PERANCANGAN *SELF-BALANCING ROBOT* MENGGUNAKAN LOGIKA *FUZZY* UNTUK *TUNING* PARAMETER KENDALI PROPORSIONAL INTEGRAL DERIVATIF

Satria Nur Cahya[*)], Wahyudi, and Sumardi

Fakultas Teknik, Universitas Diponegoro,
Jl. Prof. Sudharto, SH., Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)] *E-mail: sath.tjahya@gmail.com*

Abstrak

Self-balancing robot merupakan sebuah robot yang mempunyai dua buah roda sejajar di sisi kiri dan kanan badan robot yang dapat seimbang dengan adanya sebuah kendali. Tugas akhir ini menerapkan kendali PID sebagai sistem regulator pada plant self-balancing robot yang memiliki prinsip kerja mirip dengan pendulum terbalik. Dalam pembuatan tugas akhir self-balancing robot menggunakan logika fuzzy untuk tuning parameter kendali PID. Kecepatan putaran dua motor DC yang digunakan sebagai penggerak dapat diatur dengan mengatur tegangan masukan. Sensor Inertial Measurement Unit (IMU) MPU-6050 digunakan sebagai pendeteksi sudut kemiringan self-balancing robot. Complementary filter digunakan untuk mengurangi nilai error yang dihasilkan oleh sensor IMU MPU-6050. Penerapan kendali PID dan sistem secara keseluruhan di aplikasikan di dalam mikrokontroler ATMega16. Dari hasil pengujian didapatkan bahwa nilai konstanta logika fuzzy untuk tuning kendali PID yaitu konstanta parameter Kp (ZkpSB=10,2; ZkpB=9,8; ZkpS=9,4; dan ZkpK=9), konstanta parameter Ki (Ki=100), dan konstanta Kd (ZkdSB=62; ZkdB=58; ZkdS=54; dan ZkdK=50). Self-balancing robot dapat menyeimbangkan diri ketika tanpa gangguan dan dengan gangguan. Pada pengujian dengan gangguan searah jarum jam (CW) sudut maksimal yang masih dapat diseimbangkan oleh sistem adalah sebesar $15{,}69^0$, sedangkan pada gangguan berlawanan arah jarum jam (CCW) sebesar $-9{,}14^0$.

Kata Kunci: fuzzy, *MPU-6050, kendali PID, self-balancing robot*

Abstract

Self-balancing robot is a robot that has two wheels aligned on both side of the robot which will balanced by the the existence of a control.This final project applying PID control as a system regulator on plant self-balancing robot that has a working principle similar to the inverted pendulum. In the making of final project self-balancing robot using fuzzy logic to tuning PID control parameters. The speed of two DC motors that are used as activator can be set by regulating the input voltage.. The Inertial Measurement Unit (IMU) MPU-6050 sensor is used for detecting angle of self-balancing robot. Complementary filter is used to reduce the error that produced by the IMU MPU-6050 sensor. The implementation of PID control and overall system applied in the ATMega16 microcontroller. From the test results obtained that the constant value for the fuzzy logic to tuning PID control are parameters constants Kp (ZkpSB = 10.2; ZkpB = 9.8; ZkpS = 9.4; and ZkpK = 9), parameter constants Ki (Ki = 100), and the constant Kd (ZkdSB = 62; ZkdB = 58; ZkdS = 54; and ZkdK = 50). Self-balancing robot can balance themselves when without interference and with interference. On testing with a clockwise (CW) interference the maximum angle which is balanced by the system equal to 15.69^0, while the counter clockwise (CCW) interference equal to -9.14^0.

Keywords: fuzzy, MPU-6050, PID control, self-balancing robot

1. Pendahuluan

Dunia robotika di Indonesia baru - baru ini mengalami perkembangan yang begitu pesat. Hal ini tidak luput dari campur tangan praktisi robot yang mempunyai kreatifitas dan inovasi yang tinggi. Menurut buku *The Robot Builder's Bonanza* yang ditulis oleh Gordon McComb dan Myke Predko, difinisi robot adalah *"Robots are often modeled after human, if not in form then at least in function. For decades, scientist and experimenters have tried to duplicate the human body, to create machines with intelligence, strength, mobility, and auto-sensory mechanisms"*[1]. Bahwa robot adalah sebuah piranti mekanik yang mampu melakukan pekerjaan manusia atau berperilaku seperti manusia. *Self-balancing robot*

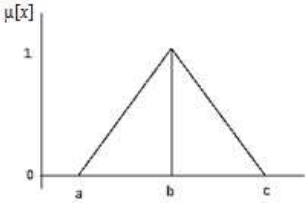
mempunyai prinsip kerja yang sama dengan pendulum terbalik yang diletakkan di atas kereta beroda[2]. Keluaran dari sensor MPU-6050 (*accelerometer* dan *gyroscope*) masih memiliki *noise* sehingga diperlukan sebuah *filter* digital untuk memperbaiki kelemahan keluaran kedua buah sensor yaitu *complementary filter*. Didalam *complementary filter* ini terdapat algoritma yang terdiri dari *low-pass filter*, integral, dan *high-pass filter* [3].

Metode kendali yang digunakan pada *self-balancing robot* ini adalah kendali Proporsional Integral Derivatif (PID) yang nilai parameter kendali PID didapat dari perhitungan logika *fuzzy*.

2. Metode

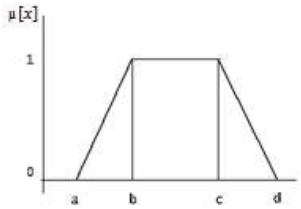
2.1 Fuzzy Logic Control

Terdapat beberapa fungsi keanggotaan yang sering digunakan dalam aplikasi logika *fuzzy* yaitu fungsi keanggotaan segitiga dan fungsi keanggotaan trapesium[4][5].



