SELF-BALANCING SCOOTER MENGGUNAKAN METODE KENDALI PID DENGAN TUNING FUZZY

Jan Pieter Candra Siahaan[*)], Sumardi, and Budi Setiyono

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

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

Abstrak

Robot mengalami perkembangan pesat dari waktu ke waktu. Salah satu jenisnya adalah *mobile robot*. *Self balancing scooter* adalah *mobile robot* yang prinsip kerjanya serupa pendulum terbalik. *Self-balancing scooter* membutuhkan kontroler untuk dapat menyeimbangkan diri dan dikendarai secara personal. Oleh karena itu diperlukan perancangan yang tepat, baik perangkat keras maupun perangkat lunak, agar *self-balancing scooter* dapat bekerja sesuai fungsinya. Pada penelitian ini dibuat *self-balancing scooter* yang dapat menyeimbangkan diri pada bidang datar dan dapat dikendarai. *Self-balancing scooter* dirancang menggunakan metode kendali PID dengan *tuning* fuzzy pada dua mode (*self-balancing* dan *ride-on*). Sensor MPU-6050 mendeteksi sudut *pitch* (θ) yang dibentuk oleh kemiringan *self-balancing scooter*. Dari data sudut diperoleh masukan *error* dan *delta error* untuk logika fuzzy. Logika fuzzy menghasilkan nilai konstanta parameter-parameter PID. Hasil pengujian menunjukkan bahwa *self-balancing scooter* dapat menyeimbangkan diri pada bidang datar serta dapat dikendarai bergerak maju, mundur, berbelok ke kanan dan kiri. *Complementary filter* memperbaiki pembacaan sudut dengan rata-rata |*error*| sebesar 0,07. Kendali PID *tuning* fuzzy mode *self-balancing* mempunyai nilai Kp sebesar KpSB=50, KpB=40, KpS=30, KpK=20, Ti sebesar 0,5, dan Td sebesar TdSB=0,05, TdB=0,045, TdS=0,042, TdK=0,04. Sedangkan mode *ride-on*, nilai Kp sebesar Kp1SB=115, Kp1B=110, Kp1S=100, Kp1K=90, Ti sebesar 0,5, dan Td sebesar Td1SB=0,095, Td1B=0,09, Td1S=0,085, Td1K=0,08.

Kata kunci: self-balancing scooter, MPU-6050, complementary filter, PID tuning fuzzy

Abstract

Robots have evolved considerably over time, including mobile robots. Self-balancing scooter is one of them which works like inverted pendulum. Self-balancing scooter requires controller in order to self-balancing and to be ridden. Therefore, it is necessary to design accurately, both hardware and software, so that self-balancing scooter operates according to its functions. A self-balancing scooter which can self-balancing on a flat horizontal plane and can be ridden is built in this final assignment. Self-balancing scooter designed using fuzzy-tuning PID control method in two modes (self-balancing and ride-on). MPU-6050 detects pitch angle (θ) formed by the slope of the self-balancing scooter. Error and delta error datas are obtained as fuzzy logic inputs. Fuzzy logic generates constant value of PID parameters. The test results showed that the self-balancing scooter can self-balancing on a horizontal plane and can be ridden to move forward, backward, turn right and left. Complementary filter improves MPU-6050 angle readings with average |error| reaches 0.07. On self-balancing mode, PID has Kp parameter constant KpSB=50, KpB=40, KpS=30, KpK=20, Ti=0.5, and the Td parmeter constant TdSB=0.05, TdB=0.045, TdS=0.042, TdK=0.04. On the ride-on mode, the Kp parameter constant Kp1SB=115, Kp1B=110, Kp1S=100, Kp1K=90, Ti=0.5, and the Td parameter constant Td1SB=0.095, Td1B=0.09, Td1S=0.085, Td1K=0.08.

Keywords: self-balancing scooter, MPU-6050, complementary filter, fuzzy-tuning PID

1. Pendahuluan

Penggunaaan sistem kontrol otomasi telah berkembang pesat menjangkau segala aspek dalam kehidupan sehari-hari. Hal ini mendorong kaum akademisi khususnya mahasiswa untuk terlibat langsung dan aktif dalam perkembangan ini. Termasuk mahasiswa konsentrasi Kontrol dan Instrumentasi Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro yang dengan giat mengaplikasikan berbagai sistem kontrol otomasi, salah satunya menghasilkan *self-balancing scooter*. *Self-balancing scooter* adalah kendaraan roda dua yang mampu menyeimbangkan diri dan dapat dikendarai.

Self-balancing scooter dikembangkan berdasarkan hasil perancangan dan pengujian dari *self-balancing robot* yang hanya dapat seimbang dengan adanya kontroler. Prinsip kerjanya hampir serupa dengan pendulum terbalik yang diletakkan di atas kereta beroda [1]. Dengan perancangan yang tepat memungkinkan *self-balancing scooter* mampu dijadikan sebagai inovasi alat transportasi personal yang stabil.

