Jurnal Teknik Elektro Vol. 13 No. 2
Juli - Desember 2021

P-ISSN 1411 - 0059
E-ISSN 2549 - 1571

Implementasi Kendali Keseimbangan Gerak *Two Wheels Self balancing robot* Menggunakan *Fuzzy Logic*

Khoirudin Fathoni*, Ababil Panji Pratama, Nur Azis Salim, dan Vera Noviana Sulistyawan

Teknik Elektro, Universitas Negeri Semarang
Kampus Sekaran, Gunungpati, Semarang, 50229, Indonesia
**Corresponding author. Email: khoirudinfathoni@mail.unnes.ac.id*

Abstract— Self balancing robot is a two-wheeled robot that only has two fulcrums so that this robot is an unbalanced system. Therefore, a control system that can maintain the stability of the robot is needed so that the robot can keep in standing position. This study aims to design a self-balancing robot and its control system which improves the robot's performance against the maximum angle of disturbance that can be overcome. The control system used is based on fuzzy logic with 9 membership functions and 81 rules. The control system is applied to the ESP-32 microcontroller with the MPU-6050 sensor as a feedback position of the robot and DC motor as an actuator. Complementary filters are added to the MPU-6050 sensor readings to reduce noise to obtain better robotic tilt angle readings. The improvement of this research compared to previous research based on fuzzy is the addition of the number of membership functions from 7 to 9 and the embedding of a complementary filter on the MPU-6050 sensor output reading. The result shows that the designed self balancing robot which has dimensions of 10cm x 18cm x 14.5cm can cope with the maximum disturbance angle up to 17.5º.

Keywords— self balancing robot, fuzzy logic control, complementary filter

Abstrak —*Self balancing robot* adalah robot beroda dua yang hanya memiliki dua titik tumpu sehingga robot ini merupakan sistem yang tidak seimbang. Oleh karena itu, diperlukan sistem kendali yang dapat menjaga kestabilan robot agar robot dapat tetap pada posisi berdiri. Penelitian ini bertujuan untuk merancang *self balancing robot* beserta sistem pengendaliannya yang meningkatkan performa robot terhadap sudut gangguan maksimal yang dapat diatasi. Sistem kendali yang digunakan berbasis logika *fuzzy* dengan 9 fungsi keanggotaan dan 81 aturan. Sistem kendali diterapkan pada mikrokontroler ESP-32 dengan sensor MPU-6050 sebagai umpan balik posisi robot dan motor DC sebagai aktuator. *Complementary filter* ditambahkan pada pembacaan sensor MPU-6050 untuk mengurangi *noise* agar diperoleh hasil pembacaan sudut kemiringan robot yang lebih baik. Peningkatan penelitian ini dibandingkan dengan penelitian sebelumnya yang berbasis *fuzzy* adalah adanya penambahan jumlah dari 7 menjadi 9 fungsi keanggotaan *fuzzy* serta disematkannya *complementary filter* pada pembacaan sensor MPU-6050. Hasil penelitian menunjukkan bahwa *self balancing robot* yang dirancang mempunyai dimensi 10cm x 18cm x 14.5cm mampu mengatasi sudut gangguan maksimal hingga 17,5º.

Kata kunci— robot *balancing*, kendali logika fuzzy, filter komplementer

I. PENDAHULUAN

Salah satu jenis robot yang ada di tengah perkembangan teknologi robotika saat ini adalah *self balancing robot*. *Self balancing robot* adalah robot yang hanya mempunyai dua roda pada kedua sisinya sehingga tidak mampu seimbang tanpa adanya kontroler [1]. *Self balancing robot* merupakan pengembangan lebih lanjut dari pendulum terbalik yang diletakkan di atas kereta beroda [2]. Karakteristik utama yang menjadi pembeda dari robot beroda tiga/lebih adalah jumlah titik tumpu robot, yang mana pada robot dua roda hanya mempunyai dua titik tumpu yang menjadikannya tidak stabil [3]. Agar robot dapat bergerak stabil dan tegak lurus terhadap bidang datar, dibutuhkan suatu rangkaian mekanik, elektronik, dan metode kontrol yang baik [4].

Beberapa penelitian telah dilakukan dalam pembuatan *hardware* dan metode kendali untuk menyeimbangkan *self balancing robot*. Diantaranya, mengimplementasikan sistem kendali *Proportional, Integral, Derivative* (PID) [5] - [7]. Pada beberapa penelitian tersebut, parameter kendali PID ditentukan secara *trial and error* [5]. Disebutkan bahwa penelitian tersebut dapat dikembangkan lebih lanjut menggunakan metode *auto tunning* agar dapat menghasilkan kontrol yang lebih optimal serta mendapatkan parameter kontrol PID yang tepat. Performa dan penentuan parameter kontrol PID juga dipengaruhi oleh desain mekanik robot seperti tinggi dan beratnya. Pada penelitian [7] kendali PID digabungkan dengan filter Kalman untuk mendapatkan sudut dari sensor *Inertial Measurement Unit* (IMU) yang digunakan sebagai acuan untuk memutar motor.

Kemudian penelitian [2], [8] - [10] telah menerapkan kendali logika *fuzzy* untuk mengatur kestabilan gerak pada *self balancing robot.* Performa yang dihasilkan dari penerapan *fuzzy logic* pada *self balancing robot* dipengaruhi oleh

Received 19 November 2021, Accepted 18 December 2021, Published 20 December 2021.
DOI: https://doi.org/10.15294/jte.v13i2.33414

beberapa faktor diantaranya bentuk dan jumlah fungsi keanggotaan *input* dan *output*, serta aturan *fuzzy*. Penelitian [2] menerapkan kontrol *fuzzy* dengan menggunakan 2 buah *input* pada proses fuzzifikasi dan menggunakan 16 aturan serta menambahkan filter Kalman pada pembacaan sensornya. Penelitian tersebut menyatakan robot mampu mencapai kesetimbangannya kembali walaupun masih terdapat *overshoot*. Penelitian [8] menggunakan menggunakan 5 fungsi keanggotaan dan 25 aturan pada kendali *fuzzy* yang diimplementasikan. Hasil perfoma robot yang dihasilkan hanya mampu dapat menyeimbangkan dengan stabil pada kemiringan hingga mencapai 10º. Penelitian [9] mengunakan inferensi *fuzzy* dengan pendekatan Takagi-Sugeno-Kang dengan menerapkan 7 variabel fungsi keanggotaan dan 49 aturan *fuzzy* dengan hasil dalam simulasi Matlab robot mampu seimbang dalam kemiringan 0,5º dalam waktu 0,2 detik. Kemudian penelitian [10] dalam simulasinya menggunakan Matlab dengan sudut gangguan 5º mampu mencapai derajat kemiringan nol dalam waktu kurang dari 2 detik dengan 5 fungsi keanggotanaan dan 25 aturan *fuzzy*.

Sementara itu penelitian [5] - [7] menerapkan *self balancing robot* berbasis PID dengan mengatasi sudut masing-masing 5º, 9º, 10º. Untuk mendapatkan hasil pembacaan sensor yang lebih baik, perlu adanya filter untuk meminimalisir *noise*. Penelitian [11] dan [12] masing-masing menerapkan kontrol PID dan *fuzzy* pada *self balancing robot* namun belum menggunakan filter pada pembacaan sensornya. Sementara [7], yang berbasis PID dan [2] berbasis *fuzzy* telah menerapkan filter Kalman pada penelitiannya. Namun karena filter Kalman membutuhkan model *state space* sistem [13] dan membutuhkan waktu eksekusi lebih lama [14], *complementary filter* lebih dipilih untuk penelitian ini. Hal ini dilakukan dengan pertimbangan bahwa *complementary filter* dapat diimplementasikan tanpa perlu mengetahui model matematis sensor dalam proses menyaring *noise* serta *complementary filter* lebih ringan sehingga tidak membutuhkan waktu proses yang lama.

Penelitian [15] menjelaskan bahwa dengan menambahkan jumlah fungsi keanggotaan pada *input* dan *output fuzzy* mampu mengurangi *overshoot* dan pencapaian *settling time* lebih cepat. Pada penelitian sebelumnya masih menggunakan 5 dan 7 fungsi keanggotaan, dengan adanya penambahan jumlah fungsi keanggotaan dapat memperbaiki kualitas performa.