Gambar 1. Grafik fungsi keanggotaan segitiga

$$\mu[x]=\begin{cases} 0; & x \le a \text{ atau } x \ge c \\ \frac{x-a}{b-a}; & a \le x \le b \\ \frac{c-x}{c-b}; & b \le x \le c \end{cases} \quad (1)$$



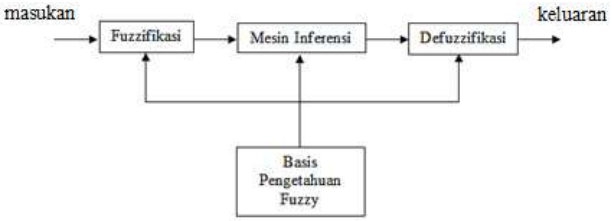
Gambar 2. Grafik fungsi keanggotaan trapesium

$$\mu[x]=\begin{cases} 0, & x \le a \text{ atau } x \ge d \\ \frac{x-a}{b-a}, & a \le x \le b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c \le x \le d \end{cases} \quad (2)$$

Model *fuzzy* sugeno (TSK) ini merupakan pendekatan sistematis pembangkit aturan *fuzzy* dari himpunan data masukan dan keluaran yang diberikan[6].

$$if\ x\ is\ A\ AND\ y\ is\ B\ then\ z = f(x,y) \quad (3)$$

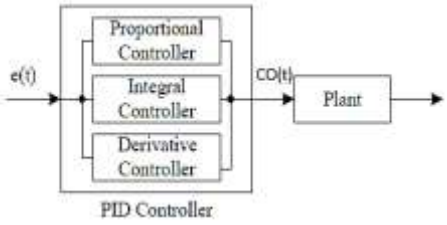
Struktur dasar logika *fuzzy* dapat dilihat pada Gambar 3.



Gambar 3. Struktur dasar logika fuzzy

2.2 Kendali Proporsional, Integral dan Derivatif (PID)

Kontroler PID merupakan gabungan dari tiga macam kendali, yaitu kendali proporsional (*proportional controller*), kendali integral (*integral controller*), dan kendali turunan (*derivative controller*)[6].



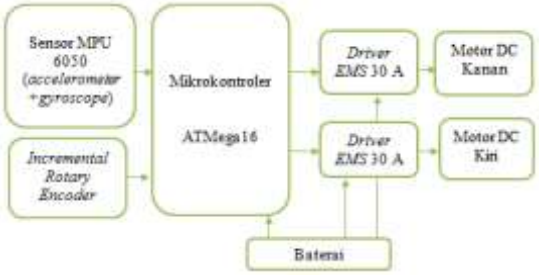
Gambar 4. Diagram blok kendali PID ideal

Bentuk umum persamaan kendali PID dapat dilihat dalam persamaan 4 berikut.

$$CO(t) = \left\{K_p\, e(t) + K_i \int_1^0 e(t) + K_d\, \frac{de(t)}{dt}\right\} \quad (4)$$

2.3 Perancangan Perangkat Keras

Secara umum perancangan perangkat keras *self-balancing scooter* dapat dilihat pada diagram blok Gambar 5.



Gambar 5. Diagram blok perancangan perangkat keras

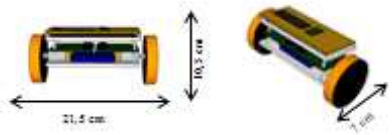
Diagram blok perancangan perangkat keras dapat dilihat pada Gambar 5. tiap-tiap bagian dari diagram blok sistem dapat dijelaskan sebagai berikut :

1. Sensor MPU-6050 merupakan sensor yang digunakan untuk mendeteksi sudut kemiringan *self-balancing robot* tegak lurus terhadap permukaan bumi.

2. *Incremental rotary encoder* merupakan sensor yang biasanya digunakan untuk menghitung sudut, posisi, kecepatan, akselerasi, dan jarak.
3. Mikrokontroler ATMega16 berfungsi sebagai pusat pengendali pada self-balancing robot.
4. Driver motor EMS 30A H-Bridge sebagai driver motor yang berfungsi sebagai pengendali untuk menggerakkan dua buah motor DC.
5. Motor DC berfungsi sebagai sistem penggerak self-balancing robot beroda dua.
6. Baterai berfungsi sebagai catu daya keseluruhan pada sistem self-balancing robot.

2.3.1 Desain Mekanik

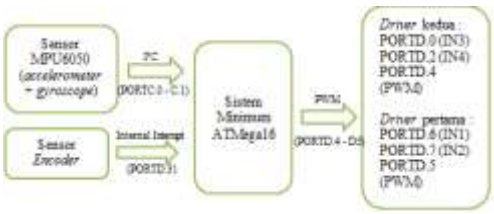
Desain mekanik dalam pembuatan *self-balancing robot* ini diusahakan seimbang pada titik pusat bebannya. Perbedaan berat antara bagian depan dan bagian belakang mempengaruhi kestabilan sistem dan akan mempersulit pengendalian *self-balancing robot.*



Gambar 6. Ilustrasi mekanik *self-balancing robot* tampak depan dan samping

2.3.2 Sistem Mikrokontroler ATMega16

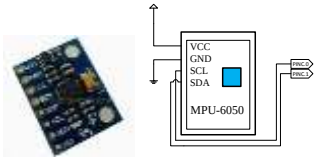
Mikrokontroler ATMega16 berfungsi sebagai pusat pemrosesan data dari sensor MPU-6050 dan sensor *encoder*. Pada Gambar 7 terlihat penggunaan (alokasi) *port* mikrokontroler yang digunakan. PORTC.0 & PORTC.1 digunakan sebagai masukan yang berasal dari sensor MPU-6050 melalui komunikasi I^2C, PORTD.3 digunakan sebagai masukan dari sensor *encoder*, sedangkan PORTD.0, PORTD.2, PORTD.4 hingga PORTD.7 dihubungkan ke dua buah *driver* motor EMS 30A *H-Bridge*.



