang dipakai untuk menentukan keluaran jaringan.

2.3 Linierisasi[6]

Beberapa hubungan diantara besaran-besaran fisik adalah tidak benar-benar linear, sekalipun sering didekati dengan persamaan - persamaan linear terutama untuk penyederhanaan matematik. Penyederhanaan ini berlaku jika jawaban persoalan yang diperoleh sesuai dengan hasil eksperimen. salah satu karakteristik yang paling penting dari sistem nonlinear adalah ketergantungan perilaku respons sistem pada besaran dan jenis masukan.

Sistem nonlinear berbeda dari sistem linear terutama dalam hal prinsip superposisi yang tidak berlaku pada sistem nonlinear. Sistem nonlinear menunjukkan beberapa fenomena yang tidak dipunyai oleh sistem linear, sehingga dalam penyelidikan sistem semacam ini kita harus mengenal fenomena-fenomena yang ditunjukkan oleh sistem nonlinear.

Perkembangan sistem nonlinear dapat didekati dengan sistem linear dengan teori linearisasi. Teori linearisasi digunakan untuk mengatasi permasalahan ketidaklinearan sistem, sehingga sistem yang nonlinear dapat didekati dengan sistem linear. Teknik yang digunakan untuk linearisai pada Tugas Akhir ini menggunakan matriks Jacobi.

Cara untuk mendapatkan model matematika linear dari sebuah sistem nonlinear, diasumsikan variabel-variabel sistem memiliki penyimpangan sangat kecil pada titik operasi tertentu. Masukan sistem dinyatakan dengan $x(t)$ dan keluaran sistem dinyatakan dengan $y(t)$, hubungan antara $y(t)$ dan $x(t)$ adalah:

$$y = f(x) \tag{2.89}$$

Diibaratkan titik operasi normal adalah $\bar{x}$ dan $\bar{y}$, dengan pendekatan linear $f(x)$ disekitar $\bar{x}$.

$$y - \bar{y} = f(x) = f(\bar{x}) = f(x - \bar{x}) \tag{2.90}$$

di mana penurunan dari Persamaan 2.89 dapat dituliskan dengan persamaan berikut:

$$\delta y = f(\delta x) = \frac{df(x)}{dx}\Big|_{x=\bar{x}} \delta x \tag{2.91}$$

Jika titik ekuilibrium sistem berada pada titik *origin* dan sistem dilinearisasi pada titik ekuilibrium, maka $f(\bar{x}) = 0$. Dengan demikian, Persamaan 2.91 menjadi:

$$y = f(x) = \frac{df(x)}{dx}\Big|_{x=\bar{x}} x \tag{2.92}$$

Sistem nonlinear $\dot{x} = f(\bar{x})$ merupakan permisalan dari nilai $f(x)$, sehingga persamaan 2.92 menjadi:

$$f(x) = \dot{x} = \frac{df(x)}{dx}\Big|_{x=\bar{x}} x \tag{2.93}$$

Sistem nonlinear *loop* terbuka dengan jumlah *state* sama dengan *n*, maka linearisasi lokal sistem pada titik ekuilibrium $\overline{x_1}, \overline{x_2}, \dots, \overline{x_n}$, dapat dijabarkan dari Persamaan 2.90 sebagai berikut:

$$\begin{bmatrix} \dot{x} \\ \vdots \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1(x_1, \dots, x_n)}{\partial x_1} & \cdots & \dfrac{\partial f_1(x_1, \dots, x_n)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n(x_1, \dots, x_n)}{\partial x_1} & \cdots & \dfrac{\partial f_n(x_1, \dots, x_n)}{\partial x_n} \end{bmatrix}_{\substack{x_1=\overline{x_1} \\ \vdots \\ x_n=\overline{x_n}}} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\dot{x} = Ax$$

Dengan (2.94)

$$\dot{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \; dan \; A = \begin{bmatrix} \dfrac{\partial f_1(x_1, \dots, x_n)}{\partial x_1} & \cdots & \dfrac{\partial f_1(x_1, \dots, x_n)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_n(x_1, \dots, x_n)}{\partial x_1} & \cdots & \dfrac{\partial f_n(x_1, \dots, x_n)}{\partial x_n} \end{bmatrix}_{\substack{x_1=\overline{x_1} \\ \vdots \\ x_n=\overline{x_n}}}$$

Persamaan 2.94 menunjukkan bahwa sistem nonlinear dapat didekati menjadi sistem linear di sekitar titik ekuilibriumnya dengan s *A* merupakan matriks Jacobi dari *f*(*x*). Dimisalkan sistem nonlinear dinyatakan dalam persamaan *state* berikut:

$$\dot{x} = f(x) + h(x, u) \tag{2.95}$$

dengan jumlah *state* sama dengan *n*, maka linearisasi lokal sistem pada titik ekuilibrium $\overline{x_1}, \overline{x_2}, \dots, \overline{x_n}, \bar{u}$, maka Persamaan 2.95 menjadi:

$$\dot{x} = Ax + Bu + Err(\bar{x}, \bar{u}) \tag{2.96}$$

$$Err(\bar{x},\bar{u}) = f(\bar{x}) + h(\bar{x},\bar{u}) + A(-\bar{x}) + B(-\bar{u}) \quad (2.97)$$

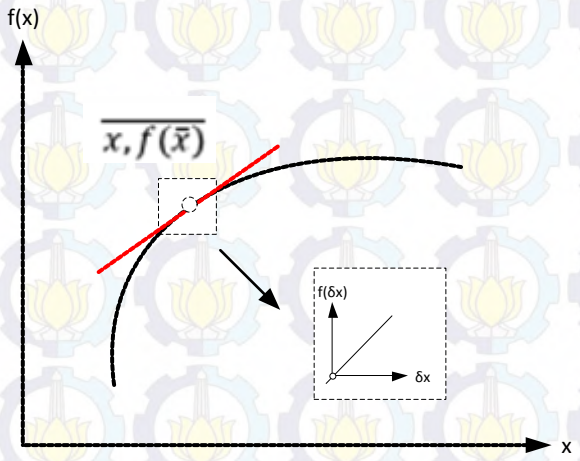
dengan mengabaikan nilai *error* pada Persamaan 2.96 dan 2.97, maka diperoleh
model linear sebagai berikut

$$\dot{x} = Ax + Bu \quad (2.98)$$

Dengan

$$x = [x_1 \quad \dots \quad x_n]^T$$

$$A = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \dots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \dots & \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}_{x=\bar{x}} \quad dan \quad B = \begin{bmatrix} \frac{\partial h_1(x,u)}{\partial u} \\ \vdots \\ \frac{\partial h_n(x,u)}{\partial n} \end{bmatrix}$$



Gambar 2. 21 Linierisasi lokal y=f(x) disekitar titik $\bar{x}$

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Bab ini akan dibahas mengenai perancangan-perancangan sistem dari quadcopter yang terdiri atas spesifikasi sistem, identifikasi kebutuhan, desain mekanik, desain elektronik, *ground station*, model matematik, identifikasi konstanta, perancangan kontroler, dan simulasi dari sistem yang akan dibuat.

3.1 Spesifikasi Sistem

Quadcopter sebagai pesawat tanpa awak yang dapat bergerak dengan 6 DOF (*degree of freedom*) terdiri dari 3 DOF rotasi dan 3 DOF translasi. Sistem akan dibuat berdasarkan spesifikasi sistem tertentu secara *hardware* maupun simulasi pada *software* MATLAB.Spesifikasi sistem yang diinginkan dapat diuraikan sebagai berikut:

a) Quadcopter dapat bergerak secara vertikal.
b) Quadcopter dapat mempertahankan posisi *hover* walaupun diberikan gangguan ekternal.
c) Quadcopter dapat digerakkan secara manual dengan menggunakan *remote control*.
d) Data-data sensor dan aktuator dikirimkan ke *ground station*.

3.2 Identifikasi Kebutuhan

Sistem quadcopter yang dibuat supaya memenuhi spesifikasi yang diharapkan maka terdapat kebutuhan yang harus dipenuhi antara lain:

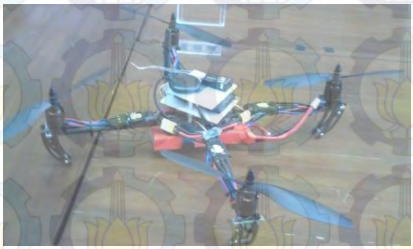
a) Quadcopter harus dibuat simetris dan ringan untuk menghindari kelembaman yang sangat besar.
b) Baterai yang digunakan minimal dapat membuat quadcopter terbang sekitar 15 menit.
c) Rangkaian mikrokontroler memiliki memori dan *port I/O* yang cukup untuk sensor, aktuator dan komunikasi.
d) Data-data selama terbang dapat dikirimkan ke *ground station*.

3.3 Desain Mekanik

Desain mekanik dari quadcopter harus dibuat simetris dan cukup ringan. *Frame* quadcopter yang digunakan adalah berbentuk *plus* (+) Beberapa contoh dari desain mekanik quadcopter disajikan pada Gambar 3.1.

Desain mekanik yang dibangun untuk implementasi kestabilan saat *hover* pada penelitian Tugas Akhir ini ditunjukkan pada Gambar 3.2.

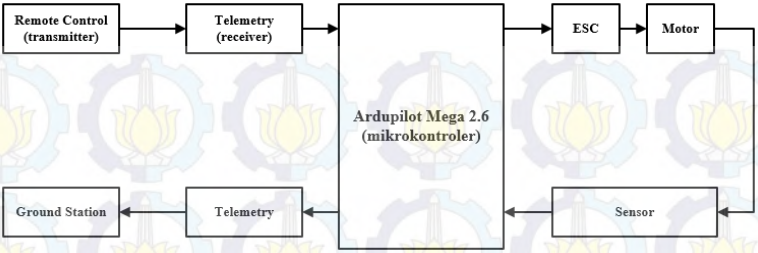
Gambar 3. 1 Contoh desain quadcopter



Gambar 3. 2 Desain quadcopter yang dibuat

3.4 Desain Elektronik

Board kontroler yang dipakai adalah Ardupilot Mega 2.6 yang sudah dilengkapi beberapa sensor dan *port* yang memudahkan untuk penggunaan sebagai komunikasi data. *Board* Ardupilot Mega 2.6 ini dilengkapi dengan ATMega 2560 sebagai mikrokontroler dari *flight controller* tersebut.



Gambar 3. 3 Perancangan sistem elektronika quadcopter

3.4.1 Sensor Gyro dan Accelerometer[1]

Sensor *gyro* dan *accelerometer* akan digunakan secara bersamaan untuk menghitung sudut yang berotasi pada sumbu x, y, dan z. Sensor *accelerometer* dan *gyro* yang digunakan adalah GY80 . Spesifikasinya adalah:

- Batas operasi : ±3,6 g,
- Sumber tegangan : 5V dan 3,3 V
- Sensitivitas : 300 mV/g,

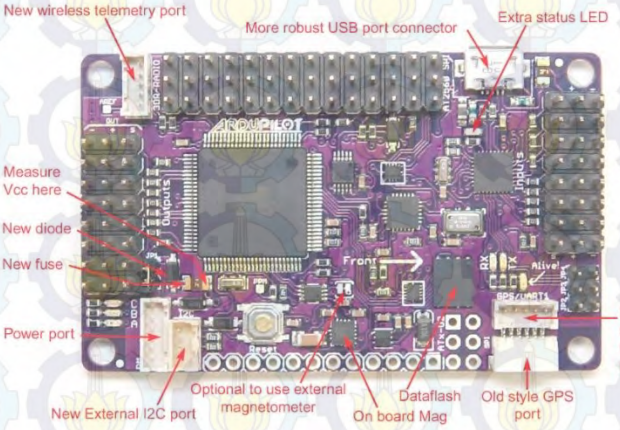
Gambar 3. 4 Sensor GY-80

3.4.2 Ardupilot mega.

Ardupilot Mega merupakan *board* yang mudah digunakan pada quadcopter. *Board ini* terdiri atas sensor *gyro* dan *accelerometer*, serta Arduino Mega 2560 sebagai mikrokontrolernya. Arduino Mega berfungsi untuk memproses sinyal dengan program instruksi yang ada. Secara umum APM terdiri atas sambungan I/O, memori internal, dan beberapa moda lain seperti ADC, *Timer/Counter* dan lainnya.

Mikrokontroler yang digunakan adalah mikrokontroler ATMega 2560. Mikrokontroler akan diprogram menggunakan bahasa C. Mikrokontroler akan dipadukan menjadi modul Ardupilot Mega dalam satu *board* seperti pada Gambar 3.5 Spesifikasi dari mikrokontroler ATMega 2560 adalah:

- Memori : 256KByte
- Frekuensi *Clock* : Maksimum 16Mhz
- I/O : 86 *Port*
- ADC : 10 bit/16 *port*
- *Timer* 8 bit : 2 buah
- *Timer* 16 Bit : 4 buah
- Supplai tegangan : 4,5-5,5 VDC



Gambar 3. 5 Ardupilot Mega 2.6

3.4.3 Transmitter dan Receiver Radio

Transmitter dan *receiver* digunakan untuk operasi manual melakukan manuver pada quadcopter. *Transmitter* berada pada sisi pengguna untuk memberikan nilai input kepada quadcopter, sinyal tersebut akan diterima *receiver* dan diteruskan ke mikrokontroler berupa pulsa. Tampilan *transmitter* dan *receiver* radio seperti pada Gambar 3.6 Spesifikasi dari radio *transmiter* adalah:

- Nama *Transmitter* : FlySky FS-TH9XB,
- *Channel* : 9 buah,

- Modulasi : 2,4GHz,
- Baterai : HK 1500mAh Li-Fe.

Disamping itu, spesifikasi untuk *receiver* adalah:

- Nama *receiver* : FlySky FS-R8B 8CH,
- Ukuran modul utama : 35mm x 25mm x 8mm,
- Berat : 25 gram,
- *Range* tegangan : 3,2-9,6V,
- Arus : 75mA,

Gambar 3. 6 *Transmitter* dan *receiver*

3.4.4 Electronic Speed Controller (ESC)

Alat ini merupakan *driver* dari motor brushless DC. ESC di-*trigger* oleh sinyal PWM yang dikendalikan oleh mikrokontroler ATMega 2560. Sinyal PWM tersebut akan men-*drive* motor dengan kecepatan yang linier dengan besar pulsa yang diberikan.

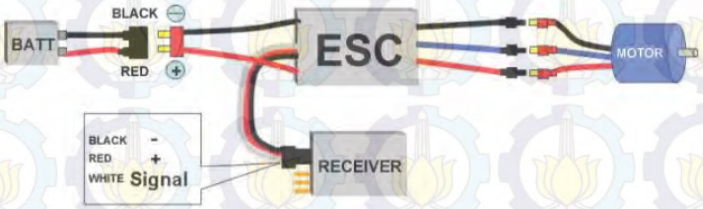
Jenis ESC yang digunakan adalah Hobby King 30A ESC 3A UBEC yang dapat dialiri arus hingga 30 Ampere, seperti pada Gambar 3.7. Quadcopter membutuhkan respon yang cepat untuk mengatur putaran motor agar lebih stabil. Spesifikasi ESC tersebut adalah:

- Arus : 30 Ampere,
- Arus *Brust* : 40 Ampere,
- BEC : 5 Volt / 3 Ampere,
- Lipo Cells : 2 – 4 buah,

- NiMH : 5 – 12 buah,
- Massa : 32 gram,
- Ukuran : 54x26x11 milimeter,
- *Firmware* : *Simon-K Firmware*,

Gambar 3. 7 *Electronic speed controller*

Pemasangan ESC ke motor *brushless* memiliki cara tersendiri. Gambar 3.8 akan menampilkan cara memasangan ESC pada quadcopter.



Gambar 3. 8 Pemasangan ESC

Gambar 3.8 menjelaskan bahwa *supply* dari baterai memiliki kabel merah dan hitam, masing – masing disambungkan ke kabel merah (+) dan hitam (-) pada sumber ESC. ESC memiliki 3 jenis kabel kontroler. Kabel merah dan hitam secara berurutan dipasang ke positif dan *ground* pin *output* APM, sedangkan untuk kabel putih disambungkan ke kabel data pin *output* APM. Kabel *output* ESC yang biru disambung ke kabel biru motor *brushless*, ini digunakan untuk pengiriman sinyal data. Pemasangan kabel merah dan hitam terserah, hal yang membedakan dari pemasangan kabel itu hanyalah arah putaran dari motor.

3.4.5 Motor BLDC dan Propeller

Motor yang digunakan sebagai aktuator pada implementasi adalah motor DC *brushless* tanpa sikat *SunnySky V2216 KV900* yang porosnya dipasang *propeller* berukuran 10x4,5. Bentuk motor *brushless* tanpa sikat dan *propeller* terlihat seperti pada Gambar 3.9

Gambar 3. 9 Motor *brushless* dan *propeller*

3.5 Ground Station

Data dari mikrokontroler dikirimkan secara *wireless* menuju komputer *ground station* dengan komunikasi serial menggunakan *telemetery* radio 433Mhz. Data serial langsung dibaca pada *software* Mission Planner pada ground station.

