JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 13 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v13i3.6940

ROBOT SELF-BALANCING DUA RODA DENGAN PID-FUZZY DAN WALL-FOLLOWING

Nastiti Dhea Imtinan[1], Dyah Ayu Ramadhani[2], Arga Abitama [3], Najmi Miftahulhuda[4], Ardy Seto Priambodo[5]

[1,2,3,4]Universitas Negeri Yogyakarta; Jl. Mandung, Pengasih, Kulon Progo, Yogyakarta; (0274) 774625

Keywords:
Self-Balancing;
Wall-Following;
Kontrol PID;
Logika Fuzzy.

Corespondent Email:
nastitidhea.2023@student.uny.ac.id

Abstrak. Penelitian ini mengembangkan robot dua roda dengan kemampuan self-balancing dan wall-following menggunakan kombinasi PID control dan fuzzy logic Mamdani. Sistem dirancang menggunakan sensor MPU6050 untuk mendeteksi sudut kemiringan dan HC-SR04 untuk mengukur jarak ke dinding, dengan Kalman filter sebagai pemroses sinyal. PID control digunakan untuk menjaga kestabilan sudut terhadap setpoint, sementara fuzzy logic mengatur arah gerak robot agar tetap sejajar dengan dinding. Pengujian dilakukan dengan beberapa konfigurasi parameter untuk mendapatkan performa optimal. Hasil menunjukkan bahwa robot mampu menjaga jarak 10–20 cm dari dinding tanpa menabrak, serta mempertahankan sudut kemiringan mendekati 0° dengan error < 2° dan waktu stabilisasi kurang dari 5 detik. Sistem mampu merespon gangguan kecil maupun besar secara adaptif. Perpaduan kontrol ini terbukti efektif dalam meningkatkan kestabilan dan responsivitas robot dalam lingkungan dinamis. Penelitian ini memberikan kontribusi dalam pengembangan robot mobile cerdas yang dapat beroperasi secara mandiri di ruang sempit atau area terbatas.

Abstract. *This research presents a two-wheeled robot capable of self-balancing and wall-following using a hybrid control system combining PID and Mamdani fuzzy logic. The system utilizes an MPU6050 sensor to detect tilt angle and an HC-SR04 ultrasonic sensor to measure wall distance, with a Kalman filter used for signal processing. The PID controller maintains balance by minimizing angular error, while fuzzy logic manages the robot's movement to remain parallel to the wall. Several parameter configurations were tested to achieve optimal performance. Results show the robot can maintain a 10–20 cm distance from the wall without collision, with a tilt angle error of less than 2° and stabilization time under 5 seconds. The system responds adaptively to both minor and major disturbances. This combined control method enhances stability and responsiveness in dynamic environments. The research contributes to the development of intelligent mobile robots capable of autonomous operation in narrow or constrained spaces.*

1. PENDAHULUAN

Dalam bidang teknik, sistem pengendalian *closed-loop* penting untuk menjaga stabilitas dan merespons perubahan secara cepat. PID *controller* sering digunakan, tetapi kurang fleksibel jika diterapkan pada sistem nonlinier. Untuk mengatasinya, digunakan pendekatan kecerdasan buatan, salah satunya fuzzy *logic*. Logika ini meniru cara manusia mengambil keputusan menggunakan bahasa alami.

Fuzzy *logic* telah banyak diterapkan dalam sistem kendali nonlinier, termasuk robotika *mobile*[1]. Salah satu tantangan utama adalah

menjaga keseimbangan pada robot dua roda yang tidak stabil seperti *inverted pendulum*. Sistem ini membutuhkan pengendali yang mampu menyesuaikan gerakan berdasarkan sudut kemiringan dan kecepatannya. Fuzzy *logic* Mamdani cocok digunakan karena memberikan koreksi yang fleksibel berdasarkan kondisi sensor[2].

Selain itu, navigasi juga penting dalam robot bergerak. Salah satu metode yang umum digunakan adalah *wall-following*, yaitu kemampuan robot untuk berjalan sejajar dengan dinding menggunakan sensor jarak seperti HC-SR04.

Penelitian ini memiliki kebaruan karena menggabungkan PID *controller* dan fuzzy *logic* secara paralel untuk robot *self-balancing* dengan fitur *wall-following* pada *platform* ESP32. Kombinasi ini belum banyak dibahas dalam penelitian sebelumnya dan diharapkan mampu meningkatkan stabilitas dan adaptivitas robot[3].