Gambar 7. Alokasi *port* mikrokontroler ATMega16

2.3.3 Sensor MPU-6050

Sensor ini mempunyai dua buah keluaran yaitu SCL dan SDA masing-masing dihubungkan ke PC.0 dan PC.1[17].



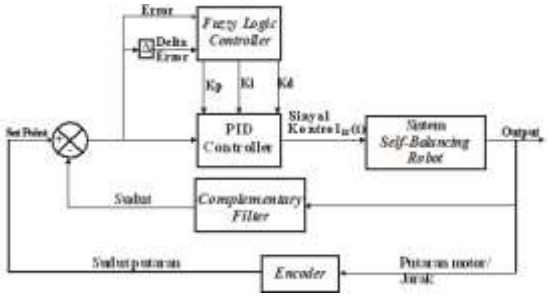
Gambar 8. Sensor MPU-6050 *3-Axis Accelerometer+ 3-Axis Gyroscope* dan Konfigurasinya

2.3.4 *Incremental Rotary Encoder*

Incremental rotary encoder dalam *self-balancing robot* ini digunakan untuk membantu sistem dalam menyeimbangkan robot. *Incremental rotary encoder* dalam robot ini menggunakan sensor optik untuk mengubah posisi dari jumlah putaran roda ke dalam kode digital yang berupa serial pulsa[7].

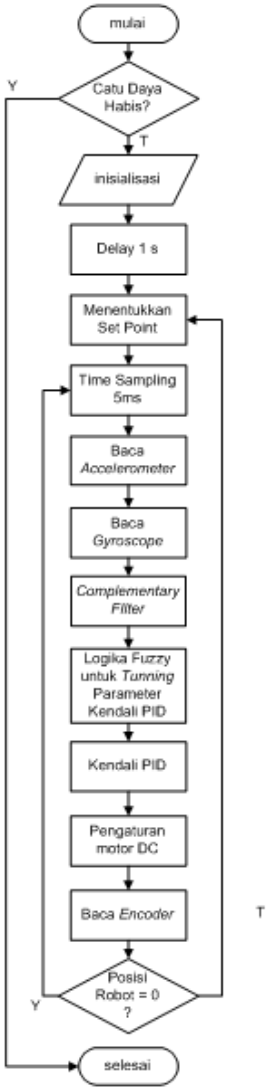
2.4 Perancangan Perangkat Lunak

Metode kendali yang digunakan untuk aplikasi ini adalah kendali Proporsional Integral Derivatif, dengan referensi berasal dari kemiringan sudut. Robot ini menggunakan logika *fuzzy* untuk *tuning* parameter kendali PID.



Gambar 9. Diagram blok sistem kendali *self-balancing robot*

2.4.1 Perancangan Perangkat Lunak



Gambar 10. Flowchart program utama

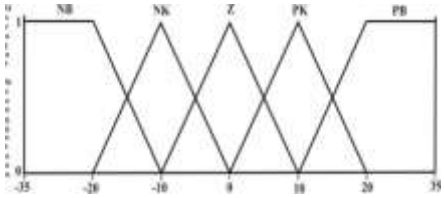
Terlihat pada Gambar 10 adalah flowchart program utama self-balancing robot menggunakan metode kendali PID yang parameter kendali PID ditentukan oleh logika fuzzy.

Program utama dari perancangan perangkat lunak *self-balancing robot* ini meliputi beberapa subrutin, yaitu :

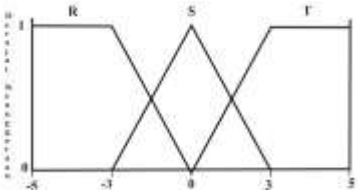
1. *Time Sampling* 5ms
2. Baca *accelerometer*
3. Baca *gyroscope*
4. *Complementary filter*
5. Logika *fuzzy* untuk *tuning* paramater kendali PID
6. Kendali PID
7. Pengaturan motor DC
8. Baca *Encoder*

2.4.2 Perancangan Logika *fuzzy* untuk *tuning* parameter kendali PID

Dalam proses logika *fuzzy* melalui beberapa tahap yaitu proses fuzzifikasi, penentuan basis aturan, dan proses defuzzifikasi. Proses fuzzifiaksi adalah proses pemetaan masukan-masukan kedalam bentuk himpunan keanggotaan *fuzzy* masukan. Dimana pada tugas akhir ini menggunakan dua buah himpunan keanggotaan masukan *fuzzy* yaitu *error* dan *delta error*.



Gambar 11. Himpunan keanggotaan masukan *error*



Gambar 12. Himpunan keanggotaan masukan *delta error*

Proses berikutnya adalah proses penentuan basis aturan *fuzzy*. Basis aturan *fuzzy* inilah yang menentukan nilai konstanta parameter kendali PID.

