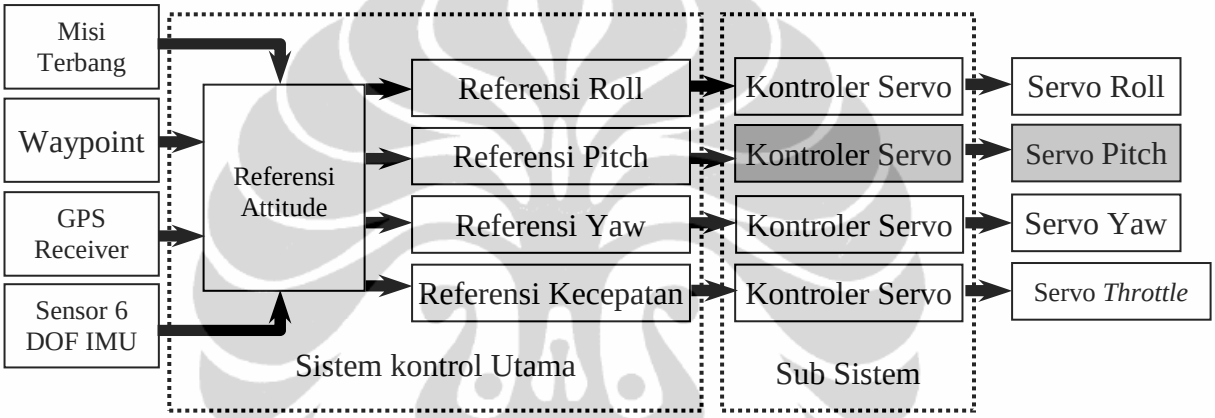
BAB 3
PERANCANGAN KONTROL DENGAN PID TUNING

3.1 Algoritma Kontrol Pada Pesawat Tanpa Awak

Pada makalah seminar dari penulis dengan judul *Pemodelan dan Simulasi Gerak Sirip Pada Pesawat Tanpa Awak* telah dihasilkan model dari gerak sirip elevator, aileron dan rudder. Algoritma kontrol untuk pesawat tanpa awak pada makalah seminar tersebut adalah sebagaimana terlihat pada gambar 3.1 berikut.



Gambar 3.1 Algoritma kontrol untuk pesawat tanpa awak. [6]

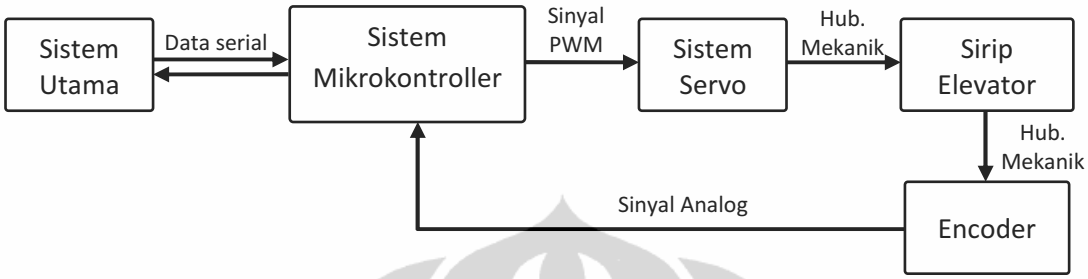
Blok dari sistem kontrol utama dari algoritma kontrol ini akan mengolah data-data dari sensor *attitude* (IMU) dan GPS, kemudian membandingkan dengan waipoint dan misi terbang. Dari data sensor-sensor tersebut, maka blok sistem kontrol utama akan mendapatkan referensi untuk *attitude*/ sikap pesawat, yaitu arah terbang dan ketinggian, yang kemudian diterjemahkan menjadi referensi untuk sikap *roll, pitch, yaw* dan kecepatan pesawat.

Masing-masing referensi dari sikap pesawat ini kemudian akan diumpankan ke masing-masing sub sistem dari bidang kontrol pesawat, yaitu referensi *roll* ke sub sistem kontrol sirip *aileron*, referensi *pitch* ke sub sistem kontrol sirip *elevator*, referensi *yaw* ke sub sistem kontrol sirip *rudder*, dan referensi kecepatan ke sub sistem kontrol *throttle*.

Pada penelitian ini kontrol sirip yang dibuat hanya pada bidang sirip *elevator*, sebagaimana pada bagian yang diarsir pada gambar 3.1. Komunikasi antara sistem utama ke sub sistem kontroler servo pada sirip *elevator*

menggunakan jalur serial, dengan kecepatan transfer data 115200 bps, 8 data bits, no parity, 1 stop bit.

Skematik dari komunikasi antara sistem utama dengan sub sistem kontrol gerak sirip *elevator* tersebut terlihat pada gambar 3.2 berikut.



Gambar 3.2 Diagram blok skematik untuk sistem kontrol sirip *elevator* pada pesawat tanpa awak.