Berdasarkan dari hasil beberapa penelitian yang telah dilakukan sebelumnya, penelitian ini mengusulkan metode kendali *fuzzy* dengan jumlah 9 fungsi keanggotaan dan 81 aturan yang dilengkapi *complementary filter* pada pembacaan sensor. Pemilihan metode kontrol logika *fuzzy* daripada kontrol PID karena PID membutuhkan model sistem dan *tuning* parameter agar mencapai performa optimal [16]. Dalam melakukan *tunning* parameter PID terdapat 2 metode yaitu manual dan *auto tunning*. Pada PID manual, penentuan parameter (Kp, Ki, Kd) dilakukan secara *trial and error*, sementara *auto tunning* dilakukan melalui pemodelan matematis menggunakan Matlab. PID *auto tuning* menghasilkan performa yang lebih lebih baik dibandingkan manual/*trial and error* [16]. Sementara itu, *fuzzy* mempunyai kelebihan tidak memerlukan *tunning* parameter dan pemodelan.

Logika *fuzzy* memiliki nilai keanggotaan antara 0 sampai 1 dengan konsep samar seperti "sedikit", "sedang", dan "sangat" [17], [18]. Dalam pengambilan keputusan sistem logika *fuzzy* menggunakan basis aturan yang berfungsi untuk menyelesaikan *problem*, di mana jika sebuah sistem sulit untuk dicari model matematisnya atau mengandung ambiguitas [19], [20]. Ada dua metode inferensi *fuzzy* yang sering digunakan yaitu metode Sugeno dan metode Mamdani [21]. Perbedaan dari kedua metode tersebut terdapat pada *output* yang dihasilkan, proses komposisi aturan, dan proses defuzzifikasinya [22]. Penelitian ini menggunakan inferensi *fuzzy* Mamdani. Inferensi Mamdani dipilih karena rentang set *input* data yang digunakan relatif kecil, namun jika rentang dari set *input* lebih lebar dan kompleks metode Sugeno akan lebih sesuai [8], [23]. Keluaran *fuzzy* ditentukan oleh proses defuzzifikasi, yang mana pada penelitian ini digunakan metode defuzzifikasi *Center of Area*. Metode *Center of Area* dipilih karena telah teruji pada [24] yang menyatakan bahwa bahwa metode defuzzifikasi yang terbaik mengikuti nilai *setpoint* adalah metode *Center of Area*. Hal ini disebabkan karena hasil nilai dari penggunaan metode defuzzifikasi diambil dari titik pusat dari fungsi keanggotaan keluaran.

Berdasarkan penjelasan di atas maka tujuan yang ingin dicapai pada penelitian ini adalah untuk memperbaiki performa *self balancing robot* berdasarkan sudut yang dapat teratasi. Tujuan dicapai dengan mengimplementasikan kendali fuzzy dengan 9 fungsi keanggotaan, 81 aturan serta *complementary filter* untuk menyaring *noise* pada MPU-6050 sebagai sensor sudut. Motor DC digunakan sebagai aktuator. Mikrokontroler ESP-32 dipilih sebagai pengendali utama dalam implementasi sistem kontrol *self balancing robot* ini.

II. METODE

Penelitian ini mengusulkan pengembangan *self balancing robot* menggunakan metode kontrol *fuzzy* dengan proses fuzzifikasi menggunakan dua *input*. Setiap *input* mempunyai 9 fungsi keanggotaan sehingga pada inferensi *fuzzy* akan mempunyai 81 aturan. Pemilihan 9 fungsi keanggotaan ini dengan mempertimbangkan kemampuan dari ESP-32 karena dengan 9 fungsi keanggotaan sudah membutuhkan waktu program 5*ms*. Jika menggunakan lebih dari 9 fungsi keanggotaan, akan membutuhkan waktu proses lebih lama sehingga kestabilan robot tidak dapat tercapai. Pada proses defuzzifikasi, metode *Center of Area* dengan 9 fungsi keanggotaan juga digunakan sebagai *output* proses *fuzzy* agar masing-masing *input* fungsi keanggotaan mendapatkan satu *output* yang bersesuaian.

A. Spesifikasi ESP-32

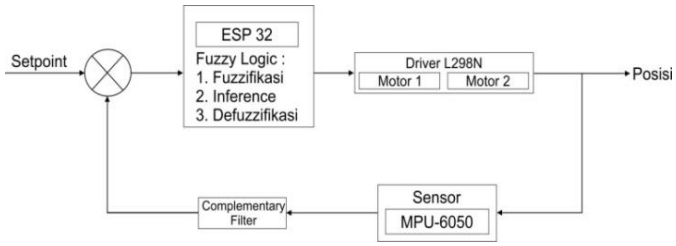
ESP-32 dikembangkan oleh *Espressif System* yang merupakan penerus dari mikrokontroler ESP8266. Mikrokontroler ESP-32 memiliki keunggulan yaitu berupa sistem berdaya lebih rendah, mempunyai arsitektur 32 bit, mempunyai *clock speed* hingga 240MHz, dan memori program hingga 16 MB. Dari kelebihan tersebut, sehingga ESP-32 lebih unggul daripada Arduino. Spesifikasi dan perbandingan dapat dilihat pada Tabel I.

B. Desain Sistem

Desain sistem ini merupakan sistem kendali loop tertutup (*closed loop*) sebagaimana ditunjukkan pada Gambar 1. Pada desain sistem ini terdapat umpan balik berupa sensor MPU-6050 yang terdiri dari *accelerometer* dan *gyroscope*. Sensor befungsi sebagai pembaca nilai sudut kemiringan robot yang nantinya akan dibandingkan dengan nilai *setpoint*.

TABEL I. PERBANDINGAN SPESIFIKASI ESP-32 DAN ARDUINO UNO

No.	Fitur	Arduino Uno	ESP-32
1.	Tegangan	5 volt	3,3 volt
2.	Arsitektur	8 bit	32 bit
3.	Flash Memory	32kB	16MB
4.	SRAM	2kB	512kB
5.	GPIO Pin (ADC/DAC)	14 (6/-)	36 (18/2)
6.	SPI/I2C/UART	1/1/1	4/2/2

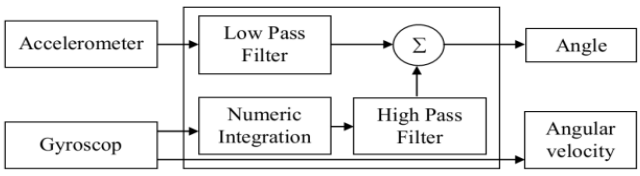


Gambar 1. Blok diagram *self balancing robot*

Setpoint sistem adalah berupa derajat yang diinginkan yaitu 0º. Nilai *output* kontroler/ESP-32 adalah tegangan dalam *Pulse Width Modulation* (PWM) untuk mengkontrol putaraan dari Motor DC yang berkorelasi langsung dengan pergerakan dan posisi sudut robot yang merupakan *output* sistem. Sensor *accelerometer* dan *gyroscope* pada MPU-6050 berfungsi sebagai umpan balik sistem untuk mengukur sudut derajat kemiringan robot saat itu. Sebelum diolah pada proses *fuzzy*, *complementary filter* digunakan sebagai penyaring sinyal hasil *output* dari sensor MPU-6050.

C. *Complementary Filter*

Saat proses pengolahan sinyal, dibutuhkan sebuah filter untuk memisahkan antara sinyal yang diinginkan serta sinyal yang tidak diinginkan (*noise*), sehingga pembacaan dari sensor akurat [25]. *Complementary filter* mengabungkan *high pass filter* untuk menyaring sinyal *output* sensor *gyroscope* dan *low pass filter* untuk menyaring sinyal *output* sensor *accelerometer* [26]. Keluaran *accelerometer* dipengaruhi oleh osilasi frekuensi tinggi yang diakibatkan karena pergerakan robot, oleh karena itu, diperlukan teknik *fusion* sensor untuk menggabungkan data terukur dari *accelerometer* dan *gyroscope* untuk mendapatkan hasil yang akurat. Blok diagram *complementary filter* dapat dilihat pada Gambar 2.