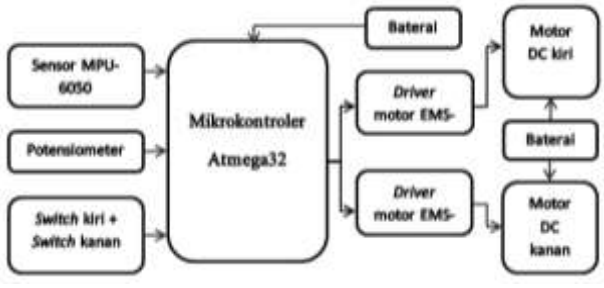
Penelitian terdahulu tentang *self-balancing scooter* yang telah dilakukan menggunakan kendali PID dalam perancangannya. Kendali PID sangat banyak ditemui pada industri proses. Faktor penting yang membuat luasnya penggunaan metode kontrol ini adalah kesederhanaan struktur kontrolnya, yaitu hanya mempunyai 3 parameter utama yang perlu diatur. Dari penelitian sebelumnya disimpulkan bahwa kendali PID bekerja dengan baik pada *self-balancing scooter* dengan parameter Kp=40, Ti=0,5, dan Td=0,01 serta dapat dikendarai dengan sudut maksimal sebesar 15º [2].

Berdasarkan hal tersebut maka dalam penelitian penelitian ini dirancang *self-balancing scooter* yang mampu menyeimbangkan diri tegak lurus permukaan bumi pada bidang datar dan juga dapat dikendarai dengan menggabungkan kendali Fuzzy dan PID [3]. Kemiringan menghasilkan *error* (selisih sudut referensi dengan sudut aktual) yang menjadi umpan balik dalam penentuan keluaran kendali. Setiap perubahan sudut kemiringan *self-balancing scooter* dideteksi oleh sensor MPU-6050. Dan dua motor DC digunakan sebagai penggerak dan mendapat masukan berupa sinyal kontrol (PWM [*Pulse Width Modulation*]) yang mengatur pergerakannya.

2. Metode

2.1. Perancangan Perangkat Keras

Secara umum perancangan perangkat keras *self-balancing scooter* dapat dilihat pada diagram blok Gambar 1.



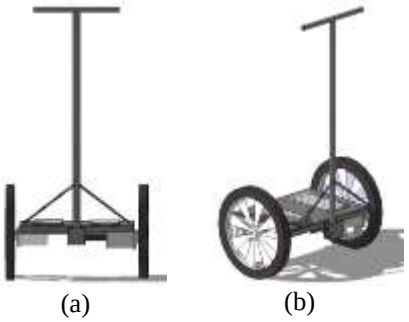
Gambar 1. Diagram blok perancangan perangkat keras

Tiap-tiap bagian dari diagram blok sistem di atas dijelaskan sebagai berikut:

1. Sensor MPU-6050 3-*axis accelerometer* + 3-*axis gyroscope* digunakan sebagai sensor yang mendeteksi perubahan sudut kemiringan *self-balancing scooter* terhadap permukaan bumi.
2. Potensiometer digunakan untuk *handling* sehingga *self-balancing scooter* dapat berbelok ke kanan maupun ke kiri pada saat dikendarai.
3. *Switch* kiri dan *Switch* kanan berperan sebagai pendeteksi keberadaan pengendara pada *self-balancing scooter* dan memastikan pengendara dalam posisi yang aman, yakni kedua kaki pengendara berapa pada posisi yang seharusnya. Selain sebagai pengaman, *switch* juga berfungsi untuk memilih mode antara *self-balancing* atau *ride on*.
4. Mikrokontroler AVR Atmega32 sebagai pusat pengendali *self-balancing scooter* secara keseluruhan.
5. Dua buah baterai digunakan sebagai catu daya bagi keseluruhan sistem *self-balancing scooter*.
6. Sepasang *driver* motor EMS-30A H-Bridge merupakan *driver* bagi masing-masing motor DC mengomunikasikan perintah keluaran mikrokontroler dalam bentuk pergerakan motor.
7. Motor DC kanan dan kiri merupakan dua buah penggerak *self-balancing scooter.*

2.2.1. Perancangan Mekanik

Perancangan mekanik dalam pembuatan *self-balancing scooter* sangat berpengaruh pada proses keseimbangan. Perancangan mekanik diusahakan dapat seimbang pada titik pusat massanya. Perbedaan berat sisi depan dan belakang dapat berpengaruh terhadap kestablian sistem dan mempersulit pengendalian *self-balancing scooter*.



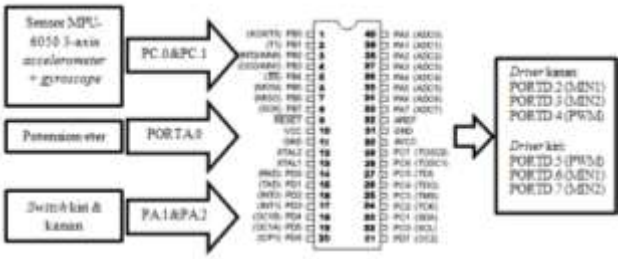
Gambar 2. Ilustrasi mekanik *self-balancing scooter* tampak depan (a) dan tampak samping (b)

2.2.2. Mikrokontroler Atmega32

Atmega32 dipilih dalam perancangan ini karena memiliki memori yang besar dan cukup untuk menjalankan program. Mikrokontroler akan memproses data keluaran sensor menghasilkan sinyal kontrol yang ditransmisikan ke driver motor DC. PORTC.0 dan PORTC.1 (PC.0 dan PC.1) digunakan sebagai port masukan sensor MPU-6050 melalui komunikasi I^2C. PORTA.0 (PA.0) terhubung dengan potensiometer yang berfungsi ketika melakukan belokan (*turning*) saat *self-balancing scooter* dalam mode *ride on*. Pemilihan mode antara *self-balancing* atau *ride on* dilakukan ketika kedua switch ditekan. *Switch* kanan

dihubungkan ke PORTA.1 dan *switch* kiri dihubungkan ke PORTA.2.