Tabel 1. Perancangan aturan *fuzzy* untuk *tuning* parameter Kp

error / *Δerror*	NB	NK	Z	PK	PB
R	ZkpSB	ZkpS	ZkpK	ZkpS	ZkpSB
S	ZkpB	ZkpS	ZkpK	ZkpS	ZkpB
T	ZkpSB	ZkpS	ZkpK	ZkpS	ZkpSB

Keterangan : ZkpSB = nilai konstanta Kp Sangat Besar
ZkpB = nilai konstanta Kp Besar
ZkpS = nilai konstanta Kp Sedang
ZkpK = nilai konstanta Kp Kecil

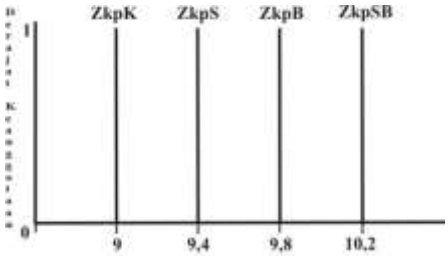
Tabel 2. Perancangan aturan *fuzzy* untuk *tuning* parameter Kd

error / *Δerror*	NB	NK	Z	PK	PB
R	ZkdSB	ZkdB	ZkdS	ZkdK	ZkdK
S	ZkdS	ZkdS	ZkdK	ZkdS	ZkdS
T	ZkdK	ZkdK	ZkdS	ZkdB	ZkdSB

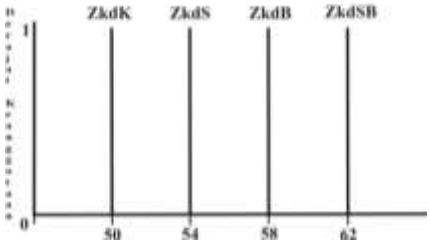
Keterangan : ZkdSB = nilai konstanta Kd Sangat Besar
ZkdB = nilai konstanta Kd Besar
ZkdS = nilai konstanta Kd Sedang
ZkdK = nilai konstanta Kd Kecil

Proses terakhir adalah proses defuzzifikasi yang merupakan proses proses pengubahan nilai keluaran *fuzzy* menjadi nilai keluaran tegas (*crisp*). Pada perancangan logika *fuzzy* ini nilai konstanta yang digunakan untuk konstanta parameter Kp sebesar ZkpSB=10,2; ZkpB=9,8; ZkpS=9,4; dan ZkpK=9, konstanta parameter Ki sebesar 100, dan konstanta Kd sebesar ZkdSB=62; ZkdB=58; ZkdS=54; dan ZkdK=50. Himpunan keanggotaan

keluaran *fuzzy* berupa *singleton*. Terlihat pada Gambar 13 dan Gambar 14 merupakan himpunan keanggotaan keluaran *fuzzy* untuk parameter kendali Kp dan parameter kendali Kd.



Gambar 13. Himpunan keanggotaan keluaran logika *fuzzy* untuk parameter Kp



Gambar 14. Himpunan keanggotaan keluaran logika *fuzzy* untuk parameter Kd

3. Hasil dan Analisis

3.1 Pengujian Perangkat Keras

3.1.1 Pengujian Sensor *Accelerometer*

Pengujian sensor *accelerometer* dilakukan dengan membandingkan keluaran dari sensor berupa sudut kemiringan dengan sudut yang diukur dengan busur derajat. Pengujian sudut kemiringan sensor *accelerometer* hanya dilakukan pada jangkauan -40^0 sampai 40^0 dengan kelipatan 10.

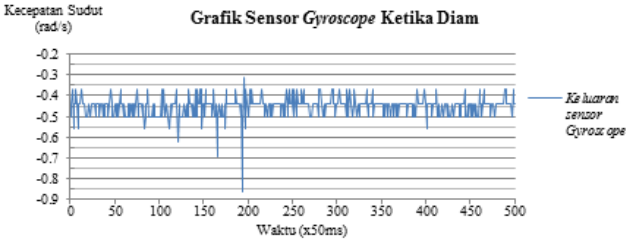
Tabel 3. Hasil pengukuran sudut kemiringan sensor *accelerometer*

No.	Sudut Pengujian (derajat)								
	-40	-30	-20	-10	0	10	20	30	40
1	-39,07	-29,73	-19,31	-9,56	0,69	11,15	21,19	30,90	41,25
2	-39,34	-29,74	-19,11	-9,67	0,63	11,22	20,59	30,74	41,83
3	-38,90	-29,58	-19,28	-9,61	0,66	11,06	20,67	31,19	40,76
4	-39,27	-29,53	-19,35	-9,43	0,57	11,10	20,67	30,94	41,55
5	-39,31	-29,75	-19,32	-9,26	0,52	10,92	20,68	30,84	41,12
6	-38,92	-29,66	-19,12	-9,46	0,54	11,02	20,70	30,98	41,47
7	-39,25	-29,56	-19,29	-9,25	0,27	11,03	20,87	30,91	41,55
8	-39,06	-29,74	-19,28	-9,34	0,39	11,22	20,79	30,96	41,30
9	-39,29	-29,73	-19,39	-9,47	0,56	10,94	20,61	30,95	40,89
10	-39,27	-29,39	-19,3	-9,41	0,61	10,89	20,82	31,37	41,21
11	-38,96	-29,56	-19,18	-9,29	0,45	11,19	20,69	30,91	41,45
12	-39,03	-29,76	-19,32	-9,83	0,55	10,97	20,79	30,91	41,22
13	-39,11	-29,73	-19,15	-9,48	0,21	10,91	20,84	31,03	41,34
14	-39,23	-29,45	-19,31	-9,27	0,09	11,00	21,07	30,99	41,37
15	-39,16	-29,92	-19,16	-9,36	0,65	11,21	20,92	30,96	41,00
16	-39,41	-29,64	-19,29	-9,09	0,4	11,28	20,63	30,69	41,04
17	-39,15	-29,89	-19,32	-9,4	0,12	11,23	20,62	30,85	41,12
18	-39,25	-29,67	-19,39	-9,33	0,45	10,97	20,82	31,29	41,10
19	-39,02	-29,58	-19,58	-9,31	0,58	10,93	20,81	30,9	40,93
20	-39,28	-29,75	-19,16	-9,36	0,39	11,03	21,47	31,06	41,41
Rata-rata error	-0,84	-0,33	-0,72	-0,59	0,47	-1,06	-0,81	-0,97	-1,25

Terlihat dari Tabel 3 variasi pengukuran derajat yang memilik nilai rata-rata *error* paling besar yaitu pada pengukuran 40^0 sebesar $-1{,}25^0$ dan yang memiliki nilai rata-rata *error* paling kecil yaitu pada pengukuran -30^0 sebesar $-0{,}33^0$.