Gambar 3. 10 *Ground station* dan *telemetry*

3.6 Identifikasi Konstanta

Model matematik yang diperoleh dengan pemodelan fisis pada bab ke dua, terlihat bahwa dibutuhkan konstanta-konstanta yang terdapat pada sistem dan menentuan parameter-parameter kontroler agar model

matematik bisa digunakan pada simulasi. Beberapa parameter yang harus diperoleh antara lain:

a) Massa

Massa keseluruhan quadcopter diukur menggunakan alat ukur massa (timbangan). Hasil pengukuran diperoleh massa total quadcopter adalah 1,26 Kg.

b) Panjang lengan quadcopter

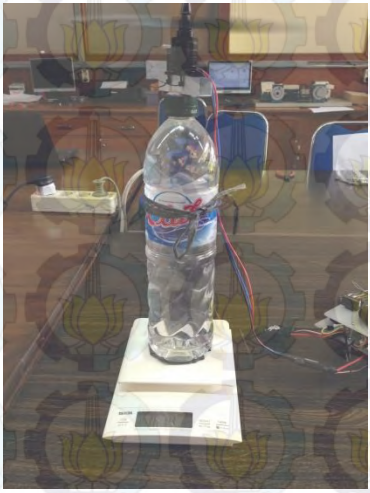
Panjang lengan quadcopter diukur menggunakan penggaris. Panjang lengan diukur dengan cara mengukur jarak dari pusat massa quadcopter (berada di tengah – tengah quadcopter) sampai titik pusat motor *brushless*. Hasil pengukurannya adalah 0,206 meter.

c) Momen inersia

Momen inersia quadcopter didapat dengan identifikasi parametrik *input output* sistem dengan cara menerbangkan quadcopter lalu didapatkan data keluaran dari sensor seperti kecepatan sudut, nilai prosentase *throttle* dari *remote control.*

d) Pemetaan gaya angkat terhadap pulsa

Percobaan ini digunakan untuk mendapatkan data pengukuran dari tiap – tiap motor *brushless*. Pengukuran dilakukan dengan cara seperti gambar 3.11 dan 3.12.



Gambar 3. 11 Pengukuran gaya angkat pada motor

Pengukuran seperti gambar 3.11 adalah pengukuran untuk menentukan gaya angkat pada setiap motor. Cara yang dilakukan dengan memberikan sinyal pwm pada ESC. Sinyal masukan PWM kemudian dinaikan sehingga kecepatan putar motor akan meningkat, akibatnya berat beban yang terbaca pada alat ukur timbangan akan berkurang. Selisih berat beban itulah yang nantinya sebagai data untuk menentukan konstanta *thrust*.

Pengukuran pada gambar 3.12 adalah pengukuran yang digunakan untuk mengetahui kecepatan motor *brushless* terhadap sinyal masukan PWM. Saat sinyal PWM dinaikan maka kecepatan putar motor juga meningkat. Disaat yang sama alat ukur rpm meter digunakan untuk mengetahui kecepatan putar motor. Hasil pengukuran ini juga digunakan untuk menentukan konstanta *thrust*.

Hasil pengukuran gaya angkat dan kecepatan motor *brushless* bisa dilihat pada tabel 3.1

Gambar 3. 12 Pengukuran kecepatan motor terhadap sinyal PWM

Tabel 3. 1 Hasil pengukuran gaya angkat dan kecepatan motor

Data (PWM)	Kecepatan Putar (RPM)	Berat (Gram)
75	1094	1618
85	1455	1602
95	2415	1534
105	2993	1473

Tabel 3. 2 Hasil pengukuran gaya angkat dan kecepatan motor (lanjutan)

Data (PWM)	Kecepatan Putar (RPM)	Berat (Gram)
115	3534	1411
125	4080	1328
135	4578	1242
145	5021	1150
155	5451	1056
165	5843	966
115	3534	1411

e) Konstanta thrust

Konstanta *thrust* didapat dari hasil perhitungan dengan menggunakan data – data hasil pengukuran gaya angkat dan kecepatan putar. Hasil perhitungan terdapat pada tabel 3.2.

$$b = \frac{\sum_1^n m.g}{\sum_1^n \Omega^2} \quad (3.1)$$

Dimana b: konstanta *thrust* (N S^2) ; m: massa (Kg); g: gaya grafitasi (ms^{-2}) ; Ω: kecepatan putar (rad s^{-1}); dan n adalah jumlah pengukuran.

f) Konstanta drag

Konstanta *drag* dihitung dengan persamaan gerak lurus berubah beraturan. Pengukuran konstanta *drag* dilakukan dengan mengambil data penerbangan quadcopter pada saat *take-off*. Hasil dari perhitungan konstanta *drag* terdapat pada tabel 3.2.

Tabel 3. 3 Hasil perhitungan konstanta *thrust* dan *drag*

Konstanta	**Nilai**	**Satuan**
Thrust	1,68198 x 10^{-5}	N S^2
Drag	4,19 x 10^{-6}	N m S^2

g) Model matematika

Model matematika yangdigunakan untuk identifikasi adalah model matematika rotasi, ini dikarenakan persamaan itu berhubungan langsung dengan konstanta – konstanta yang tidak diketahui pada quadcopter. Model matematika rotasi ditulis kembali pada persaam 3.2 demi menyederhanakan perhitungan.

$$\begin{aligned}\dot{p} &= a_1 qr - b_1 q\Omega + c_1 U_2 \\ \dot{q} &= a_2 pr + b_2 p\Omega + c_2 U_3 \\ \dot{r} &= a_3 pq + c_3 U_4\end{aligned} \tag{3.2}$$

Persamaan 3.2 diturunkan dari persamaan 2.52. Beberapa parameter didapatkatkan dengan menggunakan Pendekatan Penyelesaian Persamaan Simultan (P3LS). Hasil identifikasi dicantumkan pada tabel 3.3.

Tabel 3. 4 Tabel hasil identifikasi model matematik

Konstanta	Nilai
a_1	-0.5495
b_1	-0.0017
c_1	0.2052
a_2	0.1675
b_2	-0.0094
c_2	2.955
a_3	-2.0257
c_3	0.0594

3.7 Model Matematis Hasil Identifikasi Sistem

Setelah diperoleh konstanta-konstanta dari sistem, maka dapat dituliskan kembali model matematika dari sistem quadcopter adalah sebagai berikut:

$$\begin{aligned}X &= \frac{U_1}{1{,}26}(\sin\psi \sin\emptyset + \cos\psi \sin\theta \cos\emptyset) \\ Y &= \frac{U_1}{1{,}26}(-\cos\psi \sin\emptyset + \sin\psi \sin\theta \cos\emptyset) \\ Z &= \frac{U_1}{1{,}26}(\cos\theta \cos\emptyset) - 9{,}81\end{aligned}$$

$$\begin{aligned}\dot{p} &= -0{,}5495qr + 0{,}00017q\Omega + 0{,}2052U_2 \\ \dot{q} &= 0{,}1675pr - 0{,}0094p\Omega + 2{,}955U_3 \\ \dot{r} &= -2{,}0257pq + 0{,}0594U_4\end{aligned} \tag{3.3}$$

Dari identifikasi konstanta dilakukan pembatasan, di antaranya sudut *roll* dan *pitch* adalah maksimum 0,7 rad dan minimum -0,7 , sedangkan sudut *yaw* adalah maksimum 1 dan minimum -1.

3.8 Perancangan Kontroler

Setelah diperoleh model matematika dan konstanta quadcopter maka dapat dilakukan perancangan kontroler dengan mencari parameter-parameter kontroler yang diperlukan. Pencarian parameter – parameter kontroler dilakukan dengan cara linierisasi persamaan 3.3, ini dikarenakan karena *plant* quadcopter *non-linier* sedangkan kontroler yang digunakan tipe PID dan PD adalah tipe kontroler yang linier.

Proses linierisasi model matematika dimulai dengan menurunkan model matematika rotasi pada persamaan 3.3 terhadap fungsi kecepatan *roll*, *pitch*, dan *yaw* (p,q,r).

$$\dot{p} = -0{,}5495qr + 0{,}0017q\Omega + 0{,}2052U_2$$

$$\frac{\dot{p}}{p} = 0p$$

$$\frac{\dot{p}}{q} = -0{,}5495r + 0{,}0017\Omega$$

$$\frac{\dot{p}}{r} = -0{,}5495q \qquad (3.4)$$

$$\frac{\dot{p}}{U_2} = 0{,}2052$$

$$\dot{q} = 0{,}1675pr - 0{,}0094p\Omega + 2{,}955U_3$$

$$\frac{\dot{q}}{p} = 0{,}1675r - 0{,}0094\Omega$$

$$\frac{\dot{q}}{q} = 0q$$

$$\frac{\dot{q}}{r} = 0{,}1675p \qquad (3.5)$$

$$\frac{\dot{q}}{U_3} = 2{,}955$$

$$\dot{r} = -2{,}0257pq + 0{,}0594U_4$$

$$\frac{\dot{r}}{p} = -2{,}0257q$$

$$\frac{\dot{r}}{q} = -2{,}0257p$$

$$\frac{\dot{r}}{r} = 0r \tag{3.6}$$

$$\frac{\dot{r}}{U_4} = 0{,}0594$$

Setelah mendapatkan hasil linierisasi, persamaan 3.4, 3.5, dan 3.6 diubah dalam bentuk persamaan state.

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & -0{,}5495r - 0{,}0017\Omega & -0{,}5495q\\ 0{,}1675r - 0{,}0094\Omega & 0 & 0{,}1675p\\ -2{,}0257q & -2{,}0257p & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \tag{3.7}$$

Setelah dibentuk persamaan state seperti persamaan 3.7, kemudian masukkan titik – titik linierisasi yang ingin didapatkan. Nilai Ω=0 karena kita asumsikan disaat hover tidak ada selisih kecepatan dari keempat motor.

(p=0; q=0;r=0; Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \tag{3.8}$$

Persamaan state 3.8 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2}{s^3} \tag{3.9}$$

$$G_{22}(s) = \frac{2{,}955s^2}{s^3} \tag{3.10}$$

$$G_{33}(s) = \frac{0{,}0594s^2}{s^3} \tag{3.11}$$

Transfer function pada persamaan 3.9 ,3.10, 3.11, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$\begin{aligned} &G_{11}(s) \rightarrow K_p = 21{,}7838; K_i = 0{,}72684; K_d = -1{,}9259 \\ &G_{22}(s) \rightarrow K_p = 1{,}3024; K_i = 0{,}03784; K_d = -0{,}12224 \\ &G_{33}(s) \rightarrow K_p = 75{,}2532; K_i = 2{,}5109; K_d = -6{,}653 \end{aligned} \tag{3.12}$$

(p=0,1;q=0;r=0;Ω=0)

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0{,}01675 \\ 0 & -0{,}20257 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} 0{,}2052 & 0 & 0 \\ 0 & 2{,}955 & 0 \\ 0 & 0 & 0{,}0594 \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{3.13}$$

Persamaan state 3.13 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0007}{s^3 + 0{,}0034s} \tag{3.14}$$

$$G_{22}(s) = \frac{2{,}955s^2}{s^3 + 0{,}0034s}$$

Transfer function pada persamaan 3.14, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$\begin{aligned} &G_{11}(s) \rightarrow K_p = 16{,}35; K_i = 0{,}9575; K_d = 0 \\ &G_{22}(s) \rightarrow K_p = 2{,}489; K_i = 0{,}1398; K_d = -0{,}1107 \end{aligned} \tag{3.15}$$

(p=0; q=0,1; r=0; Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}05495\\ 0 & 0 & 0\\ -0{,}20257 & 0 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.16)$$

Persamaan state 3.16 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2}{s^3 - 0{,}0111s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}0329}{s^3 - 0{,}0111s} \quad (3.17)$$

Transfer function pada persamaan 3.17, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 62{,}8725; K_i = 6{,}1956; K_d = -1{,}5944$$
$$G_{22}(s) \rightarrow K_p = 43{,}63; K_i = 41{,}98; K_d = -0{,}1337 \quad (3.18)$$

(p=0,1;q=0,1;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}05495\\ 0 & 0 & 0{,}01675\\ -0{,}20257 & -0{,}20257 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.19)$$

Persamaan state 3.19 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0007}{s^3 - 0{,}0077s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}0329}{s^3 - 0{,}0077s} \quad (3.20)$$

Transfer function pada persamaan 3.20, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 64{,}4659; K_i = 236{,}5; K_d = 0$$
$$G_{22}(s) \rightarrow K_p = 40{,}85; K_i = 39{,}35; K_d = -0{,}1755 \quad (3.21)$$

(p=0,2;q=0;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0{,}0335\\ 0 & -0{,}4052 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.22)$$

Persamaan state 3.22 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0028}{s^3 + 0{,}0136s}$$
$$G_{22}(s) = \frac{2{,}955s^2}{s^3 + 0{,}0136s} \quad (3.23)$$

Transfer function pada persamaan 3.23, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 27{,}96; K_i = 31{,}77; K_d = 0$$
$$G_{22}(s) \rightarrow K_p = 1{,}679; K_i = 0{,}05975; K_d = -0{,}1791 \quad (3.24)$$

(p=0;q=0,2;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}1099\\ 0 & 0 & 0\\ -0{,}4051 & 0 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.25)$$

Persamaan state 3.25 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2}{s^3 - 0{,}0445s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}1316}{s^3 - 0{,}0445s} \quad (3.26)$$

Transfer function pada persamaan 3.26, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 33{,}97; K_i = 34{,}54; K_d = 0{,}6272$$
$$G_{22}(s) \rightarrow K_p = 17{,}61; K_i = 7{,}015; K_d = -0{,}1206 \quad (3.27)$$

(p=0,2;q=0,1;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}05495\\ 0 & 0 & 0{,}0335\\ -0{,}20257 & -0{,}405 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.28)$$

Persamaan state 3.28 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0028}{s^3 + 0{,}0024s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}0329}{s^3 + 0{,}0024s} \quad (3.29)$$

Transfer function pada persamaan 3.29, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 37{,}5775; K_i = 86{,}4195; K_d = 0$$
$$G_{22}(s) \rightarrow K_p = 35{,}74; K_i = 680{,}2; K_d = 0{,}05952 \quad (3.30)$$

(p=0,1;q=0,2;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}1099\\ 0 & 0 & 0{,}01675\\ -0{,}405 & -0{,}20257 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.31)$$

Persamaan state 3.31 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0007}{s^3 - 0{,}0411s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}1315}{s^3 - 0{,}0411s} \quad (3.32)$$

Transfer function pada persamaan 3.32, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 84{,}52; K_i = 347{,}3; K_d = 0$$
$$G_{22}(s) \rightarrow K_p = 83{,}89; K_i = 161; K_d = -0{,}10923 \quad (3.33)$$

(p=0,2;q=0,2;r=0;Ω=0)

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0 & 0 & -0{,}1099\\ 0 & 0 & 0{,}0335\\ -0{,}405 & -0{,}405 & 0\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} + \begin{bmatrix}0{,}2052 & 0 & 0\\ 0 & 2{,}955 & 0\\ 0 & 0 & 0{,}0594\end{bmatrix}\begin{bmatrix}U_2\\ U_3\\ U_4\end{bmatrix} \quad (3.34)$$

Persamaan state 3.34 dirubah dalam bentuk *transfer function*.

$$G_{11}(s) = \frac{0{,}2052s^2 + 0{,}0028}{s^3 - 0{,}0309s}$$
$$G_{22}(s) = \frac{2{,}955s^2 - 0{,}1315}{s^3 - 0{,}0309s} \quad (3.35)$$

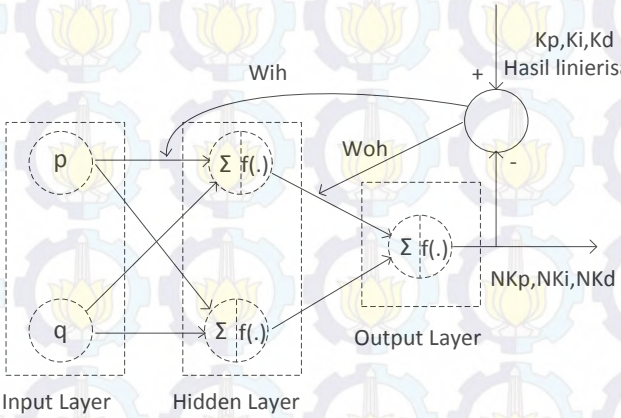
Transfer function pada persamaan 3.35, kemudian dicari nilai K_p, K_i, K_d untuk masing – masing *transfer function*.

$$G_{11}(s) \rightarrow K_p = 56{,}08; K_i = 115{,}5; K_d = 0$$
$$G_{22}(s) \rightarrow K_p = 85{,}02; K_i = 163{,}2; K_d = -0{,}1107 \quad (3.36)$$

Dimana $G_{11}(s) adalah\ nilai\ roll;\ G_{22}(s) adalah\ nilai\ pitch;$ $G_{33}(s) adalah\ nilai\ yaw$

Hasil linierisasi diatas hasilnya akan tetap sama walaupun titik – titik yang dimasukan untuk linierisasi bernilai negatif. Nilai K_p, K_i, K_d yang didapat adalah nilai PID untuk pengaturan kecepatan sudut.