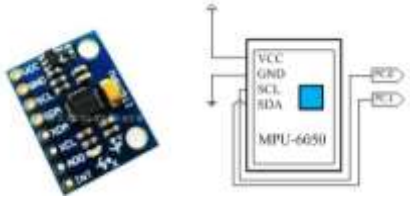
Setelah menerima masukan, khususnya masukan dari sensor MPU-6050, mikrokontroler memproses data tersebut berdasarkan algoritma yang ditanamkan dan menghasilkan keluaran berupa perintah yang ditujukan ke *driver* EMS-30A kanan dan kiri. Perintah tersebut berupa arah putar motor yang dihubungkan dari PORTD.2 dan PORTD.3 untuk *driver* kanan serta PORTD.6 dan PORTD.7 untuk *driver* kiri. Selain arah putar motor, mikrokontroler juga mengirimkan PWM (*Pulse Width Modulation*) untuk mengatur kecepatan putaran motor DC. Pengaturan PWM dihubungkan dari PORTD.4 untuk *driver* kanan dan PORTD.5 untuk *driver* kiri.



Gambar 3. Konfigurasi pin mikrokontroler Atmega32

2.2.3. Sensor MPU-6050

Dalam perancangan penelitian ini, sensor MPU-6050 dihubungkan langsung dengan mikrokontroler. Hal ini berarti sensor MPU-6050 menerima catu daya sebesar 5V dari mikrokontroler, padahal tegangan kerjanya adalah sebesar 3,3V. Sensor MPU-6050 dilengkapi dengan regulator tegangan 3,3V sehingga dapat dihubungkan langsung dengan mikrokontroler karena tegangan 5V tersebut akan diturunkan sebelum digunakan sensor. *Port* VCC dan GND-nya dihubungkan ke *port* VCC dan GND pada mikrokontroler.



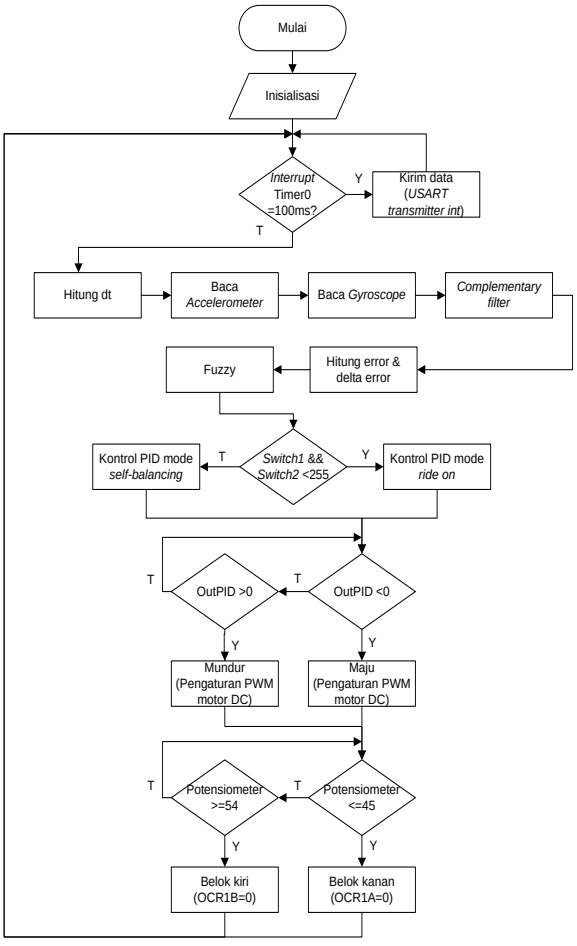
Gambar 4. Sensor MPU-6050 dan konfigurasinya

Proses pengiriman dan penerimaan data pada sensor MPU-6050 terjadi dengan komunikasi *two-wire serial interface*. Singkatnya komunikasi ini memungkinkan pertukaran informasi pada dua atau beberapa IC atau dikenal juga dengan I^2C. Komunikasi ini dapat dilakukan dengan terhubungkannya mikrokontroler dengan sensor MPU-6050, yaitu pada *port* SCL dan SDA. Dua jalur ini masing-masing berfungsi sebagai jalur *clock* (SCL) dan jalur data (SDA). Port SCL pada sensor MPU-6050 dihubungkan ke PORTC.0 dan SDA dihubungkan ke PORTC.1 pada mikrokontroler, seperti pada Gambar 3.7. Sensor MPU-6050 diletakkan di tengah dan tepat sejajar dengan as roda agar dapat dengan mudah mendeteksi sudut kemiringan *pitch* dari *self-balancing scooter*.