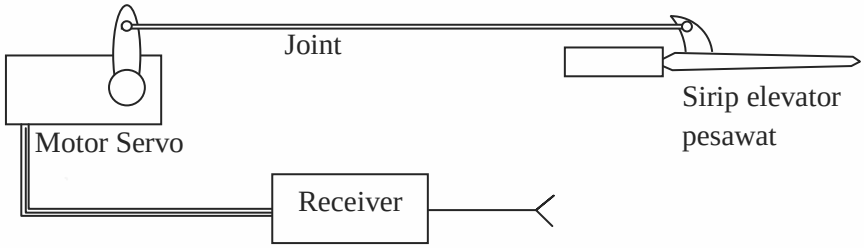
Sistem Mikrokontroller akan menerima data target posisi sudut sirip *elevator* yang dikirim dari sistem utama melalui komunikasi serial. Selanjutnya sistem mikrokontroller akan menterjemahkan data *pitch* tersebut menjadi data *set point* sudut untuk sirip *elevator* pada pesawat tanpa awak.

Setelah mengolah data sudut dengan algoritma kontrol, maka sistem mikrokontroller akan mengirim sinyal kendali berupa sinyal PWM ke sistem servo untuk menggerakkan sirip *elevator* pada pesawat tanpa awak. Perubahan sudut sirip *elevator* pada pesawat tanpa awak ini kemudian akan dibaca oleh encoder yang akan mengirimkan posisi sudut *real* dari sirip elevator pada pesawat tanpa awak ini ke sistem mikrokontroller untuk dijadikan sebagai *feedback* posisi sudut sirip. Selanjutnya sistem mikrokontroller akan mengirimkan posisi *real* sudut sirip *elevator* pada pesawat tanpa awak ke sistem utama melalui jalur komunikasi serial.

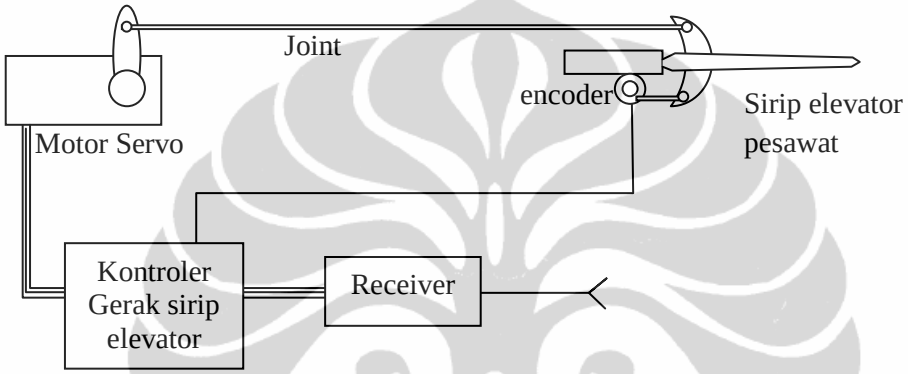
3.1.1 Implementasi Kontroler pada Pesawat Tanpa Awak

Pesawat terbang yang digunakan pada penelitian ini adalah pesawat *aeromodelling* jenis *Trainer-40*. Secara normal, pesawat ini dikendalikan melalui *remote control* oleh pilot dari bawah (*ground*).

Desain posisi pengendali sirip elevator dapat dilihat sebagaimana pada gambar 3.3 berikut.



Gambar 3.3 Posisi kontroler kendali sirip elevator pada pesawat aeromodelling tanpa kontroler

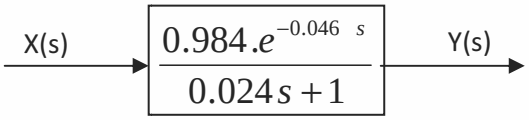


Gambar 3.4 Posisi kontroler kendali sirip elevator pada pesawat aeromodelling dengan kontroler.

Pada gambar 3.4 tersebut memperlihatkan posisi kontroler gerak sirip elevator. Receiver pada sistem ini berfungsi untuk menggantikan sistem kontroler utama dari sistem pesawat tanpa awak yang berfungsi untuk memberikan *set point* untuk posisi sudut sirip pesawat.

3.2 Model Gerak Sirip Elevator pada Pesawat Tanpa Awak

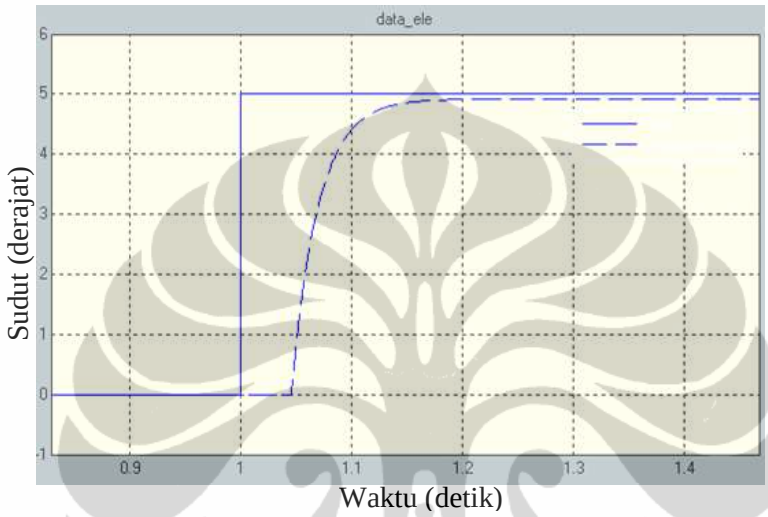
Model gerak sirip *elevator* pada pesawat tanpa awak yang digunakan pada penelitian ini sebagaimana telah didapatkan dari makalah penelitian *Pemodelan dan Simulasi Gerak Sirip Pada Pesawat Tanpa Awak,* yang diturunkan dari pengambilan respon *step* dari sirip elevator pada pesawat aeromodelling tipe *trainer*-40 dengan menggunakan metode *empirical model identification* adalah sebagai berikut: [6]



…… (3.1)

Dengan Kp = 0.984, τ = 0.024 dan θ = 0.046

Step respons untuk model gerak sirip *elevator* ini adalah sebagaimana terlihat pada gambar 3.5 berikut.