Gambar 2. Blok diagram *complementary filter*

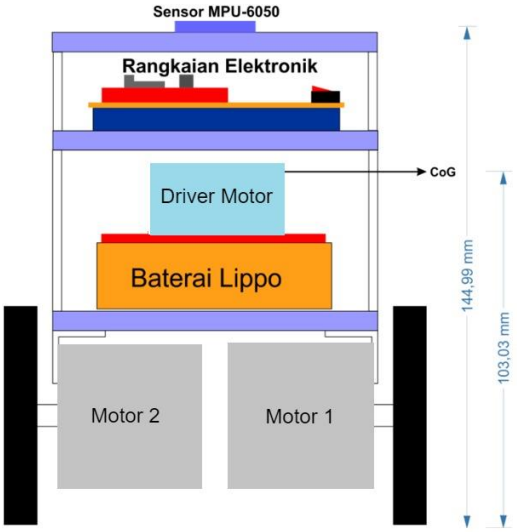
Complementary filter memiliki persamaan [27]:

$$sudut = a(\theta_{acc} + \theta_{gyro})dt + (1-a)\theta_{acc} \quad (1)$$

Di mana sudut adalah *output complementary filter*, kemudian a adalah koefisien filter, dt adalah waktu sampling, θ_{acc} adalah sudut *output* sensor *accelerometer*, dan θ_{gyro} adalah sudut *output* sensor *gyroscope*. Sementara koefisien nilai filter pada *high pass filter* yang terdapat pada sensor *gyroscope* adalah 0,98, sedangkan nilai konstanta pada *low pass filter* yang terdapat pada sensor *accelerometer* adalah 0,02 nilai tersebut didapat dari *library* MPU-6050 [28].

D. Desain Mekanik Robot

Desain mekanik pada robot menggunakan material dari akrilik dengan ketebalan 3mm. Dimensi keseluruhan dari robot ini adalah 10cm x 8cm x 14,5cm. Desain alat dan letak rangkaian elektronik dapat dilihat pada Gambar 3.



Gambar 3. Desain *self balancing robot*

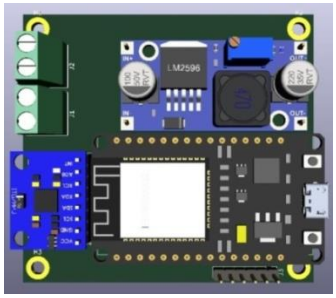
Desain *self balancing robot* yang telah dirancang memiliki *Center of Gravity* (CoG) 10,3 cm dihitung dari dasar roda. CoG diperoleh dari:

$$\text{CoG} = \frac{L \times W_f}{Wtot} = \frac{145 \times 479{,}48}{675} = 103\ mm$$

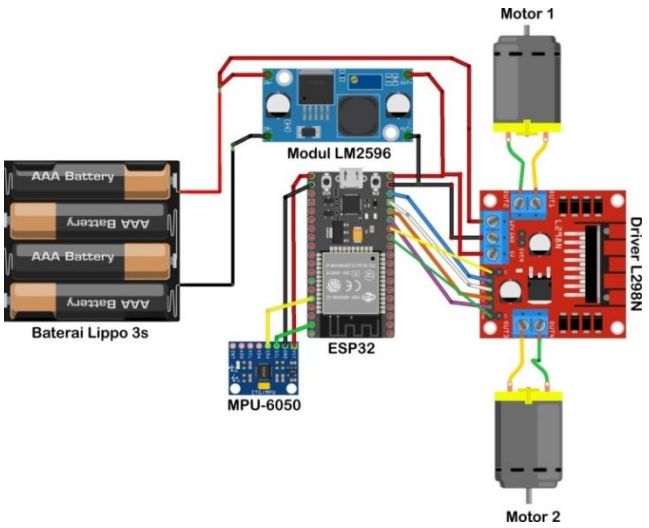
Di mana L adalah panjang vertikal robot (mm), W_f adalah berat bagian atas robot (gram), W_{tot} adalah berat total robot (gram). Titik CoG lebih dekat ke atas robot sehingga sudah cukup ideal. Meskipun demikian, karena fokus penelitian adalah menyeimbangkan robot pada titik sekitar sumbu tegak 0^0 sehingga tidak dibahas detail pengaruh posisi CoG ini.

E. Rangkaian Elektronik

Rangkaian Elektronik integrasi dari berbagai komponen-komponen elektronika yang dibutuhkan pada *self balancing robot* dapat dilihat pada Gambar 4. Rangkaian elektronika ini terdiri dari baterai lippo 3S yang berfungsi sebagai sumber tegangan, modul LM2596 sebagai *step down* DC-to-DC *regulator* karena ESP-32 memerlukan tegangan *input* 5V, sensor MPU6050 untuk umpan balik pembaca kemiringan robot, *driver* L298N sebagai *driver* motor untuk mengalirkan catu daya ke motor, dan motor DC sebagai aktuator/penggerak yang dihubungkan ke roda robot. Integrasi rangkaian elektronik robot terlihat pada Gambar 5.



Gambar 4. Tampak atas rangkaian elektronik



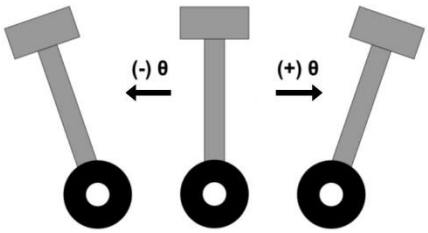
Gambar 5. Integrasi rangkaian elektronik robot

F. Perancangan Kontroler *Fuzzy*

Self balancing robot mendapatkan parameter masukan untuk fuzzifikasi yang berasal dari sensor *gyroscope* dan *accelerometer* dari MPU-6050 dalam derajat sudut kemiringn robot. Nilai sudut ini akan dijadikan himpunan *fuzzy* pada proses fuzzifikasi.

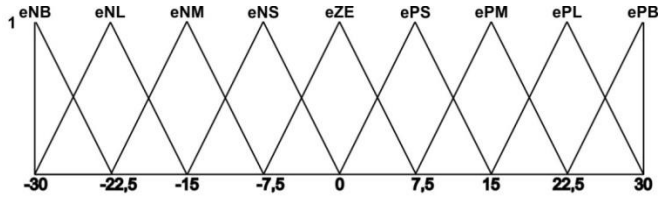
1) Fuzzifikasi

Dalam menentukan fungsi keanggotan *fuzzy*, perlu diperhatikan karakteristik dari *self balancing robot.* Robot pada kondisi stabil berada pada posisi sudut 0º. Ketika robot condong/miring ke arah kanan maka sudut berubah dari 0º s.d 90º dan ketika robot miring ke arah kiri maka sudut berubah dari 0º s.d -90º. Perubahan sudut ini akan menjadi masukan himpunan *fuzzy*. Skenario dari pergerakan *self balancing robot* terlihat pada Gambar 6.



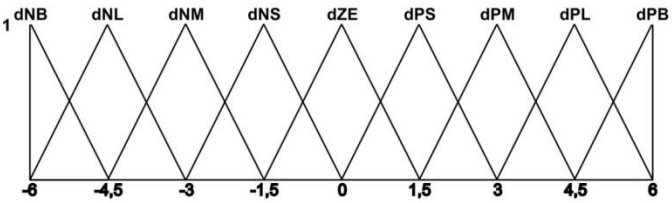
Gambar 6. Penentuan sudut positif dan negatif

Berdasarkan Gambar 6, maka dapat ditetapkan sudut kemiringan sebagai fungsi keanggotaan *error* (*error= set point* - pembacaan sudut sensor) dan perubahan nilai sudut sebagai fungsi keanggotaan *delta error (delta error = error* sekarang - *error* sebelumnya). Dari kedua fungsi tersebut akan menjadi fungsi keanggotan masukan [29]. Nilai PWM dipilih sebagai fungsi keanggotaan keluaran, yang mana nilai PWM ini nantinya disambungkan ke *driver* motor untuk menggerakkan motor DC. Pemilihan jumlah fungsi keanggotaan perlu dipertimbangkan juga berdasarkan kemampuan mikrokontroler yang digunakan. Meskipun dengan semakin banyak jumlah fungsi keanggotan membuat performa lebih baik, namun juga dapat membebani proses kerja mikrokontroler dan menyebabkan *delay*. Fungsi keanggotaan berbentuk segitiga dipilih pada penelitian ini dengan pertimbangan karena menurut penelitian [10] bentuk segitiga mampu mendapatkan performa lebih baik dari pada bentuk trapesium. Terdapat dua fungsi keanggotaan masukan yaitu *error* dan *delta error*. Fungsi keanggotan *error* terdiri dari *ErrorNegativeBig* (eNB), *ErrorNegativeLarge* (eNL), *ErrorNegative Medium* (eNM), *ErrorNegativeSmall* (eNS), *ErrorZero* (eZE), *ErrorPositiveSmall* (ePS), *ErrorPositiveMedium* (ePM), *ErrorPositiveLarge* (ePL), *ErrorPositiveBig* (ePB). Pemilihan nilai *error* sebagai masukan ini berada pada *range* (-30º - 30º), hal ini bertujuan untuk mengatasi respon sistem yang terlalu lama. Apabila rentang nilai masukan terlalu lebar/luas akan mempengaruhi waktu perhitungan *fuzzy* sehingga respon sistem lebih lama dan yang nantinya dapat mengakibatkan *delay*. Rentang nilai 9 fungsi keanggotaan *error* dapat dilihat pada Gambar 7.