Tujuan dari penelitian ini adalah merancang dan menguji robot dua roda berbasis ESP32 yang dapat menyeimbangkan diri dan mengikuti dinding. Sistem dikendalikan oleh gabungan PID dan fuzzy Mamdani yang bekerja secara bersamaan. Dengan pendekatan ini, robot diharapkan mampu bergerak stabil dan responsif di lingkungan yang dinamis.

2. TINJAUAN PUSTAKA

2.1 ESP-32

Gambar 1. Mikrokontoler ESP-32.

Mikrokontroler ESP32 dirancang untuk menangani fungsi kontrol sistem tertanam secara *real time*. Dalam sistem robot roda dua, ESP32 menggunakan sinyal PWM untuk mengontrol kecepatan motor DC melalui driver motor dan terhubung ke sensor inersia seperti MPU6050. Karena kemampuan pemrosesan yang kuat dan dukungan untuk konektivitas yang tepat, ESP32 dapat menangani data sensor dan menjalankan algoritme kontrol berbasis fuzzy *logic* secara bersamaan. Sebagai konsekuensinya, sistem dapat dengan stabil, responsif, dan kooperatif mengikuti dinding dan mempertahankan keseimbangan diri dalam lingkungan yang berubah. Arduino IDE, sebuah program dengan antarmuka yang mudah digunakan yang mendukung berbagai *library* sistem tertanam dan format kode, digunakan untuk memprogram ESP32.

ESP32 adalah mikrokontroler dengan fitur WiFi dan Bluetooth yang mendukung eksekusi kontrol secara real-time. Chip ini memiliki keunggulan dari sisi pemrosesan paralel dan konsumsi daya yang rendah, cocok untuk sistem robotik cerdas. Namun, sebagian besar penelitian sebelumnya hanya menggunakan ESP32 sebagai unit kendali tunggal tanpa pemanfaatan penuh terhadap multitasking atau kombinasi pengendali adaptif.

2.2 MPU 6050

Gambar 2. Sensor MPU 6050.

MPU6050 adalah modul sensor inersia yang mengukur kecepatan rotasi dan akselerasi linier pada saat yang sama dengan mengintegrasikan akselerometer tiga sumbu dan giroskop ke dalam satu *chip*. Tujuan sensor ini adalah untuk memberikan pelacakan *real time* yang sangat akurat terhadap orientasi dan pergerakan objek. Penghematan pin dan integrasi yang efisien dengan berbagai mikrokontroler dimungkinkan melalui transfer data melalui protokol I2C. MPU6050 sering digunakan dalam aplikasi termasuk kontrol gerakan robotik, stabilisasi perangkat, dan sistem navigasi.

MPU6050 menggabungkan akselerometer dan giroskop untuk menghasilkan data orientasi yang akurat. Penggunaan sensor ini telah menjadi standar dalam robot keseimbangan,

tetapi tantangan utamanya adalah *noise* dan *drift* jangka panjang. Beberapa studi menyarankan penggunaan Kalman Filter untuk meningkatkan stabilitas data, seperti yang dilakukan oleh Ma et al. (2024).

2.3 HC-SR04

Gambar 3. Sensor HC-SR04.

Sensor ultrasonik HC-SR04 menggunakan gelombang suara frekuensi tinggi untuk mengukur jarak suatu benda. Sensor ini, yang terdiri dari pemancar dan penerima ultrasonik, memanfaatkan konsep pantulan gelombang. Waktu tempuh setelah gelombang ultrasonik dipancarkan, mengenai objek, dan dipantulkan kembali serta ditangkap oleh sensor digunakan untuk menghitung jarak. Dengan tingkat presisi yang sangat tinggi, rentang pengukuran HC-SR04 yang umum digunakan adalah 2 cm hingga 400 cm. Karena keandalannya dalam mendeteksi objek tanpa kontak fisik, sensor ini banyak digunakan dalam robotika, sistem otomasi, penghindaran rintangan, dan perangkat pemantauan jarak dan keberadaan[4].