Setelah mendapatkan hasil linierisasi, langkah selanjutnya adalah membangun struktur jaringan syaraf tiruan untuk proses *tuning* $K_p, K_i,$ dan K_d. Strutur JST dapat dilihat seperti gambar 3.13



Gambar 3. 13 Struktur JST *backpropagation*

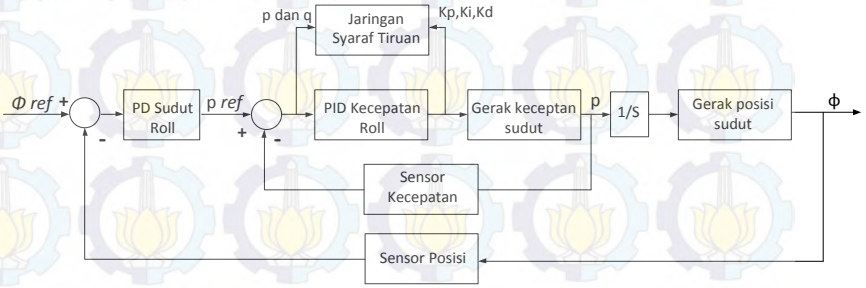
Gambar 3.13 diketahui strukut JST menggunakan satu layer input dengan 2 unit masukan yaitu p dan q. Satu *hidden layer* dengan 2 unit hiden digunakan sebelum menuju pada *output layer*.

Prinsip kerja JST pada tugas akhir ini digunakan untuk mendapatkan nilai $K_p, K_i,$ dan K_d agar sesuai dengan nilai $K_p, K_i,$ dan K_d hasil linierisasi. Jika hasil nilai PID pada JST belum sesuai,

maka nilai bobot (Wih dan Woh) akan terus direfisi sampai mendapatkan nilai yang sesuai dengan nilai PID hasil linierisasi.

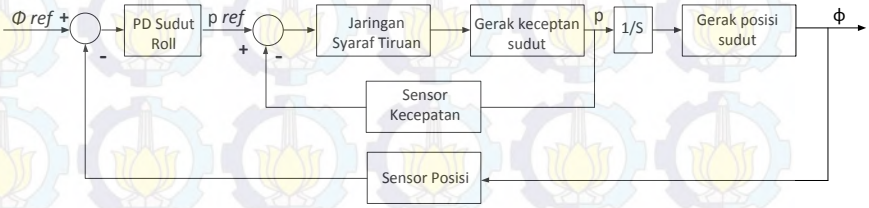
3.8.1 Kontrol Sudut Roll

Sudut *roll* pada quadcopter adalah sudut yang menyebabkan quadcopter bergerak sepanjang sumbu y. Pengaturan akan dilakukan dengan masukan sudut *roll* referensi yang dihasilkan dari sinyal kontrol sumbu y, dikarenakan dalam kasus ini hanya pergerakan *hover* maka sinyal kontrol dari sumbu y akan diatur mendekati atau sama dengan 0. Gambar 3.14 adalah diagram blok untuk kontrol posisi pada sudut *roll*. Sinyal referensi dari *phi* (ϕ) masuk ke kontroler tipe PD, keluarannya berupa kecepatan sudut referensi untuk kontrol kecepatan sudut *roll*. Kontroler tipe PID yang mengontrol kecepatan sudut untuk sementara sampai JST bisa nemukan nilai bobot yang cocok sehingga nilai PID hasil JST bisa menyerupai nilai PID hasil linierisasi.



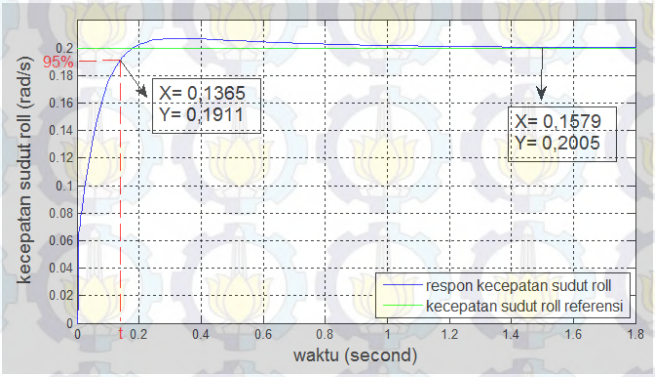
Gambar 3. 14 Kontrol posisi *roll* (proses belajar JST)

Setelah jaringan syaraf tiruan bisa mengikuti pola respon keluaran nilai PID hasil linierisasi maka kontroler JST bisa digunakan untuk proses *mapping*. Gambar 3.15 menunjukan kontroler JST digunakan untuk proses *mapping*.



Gambar 3. 15 Kontrol posisi *roll (*proses *mapping* JST)

Kontroler JST pada gambar 3.15 menggantikan peran kontroler PID hasil linierisasi untuk mengontrol kecepatan sudut *roll*. Nilai K_p dan K_d pada kontrol posisi sudut *roll* ditentukan seperti gambar 3.16.



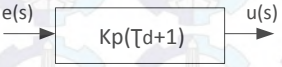
Gambar 3. 16 Respon *closed loop* kecepatan sudut *roll*

Nilai τ ditentukan dengan melihat respon *closed loop* pada kecepatan sudut *roll* saat respon mendekati 95% dari set point. Sehingga untuk menentukan τ^*

$$\tau^* = \frac{\tau}{3} = \frac{0{,}1365}{3} = 0{,}0455 \qquad (3.37)$$

Persamaan 3.37 memiliki maksud supaya respon dari *loop* dalam harus lebih cepet 3x dari respon *loop* luar. Sehingga untuk menentukan nilai K_p dan K_d untuk kontrol posisi seperti pada persamaan 3.38

$$\tau_d = \tau^* = 0{,}0455 \qquad (3.38)$$



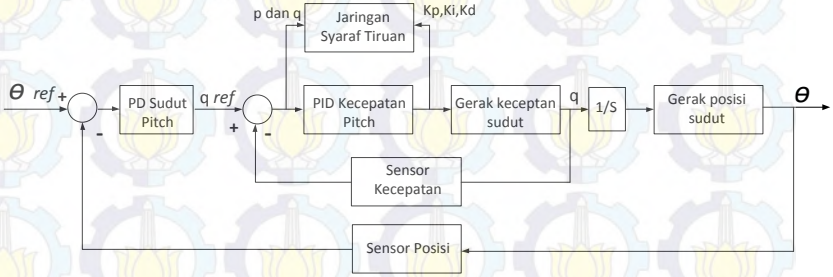
Gambar 3. 17 Diagram blok kontroler tipe PD

Nilai τ_d adalah 0,0455 dan nilai K_p bisa di *tuning* secara manual sampai respon keluaran sistem menjadi orde 1.

3.8.2 Kontrol Sudut Pitch

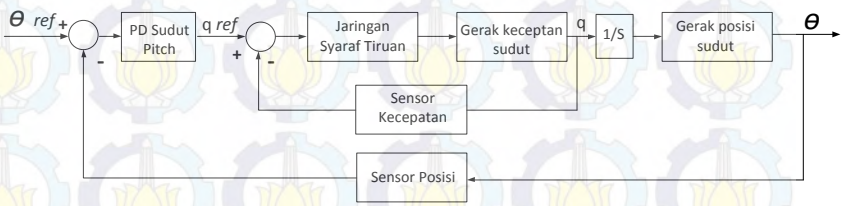
Sudut *pitch* pada quadcopter adalah sudut yang menyebabkan quadcopter bergerak sepanjang sumbu x. Pengaturan akan dilakukan dengan masukan sudut *pitch* referensi yang dihasilkan dari sinyal kontrol sumbu x, dikarenakan dalam kasus ini hanya pergerakan *hover* maka sinyal kontrol dari sumbu x akan diatur mendekati atau sama dengan 0.

Gambar 3.18 adalah diagram blok untuk kontrol posisi pada sudut *pitch*. Sinyal referensi dari *theta* (θ) masuk ke kontroler tipe PD, keluarannya berupa kecepatan sudut referensi untuk kontrol kecepatan sudut *pitch*. Kontroler tipe PID yang mengontrol kecepatan sudut untuk sementara sampai JST bisa nemukan nilai bobot yang baik sehingga nilai PID hasil JST bisa menyerupai nilai PID hasil linierisasi.



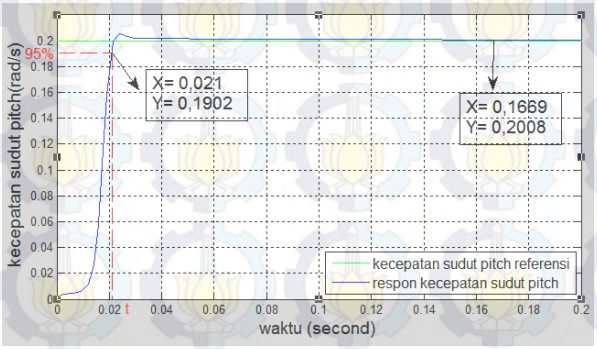
Gambar 3. 18 Kontrol posisi *pitch* (proses belajar JST)

Setelah jaringan syaraf tiruan bisa mengikuti respon keluaran nilai PID hasil linierisasi maka kontroler JST bisa digunakan untuk proses *mapping*. Gambar 3.19 menunjukan kontroler JST digunakan untuk proses *mapping*.



Gambar 3. 19 Kontrol posisi *pitch (proses* mapping JST)

Kontroler JST pada gambar 3.19 menggantikan peran kontroler PID hasil linierisasi untuk mengontrol kecepatan sudut *pitch*. Nilai K_p dan K_d pada kontrol posisi sudut *pitch* ditentukan seperti gambar 3.20



Gambar 3. 20 Respon *closed loop* kecepatan sudut *pitch*

Nilai τ ditentukan dengan melihat respon *closed loop* pada kecepatan sudut *pitch* saat respon mendekati 95% dari *set point*. Sehingga untuk menentukan τ^*

$$\tau^* = \frac{\tau}{3} = \frac{0{,}021}{3} = 0{,}007 \tag{3.39}$$

Persamaan 3.39 memiliki maksud supaya respon dari *loop* dalam harus lebih cepet 3x dari respon *loop* luar. Sehingga untuk menentukan nilai K_p dan K_d untuk kontrol posisi seperti pada persamaan 3.40

$$\tau_d = \tau^* = 0{,}007 \tag{3.40}$$

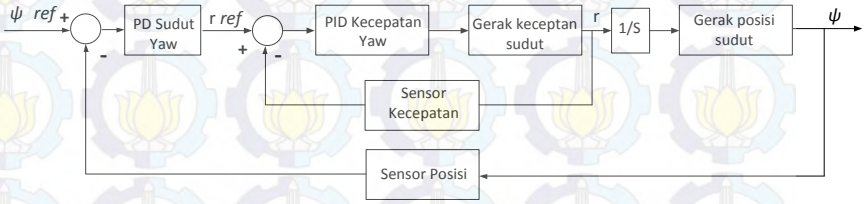
Berdasarkan gambar 3.17 ,nilai τ_d adalah 0.07 dan nilai K_p bisa di *tuning* secara manual sampai respon sistem menjadi orde 1.

3.8.3 Kontrol Sudut Yaw

Sudut *yaw* pada quadcopter adalah sudut yang menyebabkan quadcopter bergerak sepanjang sumbu z. Pengaturan akan dilakukan dengan masukan sudut *yaw* referensi ,dikarenakan dalam kasus ini hanya pergerakan *hover* maka sudut *yaw* referensi akan diatur mendekati atau sama dengan 0.

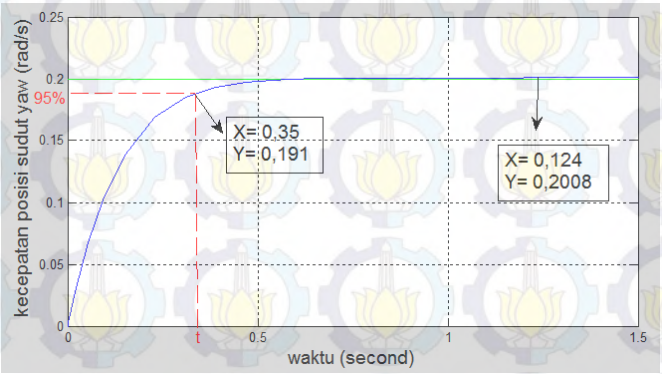
Pengaturan kecepatan sudut *yaw* pada kasus ini sedikit berbeda dengan pengaturan kecepatan sudut *roll* dan *pitch*, dikarenakan pengaturan sudut *yaw* hanya menggunakan kontroler tipe PID. Nilai

K_p, K_i dan K_d menggunakan hasil linierisasi saat nilai p, q, dan r adalah 0. Sedangkan kontrol sudut *yaw* menggunakan kontroler tipe PD. Gambar 3.22 akan menampilkan diagram blok untuk kontrol kecepatan dan posisi sudut *yaw*.



Gambar 3. 21 Kontrol posisi dan kecepatan sudut *yaw*

Dimana nilai K_p dan K_d untuk kontroler sudut *yaw* ditentukan seperti gambar 3.22.



Gambar 3. 22 Respon *closed loop* kecepatan sudut *yaw*

Nilai τ ditentukan dengan melihat respon *closed loop* pada kecepatan sudut *yaw* saat respon mendekati 95% dari set point. Sehingga untuk menentukan τ^*

$$\tau^* = \frac{\tau}{3} = \frac{0{,}35}{3} = 0{,}1167 \qquad (3.41)$$

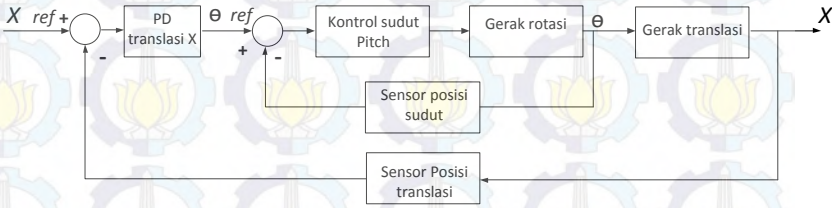
Persamaan 3.41 memiliki maksud supaya respon dari *loop* dalam harus lebih cepet 3x dari respon *loop* luar. Sehingga untuk menentukan nilai K_p dan K_d untuk kontrol posisi seperti pada persamaan 3.42

$$\tau_d = \tau^* = 0{,}1167 \quad (3.42)$$

Berdasarkan gambar 3.17 ,nilai τ_d adalah 0,1167 dan nilai K_p bisa di *tuning* secara manual sampai respon sistem menjadi orde 1.

3.8.4 Kontrol Translasi X

Kontrol translasi x sama dengan kontroler untuk posisi sudut *roll*,*pitch*,dan *yaw*. Kontroler yang digunakan adalah kontroler tipe PD. Gambar 3.25 akan menampilkan kontrol tipe PD dalam mengendalikan gerak translasi sumbu x.

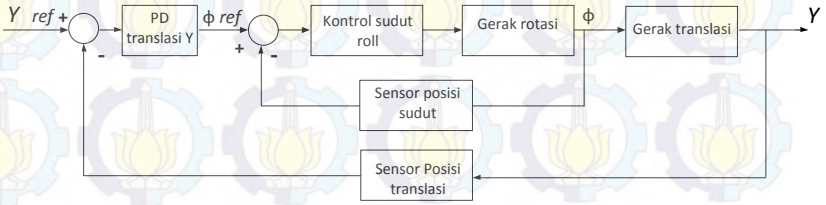


Gambar 3. 23 Kontrol translasi sumbu x

Dimana nilai K_p dan K_d pada kontroler gerak translasi x ditentukan dengan cara *tuning* manual.

3.8.5 Kontrol Translasi Y

Kontrol translasi y sama dengan kontroler untuk posisi sudut *roll*,*pitch*,dan *yaw*. Kontroler yang digunakan adalah kontroler tipe PD. Gambar 3.28 akan menampilkan kontrol tipe PD dalam mengendalikan gerak translasi sumbu y.

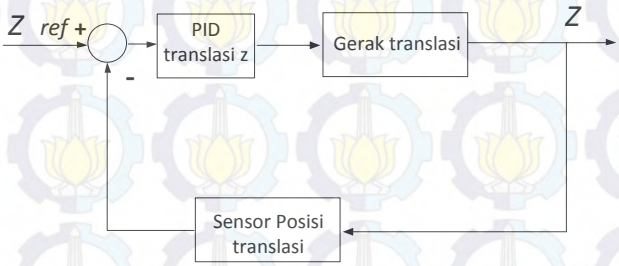


Gambar 3. 24 Kontrol translasi sumbu y

Dimana nilai K_p dan K_d pada kontroler gerak translasi y ditentukan dengan cara *tuning* manual.