Gambar 7. Fungsi keanggotaan masukan *error*

Sementara fungsi kaenggotaan *delta error* terdiri dari *Delta Error Negative Big* (dNB), *Delta Error Negative Large* (dNL), *Delta Error Negative Medium* (dNM), *Delta Error Negative Small* (dNS), *Delta Error Zero* (dZE), *Delta Error Positive Small* (dPS), *Delta Error Positive Medium* (dPM), *Delta Error Positive Large* (dPL), *Delta Error Positive Big* (dPB). Untuk nilai rentang dari masing-masing fungsi keanggotaan *delta error* dapat dilihat pada Gambar 8.



Gambar 8. Fungsi keanggotaan masukan *delta error*

2) Inferensi

Inferensi atau aturan *fuzzy* yang akan digunakan menggunakan metode Mamdani. Dari fungsi keanggotan fuzzifikasi yang terdiri dari 9 keanggotaan masing-masing dari *error* dan *delta error* maka akan menghasilkan 81 aturan *fuzzy* [30]. Berdasarkan dari aturan *fuzzy* tersebut yang nantinya akan digunakan untuk menentukan fungsi anggota keluaran *fuzzy* berdasarkan aturan-aturan yang ditunjukkan pada Tabel II.

3) Defuzzifikasi

Keluaran *fuzzy* ditentukan melalui proses defuzzifikasi yang mana pada penelitian menggunakan metode defuzzifikasi *Center of Area*. Persamaan dari *Center of Area* terdapat pada (2) dengan $\mu(z)$ nilai keanggotaan luaran (0-1) dan z nilai *output* PWM dengan rentang -255 sampai 255.

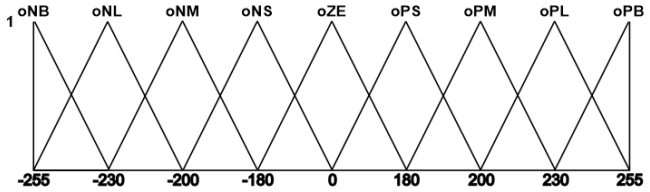
$$z = \frac{\sum_1^n \mu(z).z}{\sum_1^n \mu(z)} \quad (2)$$

Nilai *output* defuzifikasi adalah nilai PWM yang diberikan ke *driver* motor untuk memutar motor DC yang terhubung dengan roda *self balancing robot*. Pada fungsi keanggotaan *output*, juga dipilih 9 fungsi keanggotaan. Pemilihan ini bertujuan agar jumlah keluaran sebanding dengan jumlah masukan.

TABEL II. *RULE BASE FUZZY LOGIC*

Fuzzy Rule		***Error***								
		eNB	***eNL***	***eNM***	***eNS***	***eZE***	***ePS***	***ePM***	***ePL***	***ePB***
Delta Error	***dNB***	oNB	oNB	oNB	oNB	oNB	oNL	oNM	oNS	oZE
	dNL	oNB	oNB	oNB	oNB	oNL	oNM	oNS	oZE	oPS
	dNM	oNB	oNB	oNB	oNL	oNM	oNS	oZE	oPS	oPM
	dNS	oNB	oNB	oNL	oNM	oNS	oZE	oPS	oPM	oPL
	dZE	oNB	oNL	oNM	oNS	oZE	oPS	oPM	oPL	oPB
	dPS	oNL	oNM	oNS	oZE	oPS	oPM	oPL	oPB	oPB
	dPM	oNM	oNS	oZE	oPS	oPM	oPL	oPB	oPB	oPB
	dPL	oNS	oZE	oPS	oPM	oPL	oPB	oPB	oPB	oPB
	dPB	oZE	oPS	oPM	oPL	oPB	oPB	oPB	oPB	oPB

Nilai dari rentang *output* defuzzifikasi terdapat pada Gambar 9. Fungsi keanggotaan keluaran terdiri dari *Output Negative Big* (oNB), *Output Negative Large* (oNL), *Output Negative Medium* (oNM), *Output Negative Small* (oNS), *Output Zero* (oZE), *Output Positive Small* (oPS), *Output Positive Medium* (oPM), *Output Positive Large* (oPL), *Output Positive Big* (oPB).



Gambar 9. Fungsi keanggotaan keluaran

G. Perancangan *Software*

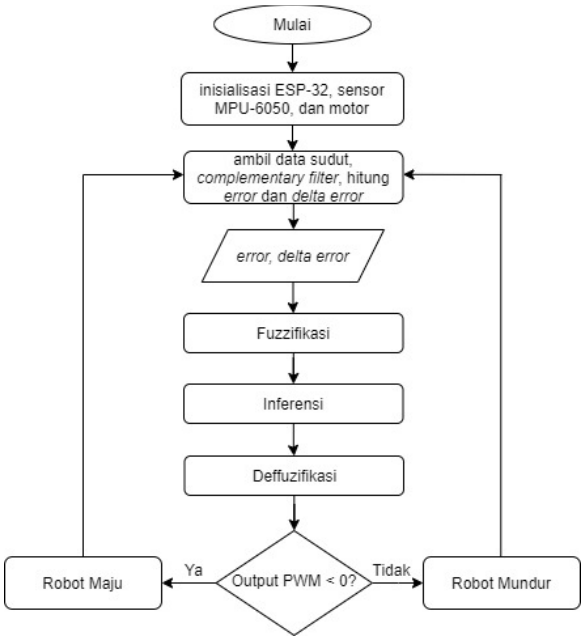
Program yang diimplementasikan ke ESP-32 dimulai dari membaca nilai sudut kemiringan dari robot dan membandingkanya dengan nilai referensi (*setpoint*), kemudian *output* program adalah PWM. PWM menentukan kecepatan motor DC yang diatur menggunakan *fuzzy logic* melalui proses fuzzifikasi, inferensi, dan defuzzifikasi. Program pengendalian meliputi pembacaan sensor, *complementary filter*, dan kendali *fuzzy* dilihat pada Gambar 10. Diagram alir pada program *self balancing robot* ini akan selalu bekerja *(looping)* dan hanya berhenti ketika saklar catu daya robot dimatikan.

III. HASIL DAN PEMBAHASAN

Hasil dari perancangan *self balancing robot* pada kondisi seimbang dapat dilihat seperti pada Gambar 11. Kemudian dilakukan beberapa pengujian yaitu pengujian dan kalibrasi sensor MPU-6050, pengujian *complementary filter*, dan pengujian performa robot dan kendali *fuzzy*.

A. Pengujian dan Kalibrasi Sensor MPU-6050

Pengujian awal dilakukan untuk mengetahui hasil konversi data mentah sensor menjadi sudut sehingga *complementary filter* belum diterapkan. Saat pengujian, data *output* sensor ditampilkan pada *serial monitor*. Hasil pembacaan sensor dikonversi dari nilai data mentah menjadi nilai sudut *accelerometer* sumbu x, y, dan z [31].