Gambar 3.5 *Step respons* model gerak sirip elevator

3.3 PID *Tuning*

Untuk menentukan parameter-parameter kendali Kc, Ti dan Td digunakan teknik dari *ciancone*-Marlin metode II. Digunakan teknik ini karena kehandalannya telah diakui [2].

Sebagaimana telah diuraikan pada bab 2 dalam sub bab 2.6, untuk dapat melakukan *tuning* dengan teknik *ciancone* untuk kontroler PID ini, maka harus mengikuti langkah-langkah sebagaimana telah disebutkan.

1) Mendapatkan nilai-nilai dari Kp, θ dan τ dari model dinamik sistem dengan menggunakan metode *empirical*.

   Dari sub bab 3.2 dapat dilihat, nilai dari :

   Kp = 0.984, τ = 0.024 dan θ = 0.046 …… (3.2)

2) Menghitung *fraction dead time*, θ/(θ+τ).

   θ/(θ+τ) = 0.046/(0.046 + 0.024)

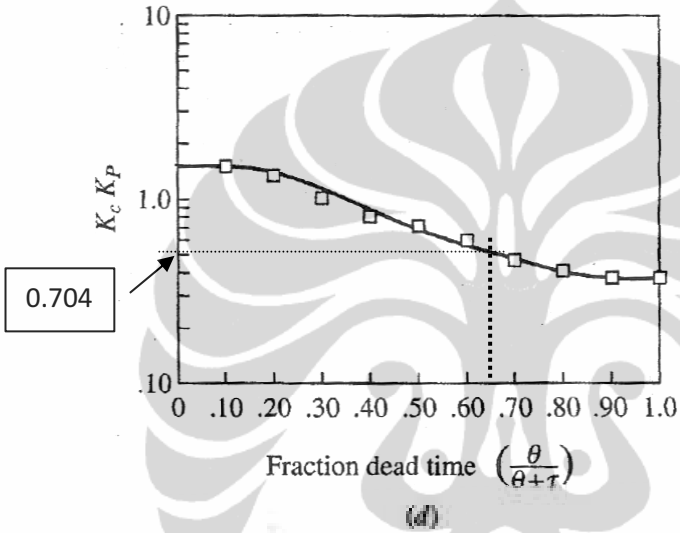
   = 0.6571 …… (3.3)

3) Pilih tabel yang sesuai, dengan *disturbance respons* atau *set point respons*.

   Model gerak sirip *elevator* ini dibuat berdasarkan *set point respons*, maka tabel *ciancone* yang digunakan adalah tabel untuk *set point respons.*

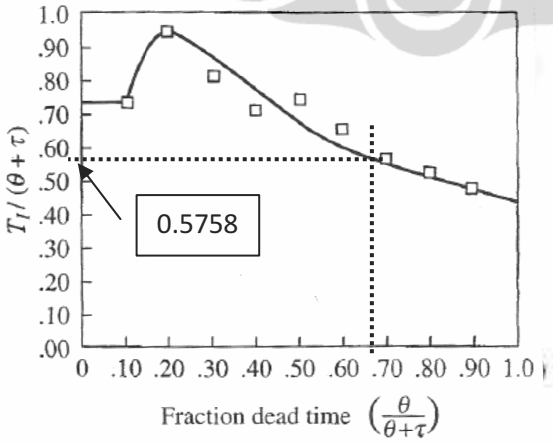
4) Tentukan nilai dari *dimensionless tuning* dari grafik untuk KcKp, Ti/(θ+τ) dan Td/(θ+τ).

   Dengan tabel ciancone,

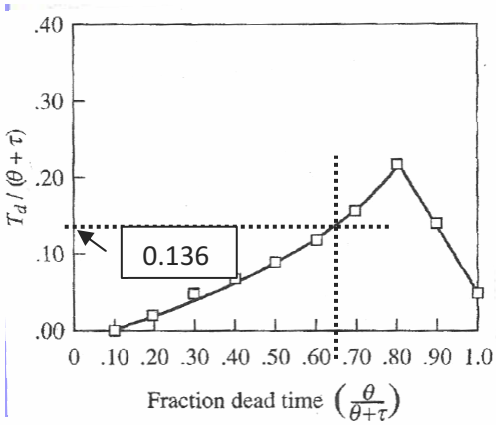


θ/(θ+τ) = 0.6571

KcKp = 0.704 …….. (3.4)



Ti/(θ+τ) = 0.5758 ……. (3.5)



$Td/(\theta+\tau) = 0.1360$ ……. (3.6)

5) Hitung *dimensional tuning controller*. Misal: Kc=(KcKp)/Kp.