3.1.2 Pengujian Sensor *Gyroscope*

Pengujian dilakukan dengan melihat keluaran dari sensor *gyroscope* melalui grafik.



Gambar 13. Grafik sensor *gyroscope* ketika diam

Sinyal yang dihasilkan mempunyai nilai dan berubah-ubah. Seharusnya jika sensor *gyroscope* dalam keadaan diam, keluarannya sama dengan 0. Hal ini dikarenakan sensor *gyroscope* mempunyai nilai bias.

3.1.3 Pengujian Sensor *Encoder*

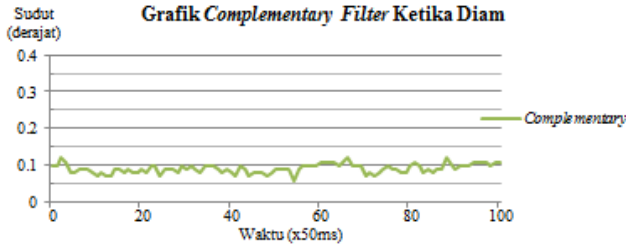
Encoder dalam *self-balancing robot* ini digunakan untuk membantu sistem dalam menyeimbangkan robot. Pengujian sensor *encoder* dilakukan dengan cara memutar roda satu putaran penuh.

Gambar 15. Kode digital pengujian pembacaan *encoder*

3.2 Pengujian Perangkat Lunak

3.2.1 Pengujian *Complementary Filter*

Pengujian dilakukan dengan memberikan nilai koefisien filter (a) sebesar 0,98. Pengujian dilakukan dengan melihat sinyal keluaran dari *complementary filter* saat sistem sedang diam dan menghitung nilai *time constant* (τ) serta melakukan pengukuran sudut aktual serta menghitung nilai rata-rata *error*.



Gambar 16. Grafik *complementary filter* saat diam

Perhitungan nilai *time constant* (τ) dengan koefisien *filter* **(*a*)** sebesar 0,98 adalah sebagai berikut.

$$\tau = \frac{a.\mathrm{dt}}{1-a}$$

$$\tau = \frac{(0{,}98).(0{,}005)}{1-0{,}98}$$

$$\tau = 0{,}245 \text{ detik}$$

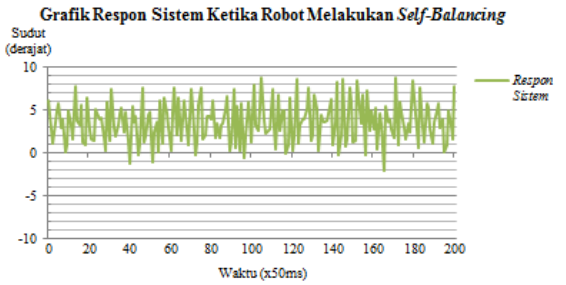
Tabel 4. Hasil pengukuran sudut kemiringan sensor dengan *complementary filter*

No.	Sudut Pengujian (derajat)								
	-40	-30	-20	-10	0	10	20	30	40
1	-40,47	-30,91	-20,46	-10,73	0,05	10,57	20,44	30,24	39,99
2	-40,47	-30,85	-20,85	-10,35	0,05	10,56	20,43	30,23	40,01
3	-40,46	-30,83	-20,45	-10,72	0,07	10,55	20,42	30,24	40,01
4	-40,45	-30,82	-20,83	-10,71	0,06	10,55	20,36	30,24	40,03
5	-40,45	-30,97	-20,47	-10,70	0,05	10,46	20,34	30,23	40,04
6	-40,43	-30,96	-20,78	-10,69	0,03	10,45	20,34	30,23	40,05
7	-40,43	-30,93	-20,81	-10,68	0,02	10,41	20,33	30,23	40,07
8	-40,42	-30,92	-20,79	-10,67	0,08	10,41	20,33	30,22	40,08
9	-40,42	-30,91	-20,78	-10,2	0,07	10,40	20,30	30,22	40,08
10	-40,41	-30,90	-20,48	-10,21	0,07	10,39	20,3	30,23	40,09
11	-40,4	-30,68	-20,47	-10,52	0,06	10,39	20,29	30,23	40,1
12	-40,39	-30,68	-20,47	-10,52	0,04	10,38	20,28	30,22	40,11
13	-40,38	-30,68	-20,49	-10,41	0,15	10,38	20,27	30,22	40,12
14	-40,37	-30,67	-20,46	-10,5	0,12	10,37	20,27	30,22	40,13
15	-40,36	-30,88	-20,50	-10,45	0,13	10,37	20,25	30,22	40,13
16	-40,36	-30,88	-20,49	-10,44	0,13	10,37	20,24	30,21	40,13
17	-40,35	-30,88	-20,54	-10,44	0,24	10,37	20,23	30,20	40,14
18	-40,35	-30,87	-20,54	-10,43	0,19	10,36	20,23	30,21	40,14
19	-40,34	-30,87	-20,52	-10,43	0,15	10,36	20,19	30,21	40,14
20	-40,34	-30,86	-20,49	-10,43	0,22	10,35	20,20	30,21	40,15
Rata-rata error	0,40	0,85	0,58	0,51	-0,1	-0,42	-0,30	-0,22	-0,09

Berdasarkan Tabel 4 terlihat bahwa nilai rata-rata *error* paling besar yaitu pada pengukuran -30^0 sebesar $0{,}85^0$ dan yang paling kecil yaitu pada pengukuran 40^0 sebesar $-0{,}09^0$.