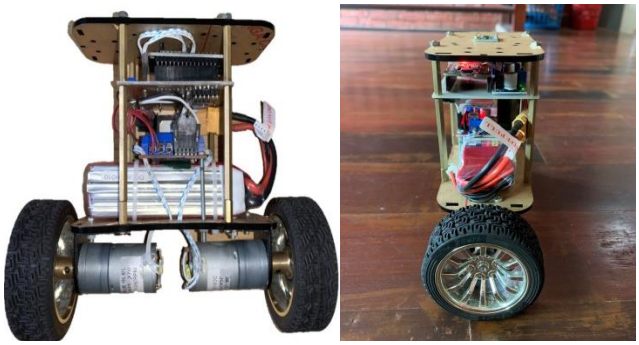
Gambar 10. Diagram alir program robot

Hasil pengujian dari sensor MPU-6050 dapat dilihat pada Gambar 12. Nilai *raw* (X, Y, Z) merupakan pembacaan data awal *accelerometer* pada setiap sumbu sementara *norm* (X, Y, Z) merupakan hasil konversi menjadi nilai sudut dalam derajat di masing-masing sumbunya. Pengambilan dan pemrosesan data mentah kemudian diubah menjadi data sudut dilakukan setiap 5 *ms*.

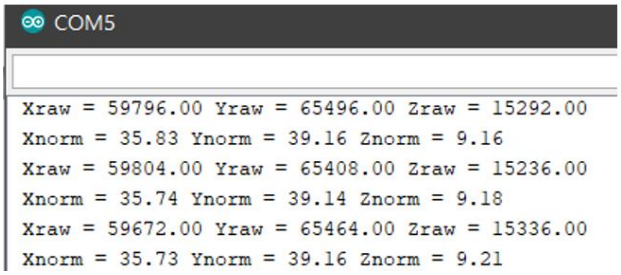
Kemudian tahap berikutnya adalah proses kalibrasi sensor MPU-6050. Hal ini bertujuan untuk memastikan bahwa antara nilai sudut pembacaan sensor dengan sudut aktual sudah mempunyai nilai yang sama sehingga diperoleh pembacaan sudut yang akurat saat robot sedang dijalankan.

Dalam melakukan kalibrasi, pertama posisikan robot ditempat yang datar dan tegak lurus. Kedua cari nilai *offset* dari sensor. Saat mencari nilai *offset* ini robot tidak boleh digerakan karena dapat mempengaruhi *nilai offset* yang di dapat. Nilai *offset* ini sangat penting dalam menentukan sudut pembacaan. Setelah mendapatkan nilai *offset*, langkah selanjutnya adalah melakukan kalibrasi dengan busur derajat. Agar mempermudah proses kalibrasi, penggunaan busur derajat digantikan dengan aplikasi busur digital *protractor*. Posisi robot dimiringkan sesuai dengan besar sudut pada busur

untuk mencocokkan nilai pembacaannya. Hasil kalibrasi sensor dapat dilihat pada Tabel III. Proses kalibrasi sensor dilakukan secara berulang pada posisi sudut yang sama. Pada proses pembacaan sudut ini sudah menggunakan *complementary filter* untuk menghasilkan pembacaan yang lebih baik. Proses kalibrasi sensor MPU-6050 dengan busur derajat didapatkan selisih *error* rata-rata pembacaan 0,172º.



Gambar 11. Hasil Implementasi *self balancing robot*



Gambar 12. *Serial monitor* pengujian MPU-6050.

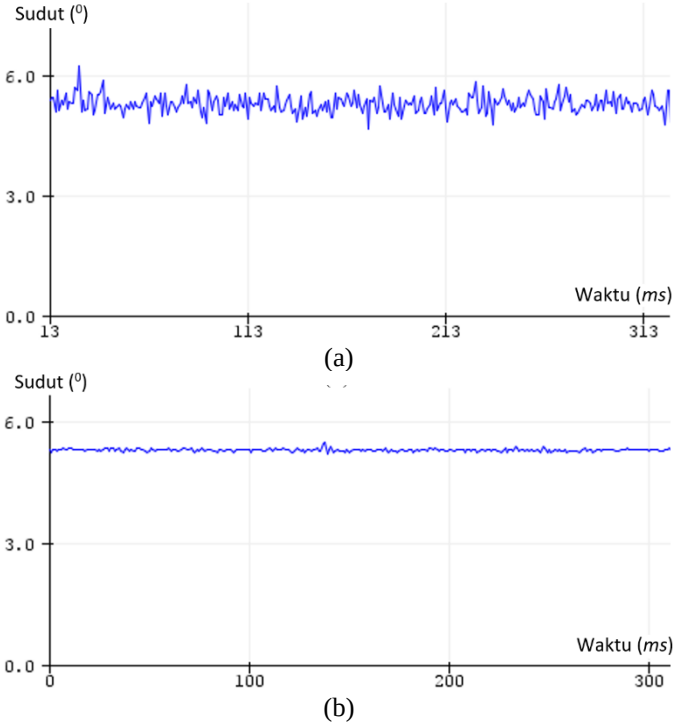
TABEL III. HASIL KALIBRASI SENSOR MPU-6050

Sudut	Pengukuran Sudut Percobaan ke-					Rerata *Error*
	1	2	3	4	5	
0º	0,15º	0,18º	0,20º	0,18º	0,2º	0,18º
15º	14,91º	14,85º	15,2º	15,17º	15,18º	0,2º
30º	30,13º	30,16º	30,18º	30,18º	30,19º	0,16º
45º	45,21º	45,19º	45,2º	45,18º	45,16º	0,18º
60º	59,88º	60,14º	60,17º	60,17º	60,19º	0,15º
90º	90,22º	90,19º	90,18º	90,14º	90,16º	0,17º
Rata-rata *error* keseluruhan						0,172º

B. Pengujian *Complementary Filter*

Proses *complementary filter* menggabungkan pembacaan hasil dari *gyroscope* dan *accelerometer*. Jika hanya menggunakan accelerometer, pembacaan sudut yang didapat masih mengandung *noise* karena terpengaruh gravitasi, sehingga dengan menggabungkan hasil pengukuran dari dua sensor, yaitu *accelerometer* dan *gyroscope* akan diperoleh pembacaan yang lebih stabil.

Gambar 13 menunjukkan respon sensor dalam sudut pada sumbu-y (derajat) terhadap waktu pada sumbu-x (ms) saat tanpa menggunakan filter (a) dan menggunakan filter (b). Terdapat osilasi pada pembacaan sensor tanpa menggunakan filter. Osilasi tersebut disebabkan adanya *noise* sehingga mengganggu dalam proses pembacaan. Sedangkan setelah dilewatkan *complementary filter*, *noise* ini dapat diminimalisir. Dengan berkurangnya *noise*, maka hasil pembacaan sudut menjadi lebih baik karena *noise* ini akan menyebabkan sensor terlalu sensitif terhadap gerakan akibat gravitasi dan getaran.



Gambar 13. Pengujian sensor MPU-6050 (a) tanpa filter dan (b) dengan *complementary filter*

C. Pengujian Performa Robot dan Kendali *Fuzzy*

Setelah melalui beberapa proses pengujian komponen dan kalibrasi, tahap selanjutnya adalah uji sistem. Uji sistem ini bertujuan untuk mengetahui apakah robot sudah dapat berfungsi secara keseluruhan. Pengujian ini mencakup dari konektivitas antar komponen dan sistem kendali pada mikrokontroler. Hal-hal yang diperhatikan dalam pengujian sistem adalah kerja sensor MPU-6050 dalam membaca sudut kemiringan serta berapa nilai *output* PWM yang akan diberikan ke motor DC. Pengujian awal dilakukan dengan menampilkan nilai-nlai tersebut ke serial monitor. Berdasarkan pengujian ini diketahui bahwa semua sistem dan komponen dalam robot dapat bekerja dengan baik. Kemudian robot dijalankan dengan hasil pengujian dapat dilihat pada [32].