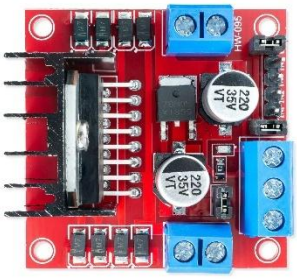
Sensor ini digunakan untuk *wall-following* karena mampu mengukur jarak dengan cukup presisi. Namun, akurasinya sangat dipengaruhi oleh permukaan pantulan dan posisi objek. Chen et al. (2020) menambahkan sistem fuzzy adaptif untuk mengatasi fluktuasi sinyal dari HC-SR04, tetapi belum banyak yang menggabungkannya secara paralel dengan kontrol keseimbangan[5].

2.4 Motor DC GearBox

Gambar 4. Motor DC *GearBox*.

Motor DC *Gearbox*, juga dikenal sebagai sistem reduksi roda gigi, memungkinkan motor DC memodifikasi torsi output dan kecepatan rotasi untuk memenuhi tuntutan aplikasinya. Motor DC menggunakan medan magnet dan arus listrik untuk mengubah energi listrik menjadi energi mekanik. *Gearbox* mengurangi kecepatan rotasi motor sambil meningkatkan torsi, memungkinkan kontrol gerakan yang lebih akurat dan kuat. Motor DC *gearbox* banyak digunakan dalam berbagai aplikasi yang membutuhkan kontrol kecepatan dan daya dorong yang tepat dan andal, termasuk robotika[6].

2.5 Driver Motor L298N



Gambar 5. *Driver M*otor L298N.

Driver motor L298N, modul pengendali motor berdasarkan sirkuit terintegrasi *H-bridge*, mengontrol kecepatan dan arah putaran motor stepper dan DC. Modul ini sempurna untuk motor berdaya tinggi karena dapat mengatur arus hingga 2A per saluran. Tanggung jawab utama driver ini termasuk mengontrol polaritas sinyal untuk menentukan arah putaran motor dan mengatur kecepatan menggunakan metode modulasi lebar pulsa (PWM). *Driver* L298N mudah diintegrasikan dengan mikrokontroler karena memiliki beberapa *port* yang menangani input logika, catu daya, dan output motor.

2.6 Kombinasi PID-Fuzzy

Penggabungan pengendali PID dan logika fuzzy telah banyak diteliti. Rachmawati et al. (2020) menggunakan fuzzy untuk menjaga keseimbangan robot dua roda, tetapi tidak menggabungkannya dengan PID sebagai koreksi adaptif.. Ma et al. (2024) mengembangkan fuzzy Kalman untuk menstabilkan sistem dua roda, namun sistemnya belum mengakomodasi navigasi berbasis *wall-following*. Ordy et al. (2020)

menyoroti bahwa integrasi fuzzy cenderung lambat dalam respon awal, sehingga penggabungan dengan PID menjadi solusi untuk mempercepat stabilisasi.

Studi terbaru oleh Fadlun dan Sugiharto (2025) menunjukkan efektivitas logika fuzzy untuk mengontrol robot dua roda secara stabil, tetapi penelitian tersebut tidak mengintegrasikan sistem wall-following maupun platform ESP32 secara eksplisit[7].

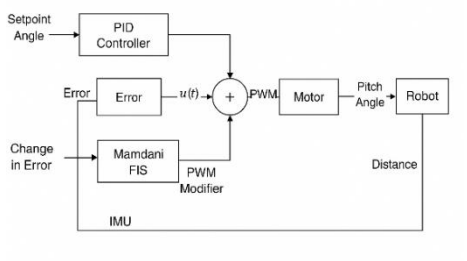
2.7 Gap dan Peran Penelitian

Kebanyakan penelitian sebelumnya hanya fokus pada satu fitur kontrol: entah keseimbangan (*self-balancing*) atau navigasi (*wall-following*), dan jarang menggabungkannya dalam satu sistem. Selain itu, sebagian penelitian tidak menggunakan arsitektur mikrokontroler ringan seperti ESP32 secara optimal[7]. Penelitian ini melengkapi kekurangan tersebut dengan:

- Menggabungkan PID *controller* dan fuzzy Mamdani secara paralel.
- Mengintegrasikan dua fitur utama robotik, yaitu *self-balancing* dan *wall-following*.
- Menggunakan *platform* ESP32 yang ringan dan efisien.
- Memproses data sensor secara *real time* untuk navigasi adaptif.

3. METODE PENELITIAN

3.1 Diagram Blok Sistem



Gambar 6. Diagram Blok Sistem.