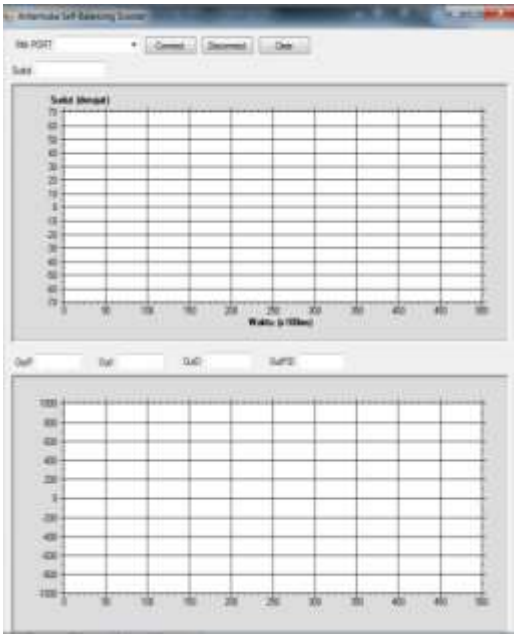
2.3. Perancangan Perangkat Lunak

Perangkat lunak *self-balancing scooter* seluruhnya diprogram ke dalam mikrokontroler Atmega32. Pemrograman mikrokontroler Atmega32 dilakukan dengan menggunakan compiler Code Vision AVR (CVAVR) yang berbasis bahasa C. Diagram alir program ditunjukkan pada Gambar 4. Program berjalan sesuai diagram alir dan terjadi pengulangan terus menerus.

Selain menggunakan CodeVision AVR sebagai compiler, juga digunakan Microsoft Visual Studio C# untuk membuat tampilan antarmuka *self-balancing scooter* sehingga mempermudah pengamatan respon sistem dalam bentuk grafik. Gambar 5 menunjukkan tampilan antarmuka *self-balancing scooter*.



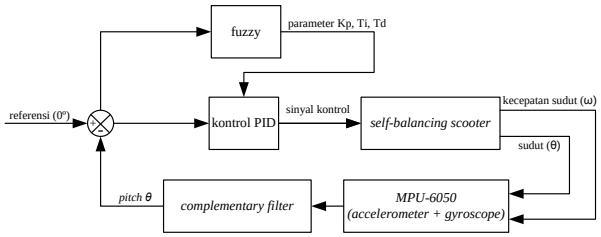
Gambar 4. Diagram alir program



Gambar 5. Antarmuka *self-balancing scooter*

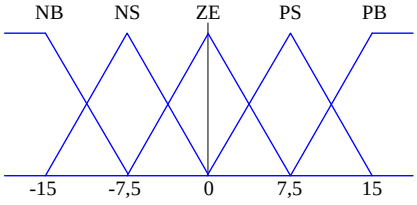
2.3.1. Kendali PID dengan *Tuning* Fuzzy

Dalam penelitian ini *self-balancing scooter* menggunakan kendali fuzzy tuning PID. Kendali fuzzy mendapat dua masukan, yaitu *error* dan *delta error*. Kedua masukan ini diperoleh berdasarkan sudut keluaran yang dihasilkan sensor MPU-6050 sehingga menghasilkan membership function. *Membership function* inilah yang nantinya akan menjadi parameter penentuan konstanta pada kendali PID. Secara umum, diagram blok sistem ditunjukkan oleh gambar dibawah ini :

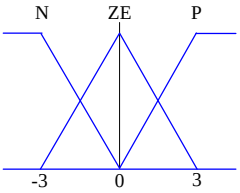


Gambar 6. Diagram blok sistem

Membership function fuzzy dari kedua masukan *error* dan *delta error* adalah sebagai berikut :



Gambar 7. Himpunan keanggotaan masukan *error*



Gambar 8. Himpunan keanggotaan masukan *delta error*

Membership function dibentuk berdasarkan basis aturan (*rule base*) fuzzy di bawah ini :

Tabel 1. Basis aturan fuzzy *tuning* Kp mode *self-balancing*

Rule	Error	Delta error	Kp
1	NB	deN	KpSB
2	NB	deZE	KpB
3	NB	deP	KpSB
4	NS	deN	KpS
5	NS	deZE	KpS
6	NS	deP	KpS
7	ZE	deN	KpK
8	ZE	deZE	KpK
9	ZE	deP	KpK
10	PS	deN	KpS
11	PS	deZE	KpS
12	PS	deP	KpS
13	PB	deN	KpSB
14	PB	deZE	KpSB
15	PB	deP	KpSB

Tabel 2. Basis aturan fuzzy *tuning* Ti mode *self-balancing* dan mode *ride on*

Rule	Error	Delta error	Ti
1	NB	deN	TiT
2	NB	deZE	TiT
3	NB	deP	TiT
4	NS	deN	TiT
5	NS	deZE	TiT
6	NS	deP	TiT
7	ZE	deN	TiT
8	ZE	deZE	TiT
9	ZE	deP	TiT
10	PS	deN	TiT
11	PS	deZE	TiT
12	PS	deP	TiT
13	PB	deN	TiT
14	PB	deZE	TiT
15	PB	deP	TiT

Tabel 3. Basis aturan fuzzy *tuning* Td mode *self-balancing*

Rule	Error	Delta error	Td
1	NB	deN	TdSB
2	NB	deZE	TdS
3	NB	deP	TdK
4	NS	deN	TdB
5	NS	deZE	TdS
6	NS	deP	TdK
7	ZE	deN	TdS
8	ZE	deZE	TdK
9	ZE	deP	TdS
10	PS	deN	TdK
11	PS	deZE	TdS
12	PS	deP	TdB
13	PB	deN	TdK
14	PB	deZE	TdS
15	PB	deP	TdSB