$Kc = KcKp/Kp$

$= 0.704/0.984$

$Kc = 0.715$ …… (3.7)

$Ti/(\theta+\tau) = 0.5758$

$Ti = Ti/(\theta+\tau) * (\theta+\tau)$

$= 0.5758 * 0.6571$

$Ti = 0.378$ …… (3.8)

$Td/(\theta+\tau) = 0.1360$

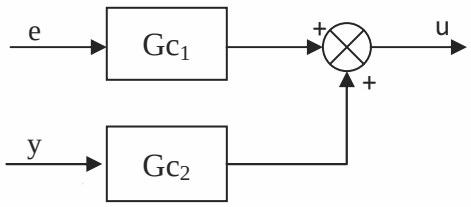
$Td = Td/(\theta+\tau) * (\theta+\tau)$

$= 0.1360 * 0.6571$

$Td = 0.089$ …… (3.9)

6) Implementasikan ke dalam kontroler.

Desain model PID *tuning* adalah sebagaimana terlihat pada gambar 3.6 berikut:



Gambar 3.6 Desain model PID tuning

Dari parameter-parameter model yang telah didapatkan tersebut maka didapat model dari PID *Tuning* tersebut:

$$Gc_1 = Kc(1+\frac{1}{(Ti+s)})$$

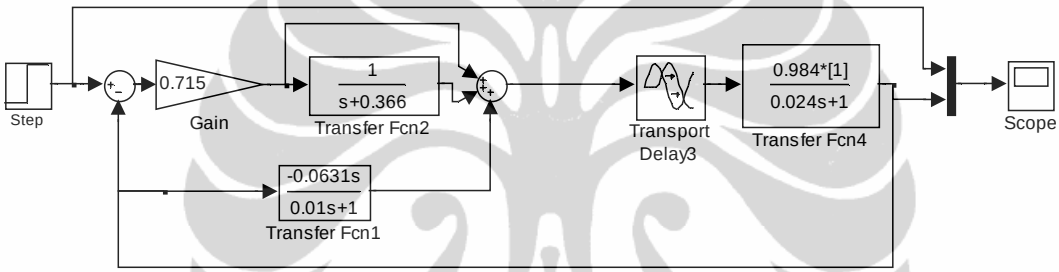
$$= 0.715 .(1 + \frac{1}{(0.366+s)}) \quad \text{..... (3.10)}$$

$$Gc_2 = -Kc \cdot Td \cdot s$$

$$= -0.715 * 0.089\ s$$

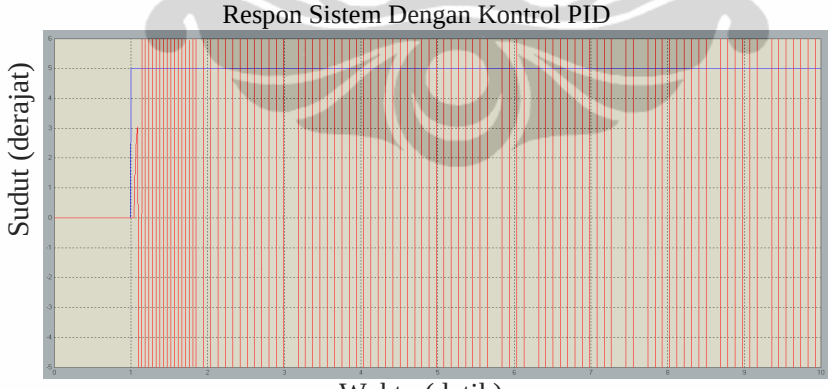
$$= -0.0631s \quad \text{..... (3.11)}$$

Dengan demikian, desain dari model sistem dengan PID *tuning* adalah sebagaimana terlihat pada gambar 3.7 berikut:



Gambar 3.7 Desain model sistem dengan PID tuning

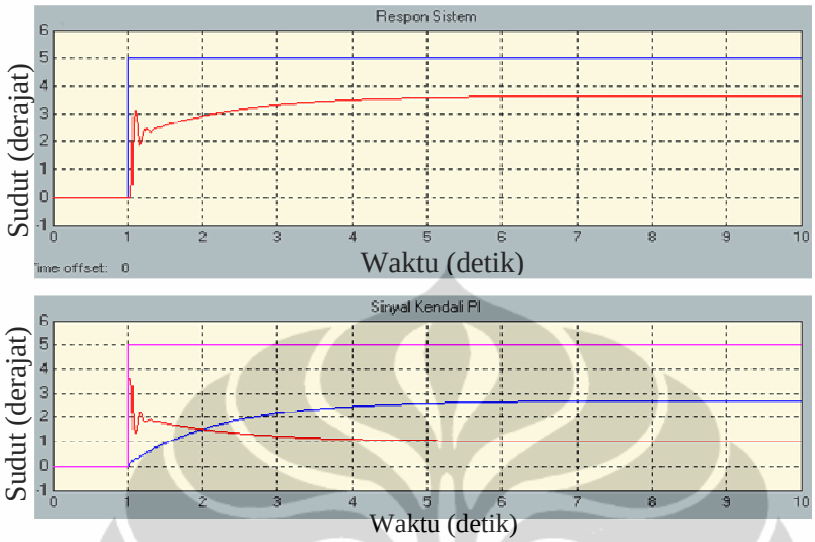
Hasil simulasi dari model tersebut dapat dilihat pada gambar 3.8 berikut.