3.8.6 Kontrol Translasi Z

Kontrol translasi z memiliki perbedaan dari kontroler translasi lainnya. Kontroler translasi x dan y menggunakan kontroler tipe PD, sedangkan kontroler translasi z menggunakan tipe PID. Kontroler tipe PID disini dipasang karena kontroler tipe PD tidak mampu membuat respon keluaran translasi z mendekati *set point*, sehingga dibutuhkan tambahan kontroler tipe I. Gambar 3.25 akan menjelaskan tentang kontrol translasi z.



Gambar 3. 25 Kontrol translasi sumbu z

Dimana nilai K_p, K_i dan K_d pada kontroler gerak translasi z ditentukan dengan cara *tuning* manual.

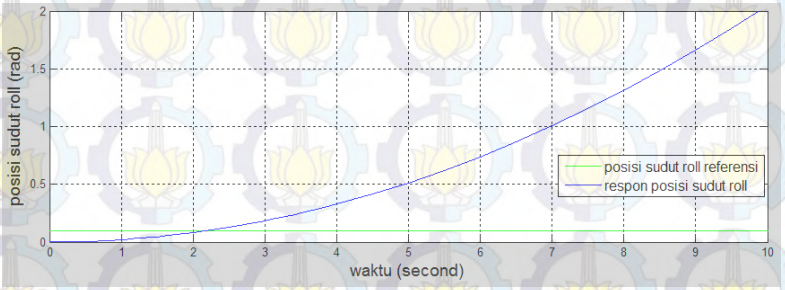
Halaman ini sengaja dikosongkan

BAB IV
HASIL SIMULASI

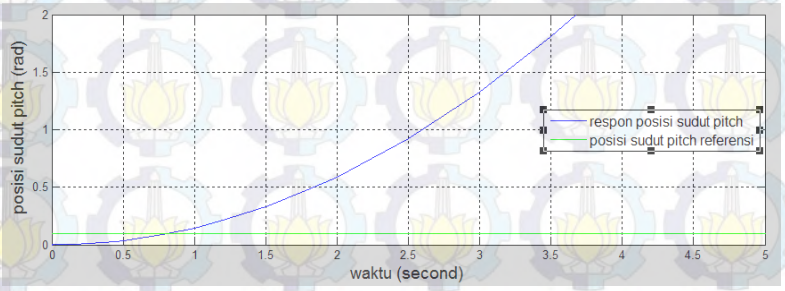
Bab ini akan ditampilkan hasil simulasi pergerakan *hover* dengan desain kontroler seperti pada bab 3. Simulasi dilakukan dengan menggunakan perangkat lunak MATLAB. Hal yang akan diuji pada simulasi ini adalah pengujian nilai PID hasil linierisasi, proses pembelajaran jaringan syaraf tiruan, dan proses *mapping* jaringan syaraf tiruan dengan ditambahkan gangguan eksternal.

4.1 Simulasi Respon *Open Loop* Sistem

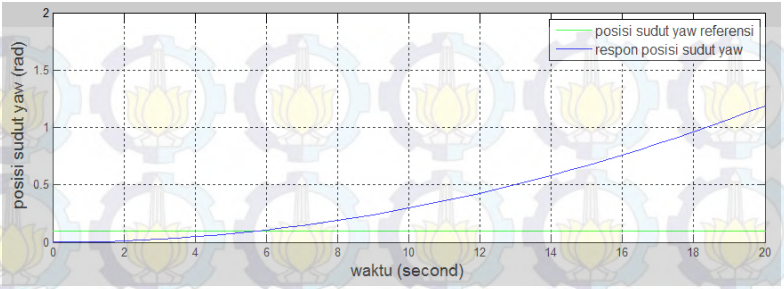
Simulasi ini diperlukan untuk mengetahui karakteristik *plant* quadcopter sebelum melakukan perancangan kontroler. Simulasi dilakukan dengan cara memberikan sinyal referensi posisi sudut *roll, pitch* dan *yaw* tanpa memberikan sinyal *feedback*. Respon posisi sudut hasil simulasi dapat dilihat pada gambar 4.1 sampai 4.3.



Gambar 4. 1 Respon posisi sudut *roll* tanpa kontroler



Gambar 4. 2 Respon posisi sudut *pitch* tanpa kontroler



Gambar 4. 3 Respon posisi sudut *yaw* tanpa kontroler

Gambar 4.1 sampai 4.3 menunjukan bahwa respon keluaran dari posisi sudut rotasi quadcopter menuju nilai tak berhingga, walaupun posisi sudut referensi berada pada posisi 0,1 rad. Ini semua menunjukan bahwa *plant* quadcopter memiliki karakteristik *plant* yang non linier sehingga mengakibatkan respon keluaran *plant* tidak stabil.

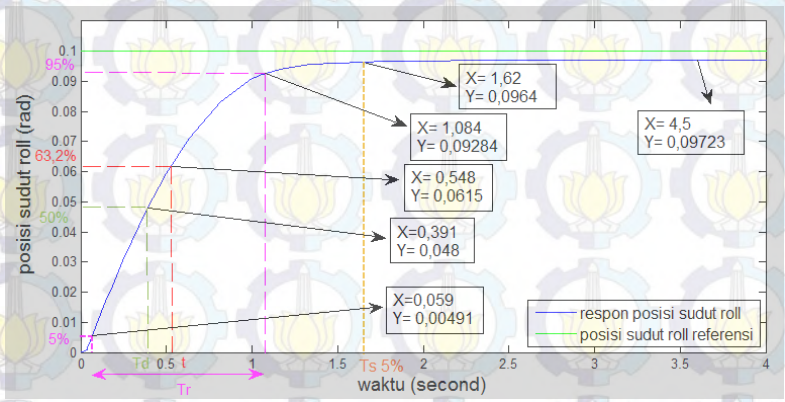
4.2 Simulasi Proses Kontrol Quadcopter Menggunakan kontrol PID dan PD

Bab III sudah didapatkan nilai – nilai K_p, K_i dan K_d untuk kecepatan sudut *roll* dan *pitch*. Sebelum digunakan sebagai proses pembelajaran perlu dicoba kemampuan nilai – nilai tersebut untuk mengontrol *plant* quadcopter. Ini dilakukan agar nilai – nilai K_p, K_i dan K_d yang nantinya digunakan untuk proses pembelajaran JST dipastikan sudah sesuai sebagai kontroler *plant* quadcopter.

4.2.1 Kontrol PID dan PD Saat Quadcopter diberi Nilai Sudut Referensi

Simulasi ini bertujuan untuk mengetahui kemampuan kontroler yang sudah didesain apakah mampu membuat respon keluaran gerak rotasi quadcopter menuju *set point* yang diinginkan dan respon keluaran juga mengalami kestabilan. Simulasi dilakukan dengan cara membuat matrik untuk masing – masing nilai K_p, K_i dan K_d yang nantinya akan dipanggil menggunakan fungsi *Interpreted Matlab Function*. Sehingga untuk mengontrol kecepatan sudut rotasi *plant* quadcopter hanya menggunakan nilai K_p, K_i dan K_d hasil linierisasi. Sedangkan kontrol posisi sudut rotasi menggunakan kontroler tipe PD. Kontroler untuk gerak translasi pada percobaan ini belum dipasang, dikarenakan simulasi ini hanya ingin mengetahui hasil linierisasi PID terhadap perubahan

posisi dan kecepatan sudut quadcopter. Hasil simulasi dapat dilihat pada gambar 4.4 sampai 4.6



Gambar 4. 4 Respon keluaran posisi sudut *roll* saat *set point* 0,1

Gambar 4.4 diketahui bahwa respon *transient* untuk posisi sudut *roll* sebagai berikut :

time constan (τ)

$$\tau(63{,}2\%) = 0{,}548 \text{ s} \quad (4.1)$$

Rise time (T_R)

$$T_R = (5\% - 95\%) = 1{,}084 - 0{,}059 = 1{,}025 \text{ s} \quad (4.2)$$

Settling time (T_S)

$$T_S = (\pm 5\%) = 3\tau = 3(0{,}548) = 1{,}644\ s \quad (4.3)$$

Delay time (T_D)

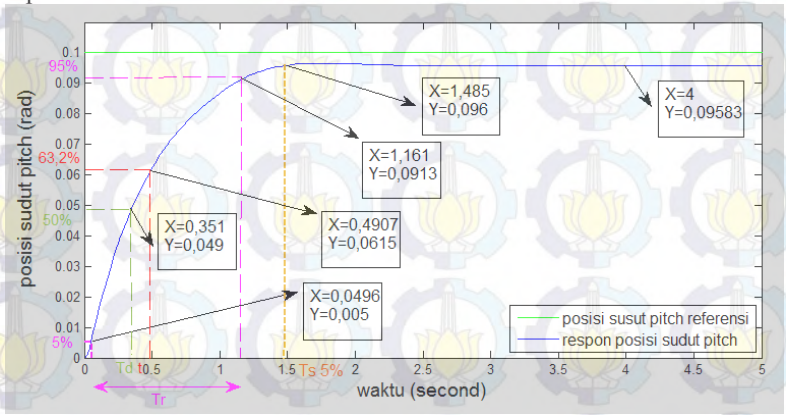
$$T_D = \tau\ \ln 2 = 0{,}712 \ln 2 = 0{,}3798\ s \quad (4.4)$$

Prosentase eror posisi (%ϵ)

$$\%\epsilon = \left(\frac{X_{SS} - Y_{SS}}{X_{SS}}\right) \times 100\% = \left(\frac{0{,}1 - 0{,}0964}{0{,}1}\right) \times 100\% = 3{,}6\% \quad (4.5)$$

Hasil spesifikasi respon diatas dapat disimpulkan bahwa, kontroler PID untuk kecepatan sudut dan PD untuk posisi sudut mampu membuat respon keluaran mendekati nilai *set point* yang diberikan. Eror posisi yang terjadi karena nilai K_p pada kontroler tipe PD yang kurang besar, tetapi jika nilai K_p diperbesar dari nilai yang dipasang sekarang, respon akan mengalami *overshoot* yang lebih dari 5 %.

Walaupun masih terdapat eror posisi, tetapi kontroler tetap dipertahankan karena eror masih dibawah 5%.



Gambar 4. 5 Respon keluaran posisi sudut *pitch* saat *set point* 0,1

Gambar 4.5 memiliki spesifikasi respon *transient* untuk posisi sudut *pitch* sebagai berikut :

time constan (τ)

$$\tau(63{,}2\%) = 0{,}491 \text{ s} \quad (4.6)$$

Rise time (T_R)

$$T_R = (5\% - 95\%) = 1{,}161 - 0{,}0496 = 1{,}1114 \text{ s} \quad (4.7)$$

Settling time (T_S)

$$T_S = (\pm 5\%) = 3\tau = 3(0{,}491) = 1{,}473\ s \quad (4.8)$$

Delay time (T_D)

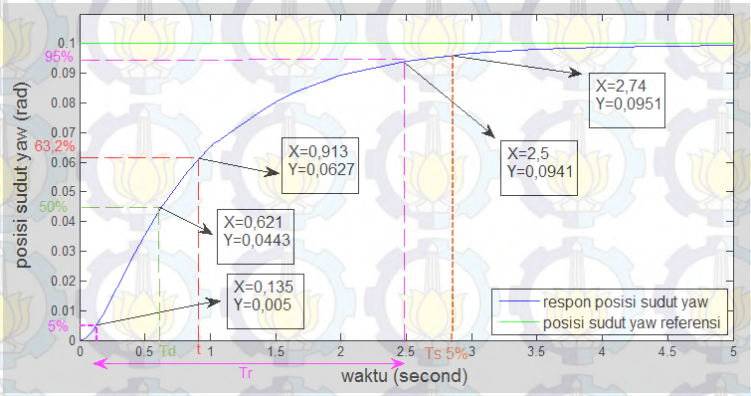
$$T_D = \tau \ \ln 2 = 0{,}491 \ \ln 2 = 0{,}34\ s \quad (4.9)$$

Prosentase eror posisi ($\%\epsilon$)

$$\%\epsilon = \left(\frac{X_{SS} - Y_{SS}}{X_{SS}}\right) \times 100\% = \left(\frac{0{,}1 - 0{,}096}{0{,}1}\right) \times 100\% = 4\% \quad (4.10)$$

Hasil spesifikasi respon diatas diketahui bahwa eror posisi untuk sudut *pitch* lebih besar, jika dibandingkan dengan eror posisi sudut *roll*. Ini disebabkan oleh nilai K_p dari kontroler tipe PD yang mengontrol posisi sudut *pitch*. Jika nilai K_p diperbesar dari nilai yang sudah ditetapkan, maka respon keluaran akan terdapat osilasi. Mengingat eror posisi yang dihasilkan nilainya masih kurang dari 5% maka kontroler ini tetap dipertahankan. Kontroler tipe PID untuk mengontrol kecepatan

sudut *pitch* dan kontroler tipe PD untuk mengontrol posisi sudut *pitch* dapat disimpulkan bahwa kontroler tersebut sudah cocok sebagai kontroler gerak rotasi dari sudut *pitch*.



Gambar 4. 6 Respon keluaran posisi sudut *yaw* saat *set point* 0,1

Gambar 4.5 memiliki spesifikasi respon *transient* untuk posisi sudut *yaw* sebagai berikut :

time constan (τ)

$$\tau(63{,}2\%) = 0{,}913 \text{ s} \quad (4.11)$$

Rise time (T_R)

$$T_R = (5\% - 95\%) = 2{,}5 - 0{,}135 = 2{,}365 \text{ s} \quad (4.12)$$

Settling time (T_S)

$$T_S = (\pm 5\%) = 3\tau = 3(0{,}913) = 2{,}739\ s \quad (4.13)$$

Delay time (T_D)

$$T_D = \tau \ \ln 2 = 0{,}913 \ \ln 2 = 0{,}633 s \quad (4.14)$$

Prosentase eror posisi ($\%\epsilon$)

$$\%\epsilon = \left(\frac{X_{SS} - Y_{SS}}{X_{SS}}\right) \times 100\% = \left(\frac{0{,}1 - 0{,}0951}{0{,}1}\right) \times 100\% = 5\% \quad (4.15)$$

Hasil spesifikasi respon diatas dapat diketahui bahwa eror posisi saat *settling time* ($\pm 5\%$) relatif besar, tetapi saat *settling time* memasuki ($\pm 0{,}5\%$) eror posisi semakin kecil. Ini disebabkan nilai *time constant* (τ) sudut *yaw* nilainya lebih besar dibandingkan nilai *time constant* pada sudut *roll* dan *pitch*. *Time constant* dalam hal ini mempengaruhi dikarenakan nilai *time constant* menjadi ukuran waktu yang menyatakan

kecepatan suatu respon. Walaupun kecepatan respon sudut *yaw* lebih lambat dibandingkan kecepatan respon sudut *roll* dan *pitch*, tetapi kontroler PID untuk kecepatan sudut *yaw* dan kontroler PD untuk posisi sudut *yaw* tetap dipertahankan, karena dianggap masih dapat untuk mengontrol gerak rotasi sudut *yaw* agar respon mendekati *set point* dan tidak terdapat osilasi.

4.2.2 Kontrol PID dan PD Saat Quadcopter diberi Gangguan

Simulasi sebelumnya bertujuan untuk mengetahui performansi kontroler PID dan PD untuk membuat respon keluaran gerak rotasi menuju ke *set point* yang diinginkan, sedangkan pada simulasi ini bertujuannya untuk mengetahui performansi kontroler PID dan PD untuk membuat respon keluaran gerak rotasi quadcopter akan tetap mengalami kestabilan jika terdapat gangguan *external*.

Simulasi dilakukan dengan cara memberikan ganguan berupa sinyal *gaussian* pada sinyal kontrol U_2. Gangguan ini diibaratkan terpaan angin yang mempengaruhi torsi *roll*. Besar kecilnya nilai gangguan yang diberikan dapat dihitung seperti persamaan 2.43 pada bab 2. Karena gangguan yang diberikan hanya pada sinyal kontrol U_2,maka persamaan 2.43 dapat ditulis kembali seperti persamaan 4.16.

$$U_2 = bl(-{\Omega_2}^2 + {\Omega_4}^2) \tag{4.16}$$

Berdasarkan percobaan yang telah dilakukan pada percobaan pengukuran kecepatan putar motor *brushless* terhadap nilai PWM seperti yang tertera di tabel 3.1 pada bab 3. Nilai maksimum yang didapat untuk nilai kecepatan putar motor *brushless* adalah 6222 Rpm. Satuan kecepatan putar motor pada tabel 3.1 berupa Rpm , sedangkan satuan kecepatan putar motor (Ω) pada persamaan 4.16 dalam rad s^{-1} , maka perlu mengkonversi satuan Rpm ke rad s^{-1} .

$$\Omega = \frac{2\pi}{60} N(Rpm) = \frac{2\times 3{,}14\times 6222}{60} = 651{,}24\ {}^{rad}/_{s} \tag{4.17}$$

Dimana N adalah kecepatan putar motor *brushless* (Rpm) dan Ω adalah kecepatan putar motor *brushless* (rad s^{-1}).