Selain melakukan pengujian sistem robot, dilakukan juga simulasi *fuzzy logic toolbox* pada aplikasi Matlab. Hasil perhitungan *fuzzy* yang telah diimplementasikan pada ESP-32 akan dibandingkan dengan hasil perhitungan menggunakan Matlab. Pengujian ini bertujuan untuk melakukan validasi apakah hasil perhitungan dan implementasi *fuzzy* ESP-32 sudah sama dengan *fuzzy logic toolbox* karena dianggap hasil dari *fuzzy logic toolbox* Matlab merupakan standar yang lebih akurat. Hasil perbandingan ini terlihat pada Tabel IV. Berdasarkan data pengujian diperoleh rata-rata selisih *output* PWM hasil defuzzifikasi di ESP-32 terhadap *output fuzzy logic toolbox* Matlab adalah 2,498. Nilai *error* dan *delta error* untuk pengujian ini diperoleh dengan menggerakkan sensor MPU-6050 ke kanan/kiri dari sumbu tegak, kemudian diamati *output* PWM ESP-32 yang ditampilkan di *serial monitor* kemudian dicatat hasilnya. Kemudian dengan *error* dan *delta error* yang sama dari hasil pengamatan *serial monitor*, dimasukkan secara manual ke *fuzzy logic toolbox* Matlab dan dicatat juga hasil *output* defuzzifikasinya.

TABEL IV. PERBANDINGAN PERHITUNGAN FUZZY

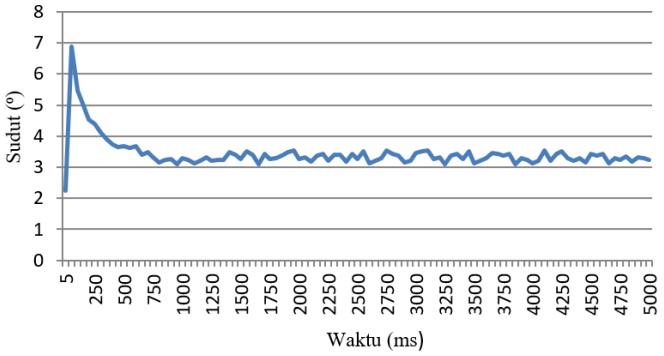
No.	*Error*	Delta *Error*	*Output* PWM		Selisih ESP32 dan Matlab
			ESP-32	Matlab	
1.	-1,26	-0,02	-10,45	-6,04	4,41
2.	-3,04	-0	-20,72	-20,4	0,32
3.	-7,85	0,01	-12,97	-12,5	0,47
4.	-11,03	-0,03	-132,24	-132	0,24
5.	-14,95	0,02	-185,87	-181	4,87
6.	-20,88	-0	-222,59	-223	0,41
7.	-26,51	0	-244,89	-233	11,89
8.	-29,01	-0,03	239,82	241	1,18
9.	6,33	0	73,72	80,5	6,78
10.	11,90	0,03	134,31	134	0,31
11.	15,61	0,01	203,51	206	2,49
12.	20,40	-0,05	201,98	201	0,98
13.	21,45	0	223,83	225	1,17
14.	23,08	0	229,26	229	0,29
Rata-rata selisih					2,498

D. Analisis Pengujian

Kendali logika *fuzzy* yang telah diimplementasikan akan diuji untuk mengetahui seberapa sudut yang dapat diatasi oleh robot sehingga dapat menyeimbangkan diri. Pada proses pengujian ini robot diberikan gangguan berupa dorongan dan dilihat bagaimana respon robot terhadap gangguan tersebut. Hasil pengujian ini ditampilkan dalam grafik yang dapat dilihat pada Gambar 14 hingga Gambar 17 untuk masing-masing sudut gangguan.

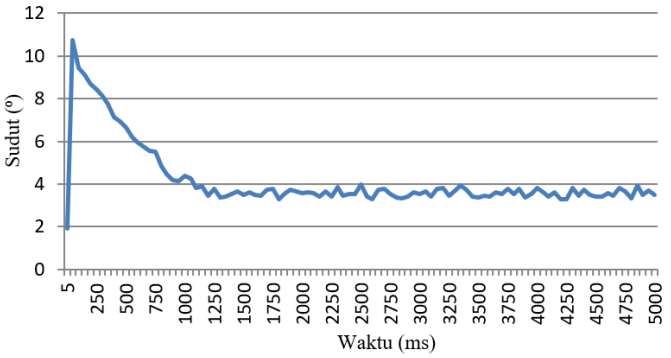
Pada pengujian ini, sudut ideal ketika robot berdiri tegak adalah pada nilai 0^0. Namun hasil yang dicapai pada penelitian ini, ketika diberi gangguan yang semakin besar terlihat pada Gambar 14 hingga Gambar 17, grafik titik setimbang robot bergeser tidak tepat di sekitar 0^0. Hal ini terjadi disebabkan beberapa hal. Grafik sensor masih dipengaruhi *error* akibat kesalahan pada saat kalibrasi nilai *offset*, *drift error*, dan kesalahan pembacaan dari sebagaimana yang ditunjukkan Tabel III. Selain itu, performa kontrol hasil dari *fuzzy* yang nilainya dapat berupa bilangan koma desimal sebagimana terlihat pada Tabel IV, tidak dapat langsung diberikan untuk menggerakkan motor karena PWM hanya dapat bernilai bilangan bulat, sehingga nilai *output fuzzy* dibulatkan ke bawah. Di samping itu, *error* perhitungan *fuzzy* ketika dibandingkan dengan Matlab juga masih terjadi. Hal ini dapat dilihat juga pada Tabel IV, yang mana jika ketiga permasalahan ini terjadi dapat mengakumulasi ketidaktepatan hasil implementasi yang mengakibatkan performa setimbang robot yang dihasilkan belum stabil di sekitar 0^0.

Gambar 14 menunjukkan grafik respon robot dengan gangguan 5º. Robot diberikan dorongan sekitar 6º. Dengan gangguan tersebut robot mampu menyeimbangkan diri pada detik ke 0,7 pada posisi stabil di rentang sudut sekitar 3,31º. Pada pengujian ini torsi dari motor DC mampu memberikan gaya perlawanan saat robot mulai miring akibat gangguan yang menyebabkan robot dapat seimbang dalam waktu singkat. Terlihat dari grafik diatas kemiringan robot dapat diimbangi dengan torsi yang diberikan motor DC dalam waktu kurang dari 250*ms* sehingga terjadi penuruan derajat dari 6º menjadi di sekitar 3º.



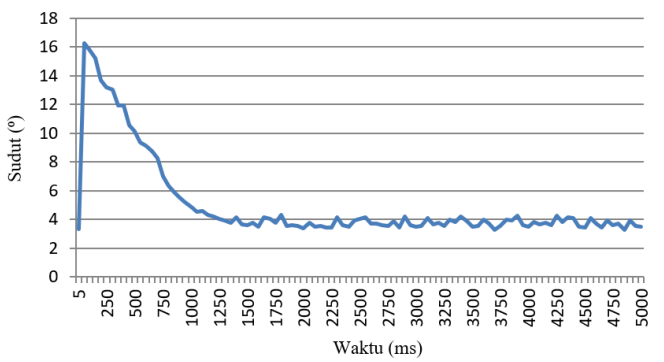
Gambar 14. Grafik respon robot pada sudut 5º.

Gambar 15 menunjukkan grafik respon robot terhadap gangguan 10º. Robot diberi dorongan sekitar 11º. Dengan gangguan tersebut robot mampu menyeimbangkan diri pada detik ke 1,2 pada posisi stabil direntang sudut 3,56º. Pada pengujian ini torsi dari motor DC masih mampu memberikan gaya perlawanan dan menyeimbangkan robot namun tidak secepat pada pengujian sebelumnya. Kemiringan sudut yang lebih besar mengakibatkan torsi dari motor DC memerlukan waktu sedikit lebih lama untuk menyeimbangkan robot. Dari grafik terlihat bahwa kemiringan robot dapat diimbangi dengan torsi yang diberikan motor DC dalam waktu 800*ms* sehingga terjadi penuruan derajat dari 11º menjadi sekitar 4º.



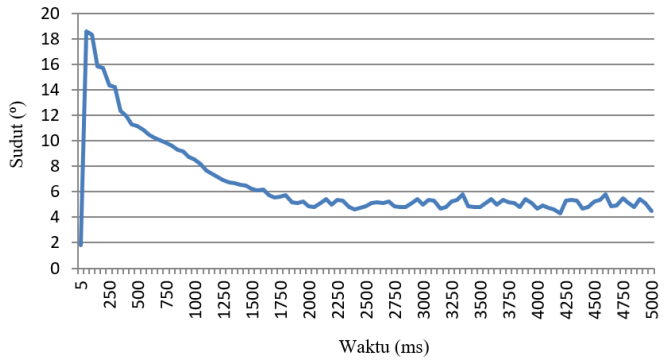
Gambar 15. Grafik Respon Robot pada sudut 10º.