Gambar 3.8 Hasil simulasi kontrol PID

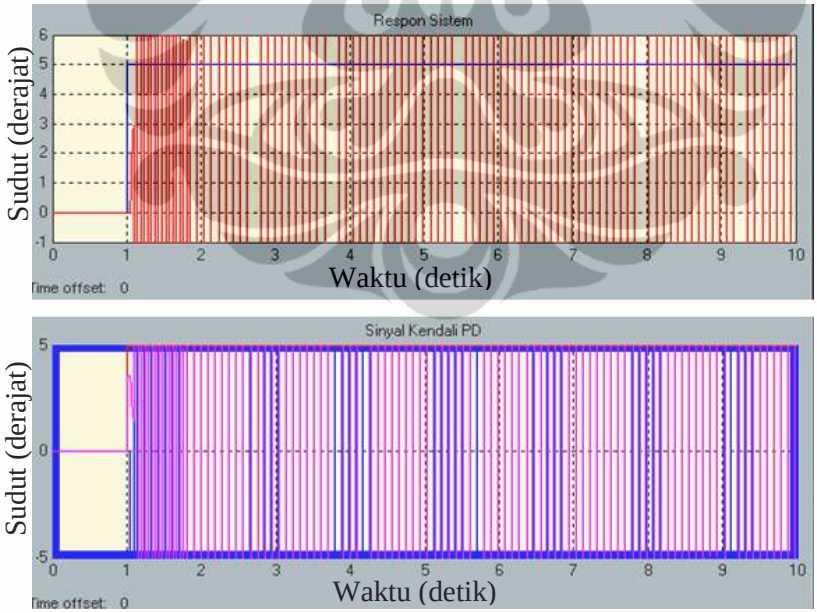
Dari hasil simulasi terlihat kontrol PID kurang optimal, bahkan menyebabkan sistem menjadi tidak stabil. Untuk mengetahui penyebab ketidakstabilan sistem, maka dicoba dengan menghilangkan salah satu dari komponen pengendali.

Langkah pertama dengan menghilangkan komponen *derivative* dari pengendali. Hasil plot sistem dan sinyal kendali dapat dilihat pada gambar 3.9 berikut.



Gambar 3.9 Hasil simulasi pengendali PI

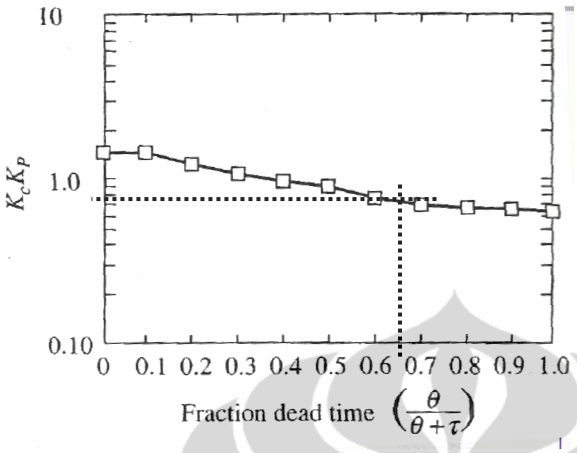
Langkah kedua dengan menghilangkan unsur integral dari pengendali, sehingga respons sistem menjadi seperti pada gambar 3.10 berikut.



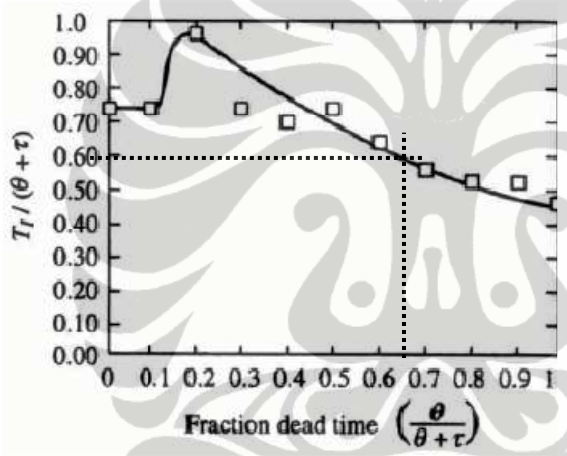
Gambar 3.10. Hasil simulasi pengendali PD

Dari analisa sinyal kendali, maka terlihat bahwa yang menyebabkan sistem menjadi tidak stabil adalah dari kontrol *derivative*. Dengan demikian, maka kontrol PID tidak cocok untuk sistem ini. Mencoba mengulang dari langkah ke tiga dengan menggunakan tabel ciancone untuk kontrol PI.

4) Tentukan nilai dari *dimensionless tuning* dari grafik untuk KcKp, Ti/(θ+τ) dan Td/(θ+τ).



KcKp = 0.7



Ti/(θ+τ) = 0.585

5) Hitung *dimensional tuning controller*. Misal: Kc=(KcKp)/Kp.

Kc = KcKp/Kp

= 0.7/0.984

Kc = 0.711 …… (3.12)

Ti/(θ+τ) = 0.585

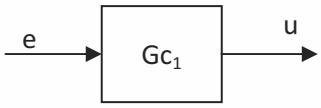
Ti = Ti/(θ+τ) * (θ+τ)

= 0.585 * 0.6571

Ti = 0.384 …… (3.13)

6) Implementasikan ke dalam kontroler.