Tabel 4. Basis aturan fuzzy *tuning* Kp mode *ride on*

Rule	Error	Delta error	Kp
1	NB	deN	Kp1SB
2	NB	deZE	Kp1B
3	NB	deP	Kp1SB
4	NS	deN	Kp1S
5	NS	deZE	Kp1S
6	NS	deP	Kp1S
7	ZE	deN	Kp1K
8	ZE	deZE	Kp1K
9	ZE	deP	Kp1K
10	PS	deN	Kp1S
11	PS	deZE	Kp1S
12	PS	deP	Kp1S
13	PB	deN	Kp1SB
14	PB	deZE	Kp1SB
15	PB	deP	KpSB

Tabel 5. Basis aturan fuzzy tuning Td mode ride on

Rule	Error	Delta error	Td
1	NB	deN	Td1SB
2	NB	deZE	Td1S
3	NB	deP	Td1K
4	NS	deN	Td1B
5	NS	deZE	Td1S
6	NS	deP	Td1K
7	ZE	deN	Td1S
8	ZE	deZE	Td1K
9	ZE	deP	Td1S
10	PS	deN	Td11K
11	PS	deZE	Td1S
12	PS	deP	Td1B
13	PB	deN	Td1K
14	PB	deZE	Td1S
15	PB	deP	Td1SB

Variabel-variabel dalam *rule base* tersebut memiliki nilai sebagai berikut:

Tabel 6. Nilai variabel *rule base*

Variabel	Nilai
NB	-15
NS	-7,5
ZE	0
PS	7,5
PB	15
deN	-3
deZE	0
deP	3
KpSB	50
KpB	40
KpS	30
KpK	20
Kp1SB	115
Kp1B	110
Kp1S	100
Kp1K	90
TiT	0,5
TdSB	0,05
TdB	0,045
TdS	0,042
TdK	0,04
Td1SB	0,095
Td1B	0,09
Td1S	0,085
Td1K	0,08

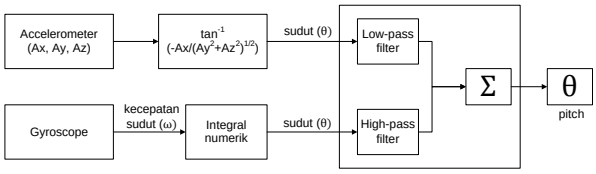
Kemudian nilai-nilai ini akan dikeluarkan dalam bentuk *crisp* pada proses defuzzyikasi. Metode defuzzyfikasi yang digunakan adalah *weighted average* yang dirumuskan pada persamaan berikut:

$$z = \frac{w_1 z_1 + w_2 z_2 + \cdots + w_n z_n}{w_1 + w_2 + w_n} \quad (1)$$

Hasil akhir dari defuzzyfikasi adalah konstanta parameter Kp, Ti, dan Td yang kemudian digunakan pada kendali PID.

2.3.2. Complementary filter

Complementary filter digunakan untuk menggabungkan data sensor *accelerometer* dan *gyroscope* dengan menghilangkan kelemahan masing-masing sensor. Pertimbangan penggunaan *complementary filter* adalah karena mudah diimplementasikan ke dalam bahasa pemrograman. Pada *complementary filter* terdapat dua nilai koefisien masing-masing untuk *accelerometer* dan *gyroscope*, yaitu KA dan KG yang masing-masing bernilai 0,1 dan 0,9.



Gambar 9. Diagram blok *complementary filter*

3. Hasil dan Analisis

3.1. Pengujian Perangkat Keras

3.3.1. Pengujian Sensor *Accelerometer*

Self-balancing scooter dalam pengoperasiannya tidak membutuhkan sudut yang terlalu besar, cukup sekitar -50° sampai 50°. Maka untuk pengujian sensor *accelerometer* dilakukan pada jangkauan -60° sampai 60° dengan kelipatan 10.

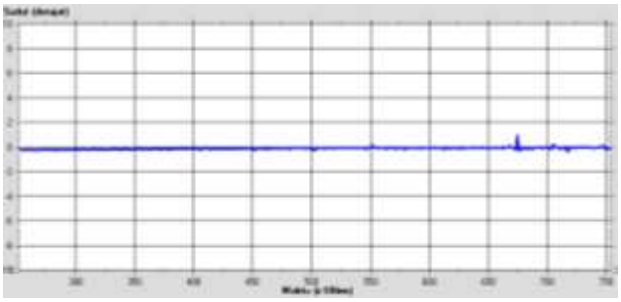
Tabel 7. Hasil pengujian sensor *accelerometer*

Sudut aktual (°)	Sudut *accelerometer* (°)	*Error*	| *Error* |
-60	-59,7	-0,3	0,3
-50	-49,95	-0,05	0,05
-40	-40,27	0,27	0,27
-30	-29,96	-0,04	0,04
-20	-20,35	0,35	0,35
-10	-10,19	0,19	0,19
0	-0,91	0,91	0,91
10	10,15	-0,15	0,15
20	20,31	-0,31	0,31
30	30,19	-0,19	0,19
40	39,98	0,02	0,02
50	50,14	-0,14	0,14
60	59,86	0,14	0,14
	Rata-rata | *error* |		0,235385