Gambar 16 menunjukkan grafik respon robot dengan gangguan 15º. Robot diberikan gangguan berupa dorongan sekitar 16º. Dengan gangguan tersebut robot mampu menyeimbangkan diri pada detik ke 1,2 pada posisi stabil direntang sudut 3,74º. Pada pengujian ini torsi dari motor DC juga masih mampu memberikan gaya perlawanan saat robot dan menyeimbangkan robot meskipun dengan waktu yang lebih lambat dibandingkan dengan dua pengujian sebelumnya. Dari grafik terlihat bahwa kemiringan robot dapat diimbangi dengan torsi yang diberikan motor DC dalam waktu 950*ms* sehingga terjadi penuruan derajat dari 16º menjadi sekitar 4º.



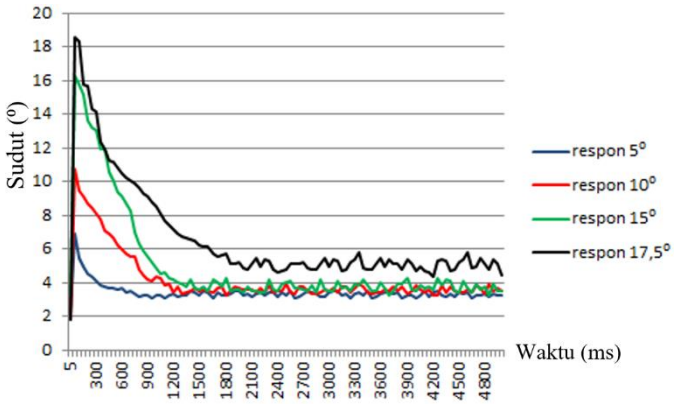
Gambar 16. Grafik respon robot pada sudut 15º

Gambar 17 menunjukkan grafik respon robot dengan gangguan 17,5º. Robot diberikan gangguan berupa dorongan sekitar 18º. Pada grafik terlihat dengan gangguan tersebut robot mampu menyeimbangkan diri pada detik ke 2 pada posisi stabil direntang sudut sekitar 5,05º. Pada pengujian motor DC membutuhkan waktu lebih lama untuk mengatasi kemiringan robot, hal ini terlihat pada grafik dengan penurunan sudut yang lebih landai. Kemiringan sudut yang lebih besar mengakibatkan proses penyeimbangan memerlukan waktu lebih lama. Dari grafik terlihat bahwa kemiringan robot dapat diimbangi dengan torsi yang diberikan motor DC dalam waktu 1,7 detik sehingga terjadi penuruan derajat dari 18º menjadi di sekitar 5º.



Gambar 17. Grafik respon robot pada sudut 17,5º

Grafik gabungan sebagai perbandingan performa pengujian untuk setiap sudut, dapat dilihat pada Gambar 18. Kemudian saat robot diberikan gangguan berupa dorongan 20º robot berusaha menstabilkan diri namun akhirnya terjatuh pada 2,8 detik. Kemudian pada pengujian gangguan 25º robot hanya bertahan sekitar 0,4 detik kemudian terjatuh. Hal tersebut terjadi karena motor DC tidak cukup menghasilkan torsi lawan yang dibutuhkan robot untuk menyetabilkan diri kembali ke posisi tegak.



Gambar 18. Grafik gabungan respon robot terhadap gangguan

E. Pembahasan

Berdasarkan dari hasil pengujian komponen, sistem dan pengambilan data, *self balancing robot* yang telah dibuat dapat bekerja dengan baik. Jika dibandingkan dengan penelitian terdahulu tentang *self balancing robot* berbasis kontrol logika *fuzzy* yaitu [2], [8] - [10], maupun berbasis PID [5] - [7], terdapat perbedaan dan perbaikan performa dari penelitian ini.

Penelitian [2] berbasis *fuzzy* dengan 4 fungsi keanggotaan dan 16 aturan *fuzzy*, fokus utama pengujian adalah pada performa filter Kalman dan tidak dilakukan uji sudut maksimal robot. Sementara [8] dan [10] keduanya menggunakan 5 fungsi keanggotaan, 25 aturan, dan tanpa filter sensor. Penelitian [8] hanya mampu menyeimbangkan dengan stabil pada kemiringan hingga sekitar 10º dan robot terjatuh ketika diberi gangguan 15^0. Sementara penelitian [10] hanya berupa simulasi *fuzzy* di Matlab dan hanya diuji pada sudut gangguan 5º dan mampu mencapai nol dalam waktu kurang dari 2 detik. Kemudian penelitian [9] menggunakan 7 fungsi keanggotaan dan 49 aturan *fuzzy* namun masih berupa simulasi di Matlab.

Sementara itu, jika dibandingkan dengan beberapa penelitian berbasis PID misalnya [5] - [7] menyatakan bahwa sudut maksimum yang dapat diatasi dalam penelitiannya adalah pada sudut 5º, 9º, dan 10º. Namun kelemahan PID adalah *tuning* parameter (Kp, Ki, Kd) akan sangat dipengaruhi oleh dimensi dan berat robot sehingga performanya tidak stabil. Sehingga penelitian ini telah menunjukkan hasil perbandingan dengan penelitian sebelumnya, dan mencapai tujuan yang ingin dicapai yaitu mampu mengatasi sudut kemiringan hingga 17,5º.

IV. Penutup

Dari penelitian ini robot mampu mempertahankan posisi agar tidak terjatuh dengan sudut maksimal yang teratasi yaitu hingga 17,5º dalam waktu 1,7 detik pada posisi stabil direntang sudut di sekitar 5º. Penelitian ini menggunakan metode *fuzzy* 9x9 fungsi keanggotaan dan 81 aturan. *Complementary filter* digunakan untuk menyaring *noise* dari MPU-6050. Pemilihan ESP-32 sebagai pengendali dilakukan dengan pertimbangan antara lain karena memiliki spesifikasi lebih tinggi dari seri Arduino seperti *clock speed*, kapasitas memori, dan kecepatan CPU yang lebih baik. Dapat disimpulkan dari hasil pengujian seluruh sistem *self balancing robot* dapat berjalan secara stabil meskipun masih terdapat *error* akan tetapi telah mencapai performa yang lebih baik dari beberapa penelitian terdahulu. Saran untuk penelitian selanjutnya adalah meningkatkan spesifikasi *hardware* yang digunakan serta pemilihan motor DC dengan torsi lebih besar, sehingga dapat digunakan metode kontrol gabungan antara PID dan *fuzzy*, *predictive control*, dan lain sebagainya, agar performa sudut maksimal lebih tinggi dan *error* dapat teratasi. Perlu diperhatikan juga dalam penentuan *offset* dan pemilihan sensor yang lebih presisi agar robot dapat mencapai titik stabil 0º. Selain itu, kendali kecepatan putaran pada kedua roda juga dapat ditambahkan agar kecepatan yang dihasilkan sama antara dua roda.

Ucapan Terima Kasih

Terima kasih kepada Fakultas Teknik Universitas Negeri Semarang atas hibah penelitian dosen pemula dalam Daftar Isian Pelaksanaan Anggaran (DIPA) Universitas Negeri Semarang (UNNES) Nomor: 023.17.2.677507/2021 tanggal 23 November 2020 sesuai dengan Kontrak No. 92.28.4/UN37/PPK.4.5/2021 tanggal 28 April 2021.

Referensi

[1] Raranda and P. W. Rusimamto, "Implementasi Kontroler Pid Pada Two Wheels Self Balancing Robot Berbasis Arduino Uno," *J. Tek. Elektro*, vol. 6, no. 2, pp. 89–96, 2017.

[2] F. Y. Bobby, Grace Susanto, Erwin Suratman, "Implementasi Robot Keseimbangan Beroda Dua Berbasis Mikrokontroler," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 3, no. 2, p. 142, 2015.