Sehingga dapat dihitung nilai maksimum yang didapat quadcopter dengan menulis kembali persamaan 4.16 ke persamaan 4.18.

$$\begin{aligned} U_2 &= bl(-{\Omega_2}^2 + {\Omega_4}^2) \\ &= 1{,}68198\,.\,10^{-5} \times 0{,}206(-0^2 + 651{,}24^2) \\ &= 1{,}4695\ \text{Nm} \end{aligned} \tag{4.18}$$

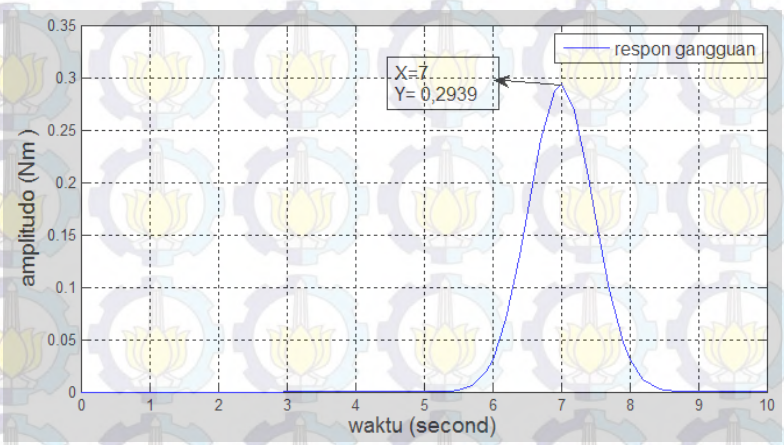
Persamaan 4.18 mengisyaratkan saat quadcopter diterpa angin pada salah satu sisinya sehingga mengakibatkan quadcopter menghasilkan torsi *roll* maksimum.

Gangguan yang diberikan bisa bervariasi dari nilai minimum U_2 sampai maksimum U_2. Ini semua dilakukan agar kondisi gangguan diberikan dapat disesuaikan dengan kondisi *plant* quadcopter secara nyata. Gangguan yang diberikan pada simulasi ini bernilai 40% dari kondisi maksimum.

$$Gangguan = 20\%\ maksimum\ torsi\ roll$$

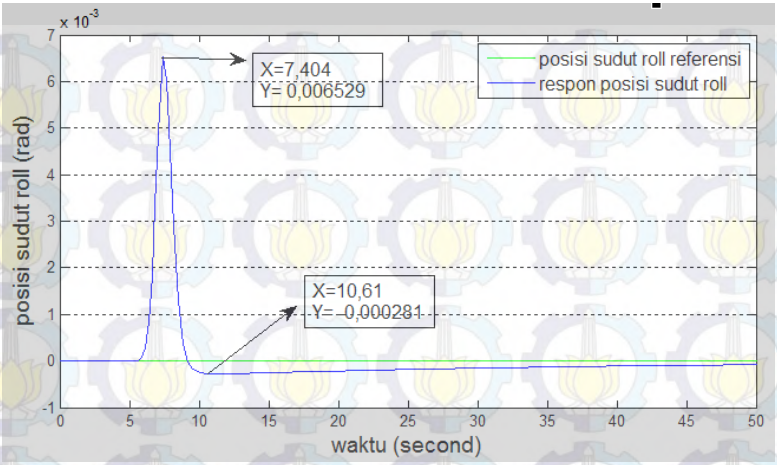
$$= \frac{20}{100} \times 1{,}4695 = 0{,}2939 \text{ Nm} \qquad (4.19)$$

Sehingga gangguan yang diberikan pada U_2 bernilai 0,2939 Nm. Hasil respon gangguan gambar 4.7 dan hasil simulasi dapat dilihat pada gambar 4.8 sampai 4.13.

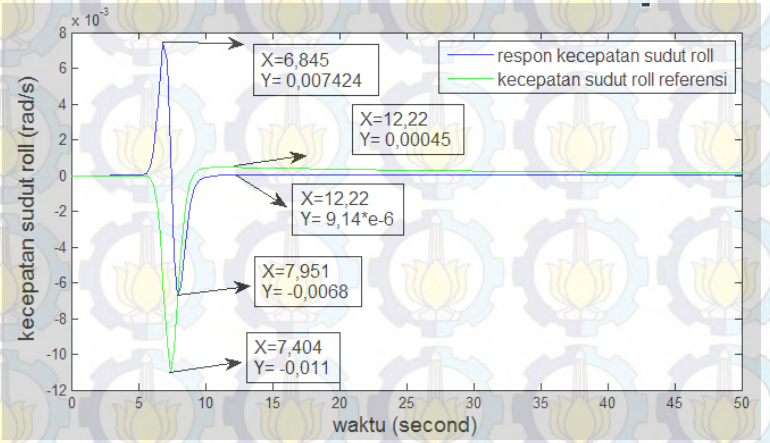


Gambar 4. 7 Respon gangguan

Gambar 4.7 menjelaskan bahwa saat t=7s, simpangan gangguan untuk U_2 bernilai tertinggi dengan nilai 0,2939 Nm. Ini disesuaikan dengan hasil perhitungan yang telah diperoleh pada persamaan 4.19. Lebar simpangan selama 2 *second* . Hal ini mengibaratkan bahwa *plant* quadcopter mengalami gangguan berupa terpaan angin selama 2 second dengan besarnya nilai gangguan tertinggi adalah 0,2939 Nm yang berdampak langsung pada torsi *roll*.



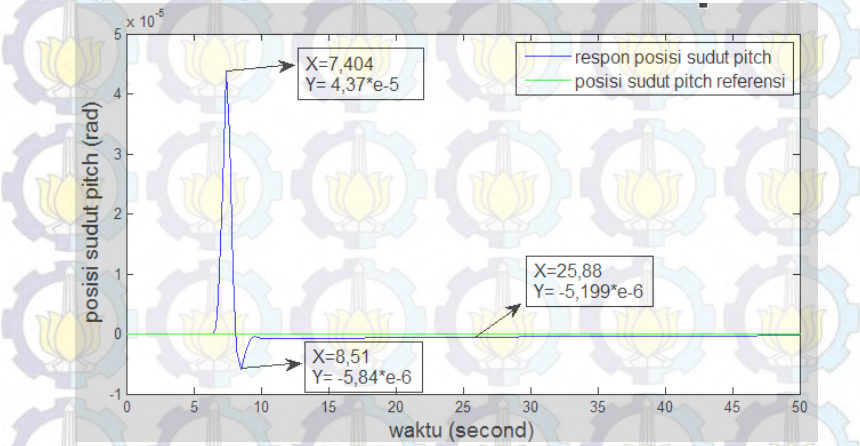
Gambar 4. 8 Respon posisi sudut *roll* saat diberi gangguan



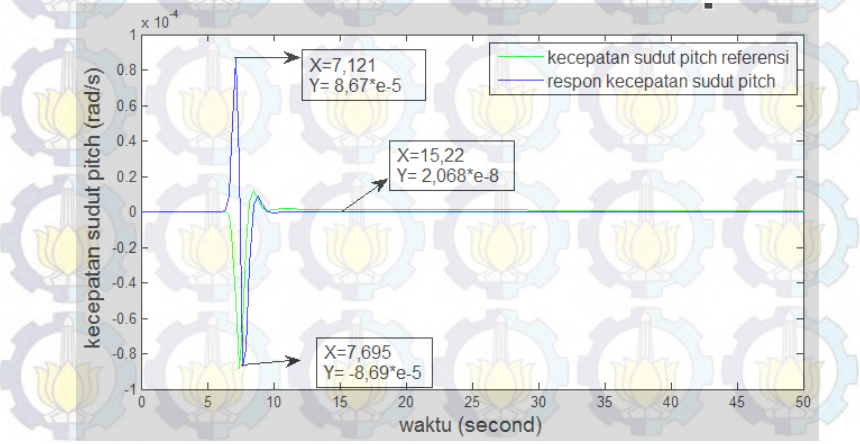
Gambar 4. 9 Respon kecepatan sudut *roll* saat diberi gangguan

Gambar 4.8 dan 4.9 adalah hasil kontroler tipe PD untuk posisi sudut *roll* dan kontroler tipe PID untuk kecepatan sudut *roll*. Simpangan yang terjadi pada posisi sudut akibat gangguan yang diberikan, tetapi kontroler PD mampu mengontrol agar simpangan tidak terlalu besar. Jika dilihat respon kecepatan sudut maka saat simpangan posisi sudut *roll* menuju ke 0,0065 rad kecepatan sudutnya ikut bertambah dan saat

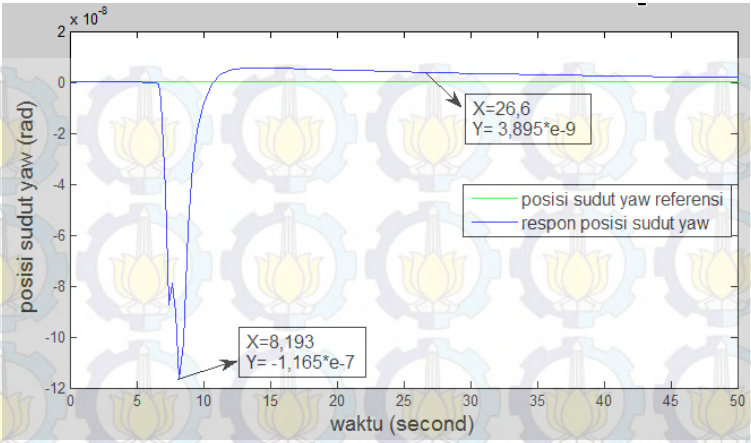
posisi sudut meninggalkan posisi 0,0065 rad maka kecepatannya ikut turun. Semua ini dikontrol dengan nilai PID hasil linierisasi saat nilai p dan q diantara 0 sampai 0,2 rad yang menyebabkan respon kecepatan sudut *roll* kembali ke *set point*. Jadi dapat disimpulkan bahwa hasil desain kontroler PD untuk posisi sudut *roll* dan nilai K_p, K_i dan K_d hasil linierisasi cocok untuk *plant* quadcopter ini.



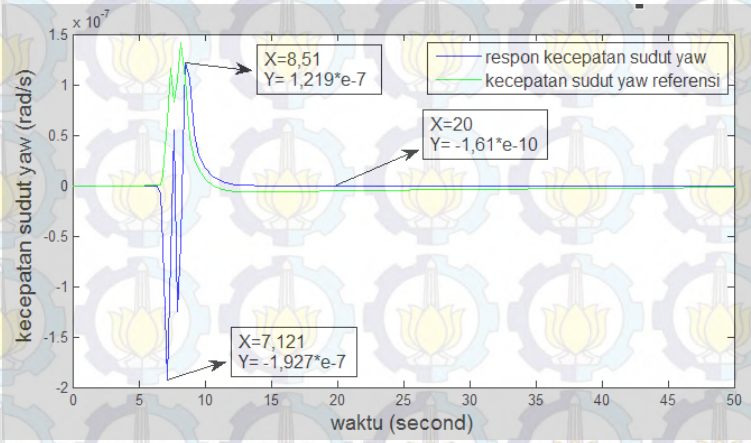
Gambar 4. 10 Respon posisi sudut *pitch* saat diberi gangguan



Gambar 4. 11 Respon kecepatan sudut *pitch* saat diberi gangguan



Gambar 4. 12 Respon posisi sudut *yaw* saat diberi gangguan



Gambar 4. 13 Respon kecepatan sudut *yaw* saat diberi gangguan

Gambar 4.10 sampai 4.13 adalah respon keluaran untuk posisi sudut dan kecepatan sudut *pitch* dan *yaw*. Perubahan sudut yang terjadi memanglah tidak terlalu besar, karena gangguan yang diberikan pada U_2 tidak terlalu berdampak pada sudut *pitch* dan *yaw*. Walaupun tidak terlalu berdampak tetapi tetap saja jika kontroler PD untuk posisi sudut dan kontroler PID untuk kontroler kecepatan sudut tidak memiliki nilai

K_p, K_i dan K_d yang cocok maka respon keluaran tetap tidak stabil. Sehingga dapat ditarik kesimpulan bahwa hasil desain kontroler tipe PD untuk kontrol posisi sudut dan nilai Kp, Ki dan Kd hasil linierisasi untuk kontroler kecepatan sudut *pitch* dan *yaw* dapat digunakan pada *plant* quadcopter.

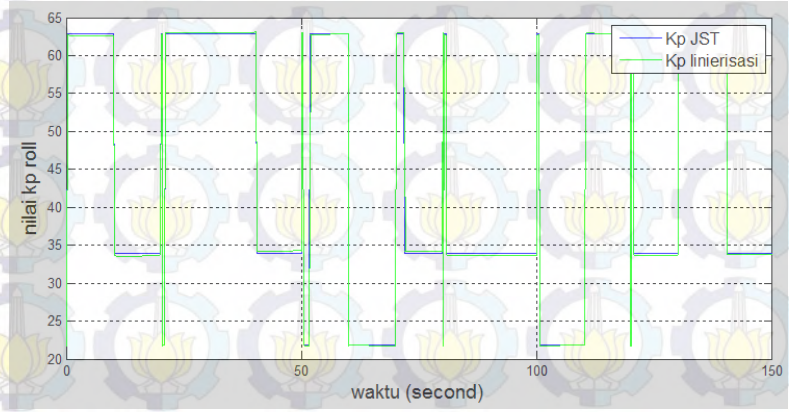
4.3 Simulasi Proses Pembelajaran Jaringan Syaraf Tiruan Terhadap Nilai PID Hasil linierisasi

Sebelum jaringan syaraf tiruan diimplementasikan untuk mengambil keputusan (*mapping*) perlu dilakukan proses belajar. Proses belajar yang dimaksud adalah proses merevisi bobot – bobot sehingga respon keluaran (Kp,Ki,Kd) pada jaringan syaraf tiruan bisa mengikuti pola keluaran nilai PID hasil linierisasi.

Simulasi ini akan mencoba menampilkan proses belajar untuk tiap – tiap nilai PID pada kontroler kecepatan sudut *roll* dan *pitch*. Cara yang dilakukan untuk proses belajar ini adalah dengan memberikan nilai random dari -0,2 sampai 0,2 pada nilai p (kecepan sudut *roll*) dan q (kecepatan sudut *pitch*) . Nilai p dan q nantinya dimasukan pada kontroler PID hasil linierisasi dan kontroler jaringan syaraf tiruan.

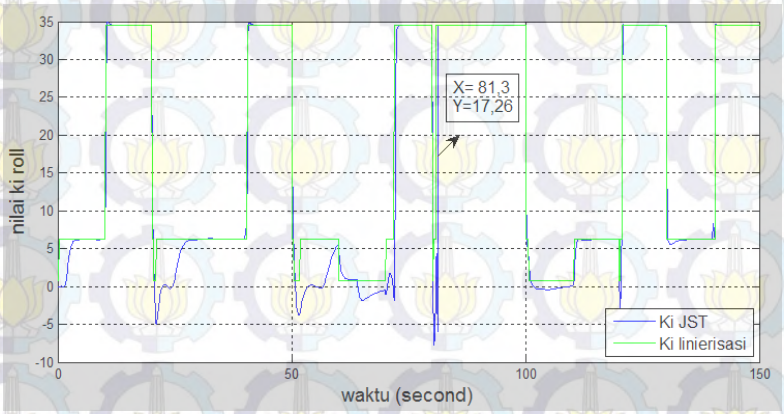
4.3.1 Proses Pembelajaran Jaringan Syaraf Tiruan untuk Nilai K_p, K_i, dan K_d Sudut *Roll*

Simulasi ini akan dicoba untuk membuat suatu proses pembelajaran untuk nilai K_p, K_i dan K_d pada kontroler kecepatan sudut *roll*. Hasil simulasi dapat dilihat pada gambar 4.14 sampai 4.16.