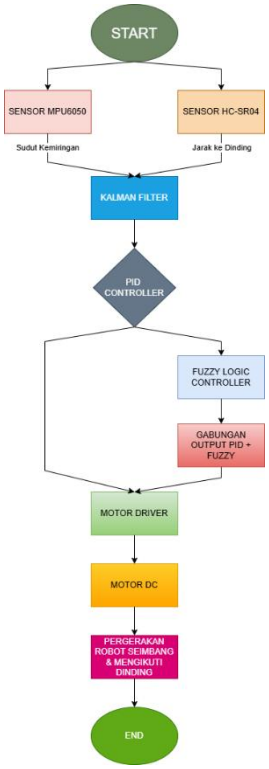
Diagram tersebut mengilustrasikan bagaimana sistem kontrol *closed-loop* berdasarkan fuzzy *logic* dan PID menjaga keseimbangan dan navigasi robot *wall-follower* yang dapat menyeimbangkan diri sendiri. Mikrokontroler, seperti ESP32, berfungsi sebagai unit pemrosesan utama dalam sistem ini[7][8]. Nilai referensi atau disebut *setpoint*, ditentukan dengan menghitung sudut ideal agar robot tetap tegak. Hal ini sering kali ditetapkan secara konstan atau dinamis sesuai kebutuhan.

Sensor IMU (MPU6050) membaca sudut kemiringan dan data akselerasi sudut tubuh robot. Dengan membandingkan data dari sensor ini dengan nilai *setpoint*, tingkat ketidakakuratan atau penyimpangan dapat dipastikan. Setelah jumlah kesalahan ini diproses, pengontrol PID menghitung sinyal kontrol dengan mempertimbangkan jumlah kesalahan saat ini, jumlah kesalahan sebelumnya, dan perubahan kesalahan dari waktu ke waktu.

Nilai *error* dan modifikasi *error* merupakan input tambahan untuk sistem fuzzy *logic*. Fuzzy *logic* menentukan seberapa banyak koreksi yang harus dilakukan pada sinyal kontrol utama berdasarkan kriteria linguistik. Sinyal kontrol akhir yang lebih tahan terhadap gangguan mendadak atau perubahan lingkungan dihasilkan dengan menggabungkan perubahan fuzzy *logic* ini dengan output PID.

Sinyal kontrol akhir diubah menjadi sinyal PWM dan dikirim ke *driver motor* (seperti L298N), yang mengatur arah dan kecepatan rotasi motor DC *gearbox*. Untuk menjaga keseimbangan dan mengikuti dinding pada jarak yang konstan, maka untuk menjaga keseimbangan dan mengikuti dinding pada jarak yang konstan, motor akan menyesuaikan gerakan robot dalam pengaturan ini.

3.2 Alur Kerja Sistem



Gambar 7. Diagram Alir Kerja Sistem.

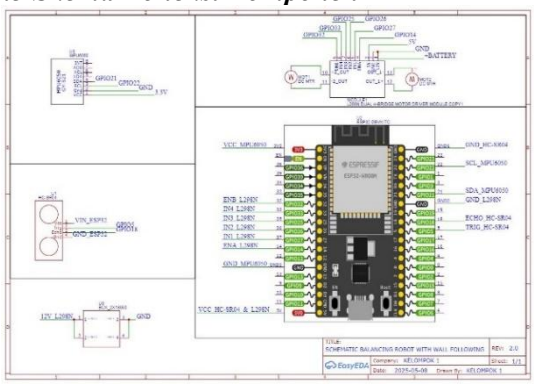
Sistem robot terdiri dari dua fungsi utama:

- Menjaga keseimbangan (*self-balancing*).
- Menavigasi sejajar dinding (*wall-following*)

Kedua fungsi dikendalikan oleh kombinasi PID *controller* dan fuzzy *logic* secara paralel. Alir kerja sistem :