3.2.2 Pengujian *Self-Balancing Robot* Tanpa Gangguan

Pengujian *self-balancing robot* tanpa gangguan dilakukan setelah kendali PID berfungsi mengendalikan robot.



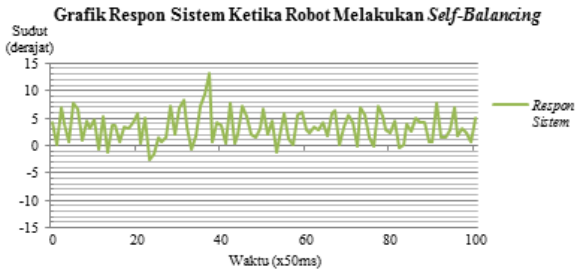
Gambar 17. Pengujian *self-balancing robot* tanpa gangguan

Terlihat pada Gambar 17 *self-balancing robot* dapat menyeimbangkan diri tanpa diberi gangguan dari luar. Respon sistem *self-balancing robot* dapat seimbang mendekati nilai *set point* 0^0 yaitu diantara 0^0 hingga 5^0.

3.2.3 Pengujian *Self-Balancing Robot* dengan Gangguan

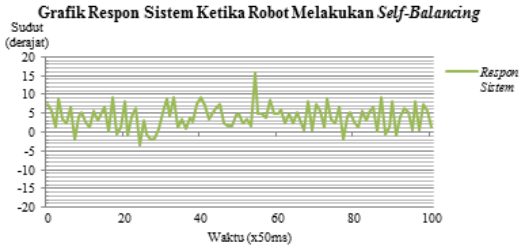
Gangguan saat kondisi sedang menyeimbangkan diri adalah memberikan dorongan searah jarum jam/*clock wise* (CW) atau berlawanan arah jarum jam/*counter clock wise* (CCW) secara perlahan sehingga mengakibatkan adanya perubahan sudut.

A. Pengujian dengan Gangguan Searah Jarum Jam (CW)
Dalam pengujian ini *self-balancing robot* diberi sebuah gangguan secara perlahan searah jarum jam. Pengujian dengan gangguan searah jarum jam (CW) dilakukan sebanyak dua kali pengujian. Dari dua kali pengujian tersebut maka didapat dua buah grafik respon sistem *self-balancing robot.*



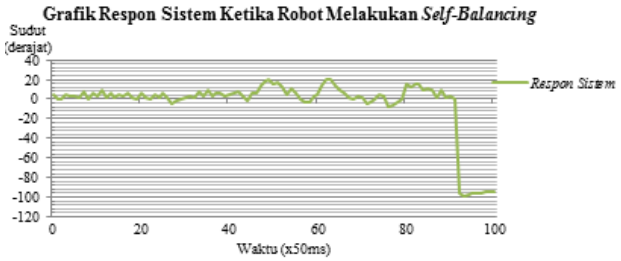
Gambar 18. Grafik respon *self-balancing robot* penguijian pertama CW

Dalam pengujian pertama terjadi gangguan pada *self-balancing robot* saat *time sampling* ke 37 dengan gangguan sebesar $13{,}23^0$.



Gambar 19. Grafik respon *self-balancing robot* penguijian kedua CW

Dalam pengujian kedua terjadi gangguan pada *self-balancing robot* saat *time sampling* ke 54 dengan gangguan sebesar $15{,}69^0$.



Gambar 20. Grafik respon *self-balancing robot* penguijian ketiga CW

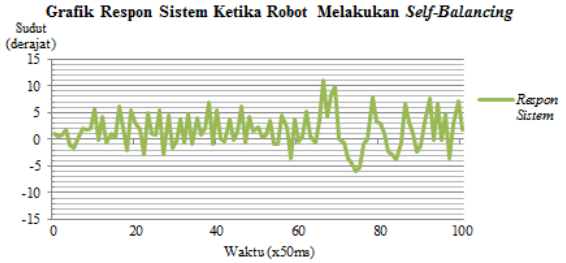
Pada waktu sampling ke 47 gangguan yang didapat *self-balancing robot* yaitu sebesar $14{,}26^0$. Setelah diberikan sebuah gangguan, *self-balancing robot* langsung berusaha menyeimbangkan diri dan ternyata dapat menyeimbangkan diri. Kemudian pada *time sampling* ke 63 diberikan gangguan sebesar $20{,}54^0$. Setelah diberi gangguan sistem *self-balancing robot* terus melakukan osilasi dan akhirnya terjatuh. Hal ini dikarenakan sistem *self-balancing robot* tidak dapat menyeimbangkan diri ketika di beri gangguan sebesar $20{,}54^0$.

B. Pengujian dengan Gangguan Berlawanan Arah Jarum Jam (CCW)

Dalam pengujian ini *self-balancing robot* diberi sebuah gangguan secara perlahan berlawanan arah jarum jam. Pengujian dengan gangguan berlawanan arah jarum jam (CCW) dilakukan sebanyak dua kali pengujian. Dari dua kali pengujian tersebut maka didapat dua buah grafik respon sistem *self-balancing robot.*

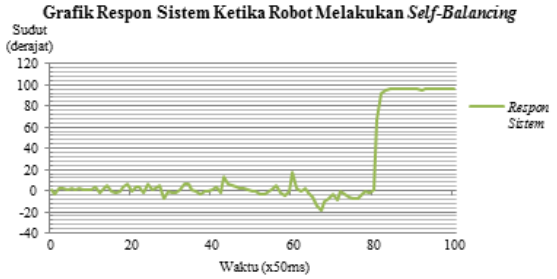
Gambar 21. Grafik respon *self-balancing robot* penguijian pertama CCW

Dalam pengujian pertama terjadi gangguan pada *self-balancing robot* saat *time sampling* ke 56 dengan gangguan sebesar $-9{,}14^0$.