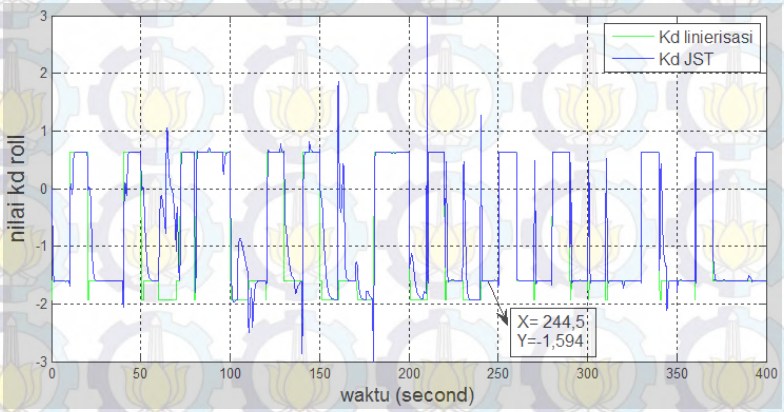


Gambar 4. 14 Proses pembelajaran nilai K_p *roll*

Gambar 4.10 menunjukan bahwa pada saat awal $t = 0$ hingga $t = 150$ pola respon keluaran nilai K_p, K_i dan K_d pada jaringan syaraf tiruan dapat mengikuti pola respon keluaran PID hasil linierisasi, ini dikarenakan nilai *learning rate* yang diperbesar hingga 0,001. Sehingga tidak membutuhkan waktu yang cukup lama untuk mendapatkan nilai bobot – bobot supaya menghasilkan pola respon yang menyerupai pola keluaran nilai Kp hasil linierisasi. Nilai bobot – bobot sudah dapat dicoba untuk proses *mapping*.



Gambar 4. 15 Proses pembelajaran nilai K_i *roll*

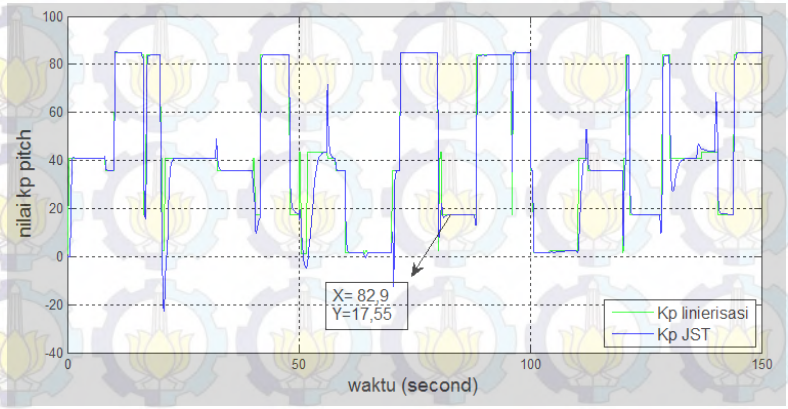


Gambar 4. 16 Proses pembelajaran nilai K_d *roll*

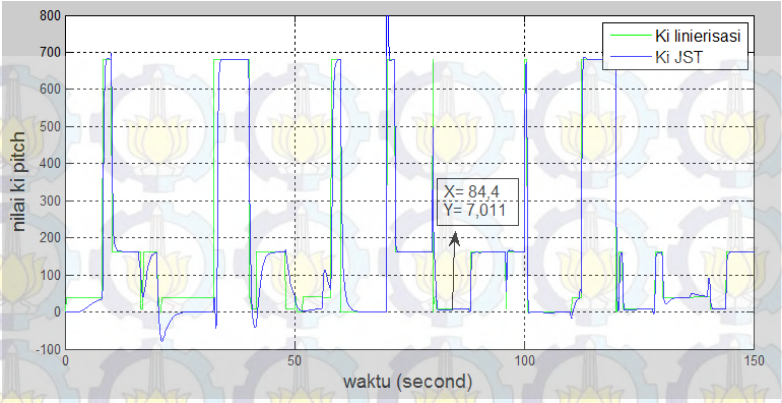
Gambar 4.15 dan gambar 4.16 adalah proses pembelajaran untuk nilai K_i dan K_d *roll*. Gambar 4.15 terjadi beberapa ketidaksesuaian pola respon antara nilai K_i hasil proses pembelajaran jaringan syaraf tiruan dengan pola respon K_i hasil linierisasi pada saat $t < 81{,}3$. Gambar 4.16 juga terjadi ketidaksesuaian antara pola respon nilai K_d hasil proses pembelajaran jaringan syaraf tiruan dengan pola respon nilai K_d hasil linierisasi saat $t < 244{,}5$. Semuai ini dikarenakan nilai bobot – bobot yang terdapat dalam jaringan syaraf tiruan belum menemukan nilai yang sesuai, sehingga diperlukan waktu lagi untuk jaringan syaraf tiruan dalam melakukan proses pembelajaran. Pola respon keluaran jaringan syaraf tiruan jika sudah menyerupai pola respon keluaran K_i dan K_d hasil linierisasi, maka nilai – nilai bobot dapat dicoba untuk proses *mapping*.

4.3.2 Proses Pembelajaran Jaringan Syaraf Tiruan untuk Nilai Kp,Ki, dan Kd Sudut *Pitch*

Proses simulasi pembelajaran jaringan syaraf tiruan untuk nilai K_p, K_i dan K_d kecepatan sudut *pitch* pada dasarnya sama dengan proses pembelajaran nilai K_p, K_i dan K_d untuk sudut *roll*. Hasil proses pembelajaran dapat dilihat pada gambar 4.17 sampai 4.19.

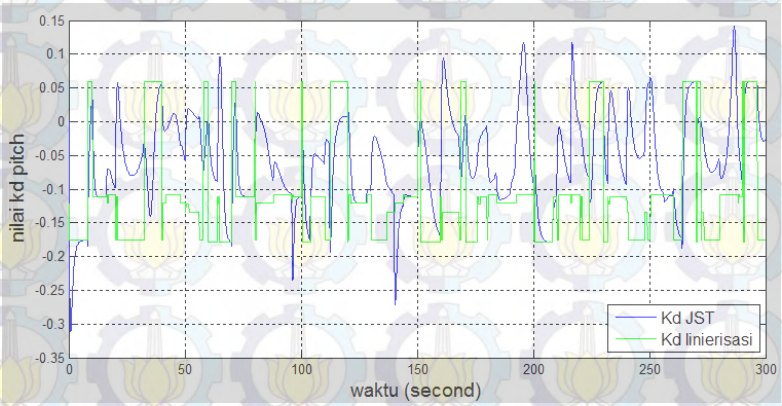


Gambar 4. 17 Proses pembelajaran nilai K_p *pitch*



Gambar 4. 18 Proses pembelajaran nilai K_i *pitch*

Gambar 4.17 dan gambar 4.18 adalah proses pembelajaran untuk Kp dan Ki *pitch*. Gambar 4.17 saat $t < 82{,}9$ dan gambar 4.18 saat $t < 84{,}4$ terjadi ketidaksesuaian antara pola respon K_p dan K_i hasil pembelajaran jaringan syaraf tiruan dengan pola respon K_p dan K_i hasil linierisasi. Semua ini dikarenakan nilai bobot jaringan syaraf tiruan yang belum sesuai, sehingga mengakibatkan kedua pola respon jaringan syaraf tiruan tersebut belum dapat menyerupai pola respon hasil linierisasi.



Gambar 4. 19 Proses pembelajaran nilai K_d *pitch*

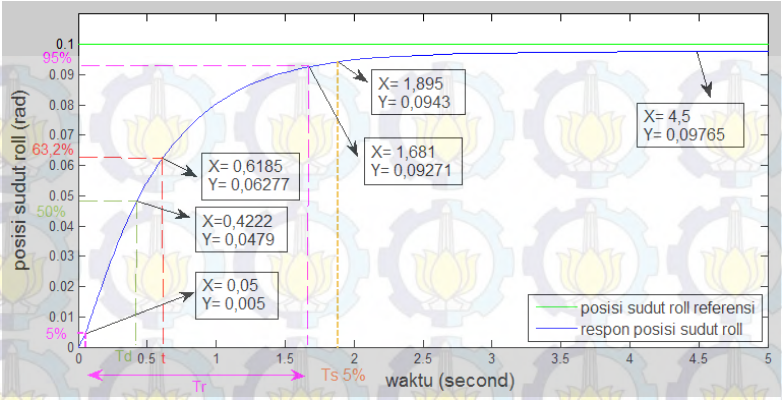
Gambar 4.19 adalah gambar hasil proses pembelajaran K_d *pitch*. Bobot – bobot yang dihasilkan selama proses pembelajaran tersebut $(0 < t < 300)$ belum dapat digunakan untuk proses pengambilan keputusan (*mapping*). Hal ini terjadi dikarenakan dua penyebab. Pertama nilai *learning rate* kecil yang menyebabkan proses pembelajaran dibutuhkan waktu yang lebih lama, jika *learning rate* diperbesar akan menyebabkan proses belajar lebih cepat tetapi bobot yang dihasilkan belum tentu sesuai. Kedua adalah nilai variasi dari K_d *pitch* ini yang memiliki *range* terlalu kecil, sehingga eror yang dihasilkan selama proses pembelajaran juga kecil. Eror yang terlalu kecil akan mengakibatkan proses untuk mencapai ke *set point* (pola respon Kd linierisasi) akan semakin lama.

4.4 Simulasi *Mapping* Jaringan Syaraf Tiruan Sebagai Kontrol Quadcopter

Bobot – bobot yang sudah didapat pada hasil pembelajaran jaringan syaraf tiruan akan digunakan untuk proses *mapping*. Simulasi ini digunakan untuk mengetahui kemampuan Jaringan Syaraf Tiruan sebagai kontroler dalam menjaga kestabilan gerak *hover* quadcopter. Proses simulasi pada tahap ini akan dibagi menjadi 2. Pertama proses simulasi saat quadcopter diberi sudut referensi dan yang kedua saat quadcopter diberi gangguan.

4.4.1 Kontrol Jaringan Syaraf Tiruan Saat Quadcopter diberi Nilai Sudut Referensi

Simulasi ini bertujuan untuk menguji bobot – bobot hasil proses pembelajaran jaringan syaraf tiruan apakah mampu membuat respon keluaran gerak rotasi quadcopter menuju *set point* yang diinginkan dan respon keluaran mengalami kestabilan. Simulasi dalam proses ini dilakukan dengan cara membuat program jaringan syaraf tiruan yang berisi perhitungan maju saja dengan nilai bobot sesuai yang didapatkan selama proses pembelajaran jaringan syaraf tiruan. Program nantinya akan dipanggil menggunakan fungsi *Interpreted Matlab Function.* Sehingga untuk mengontrol kecepatan sudut rotasi *plant* quadcopter hanya menggunakan nilai K_p, K_i dan K_d hasil kontroler jaringan syaraf tiruan. Sedangkan kontrol posisi sudut rotasi menggunakan kontroler tipe PD. Kontroler untuk gerak translasi pada percobaan ini belum dipasang, dikarenakan simulasi ini hanya ingin mengetahui hasil bobot – bobot yang didapat selama proses pembelajaran. Hasil simulasi dapat dilihat pada gambar 4.22 sampai 4.21.



Gambar 4. 20 Respon posisi sudut *roll* dengan kontroler JST

Gambar 4.20 diketahui bahwa respon *transient* untuk posisi sudut *roll* sebagai berikut :

time constan (τ)

$$\tau(63{,}2\%) = 0{,}6185 \text{ s} \quad (4.20)$$

Rise time (T_R)

$$T_R = (5\% - 95\%) = 1{,}681 - 0{,}05 = 1{,}631 \text{ s} \quad (4.21)$$

Settling time (T_S)

$$T_S = (\pm 5\%) = 3\tau = 3(0{,}6185) = 1{,}8555\ s \quad (4.22)$$

Delay time (T_D)

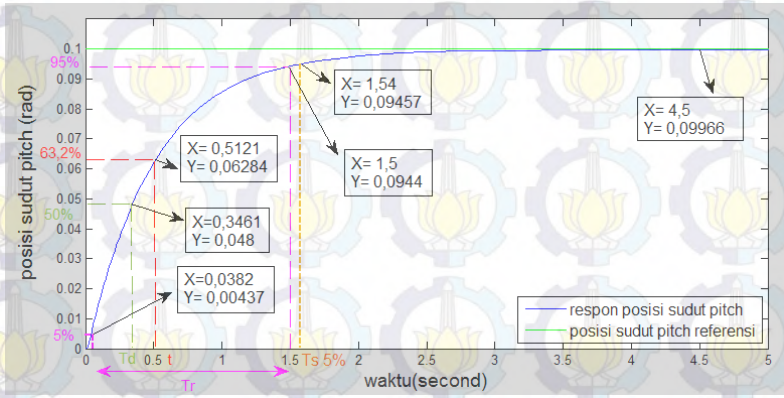
$$T_D = \tau \ln 2 = 0{,}6185 \ln 2 = 0{,}4287\ s \quad (4.23)$$

Prosentase eror posisi (%ϵ)

$$\%\epsilon = \left(\frac{X_{SS} - Y_{SS}}{X_{SS}}\right) \times 100\% = \left(\frac{0{,}1 - 0{,}09765}{0{,}1}\right) \times 100\% = 2.35\% \quad (4.24)$$

Hasil spesifikasi respon diatas dapat diketahui bahwa kecepatan respon *transient* kontroler jaringan syaraf tiruan lebih lambat jika dibandingkan dengan kontroler PID hasil linierisasi. Ini dikarenakan nilai *time constant* (τ) yang dimiliki kontroler jaringan syaraf tiruan lebih besar. Respon *transient* yang dihasilkan juga lebih halus jika dibandingkan dengan respon *transient* PID hasil linierisasi.Eror posisi yang dihasilkan kontroler jaringan syaraf tiruan relatif lebih kecil dibandingkan kontroler PID hasil linierisasi. Ini semua dikarenakan pada kontroler jaringan syaraf tiruan disetiap perubahan kecepatan sudut

roll, jaringan syaraf tiruan mampu memberikan nilai K_p, K_i dan K_d yang sesuai untuk *plant* quadcopter, sedangkan pada kontroler PID hasil linierisasi jika terjadi perubahan kecepatan sudut, nilai K_p, K_i dan K_d yang diberikan hanya pada saat titik kecepatan sudut yang dilinierisasi saja. Walaupun masih terdapat eror posisi, tetapi kontroler gerak rotasi sudut *roll* layak untuk diimplementasikan pada *plant* quadcopter.



Gambar 4. 21 Respon posisi sudut *pitch* dengan kontroler JST

Gambar 4.20 diketahui bahwa respon *transient* untuk posisi sudut *pitch* sebagai berikut :

time constan (τ)

$$\tau(63{,}2\%) = 0{,}5121 \text{ s} \quad (4.25)$$

Rise time (T_R)

$$T_R = (5\% - 95\%) = 1{,}5 - 0{,}0382 = 1{,}462\text{s} \quad (4.26)$$

Settling time (T_S)

$$T_S = (\pm 5\%) = 3\tau = 3(0{,}5121) = 1{,}5363\ s \quad (4.27)$$

Delay time (T_D)

$$T_D = \tau\ \ln 2 = 0{,}5121 \ln 2 = 0{,}355\ s \quad (4.28)$$

Prosentase eror posisi ($\%\epsilon$)

$$\%\epsilon = \left(\frac{X_{SS} - Y_{SS}}{X_{SS}}\right) \times 100\% = \left(\frac{0{,}1 - 0{,}09966}{0{,}1}\right) \times 100\% = 0{,}34\% \quad (4.29)$$

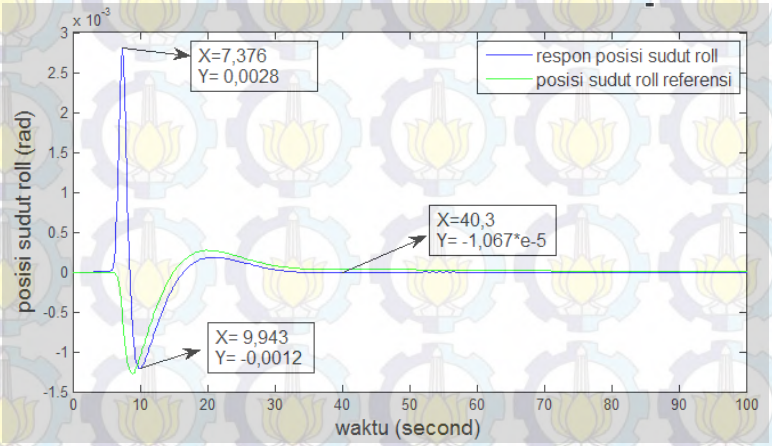
Hasil dari spesifikasi respon *transient* kontroler gerak rotasi sudut *pitch* diketahui bahwa respon kecepatan kontroler jaringan syaraf tiruan lebih lambat dibandingkan respon kecepatan kontroler PID hasil

linierisasi. Respon *transient* yang dihasilkan juga lebih halus jika dibandingkan dengan respon *transient* PID hasil linierisasi. Eror posisi yang dihasilkan juga jauh lebih kecil jika dibandingkan dengan eror posisi kontroler PID hasil linierisasi. Semua ini terjadi dengan analisa yang sama seperti kontroler gerak rotasi sudut *roll*. Sehingga dapat disimpulkan bahwa kontroler untuk gerak rotasi sudut *pitch* dapat digunakan pada *plant* quadcopter.