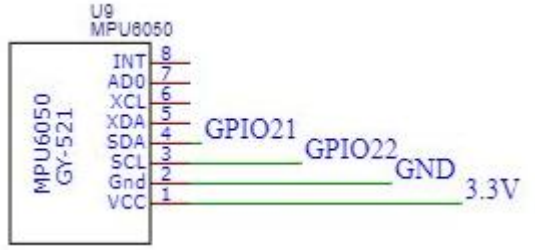
- START
  Sistem mulai bekerja saat robot dinyalakan.
- Sensor MPU6050
  Membaca sudut kemiringan robot (*pitch*) sebagai masukan untuk menjaga keseimbangan.
- Sensor HC-SR04
  Mengukur jarak ke dinding untuk mendukung fitur *wall-following*.
- Kalman Filter
  Menggabungkan data sensor untuk menghasilkan estimasi sudut yang halus dan akurat.
- PID *Controller*
  Menghitung sinyal koreksi berdasarkan *error* sudut dan kecepatan sudut untuk menjaga keseimbangan robot.
- Fuzzy *Logic Controller*
  Memberikan koreksi tambahan yang adaptif berdasarkan pola perubahan error, untuk menyempurnakan output PID.
- Gabungan Output PID + Fuzzy
  Menghasilkan sinyal kontrol akhir yang lebih stabil dan responsif terhadap gangguan.
- Motor Driver (L298N)
  Menerjemahkan sinyal kontrol menjadi aksi fisik pada motor.
- Motor DC
  Menggerakkan roda berdasarkan sinyal dari motor driver.
- Pergerakan Robot
  Robot bergerak secara seimbang dan menjaga jarak terhadap dinding secara otomatis.
- END
  Proses berulang secara terus-menerus selama robot aktif, menjaga performa stabil di lingkungan dinamis.

3.3 Skema Koneksi Komponen



Gambar 8. Skema Pengkabelan angkaian.

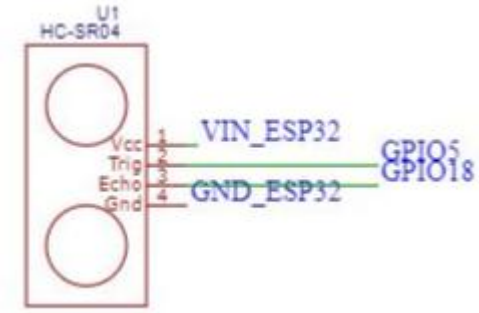
Gambar 8 menunjukkan skema rangkaian, di mana sumber tegangan utama untuk motor DC diperoleh dari baterai 18650 sebesar 12V. Berikut adalah konfigurasi pin dan sumber tegangan yang digunakan pada motor driver L298N, ESP32, MPU6050, dan HC-SR04 untuk mengontrol motor DC dan sistem keseluruhan:



Gambar 9. Skema Pengkabelan MPU.

Tabel 1. Konfigurasi Pin MPU 6050, ESP-32.

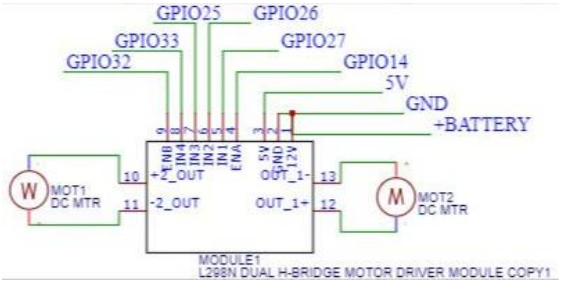
MPU6050	ESP32
SDA	GPIO21
SCL	GPIO22
VCC	3.3V
GND	GND



Gambar 10. Skema Pengkabelan HC-SR04.

Tabel 2. Konfigurasi Pin HC-SR04, ESP-32.

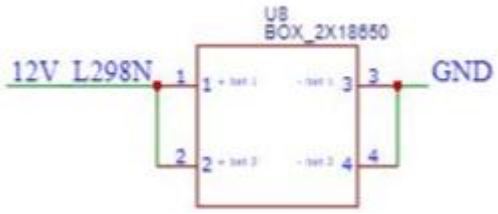
HC-SR04	ESP32
Trig	GPIO5
Echo	GPIO18
VCC	VIN (5V)
GND	GND



Gambar 11. Skema Pengkabelan L298N.

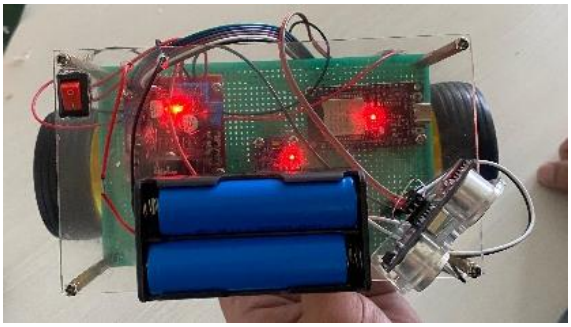
Tabel 3. Konfigurasi Pin L298N, ESP-32.