Gambar 22. Grafik respon *self-balancing robot* penguijian kedua CCW

Dalam pengujian kedua terjadi gangguan pada *self-balancing robot* saat *time sampling* ke 74 dengan gangguan sebesar $-5{,}98^0$.

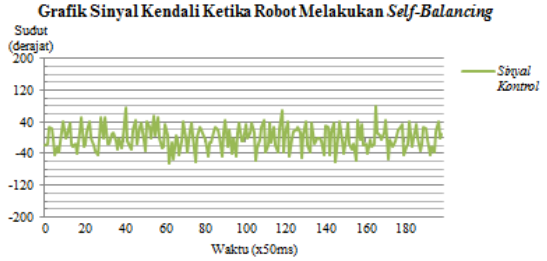


Gambar 23 Grafik respon self-balancing robot penguijian ketiga CCW

Pada time sampling ke 28 gangguan yang didapat *self-balancing robot* yaitu sebesar $-7{,}39^0$. Setelah diberikan sebuah gangguan, *self-balancing robot* langsung berusaha menyeimbangkan diri dan ternyata dapat menyeimbangkan diri. Kemudian pada *time sampling* ke 66 diberikan gangguan sebesar $-14{,}68^0$. Setelah diberi gangguan sistem *self-balancing robot* terus melakukan osilasi dan akhirnya terjatuh.

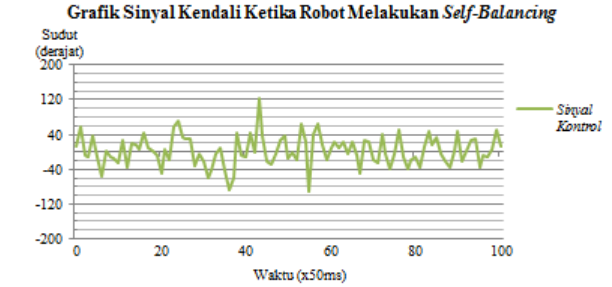
3.2.4 Pengujian Sinyal Kendali PID

Pengujian dilakukan dengan melihat keluaran sinyal kendali PID yang diumpankan ke motor DC pada saat *self-balancing robot* ketika tanpa gangguan dan ketika ada sebuah gangguan serah jarum jam (CW) maupun berlawanan arah dengan jarum jam (CCW).



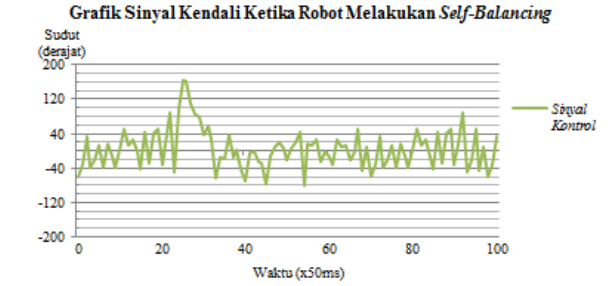
Gambar 24. Grafik sinyal kendali PID pengujian tanpa gangguan

Berdasarkan Gambar 24, jangkauan kerja sinyal kendali PID saat *self-balancing robot* menyeimbangkan diri tanpa gangguan adalah sebesar -65 hingga 79.



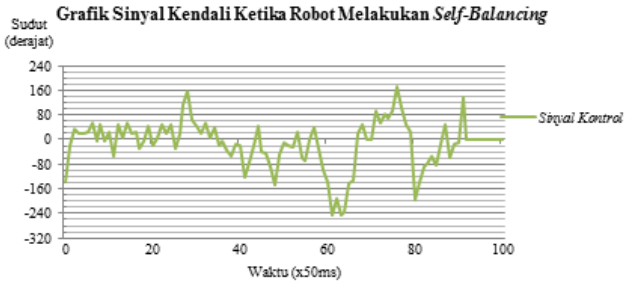
Gambar 25. Grafik sinyal kendali PID pengujian pertama dengan gangguan CW

Terlihat pada pengujian pertama dengan gangguan searah jarum jam (CW) tersebut berdampak pada sinyal kendali PID yang mencapai nilai 122 pada *time sampling* 43. Kemudian setelah beberapa lama sinyal kendali PID berangsur mengecil seperti semula.



Gambar 26. Grafik sinyal kendali PID pengujian kedua dengan gangguan CW

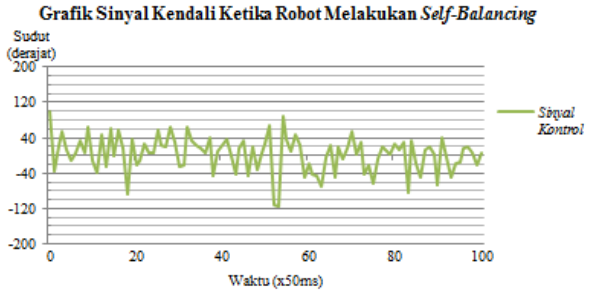
Dalam pengujian kedua dengan gangguan searah jarum jam (CW). Pada pengujian kedua, sinyal kendali PID mencapai nilai 163 pada *time sampling* 25.