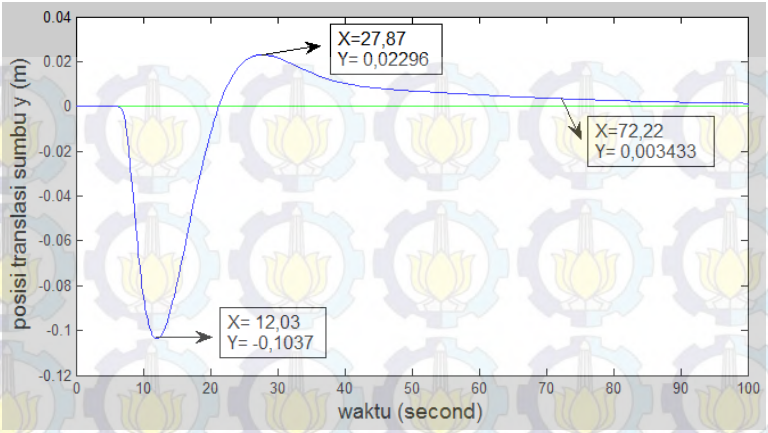
Kontroler gerak rotasi sudut *yaw* masih sama. Ini dikarenakan jaringan syaraf tiruan tidak dipasang untuk gerak rotasi sudut *yaw*. Sehingga kontroler yang digunakan tetap menggunakan tipe PID hasil linierisasi untuk kontrol kecepatan sudut dan kontroler tipe PD untuk posisi sudut.

4.4.2 Kontrol Jaringan Syaraf Tiruan Saat Quadcopter diberi Gangguan

Simulasi ini akan dicoba memberikan gangguan pada *plant* quadcopter. Percobaan ini digunakan untuk mengetahui apakah kontroler jaringan syaraf tiruan dapat menjaga kestabilan *hover* quadcopter saat diberi gangguan. Besarnya gangguan yang diberikan nilainya sama dengan perhitungan pada persamaan 4.19 dan gangguan diberikan pada U_2. Kontroler translasi pada simulasi ini sudah dapat dipasang. Hasil dari simulasi dapat dilihat pada gambar dibawah.

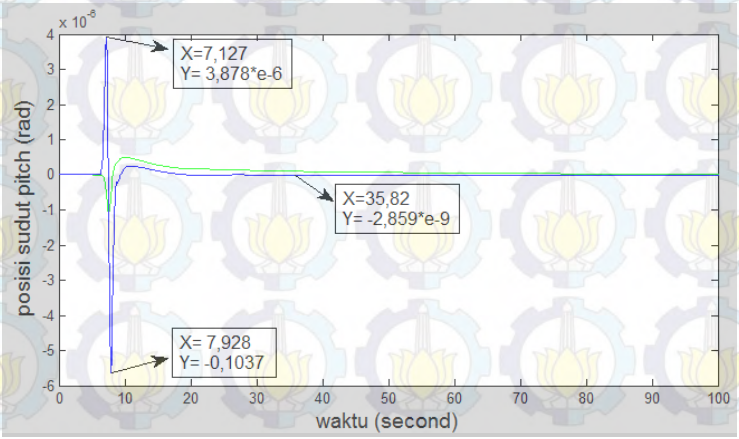


Gambar 4. 22 Respon posisi sudut *roll* dengan gangguan

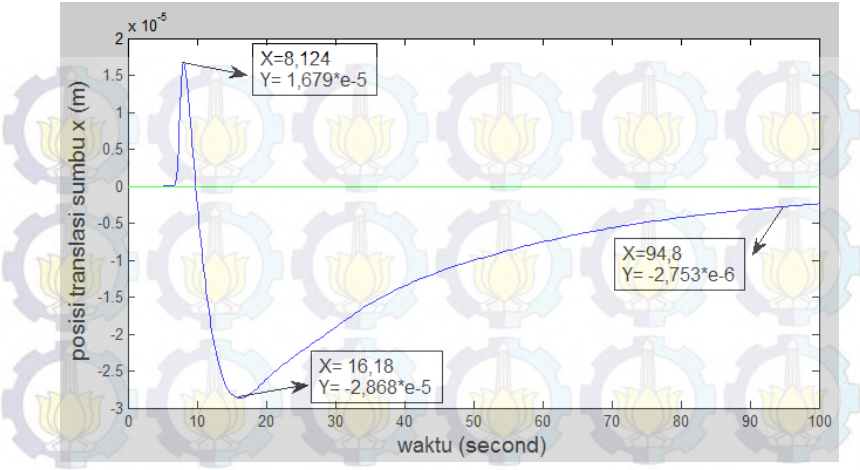


Gambar 4. 23 Respon posisi translasi searah sumbu y

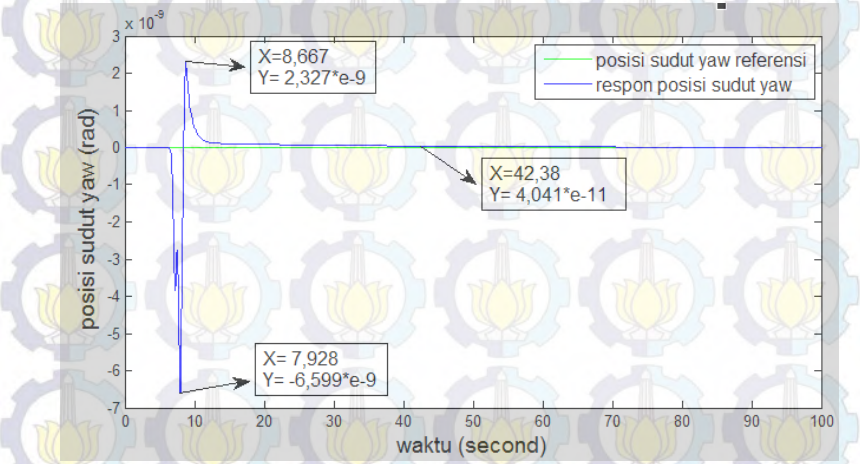
Gambar 4.22 dan 4.23 adalah respon posisi sudut *roll* dan posisi translasi searah sumbu y saat diberi gangguan. Gambar 4.22 menunjukan bahwa respon posisi sudut *roll* membutuhkan waktu ± 30 s untuk kembali ke *set point* setelah mengalami gangguan pada detik ke-7. Kondisi ini diakibatkan oleh kontroler sumbu y. Kontroler tipe PD ini menghasilkan respon yang lama menuju *set point*, sehingga saat diberi gangguan, respon membutuhkan waktu yang lama untuk kembali menuju *set point*.



Gambar 4. 24 Respon posisi sudut *pitch* dengan gangguan



Gambar 4. 25 Respon posisi translasi sumbu x



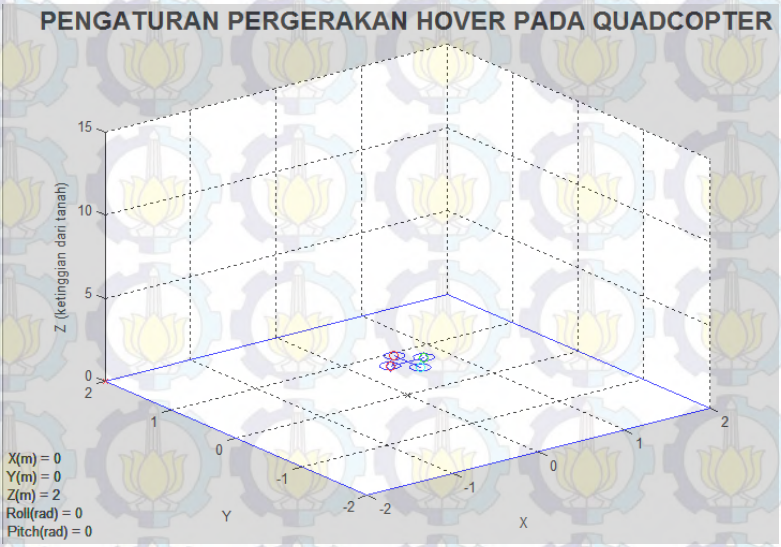
Gambar 4. 26 Respon posisi sudut *yaw* dengan gangguan

Gambar 4.24, gambar 4.25 dan gambar 4.26 adalah gambar respon posisi sudut *pitch*, respon posisi translasi sumbu x dan posisi sudut *yaw*. Simpangan yang terjadi pada ketiga gambar tidaklah terlalu besar. Ini dikarenakan gangguan yang diberikan pada U_2 tidak terlalu berpengaruh terhadap gerak rotasi sudut *roll* dan *pitch*. Walaupun tidak

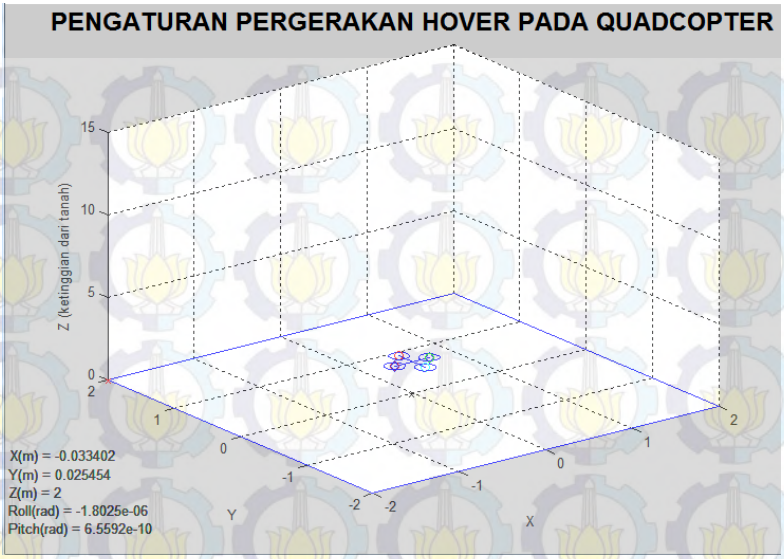
berpengaruh, jika kontroler tidak memiliki nilai K_p, dan K_d yang sesuai untuk kontroler posisi sudut *pitch, yaw* dan posisi translasi sumbu x, maka respon tetap tidak stabil.

4.5 Simulasi 3D Kontroler Jaringan Syaraf Tiruan untuk Kestabilan *Hover*

Pengujian simulasi ini akan menggunakan fasilitas simulasi 3D yang dapat menunjukan penerbangan quadcopter secara visual. Simulasi ini bertujuan untuk mengetahui gambaran yang terjadi jika quadcopter diterbangkan secara nyata. Hasil simulasi ini dapat dilihat pada gambar 4.27 dan gambar 4.28.



Gambar 4. 27 Tampilan 3D saat keadaan *hover* tanpa gangguan



Gambar 4. 28 Gambar 3D saat keadaan *hover* dengan gangguan

Gambar 4.27 dan 4.28 menunjukan hasil simulasi pengaturan pergerakan *hover* quadcopter menggunakan kontroler jaringan syaraf tiruan. Gambar 4.27 adalah posisi awal quadcopter sebelum mendapatkan gangguan. Posisi berada disekitar ketinggian 2 meter, karena *set point* yang diberikan pada kontrol gaya *thrust* adalah 2. Gambar 4.28 adalah gambar disaat quadcopter mendapatkan gangguan di U_2 sebesar 0,2939 Nm. Terlihat posisi ketinggian quadcopter tidak berubah, tetapi posisi translasi sumbu x, y , posisi sudut *roll* dan *pitch* sedikit berubah. Walaupun mengalami perubahan, tetapi perubahannya sangat kecil dan masih membuat quadcopter dalam kondisi stabil. Sehingga kontroler yang telah didesain dapat dicoba untuk diimplementasikan secara nyata.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil pengujian dan analisis, maka diperoleh kesimpulan sebagai berikut:

a) Pola respon keluaran metode jaringan syaraf tiruan dengan 2 bobot input dan 1 *hidden layer* dapat menyerupai pola respon keluaran PID hasil linierisasi.
b) Metode Jaringan Syaraf Tiruan hasil pembelajaran dapat diterapkan sebagai kontroler untuk memilih nilai Kp,Ki dan Kd, sehingga mampu untuk mengontrol kestabilan pergerakan *hover* quadcopter. Ini dibuktikan dengan nilai prosentase posisi eror terhadap *set point* sebesar 2,35% untuk kontrol posisi sudut *roll* dan 0,34% untuk kontrol posisi sudut *pitch.*
c) Hasil respon keluaran jaringan syaraf tiruan memiliki respon yang lebih halus dan memiliki eror posisi yang lebih kecil jika dibandingkan respon keluaran dari PID hasil linierisasi. Ini dibuktikan dengan prosentase eror posisi PID yang lebih besar sekitar 3,6% untuk kontrol posisi sudut *roll* dan 4% untuk kontrol posisi sudut *pitch.*

5.2. Saran

Dari hasil penelitian yang dilakukan, untuk pengembangan berikutnya, disarankan beberapa hal berikut ini:

a) Kontroler Jaringan Syaraf Tiruan perlu ditambahkan juga untuk mengontrol gerak translasi dari sumbu x,y, dan z agar respon kecepatan yang dihasilkan semakin cepat.
b) Jika masih menggunakan kontroler APM perlu mengetahui *open source* programnya agar dapat dipelajari dan menulis program yang tepat pada APM.
c) Jika mengganti kontroler APM dengan kontroler lainnya seperti arduino, perlu tambahan sensor seperti GY-80 untuk membaca posisi, kecepatan, percepatan dan arah dari quadcopter .

DAFTAR PUSTAKA

[1] *GY-80 orientation sensor on a raspberry Pi*. (2014, Januari 10). Dipetik 12 9, 2015, dari Astro-Beano: astrobeano.blogspot.co.id

[2] *Quadcopter*. (2014, November Sabtu). Dipetik Desember Rabu, 2015, dari Zona Elektro: zonaelektro.net

[3] Bresciani, T. (2008). Modelling, Identification and Control of a Quadrotor Helicopter. Lund Sweden: Lund University.

[4] Drs.Jong Jek Siang, M. (2004). Jaringan Syaraf Tiruan & Pemrogramannya menggunakan MATLAB. Yogyakarta: Penerbit ANDI.

[5] Gamayanti, N. (t.thn.). *Kontroler tipe PD dan PID.* Surabaya: ITS.

[6] Sarwono, D. F. (2014). *AUTOMATIC SAFE LANDING MENGGUNAKAN KONTROL OPTIMAL LINEAR QUADRATIC TRACKING PADA UNMANNED AERIAL VEHICLE QUADCOPTER.* Surabaya: ITS.

[7] Mark W.Spong, M. (1976). Robot Dynamics and Control. United States.