L298N	ESP32
ENA (Enable Motor A)	GPIO14
IN1 (Arah Motor A)	GPIO27
IN2 (Arah Motor A)	GPIO26
IN3 (Arah Motor B)	GPIO25
IN4 (Arah Motor B)	GPIO33
ENB (Enable Motor B)	GPIO32



Gambar 12. Skema Pengkabelan Motor.

Mikrokontroler ESP32 mengendalikan robot, memproses data sensor, dan mengatur motor DC untuk keseimbangan serta *wall-following* melalui pin GPIO. *Motor driver* L298N menggerakkan roda kiri dan kanan; ENA (GPIO14) dan ENB (GPIO32) ESP32 mengatur kecepatan via PWM, IN1 (GPIO27), IN2 (GPIO26), IN3 (GPIO25), dan IN4 (GPIO33) mengontrol arah. L298N menggunakan 5V dari ESP32 untuk logika dan 12V dari baterai 18650 untuk motor, dengan OUT1-OUT2 dan OUT3-OUT4 menggerakkan motor. Sensor MPU6050 (3.3V dari ESP32) membaca data giroskop dan akselerometer untuk keseimbangan, terhubung via I2C (SDA ke GPIO21, SCL ke GPIO22). Sensor HC-SR04 (5V dari ESP32) untuk *wall-following*, menggunakan TRIG (GPIO5) dan ECHO (GPIO18) untuk mengukur jarak. GND semua komponen disatukan. Berikut *wiring* rangkaian pada *hardware*:



Gambar 13. Implementasi *Wiring* Rangkaian.

3.4 Pengumpulan Data

Pengumpulan data pada robot *self-balancing* dan *wall-following* yang menggunakan motor

DC dengan *gearbox* dilakukan dengan mengandalkan sensor IMU dan sensor jarak ultrasonik. Sensor IMU digunakan untuk membaca sudut kemiringan (*pitch*) dan percepatan sudut (*angular velocity*) dari tubuh robot. Data dari akselerometer dan *gyroscope* dikombinasikan menggunakan metode kalman filter untuk menghasilkan estimasi sudut yang stabil dan responsif terhadap perubahan. Estimasi sudut kemiringan $\theta(t)$ dihitung dengan persamaan:

$$\theta(t) = \alpha \cdot (\theta(t - \Delta t) + \omega \cdot \Delta t) + (1 - \alpha) \cdot \theta acc$$

Di mana ω adalah percepatan sudut dari *gyroscope*, θ_{acc} adalah sudut dari *akselerometer*, Δt adalah interval waktu pembacaan sensor, dan α adalah konstanta filter antara 0 dan 1. Nilai sudut ini sangat penting dalam sistem kendali keseimbangan, karena digunakan sebagai umpan balik dalam algoritma pengendalian, seperti PID, untuk menyesuaikan kecepatan dan arah putaran motor agar robot tetap dalam posisi tegak.

Selain sensor IMU, sistem *wall-following* pada robot ini juga bergantung pada sensor jarak ultrasonik untuk mendeteksi jarak antara robot dan dinding. Sensor ini bekerja dengan mengukur waktu tempuh gelombang ultrasonik dari pemancar ke permukaan dinding dan kembali ke penerima. Jarak dihitung dengan menggunakan rumus:

$$d = \frac{v \times \Delta t}{2}$$

Di mana d adalah jarak ke dinding, v adalah kecepatan suara di udara (sekitar 343 m/s), dan Δt adalah selisih waktu antara pengiriman dan penerimaan sinyal. Nilai jarak ini digunakan sebagai referensi untuk menjaga agar robot tetap bergerak sejajar dengan dinding. Ketika jarak terlalu dekat atau terlalu jauh, sistem akan menyesuaikan kecepatan motor pada salah satu sisi untuk mengoreksi arah gerak.