Desain model PI *tuning* adalah sebagaimana terlihat pada gambar 3.11 berikut:



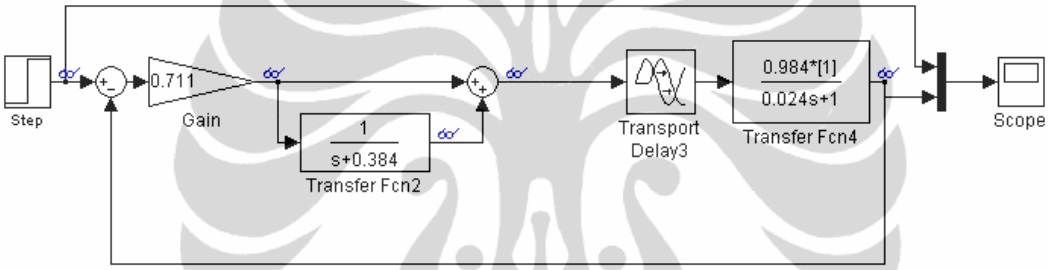
Gambar 3.11 Desain model PI tuning

Dari parameter-parameter model yang telah didapatkan tersebut maka didapat model dari PI *Tuning* tersebut:

$$Gc_1 = Kc(1+\frac{1}{(Ti+s)})$$

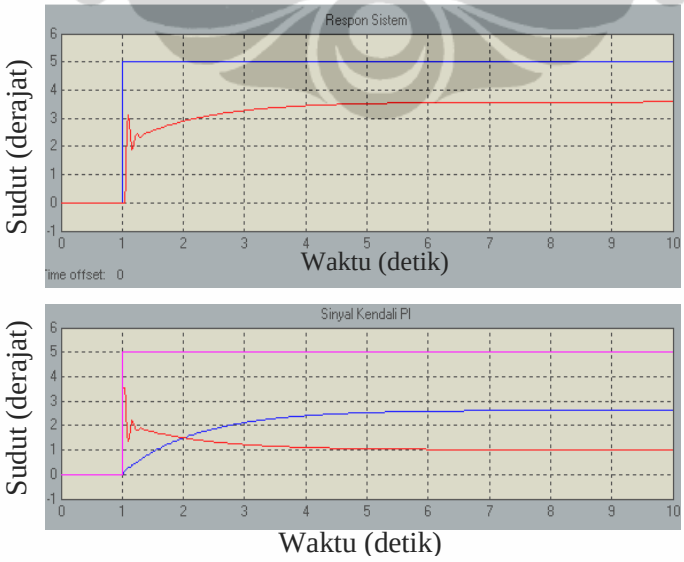
$$= 0.711\ .(1+\frac{1}{(0.384+s)}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.14)$$

Sehingga desaim dari model kontrol PI tersebut adalah sebagaimana terlihat pada gambar 3.12 berikut.



Gambar 3.12 Desain model sistem dengan kontrol PI

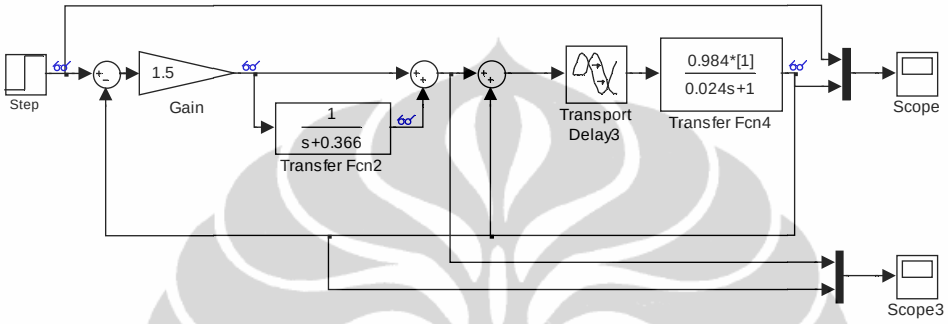
Hasil simulasi dari kontrol PI tersebut adalah sebagaimana terlihat pada gambar 3.13 berikut.



Gambar 3.13 Hasil simulasi kendali PI

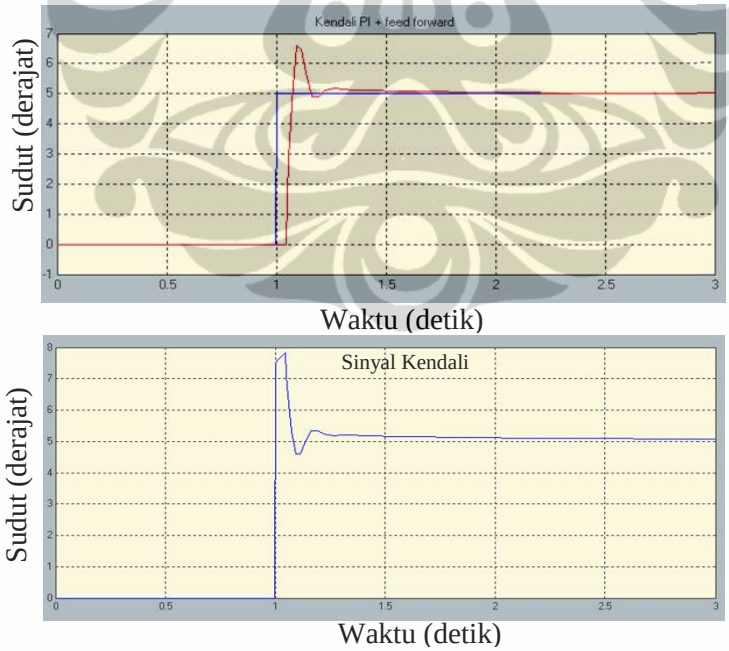
Dari simulasi kendali tersebut dapat dilihat bahwa desain kendali PI pada gerak sirip *elevator* ini belum maksimal, dikarenakan kontrol tersebut tidak dapat memacu sinyal keluaran menuju ke set point.