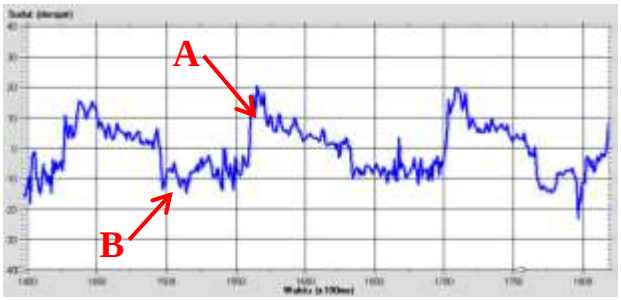
Dari Tabel 7 terlihat bahwa masih terdapat *error* dalam pembacaan sudut oleh *accelerometer*, yaitu rata-rata | *error* | sebesar 0.23°. Hal ini disebabkan oleh cara pengukuran yang kurang akurat, pembulatan pecahan dalam pemrograman, dan noise yang dihasilkan oleh sensor *accelerometer* sendiri.

3.3.2. Pengujian Sensor Gyroscope

Pengujian sensor *gyroscope* dilakukan dengan melihat grafik keluaran *gyroscope* saat diam dan berotasi (rotasi berlawanan arah jarum jam dan searah jarum jam). Sensor *gyroscope* menunjukkan grafik yang mendekati 0° ketika kondisi diam atau tidak berotasi. menunjukkan grafik keluaran *gyroscope* ketika berotasi. Keterangan label A pada Gambar menunjukkan *gyroscope* berotasi searah jarum jam (*clockwise*). Sedangkan label B menunjukkan *gyroscope* berotasi berlawanan arah dengan jarum jam (*counterclockwise*).



Gambar 10. Grafik sensor *gyroscope* keadaan diam



Gambar 11. Grafik sensor *gyroscope* ketika berotasi

3.2. Pengujian Perangkat Lunak

3.2.1. Pengujian *Complementary filter*

Complementary filter dimasukkan ke dalam program untuk menggabungkan *accelerometer* dan *gyroscope*. Kemudian dilakukan pembacaan sudut sama seperti pada pengujian sensor *accelerometer* (jangkauan sudut -60° sampai 60° dengan kelipatan 10).

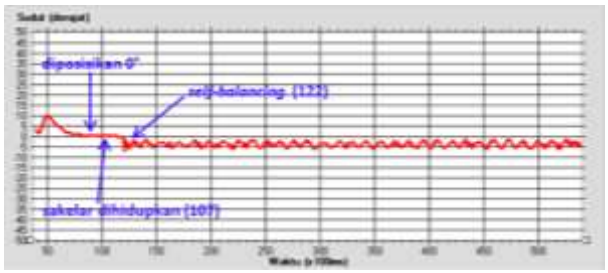
Tabel 8 Hasil pengujian *complementary filter*

Sudut aktual (°)	Sudut *pitch* MPU-6050 (°)	*Error*	| *Error* |
-60	-60,09	0,09	0,09
-50	-50,01	0,01	0,01
-40	-40,17	0,17	0,17
-30	-30,01	0,01	0,01
-20	-20,05	0,05	0,05
-10	-10,11	0,11	0,11
0	-0,02	0,02	0,02
10	10,08	-0,08	0,08
20	20,16	-0,16	0,16
30	30,13	-0,13	0,13
40	40,04	-0,04	0,04
50	50,02	-0,02	0,02
60	60,03	-0,03	0,03
	Rata-rata | *error* |		0,070769

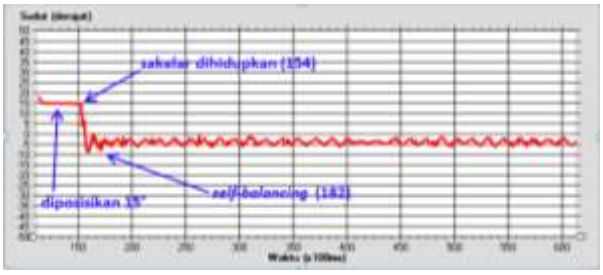
Berdasarkan Tabel 8 pembacaan sudut *pitch* (θ) hasil *complementary filter* memiliki nilai rata-rata | *error* | sebesar 0,07. Nilai ini lebih baik jika dibandingkan dengan pembacaan sudut pada *accelerometer* yang memiliki nilai rata-rata | *error* | sebesar 0,23.

3.2.2. Pengujian Mode *Self-balancing*

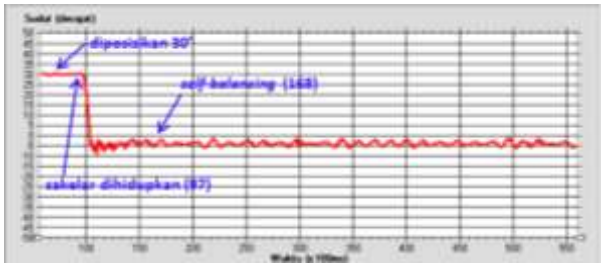
Pengujian ini dilakukan ketika *self-balancing scooter* pada mode *self-balancing*. Pengujian mode *self-balancing* ini dilakukan dengan melihat respon sistem dalam bentuk grafik pada beberapa posisi awal, yaitu posisi di tengah, posisi 15°, posisi 30°, dan posisi jatuh ke depan. Cara tersebut dilakukan untuk melihat respon sistem *self-balancing scooter* dalam menyeimbangkan diri ketika berada pada posisi awal yang berbeda (dalam pengujian ini dilakukan pada posisi tengah (0°), 15°, 30°, dan jatuh ke depan).