Gambar 27. Grafik sinyal kendali PID pengujian ketiga dengan gangguan CW

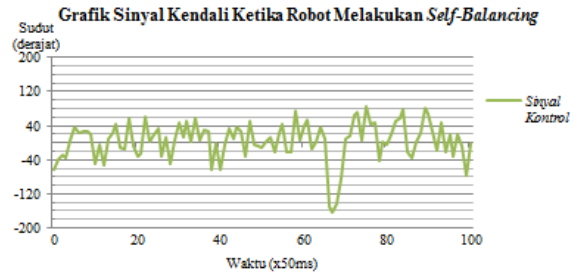
Nilai sinyal kendali PID pada saat pengujian ketiga gangguan searah jarum jam (CW) pada saat diberikan gangguan pada *time sampling* 47 bernilai -146 dan pada time sampling 63 sebesar -245 serta pada saat robot terjatuh sinyal kendali PID bernilai 0.

Selanjutnya grafik sinyal kendali PID ketika *self-balancing robot* diberi gangguan berlawaan arah jarum jam (CCW) dilakukan sebanyak tiga kali seperti dengan pengujian searah jarum jam (CW).



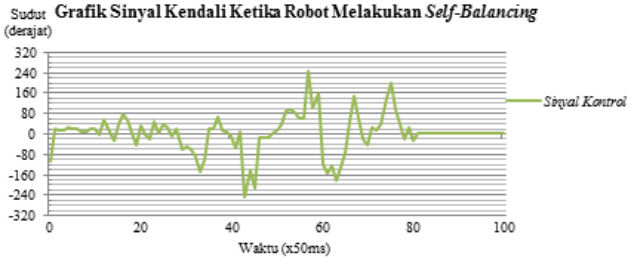
Gambar 28. Grafik sinyal kendali PID pengujian pertama dengan gangguan CCW

Terlihat pada Gambar 28 grafik sinyal kendali PID mengalami osilasi setelah diberi gangguan berlawanan arah jarum jam (CCW). Nilai sinyal kendali PID paling besar pada saat pengujian pertama gangguan berlawanan arah jarum jam (CCW) adalah sebesar -111 pada waktu sampling 52.



Gambar 29. Grafik sinyal kendali PID pengujian kedua dengan gangguan CCW

Berdasarkan Gambar 29 sinyal kendali PID memiliki nilai paling besar pada *time sampling* 67 dengan nilai sinyal kendali sebesar -149.



Gambar 30 Grafik sinyal kendali PID pengujian ketiga dengan gangguan CCW

Terlihat pada Gambar 30 grafik sinyal kendali PID mengalami osilasi setelah diberi gangguan berlawanan arah jarum jam (CCW). Nilai sinyal kendali PID pada saat pengujian ketiga gangguan berlawanan arah jarum jam (CCW) pada saat diberikan gangguan pada time sampling 28 bernilai -62 dan pada time sampling 66 sebesar -145.

Namun nilai terbesar sinyal kendali PID berada pada time sampling 57 dan bernilai 245, pada time sampling tersebut sistem self-balancing robot masih berusaha menyeimbangkan diri dan mengakibatkan terjadinya osilasi. Ketika self-balancing robot terjatuh sinyal kendali PID bernilai 0.

4. Kesimpulan

Self-Balancing robot dapat menyeimbangkan diri pada permukaan yang datar ketika tanpa gangguan. Pada pengujian dengan gangguan searah jarum jam (CW) sudut maksimal yang masih dapat diseimbangkan oleh sistem adalah sebesar $15{,}69^0$, sedangkan pada gangguan berlawanan arah jarum jam (CCW) sebesar $-9{,}14^0$. Nilai koefiisien *filter* (α) untuk *complementary filter* yang paling optimal pada *self-balancing robot* yaitu bernilai 0,98, sehingga menghasilkan *time constant* (τ) *complementary filter* sebesar 0,245 detik. Nilai konstanta logika *fuzzy* yang digunakan untuk *tuning* parameter kendali PID yaitu konstanta parameter Kp (ZkpSB=10,2; ZkpB=9,8; ZkpS=9,4; dan ZkpK=9), konstanta parameter Ki (Ki=100), dan konstanta Kd (ZkdSB=62; ZkdB=58; ZkdS=54; dan ZkdK=50).

Untuk penelitian selanjutnya, dapat digunakan metode kontrol lain, seperti Jaringan Saraf Tiruan (JST), algoritma genetik atau *Linear Quadratic Regulator* (LQR) untuk mengendalikan *self-balancing robot.*

Referensi

[1] McComb, Gordon, *and* Predko, Myke, *Robot Builder's Bonanza, Third Editon*, McGraw-Hill, New York, 2006.

[2] Grasser, F., D'arrigo, A., Colombi, S., Rufer, A.C., JOE: A Mobile, Inverted Pendulum, IEEE Transactions on Industrial Electronics, 49(1): 107-114, 2002.

[3] Colton, Shane., The Balance Filter: A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform, Massachuesetts Institute of Technology, Cambridge, MA, Rev.1: Submitted as a Chief Delphi white paper, June 2007.

[4] Sutojo, T,dkk., Kecerdasan Buatan, Penerbit ANDI Yogyakarta, Yogyakarta, 2010.

[5] Kusumadewi, Sri dan Sri Hartati, Neuro-Fuzzy Integrasi Sistem Fuzzy & Jaringan Syaraf Edisi2, Penerbit Graha Ilmu, Yogyakarta, 2010.

[6] Jang, J-S, dkk., Neuro-Fuzzy and Soft Computing, Prentice-Hall International, Inc., 1997.Setiawan, Iwan, Kontrol PID Untuk Proses Industri, Elex Media Komputindo, Jakarta, 2008.

[7] Cetinkunt, Sabri, Mechatronics with Experiment, 2nd Edition, John Wiley & Sons Ltd , 2015.