Untuk dapat memaksimalkan kendali PI ini, maka ditambahkan sebuah sinyal *feed forward* dari keluaran sistem ke sinyal kendali, serta menaikkan nilai gain menjadi 1.5. Dengan demikian desain model sistem untuk kendali PI dengan *feed forward* ini sebagaimana terlihat pada gambar 3.14 berikut.



Gambar 3.14 Desain kendali PI dengan *feed forward* sinyal keluaran sistem

Hasil simulasi dari kendali PI plus *feed forward signal* tersebut sebagaimana terlihat pada gambar 3.15 berikut.



Gambar 3.15 Hasil simulasi kendali PI + *feed forward signal*

Dari hasil simulasi tersebut dapat dilihat bahwa kendali PI dengan *feed forward* ini mampu mengendalikan model sistem gerak sirip *elevator* pesawat dengan baik. Keluaran sinyal kendali pada gambar 3.15 juga menunjukkan bahwa

sinyal kendali ini tidak terlalu besar sehingga mampu untuk diterapkan pada sistem gerak sirip *elevator*.

3.4 Implementasi Kontroler PI Pada Mikrokontroller

Pada implementasinya di mikrokontroler, karena mode pengambilan data pada mikrokontroler adalah mode diskrit, dimana pada proses pengambilan dan pengolahan data terdapat sebuah *delay time*, sehingga dilakukan perhitungan Kc dan Ti dengan memperhatikan waktu *delay time* ini. Untuk perhitungan nilai Kc dan Ti digunakan nilai θ yang baru [2]:

$\theta' = \theta + \Delta t/2$ …….. 3.15

Dengan:

θ’ = nilai θ yang baru

Δt = nilai *delay time* proses pengambilan dan pengolahan data

Sehingga nilai untuk θ’ adalah:

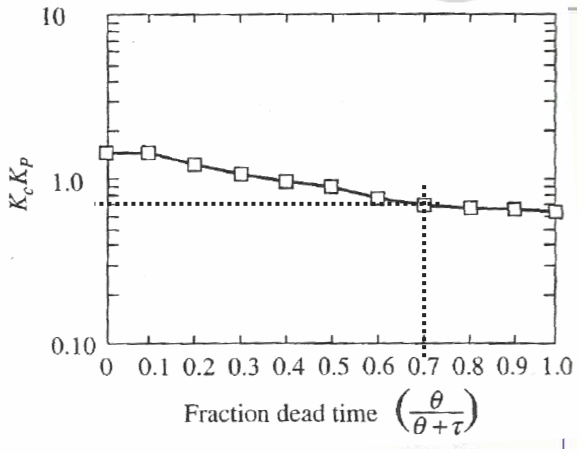
Δt = 0.02 s

θ’ = 0.046 + 0.01

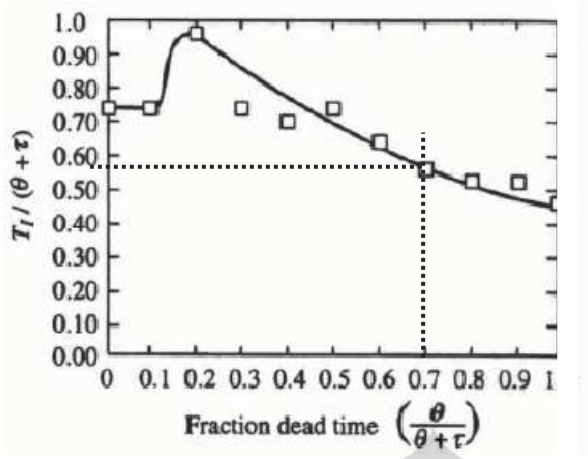
θ’ = 0.056 …… 3.16

$\theta'/(\theta' + \tau) = 0.7$ …… 3.17

Dengan demikian, harus dilakukan perhitungan ulang untuk nilai Kc dan Ti untuk implementasi pada mikrokontroler. Menentukan nilai dari *dimensionless tuning* dari grafik untuk KcKp, Ti/(θ+τ) dan Td/(θ+τ).



KcKp = 0.6



Ti/(θ+τ) = 0.56

7) Hitung *dimensional tuning controller*. Misal: Kc=(KcKp)/Kp.

Kc = KcKp/Kp

= 0.6/0.984

Kc = 0.61 …… (3.18)

Ti/(θ+τ) = 0.56

Ti = Ti/(θ+τ) * (θ+τ)

= 0.56 * 0.6571

Ti = 0.368 …… (3.19)

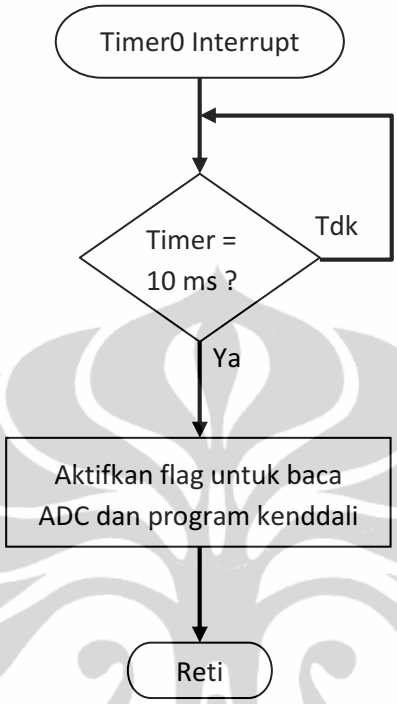
3.5 Desain Diagram Alir Pemrograman pada Mikrokontroler ATMega32