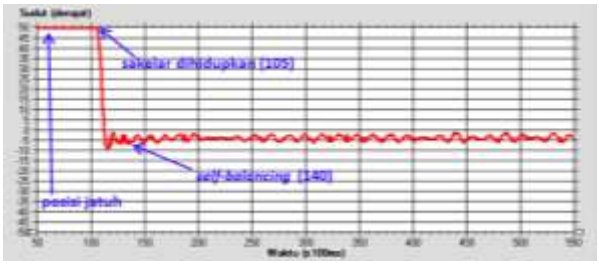
Gambar 12. Respon sistem posisi awal 0°



Gambar 13. Respon sistem posisi awal 15°



Gambar 14. Respon sistem posisi awal 30°

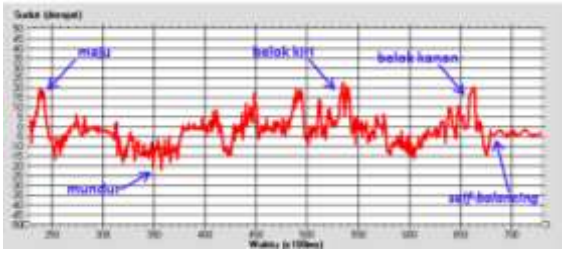


Gambar 15. Respon sistem posisi awal jatuh ke depan

Pengujian mode *self-balancing* posisi awal 0° menyeimbangkan diri dengan Ts=1,5 detik. Pengujian mode *self-balancing* posisi awal 15° menyeimbangkan diri dengan Ts=2,8 detik. Pengujian mode *self-balancing* posisi awal 30° menyeimbangkan diri dengan Ts=7,1 detik. Pengujian mode *self-balancing* posisi awal jatuh ke depan (sekitar 50°) menyeimbangkan diri dengan Ts=3,5 detik, namun dengan adanya bantuan gaya dorong awal.

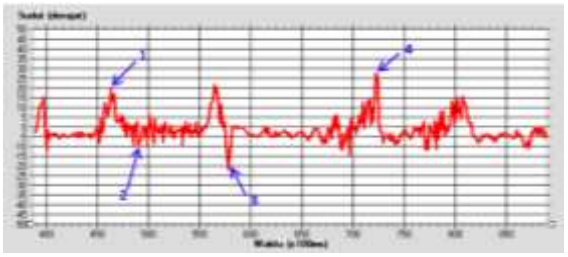
3.2.3. Pengujian Mode *Ride on*

Pengujian mode *ride on* dilakukan dengan melihat respon sistem dalam bentuk grafik saat *self-balancing scooter* berada dalam mode *ride on*. Pengujian yang dilakukan antara lain pengujian untuk mengetahui pergerakan *self-balancing scooter*, jangkauan maju dan mundur yang aman, serta pengujian dengan variasi beban pada mode *ride on*. Saat mode *ride on*, nilai konstanta parameter Kp Kp1SB=115, Kp1B=110, Kp1S=100, Kp1K=90, konstanta parameter Ti=0,5, dan konstanta parameter Td sebesar Td1SB=0,095, Td1B=0,09, Td1S=0,085, Td1K=0,08.



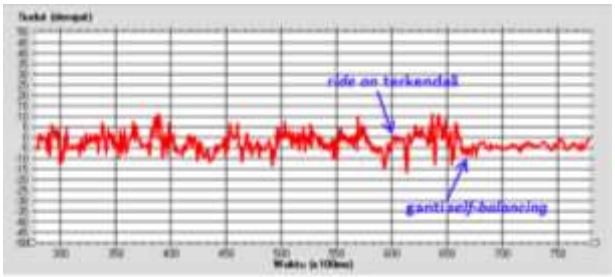
Gambar 16. Respon sistem mode *ride on* saat dikendarai

Berdasarkan respon sistem mode ride in saat dikendarai diketahui bahwa *self-balancing scooter* yang dirancang dapat dikendarai bergerak maju, mundur, berbelok ke kiri dan ke kanan.

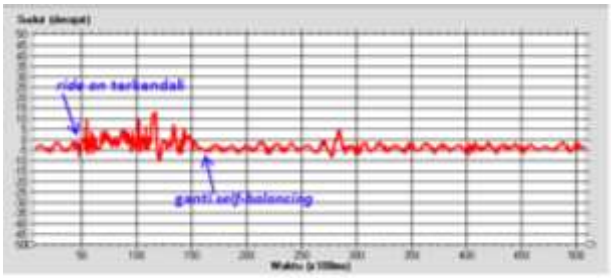


Gambar 17. Respon sistem jangkauan maju dan mundur

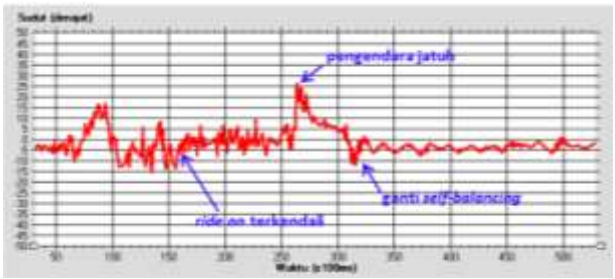
Tanda panah 1 (waktu *sampling* 462) menunjukkan kondisi maju dengan sudut 20°. Tanda panah 2 (waktu *sampling* 486) menunjukkan kondisi mundur dengan sudut 10°. Tanda panah (waktu *sampling* 579) menunjukkan kondisi mundur dengan sudut 22°. Tanda panah 4 (waktu *sampling* 723) menunjukkan kondisi maju dengan sudut 27°. Berdasarkan pengujian jangkauan maju dan mundur diperoleh jangkauan aman maju sebesar 20° dan jangkauan aman mundur sebesar -10°.