Halaman ini sengaja dikosongkan

LAMPIRAN A

A1. Program MATLAB Matrik PID linierisasi

```
function out=konstanta_PID(in)
global p q k l
p=in(1);
q=in(2);
k=1;
if p>0.04 && p<0.13 || p<-0.04 && p>-0.13
   k=2;
elseif p>=0.13 || p<=-0.13
   k=3;
end
l=1;
if q>0.04 && q<0.13 || q<-0.04 && q>-0.13
   l=2;
elseif q>=0.13 || q<=-0.13
   l=3;
end

KpRoll=[21.7838 62.873 33.97;16.35  64.466 84.52;27.96
37.5775 56.08];
KiRoll=[0.72684 6.1956 34.54;0.9575 236.5  347.3;31.77
86.4195 115.5];
KdRoll=[-1.9259 -1.594 0.627;0      0      0    ;0     0       0];

KpPitch=[1.3024 43.63 17.61;  2.489  40.85  83.89 ;1.679  35.74
85.02];
KiPitch=[0.0378 41.98 7.015;  0.1398 39.35  161   ;0.0598 680.2
163.2];
KdPitch=[-0.122 -0.134 -0.121;-0.111 -0.176 -0.109;-0.179
0.0595 -0.111];
out(1)=KpRoll(k,l);
out(2)=KiRoll(k,l);
out(3)=KdRoll(k,l);
out(4)=KpPitch(k,l);
```

```
out(5)=KiPitch(k,l);
out(6)=KdPitch(k,l);
```

A2. Program MATLAB Proses Belajar JST Kp *Roll*

```
function KpRoll=jstKpRoll(in1)
global tt jin1 jeh1 wih1 woh1 lmdh lmdo  p1 q1 KpPIDRoll
KpRollOut biasKpRoll alph1 alpo1 zh1 zo1
KpPIDRoll=in1(3);
tt = in1(4);

if tt==0
   wih1 =ones(jin1,jeh1);
   woh1 =ones(jeh1,1);
   jin1 = 2;
   jeh1 = 2;
   lmdh=1;
   lmdo=1;
   alph1=0.001;
   alpo1=0.001;
   biasKpRoll=21.7838;
end

   %perhitungan unit masukan.
   for x=1:jin1
      xx1(x)=in1(x)
   end
   % Perhitungan forwArd
   % menghitung output layer hiddent
   for j=1:jeh1
     zh1(j)=0;
      for i=1:jin1
         zh1(j)=zh1(j)+wih1(i,j)*xx1(i);
      end
      yh1(j)=lmdh*zh1(j); % fungsi aktivasi linear
   end
   % menghitung output neuron
   zo1=0;
   for j=1:jeh1
      zo1=zo1+woh1(j)*yh1(j);
```

```
end
zo1=zo1+KpPIDRoll;
yn1=lmdo*zo1; % fungsi aktivasi linear
% menghitung eror output
er1=KpPIDRoll-yn1;
% perhitungan backward (revisi bobot)
% revisi bobot dari layer hiddent ke layer output
for j=1:jeh1
   woh1(j)=woh1(j)+alpo1*er1*lmdo*yh1(j);
  % assignin('base','woh11',woh1(1))
  % assignin('base','woh12',woh1(2))

end
% menghitung perambatan eror
for j=1:jeh1
   if zo1==0
      erh1(j)=er1/jeh1;
   else
      erh1(j)=(woh1(j)*yh1(j)/zo1)*er1;
   end
end

% revisi bobot dari layer input ke layer hiddent ke
for j=1:jeh1
   for i=1:jin1
      wih1(i,j)=wih1(i,j)+alph1*erh1(j)*lmdh*xx1(i);
    end
end
KpRoll=yn1
```

A3. Program MATLAB Proses Belajar JST Kd *Roll*

```
function KdRoll=jstKdRoll(in2)
global tt  lmdh lmdo alph2 alpo2  p2 q2  KdRollOut KdPIDRoll
jin2 jeh2 wih2 woh2 biasKdRoll
p2  = in2(1);
%q2  = in2(2);
KdPIDRoll=in2(3);
tt = in2(4);
```

```
% proses inisialisasi
 if tt==0
    jin2 = 2;
    jeh2 = 2;
    wih2 = ones(jin2,jeh2);
    woh2 = ones(jeh2,1);
    lmdh=1;
    lmdo=1;
    alph2=0.4
    alpo2=0.4
    biasKdRoll=-1.9259;
end
    %perhitungan unit masukan.
    for x=1:jin2
        xx2(x)=in2(x)
    end
    % Perhitungan forwArd
    % menghitung output layer hiddent
    for j=1:jeh2
        zh2(j)=0;
        for i=1:jin2
            zh2(j)=zh2(j)+wih2(i,j)*xx2(i);
        end
        yh2(j)=lmdh*zh2(j); % fungsi aktivasi linear
    end
    % menghitung output neuron
    zo2=0;
    for j=1:jeh2
        zo2=zo2+woh2(j)*yh2(j);
    end
    zo2=zo2+biasKdRoll;
    yn2=lmdo*zo2; % fungsi aktivasi linear
    % menghitung eror output
    er2=KdPIDRoll-yn2;

    % perhitungan backward (revisi bobot)
    % revisi bobot dari layer hiddent ke layer output
    for j=1:jeh2
        woh2(j)=woh2(j)+alpo2*er2*lmdo*yh2(j);
```

```
    end
    % menghitung perambatan eror
    for j=1:jeh2
        if zo2==0
            erh2(j)=er2/jeh2;
        else
            erh2(j)=(woh2(j)*yh2(j)/zo2)*er2;
        end
    end

    % revisi bobot dari layer input ke layer hiddent ke
    for j=1:jeh2
        for i=1:jin2
            wih2(i,j)=wih2(i,j)+alph2*erh2(j)*lmdh*xx2(i);
        end
    end
    if p2> 0.05 || p2<-0.05
        yn2=0;
    end
KdRoll=yn2;
```

A4. Program MATLAB Proses Belajar JST Ki *Roll*

```
function KiRoll=jstKiRoll(in3)
global tt  lmdh lmdo alph3 alpo3 wih3 jin3 jeh3 woh3 KiPIDRoll
KiRollOut biasKiRoll
KiPIDRoll=in3(3)
tt = in3(4);
% proses inisialisasi
 if tt==0
    jin3 = 2;
    jeh3 = 2;
    wih3 = ones(jin3,jeh3);
    woh3 = ones(jeh3,1);
    lmdh=1;
    lmdo=1;
    alph3=0.1;
    alpo3=0.1;
    biasKiRoll=0.72684;
end
```

```
%perhitungan unit masukan.
for x=1:jin3
   xx3(x)=in3(x)
end
% Perhitungan forwArd
% menghitung output layer hiddent
for j=1:jeh3
   zh3(j)=0;
   for i=1:jin3;
      zh3(j)=zh3(j)+wih3(i,j)*xx3(i);
   end
   yh3(j)=lmdh*zh3(j);% fungsi aktivasi linear
end
% menghitung output neuron
zo3=0;
for j=1:jeh3
   zo3=zo3+woh3(j)*yh3(j)
end
zo3=zo3+biasKiRoll;
yn3=lmdo*zo3 % fungsi aktivasi linear
% menghitung eror output
er3=KiPIDRoll-yn3

% perhitungan backward (revisi bobot)
% revisi bobot dari layer hiddent ke layer output
for j=1:jeh3
   woh3(j)=woh3(j)+alpo3*er3*lmdo*yh3(j);
end

% menghitung perambatan eror
for j=1:jeh3
   if zo3==0
      erh3(j)=er3/jeh3;
   else
      erh3(j)=(woh3(j)*yh3(j)/zo3)*er3;
   end
end
```

```
    % revisi bobot dari layer input ke layer hiddent ke
    for j=1:jeh3
        for i=1:jin3
            wih3(i,j)=wih3(i,j)+alph3*erh3(j)*lmdh*xx3(i);
        end
    end
KiRoll=yn3;
```

A5. Program MATLAB Proses Belajar JST Kp *Pitch*

```
function KpPitch=jstKpPitch(in4)
global tt jin4 jeh4 wih4 woh4 lmdh lmdo  biasKpPitch
KpPitchOut KpPIDPitch alph4 alpo4
KpPIDPitch=in4(3);
tt = in4(4);
% proses inisialisasi
 if tt==0
    jin4 = 2;
    jeh4 = 2;
    wih4 = ones(jin4,jeh4);
    woh4 = ones(jeh4,1);
    lmdh=1;
    lmdo=1;
    alph4=0.08
    alpo4=0.08
    biasKpPitch=1.3024;
end

    %perhitungan unit masukan.
    for x=1:jin4
        xx4(x)=in4(x);
    end
    % Perhitungan forwArd
    % menghitung output layer hiddent
    for j=1:jeh4
        zh4(j)=0;
        for i=1:jin4
            zh4(j)=zh4(j)+wih4(i,j)*xx4(i);
        end
```

```
    yh4(j)=lmdh*zh4(j); % fungsi aktivasi linear
end
% menghitung output neuron
zo4=0;
for j=1:jeh4
    zo4=zo4+woh4(j)*yh4(j);
end
zo4=zo4+biasKpPitch;
yn4=lmdo*zo4; % fungsi aktivasi linear
% menghitung eror output
er4=KpPIDPitch-yn4;

% perhitungan backward (revisi bobot)
% revisi bobot dari layer hiddent ke layer output
for j=1:jeh4
    woh4(j)=woh4(j)+alpo4*er4*lmdo*yh4(j);
end

% menghitung perambatan eror
for j=1:jeh4
    if zo4==0
        erh4(j)=er4/jeh4;
    else
        erh4(j)=(woh4(j)*yh4(j)/zo4)*er4;
    end
end

% revisi bobot dari layer input ke layer hiddent ke
for j=1:jeh4
    for i=1:jin4
        wih4(i,j)=wih4(i,j)+alph4*erh4(j)*lmdh*xx4(i);
    end
end

KpPitch=yn4;
```

A6. Program MATLAB Proses Belajar JST Ki *Pitch*

```
function KiPitch=jstKiPitch(in6)
global tt  lmdh lmdo  p6 q6 KiPitchOut KiPIDPitch jin6 jeh6
wih6 woh6 biasKiPitch alpo6 alph6
%p6 = in6(1);
%q6 = in6(2);
KiPIDPitch=in6(3);
tt = in6(4);
% proses inisialisasi

if tt==0
   jin6 = 2;
   jeh6 = 2;
   wih6 = ones(jin6,jeh6);
   woh6 = ones(jeh6,1);
   lmdh=1;
   lmdo=1;
   alph6=0.1%0.001;
   alpo6=0.1%0.001;
   biasKiPitch=0;
end

   %perhitungan unit masukan.
   for x=1:jin6
      xx6(x)=in6(x);
   end
   % Perhitungan forwArd
   % menghitung output layer hiddent
   for j=1:jeh6
      zh6(j)=0;
      for i=1:jin6
         zh6(j)=zh6(j)+wih6(i,j)*xx6(i);
      end
      yh6(j)=lmdh*zh6(j); % fungsi aktivasi linear
   end
   % menghitung output neuron
   zo6=0;
   for j=1:jeh6
      zo6=zo6+woh6(j)*yh6(j);
```

```
    end
    zo6=zo6+biasKiPitch;
    yn6=lmdo*zo6; % fungsi aktivasi linear
    %KiPitchOut=yn6;

    % menghitung eror output
    er6=KiPIDPitch-yn6

    % perhitungan backward (revisi bobot)
    % revisi bobot dari layer hiddent ke layer output
    for j=1:jeh6
       woh6(j)=woh6(j)+alpo6*er6*lmdo*yh6(j);
       %assignin('base','woh61',woh6(1))
       %assignin('base','woh62',woh6(2))
    end
    %revisi bias
   % biasKiPitch=biasKiPitch+alpo6*er6;

    % menghitung perambatan eror
    for j=1:jeh6
       if zo6==0
          erh6(j)=er6/jeh6;
       else
          erh6(j)=(woh6(j)*yh6(j)/zo6)*er6;
       end
    end

    % revisi bobot dari layer input ke layer hiddent ke
    for j=1:jeh6
       for i=1:jin6
          wih6(i,j)=wih6(i,j)+alph6*erh6(j)*lmdh*xx6(i);
        %  assignin('base','wih61',wih6(1,1))
        %  assignin('base','wih62',wih6(1,2))
        %  assignin('base','wih63',wih6(2,1))
        %  assignin('base','wih64',wih6(2,2))
       end
    end
KiPitch=yn6
```

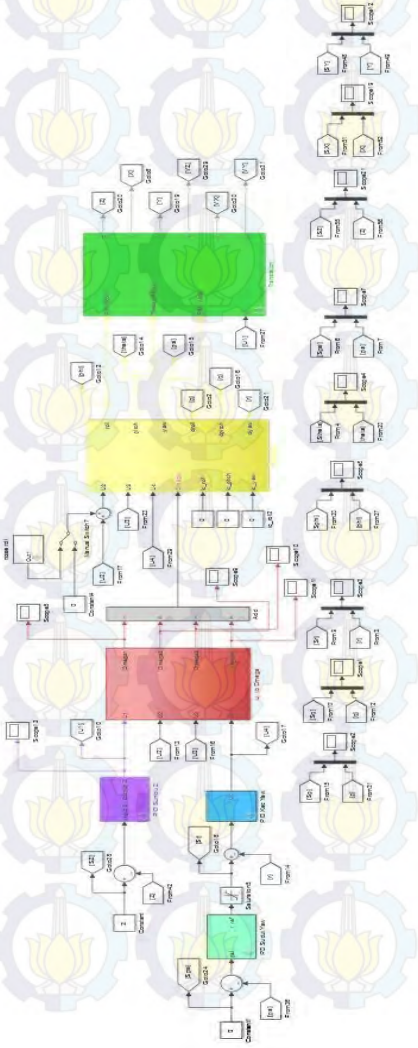
A7. Program MATLAB Proses Belajar JST Kd *Pitch*

```
function KdPitch=jstKdPitch(in5)
global tt jin5 jeh5 wih5 woh5 lmdh lmdo alph5 alpo5  p5 q5
biasKdPitch KdPIDPitch KdPitchOut
KdPIDPitch=in5(3);
tt = in5(4);
% proses inisialisasi
 if tt==0
   jin5 = 2;
   jeh5 = 2;
   wih5 = ones(jin5,jeh5);
   woh5 = ones(jeh5,1);
   lmdh=1;
   lmdo=1;
   alph5=0.6
   alpo5=0.6
   biasKdPitch=-0.122;
end
   %perhitungan unit masukan.
   for x=1:jin5
      xx5(x)=in5(x)
   end
   % Perhitungan forwArd
   % menghitung output layer hiddent
   for j=1:jeh5
      zh5(j)=0;
      for i=1:jin5
         zh5(j)=zh5(j)+wih5(i,j)*xx5(i);
      end
      yh5(j)=lmdh*zh5(j); % fungsi aktivasi linear
   end
   % menghitung output neuron
   zo5=0;
   for j=1:jeh5
      zo5=zo5+woh5(j)*yh5(j);
   end
```

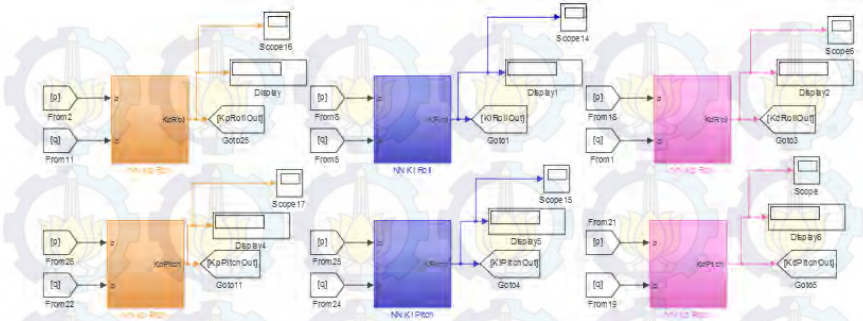
```
    zo5=zo5+biasKdPitch;
    yn5=lmdo*zo5; % fungsi aktivasi linear

    % menghitung eror output
    er5=KdPIDPitch-yn5;
    % perhitungan backward (revisi bobot)
    % revisi bobot dari layer hiddent ke layer output
    for j=1:jeh5
       woh5(j)=woh5(j)+alpo5*er5*lmdo*yh5(j);
    end
    % menghitung perambatan eror
    for j=1:jeh5
       if zo5==0
          erh5(j)=er5/jeh5;
       else
          erh5(j)=(woh5(j)*yh5(j)/zo5)*er5;
       end
    end

    % revisi bobot dari layer input ke layer hiddent ke
    for j=1:jeh5
       for i=1:jin5
          wih5(i,j)=wih5(i,j)+alph5*erh5(j)*lmdh*xx5(i);
       end
    end
KdPitch=yn5;
```

LAMPIRAN B

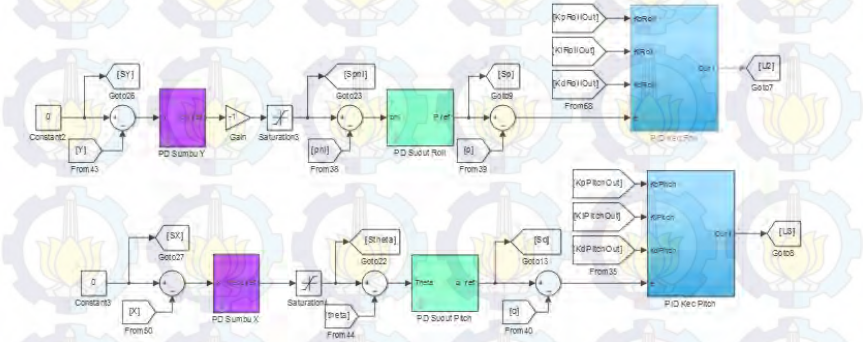
B.1 Simulink *Plant Quadcopter*



B.2 Simulink Jaringan Tiruan



B.3 Simulink Kontroler PID dan PD



RIWAYAT PENULIS

Prihatama Kunto Wicaksono yang biasa dipanggil Tama oleh teman-temannya lahir di Semarang, 29 Mei 1991. Tama merupakan anak kedua dari pasangan Sarwono dan Jimah Sumanti. Lulus dari SDN Tlogosari Kulon 04, kemudian melanjutkan studi ke jenjang lebih lanjut di SMPN 9 Semarang dan lulus pada tahun 2006. Kemudian melanjutkan ke SMAN 11 Semarang dan lulus pada tahun 2009. Pada tahun yang sama penulis melanjutkan kuliah di Politeknik Negeri Semarang dan lulus pada tahun 2012. Setelah lulus kuliah, penulis memutuskan untuk bekerja selama 1 tahun dan melanjutkan kuliah S1 di Institut Teknologi Sepuluh Nopember Surabaya jurusan Teknik Elektro pada tahun 2013. Pada bulan Januari 2015 penulis mengikuti seminar dan ujian Tugas Akhir sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Elektro dari Institut Teknologi Sepuluh Nopember Surabaya.