Mikrokontroler yang digunakan dalam penelitian ini adalah mikrokontroler jenis ATMega 32, dengan frekwensi kristal 16 MHz. Pin yang digunakan adalah:

- PortA.0 : masukan ADC dari *encoder* sirip *elevator*.
- PortD.0 : masukan data serial dari sistem utama (komputer) berupa data untuk *set point.*
- PortD.1 : keluaran data serial menuju ke sistem utama, berupa data untuk posisi sudut sirip.
- PortD.5 : Keluaran sinyal PWM menuju ke motor servo elevator.

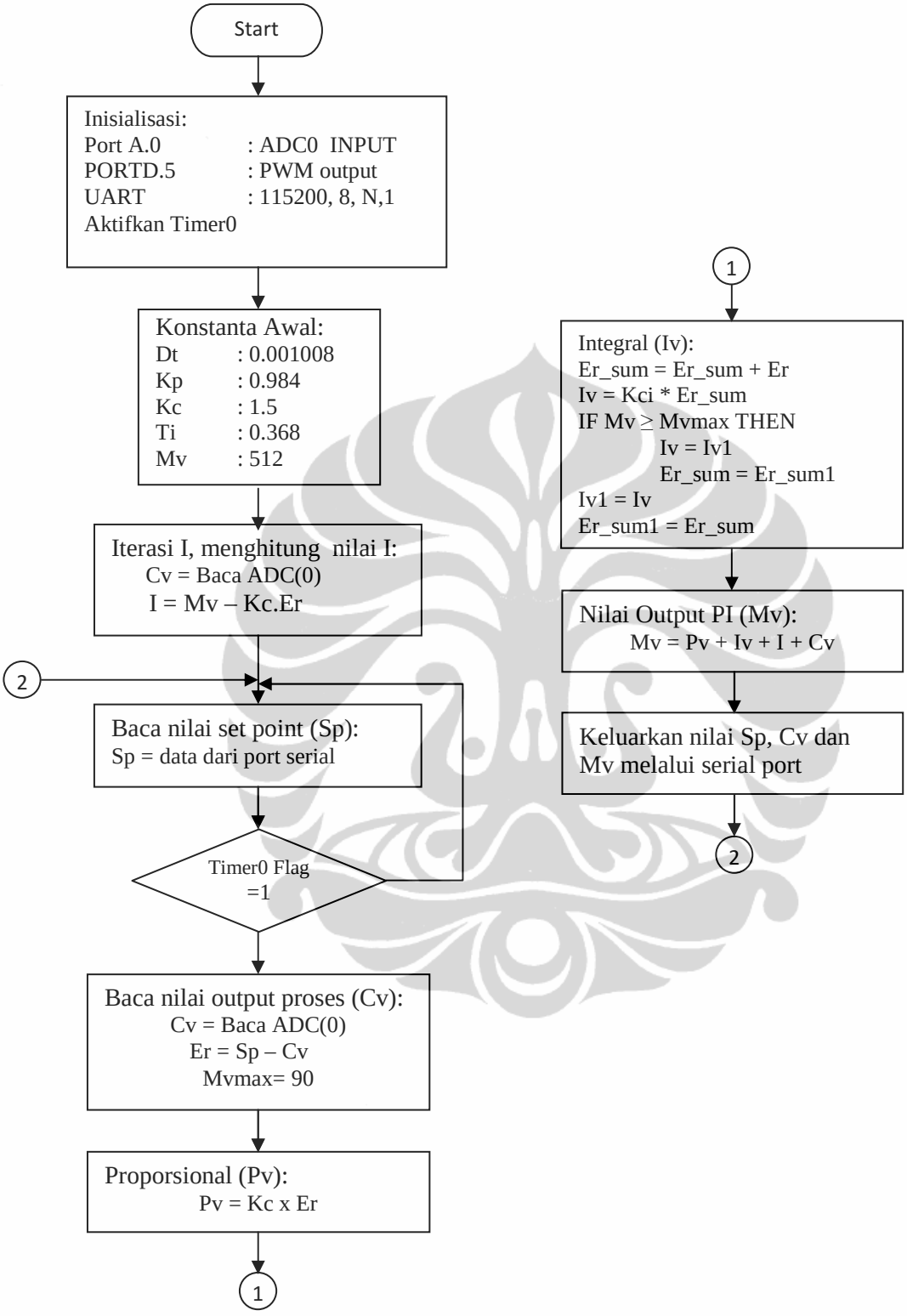
Diagram alir untuk pemrograman mikrokontoler ini dibagi dalam dua bagian, yaitu diagram alir untuk penentuan waktu (pewaktu/ timer) untuk periode

pengambilan data serta diagram alir untuk pengolahan data dan kendali PI. Diagram alir untuk pewaktu periode pengambilan data sebagaimana terlihat pada gambar 3.16 berikut.



Gambar 3.16 Diagram alir pewaktu periode pengambilan data

Diagram alir untuk pengolahan data dan kendali PI dapat dilihat pada gambar 3.17 berikut.



Gambar 3.17 Diagram alir pengolahan data dan kendali PI