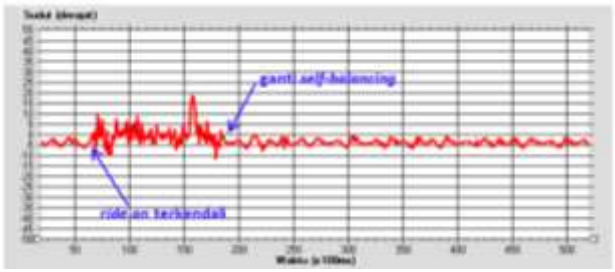
Gambar 18. Respon sistem mode *ride on* beban 42 kg



Gambar 19. Respon sistem mode *ride on* beban 68 kg



Gambar 20. Respon sistem mode *ride on* beban 73 kg



Gambar 21. Respon sistem mode *ride on* 80 kg

Pengujian variasi beban dilakukan dengan variasi beban pengendara 42 kg, 68 kg, 73 kg, dan 80 kg. Pengendara 42 kg menyeimbangkan diri di atas *self-balancing scooter* selama 6,9 detik, pengendara 68 kg selama 10,3 detik, pengendara 73 kg selama 10,4 detik, pengendara 80 kg selama 11,7 detik. Dari pengujian ini diketahui bahwa pengendara perlu menjaga keseimbangan ketika mengendarai *self-balancing scooter*.

4. Kesimpulan

Sudah dibuat sebuah *self-balancing scooter* yang dapat menyeimbangkan diri pada bidang datar dengan Ts=1,5 detik saat posisi awal dihidupkan 0° dan dapat dikendarai bergerak maju, mundur, berbelok ke kiri dan kanan. *Self-balancing scooter* pada mode *self-balancing* dapat menyeimbangkan diri dengan kendali PID *tuning* fuzzy dengan nilai konstanta parameter Kp sebesar KpSB=50, KpB=40, KpS=30, dan KpK=20, konstanta parameter Ti sebesar 0,5, dan konstanta parameter Td sebesar TdSB=0,05, TdB=0,045, TdS=0,042, dan TdK=0,04. Sedangkan pada mode *ride on*, *self-balancing scooter* dapat dikendarai dengan nilai konstanta parameter Kp sebesar Kp1SB=115, Kp1B=110, Kp1S=100, Kp1K=90, konstanta parameter Ti sebesar 0,5, dan konstanta parameter Td sebesar Td1SB=0,095, Td1B=0,09, Td1S=0,085, Td1K=0,08.

Referensi

[1]. Elvin, Toh Boon Heng., *Development of a Self Balancing scooter*, Nanyang Technologycal University, Mei 2007.

[2]. Cahyono, Bambang Nur, *Self-balancing Scooter Menggunakan Kendali Proporsional Integral Derivatif*, Penelitian Jurusan Teknik Elektro Universitas Diponegoro, Semarang, 2013.

[3]. Jantzen, Jan, *Tuning Of Fuzzy PID Controllers*, Technical University of Denmark, Department of Automation, 1999.

[4]. ---------, *Atmega32 Data Sheet*, http://www.atmel.com.

[5]. --------, *MPU-6050 3-axis accelerometer + 3-axis gyroscope Module Data Sheet*, http://www.invensense.com/mems/gyro/mpu6050.html, September 2015.

[6]. Ronnback, Sven, *Development of a INS/GPS navigation loop for an UAV*, Lulea University of Technology, Februari 2000.

[7]. --, *Low-Pass filter* http://en.wikipedia.org/wiki/Low-pass_filter, Diakses November 2015

[8]. Bejo, Agus., C&AVR Rahasia Kemudahan Bahasa C dalam Mikrokontroler ATMega8535. Graha Ilmu, Yogyakarta, 2008.

[9]. Roger's Hobby Center, *A Guide to Understanding LiPo Batteries*. Tersedia: www.rogershobbycenter.com. Diakses 18 November 2015.

[10]. Fahmizal, Driver Motor DC pada Robot Beroda dengan Konfigurasi H-Bridge MOSFET. Tersedia: www. fahmizaleeits.wordpress.com. Diakses Desember 2015.

[11]. ----------, Solitronics Engineering Ltd. RL5-2 Door Switch, http://www.solitronics.com

[12]. Setiawan, Iwan, *Kontrol PID untuk Proses Industri*, Elex Media Komputindo, Jakarta, 2008.

[13]. Pappis, Costas P, *Fuzzy Reasoning*, University of Piraeus, Greece.

[14]. Jang, Jyh Shing Roger, Chuen Tsai Sun, Eiji Mizutani. Neuro Fuzzy and Soft Computing, Prentice-Hall International, Inc, 1997