[3] C.-H. Kuo, F. Zal, and S.-L. Wu, "Development of *Fuzzy* Logic Controllers for Controlling Bipedal Robot Locomotion on Uneven Terrains with IMU Feedbacks," *Indian J. Sci. Technol.*, vol. 9, no. 28, 2016.

[4] B. Firman, "Implementasi Sensor IMU MPU6050 Berbasis Serial I2C pada Self-Balancing Robot.," *J. Teknol. Technoscientia*, vol. 9, no. 1, pp. 18–24, 2016.
[5] A. Pratama, A., & Hernawan, "Implementasi PID Controller pada Self Balancing Robot," Universitas Teknologi Yogyakarta, 2019.
[6] A. Chairunnas and T. G. Pamungka, "Sistem Kontrol Robot Penyeimbang Berbasis Arduino Menggunakan Metode Pid Dengan Komunikasi Bluetooth Hc-05," *Komputasi J. Ilm. Ilmu Komput. dan Mat.*, vol. 15, no. 2, pp. 140–151, 2019.
[7] R. S. Martins and F. Nunes, "Control system for a self-balancing robot," *Proc. 2017 4th Exp. Int. Conf. Online Exp. exp.at 2017*, no. Project I, pp. 297–302, 2017.
[8] M. S. Rachmawati, I. D., Rusimamto, P. W., & Zuhrie, "Perancangan dan Implementasi *Fuzzy* Logic Control untuk Pengaturan Kestabilan Gerak pada Two Wheels Self Balancing Robot Berbasis Arduino Uno," *J. Tek. Elektro, Unesa*, vol. Vol 9, No, pp. 717–723, 2020.
[9] J. R. Cao, C. P. Huang, and J. C. Hung, "Stabilizing controller design using *fuzzy* T-S model on two wheeled self-balancing vehicle," *Proc. IEEE Int. Conf. Adv. Mater. Sci. Eng. Innov. Sci. Eng. IEEE-ICAMSE 2016*, pp. 520–523, 2017.
[10] M. A. Akmal, N. F. Jamin, and N. M. A. Ghani, "*Fuzzy* logic controller for two wheeled EV3 LEGO robot," *Proc. - 2017 IEEE Conf. Syst. Process Control. ICSPC 2017*, vol. 2018-Janua, no. December, pp. 134–139, 2017.
[11] O. B. Kharisma, A. Wildan, Auliaullah, and F. E. Laumal, "Implementasi Sensor MPU 6050 untuk Mengukur Kesetimbangan Self Balancing Robot Menggunakan Kontrol PID," *Semin. Nas. Teknol. Informasi, Komun. dan Ind.*, no. November, pp. 357–364, 2018.
[12] A. Najmurrokhman and B. H. S. R. Wibowo, "Desain Pengendali Logika *Fuzzy* Tipe Takagi-Sugeno- Kang Untuk Mengatur Kecepatan Gerak Mobile Robot," pp. 1–8, 2018.
[13] K. Fathoni and D. Prastiyanto, "Pengenal Gerakan dengan Joystick Akselerometer Menggunakan Filter Kalman," *J. Rekayasa Elektr.*, vol. 13, no. 3, p. 172, 2017.
[14] K. Madhira, A. Gandhi, and A. Gujral, "Self balancing Robot using *Complementary filter*," *Int. Conf. Electr. Electron. Optim. Tech.*, pp. 2950–2954, 2016.
[15] I. Dwisaputra, T. Mahmoud, M. Megayanti, I. Budiawan, and P. H. R, "Pengaruh Jumlah *Input* Dan Fungsi keanggotaan *Fuzzy* Logic Control Pada Robot Keseimbangan Beroda Dua," *Manutech J. Teknol. Manufaktur*, vol. 8, no. 02, pp. 19–24, 2019.
[16] A. I. Roose, S. Yahya, and H. Al-Rizzo, "*Fuzzy*-logic control of an inverted pendulum on a cart," *Comput. Electr. Eng.*, vol. 61, pp. 31–47, 2017.
[17] F. Wahab, A. Sumardiono, A. R. Al Tahtawi, and A. F. A. Mulayari, "Desain dan Purwarupa *Fuzzy* Logic Control untuk Pengendalian Suhu Ruangan," *J. Teknol. Rekayasa*, vol. 2, no. 1, p. 1, 2017.
[18] B. F. Butar Butar, "Pemodelan dan Kendali *Fuzzy* pada DC Drive," *Setrum Sist. Kendali-Tenaga-elektronika-telekomunikasi-komputer*, vol. 4, no. 1, p. 24, 2016.
[19] F. Fahmizal, G. Setyawan, M. Arrofiq, and A. Mayub, "Logika *Fuzzy* pada Robot Inverted Pendulum Beroda Dua," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 4, p. 244, 2017.
[20] H. Putra, M. Kelviandy, and B. Eka Putera, "Penerapan Kontrol *Fuzzy* Logic Berbasis Matlab Pada Perangkat Mesin Cuci," *J. Multinetics*, vol. 4, no. 2, pp. 14–21, 2018.
[21] W. S. Rafi'ah, Almira Nindya; Pambudi, "Implementasi Sistem Kendali *Fuzzy* pada Arah Gerak Robot Finoid," *J. Inform. Rekayasa Elektron.*, vol. 3, no. 2, pp. 48–57, 2020.
[22] S. F. Riski Rullah and N. F. Prebianto, "Lampu Cerdas Multimode Menggunakan Arduino dengan Kontrol *Fuzzy* Berbasis Android," *J. Appl. Electr. Eng.*, vol. 4, no. 1, pp. 10–15, 2020.
[23] M. S. S. Virdaus and E. Ihsanto, "Rancang Bangun Monitoring Dan Kontrol Kualitas Udara Dengan Metode *Fuzzy* Logic Berbasis Wemos," *J. Teknol. Elektro*, vol. 12, no. 1, p. 22, 2021.
[24] S. Sutikno and I. Waspada, "Perbandingan Metode Defuzzifikasi Sistem Kendali Logika *Fuzzy* Model Mamdani Pada Motor Dc," *J. Masy. Inform.*, vol. 2, no. 3, pp. 27–38, 2012.
[25] I. D. Cahyo, W. Kurniawan, M. Hannats, and H. Ichsan, "Implementasi *Complementary filter* Pada Perancangan Alat Bantu Makan Penderita Parkinson," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 3, no. 1, pp. 770–773, 2019.
[26] M. Engin, "Embedded LQR Controller Design for Self-Balancing Robot," *2018 7th Mediterr. Conf. Embed. Comput.*, no. June, pp. 1–4, 2018.
[27] V. Y. Philippart, K. O. Snel, A. M. De Waal, J. S. Y. Jeedella, and E. Najafi, "Model-Based Design for a Self-Balancing Robot using the Arduino Micro-Controller Board," *2019 23rd Int. Conf. Mechatronics Technol. ICMT 2019*, pp. 1–6, 2019.
[28] R. J. L. Fetick, "MPU-6050 Light Library Documentation," January, pp. 1–8, 2021. https://github.com/rfetick/MPU6050_light/blob/master/documentation_MPU6050_light.pdf
[29] F. R. Djuliandri, A. Triwiyatno, and B. Setiyono, "Desain Sistem Kontrol *Fuzzy* Untuk Kendali Sudut Pitch Pada Model Pesawat Konvensional Dengan Tipe Fixed Wing," *J. Ilm. Tek. Elektro*, vol. 5, no. 2, p. 245, 2016.
[30] F. N. Arieni, D. Halimah, and I. Audita, "Implementasi Metode *Fuzzy* Sugeno Pada Penentuan Harga Emas 24 Karat pada Kota Medan," *Brahmana J. Penerapan Kecerdasan Buatan*, vol. 1, no. 2, pp. 116–120, 2020.
[31] V. M. T. Mubarak, D. Syauqy, and M. H. H. Ichsan, "Implementasi Wearable Device Untuk Klasifikasi Postur Keadaan Tubuh Berbasis Data Sensor MPU6050 Menggunakan Metode Naive Bayes," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, no. 12, p. 964X, 2018.
[32] A. P. Pratama, "Demo Robot." [Online]. Available: https://drive.google.com/file/d/1XsyzDkvCBs3HTPc7hqoukOIhr_GD4quT/view?usp=sharing.