Karena tidak menggunakan sensor kecepatan mekanis, kecepatan motor diperkirakan berdasarkan nilai PWM (*Pulse Width Modulation*) yang dikirimkan ke motor. Hubungan antara nilai PWM dan kecepatan motor diasumsikan linier dan ditentukan melalui proses kalibrasi awal. Hubungan ini dapat dinyatakan dengan persamaan:

$$vmotor \approx k \cdot PWM$$

Di mana $_{Vmotor}$ adalah estimasi kecepatan linier motor, dan k adalah konstanta kalibrasi. Dengan memantau perubahan sudut dari IMU dan jarak dari sensor ultrasonik, sistem dapat mengontrol arah dan kecepatan motor secara *real-time* tanpa perlu mengukur jumlah putaran secara langsung.

3.5 Implementasi PID + Fuzzy Logic

Penggunaan kombinasi PID *controller* dan Fuzzy Logic dengan metode Mamdani dalam sistem robot *self-balancing* dan *wall-following* bertujuan untuk meningkatkan stabilitas dan respons adaptif robot terhadap gangguan maupun variasi lingkungan. Sistem ini mengandalkan sensor MPU6050 untuk memperoleh informasi sudut kemiringan (*pitch*) dan sensor ultrasonik HC-SR04 untuk mengukur jarak dari dinding, serta motor DC *gearbox* yang dikendalikan melalui *driver* L298N.

Untuk menjaga keseimbangan, sistem menggunakan pengendali PID dengan struktur klasik, yang dihitung berdasarkan kesalahan antara sudut *setpoint* (misalnya -3.0°) dan sudut aktual yang terukur. Estimasi sudut diperoleh dari kombinasi data *akselerometer* dan *gyroscope* yang telah difilter menggunakan Kalman Filter. Persamaan PID yang digunakan adalah sebagai berikut:

$$u(t) = Kp \cdot e(t) + Ki \cdot \int^{e} (\tau) d\tau + Kd \cdot dtde(t)$$

Di mana e(t) adalah selisih antara sudut *setpoint* dan sudut aktual. Parameter Kp, Ki, dan Kd mengatur bobot komponen proporsional, *integral,* dan *derivatif* untuk menghasilkan output pengendali berupa nilai PWM. Output ini kemudian dikirimkan ke motor kiri dan kanan untuk menjaga keseimbangan robot secara dinamis terhadap gaya gravitasi atau gangguan eksternal.

Namun, untuk menghadapi kondisi non-linear seperti perubahan permukaan lantai, variasi jarak ke dinding, atau beban tidak

seimbang, digunakan pendekatan tambahan berupa Fuzzy Logic Controller (FLC). Logika fuzzy dirancang menggunakan metode Mamdani dengan dua input utama: error sudut kemiringan dan perubahan error sudut (rate of error), serta satu output berupa fuzzy PWM modifier yang akan mengubah sinyal PWM dari PID controller secara adaptif.

Input fuzzy dinyatakan dalam tiga himpunan linguistik:

- Untuk error sudut: *Negative* (N), *Zero* (Z), *Positive* (P)
- Untuk perubahan error: *Decreasing* (D), *Stable* (S), *Increasing* (I)

Output fuzzy PWM menggunakan tiga kategori juga: *Low* (L), *Medium* (M), dan *High* (H). Aturan fuzzy *(fuzzy rule base)* sebagai berikut:

- *IF error is Negative AND change is Decreasing THEN output is Low*
- *IF error is Zero AND change is Stable THEN output is MediumIF error is Positive AND change is Increasing THEN output is High*

Proses inferensi dilakukan menggunakan metode Mamdani, dengan fungsi keanggotaan berbentuk segitiga (*triangular*) dan proses defuzzifikasi menggunakan metode centroid. Output dari sistem fuzzy ini tidak menggantikan PID, melainkan mengubah bobot atau menggeser output PWM PID agar lebih adaptif.

Rumus akhir penggabungan dapat dituliskan sebagai:

$$PWMfinal = PWMPID + PWMfuzzy$$

di mana PWM PID merupakan output dari persamaan PID (1), dan PWM fuzzy adalah hasil defuzzifikasi dari sistem fuzzy Mamdani. Pendekatan ini memberikan fleksibilitas tambahan, memungkinkan sistem tetap seimbang saat menghadapi kondisi dinamis yang sulit ditangani oleh PID konvensional saja.

3.6 Tuning Parameter PID

Tuning PID dilakukan menggunakan pendekatan eksperimental dan heuristic tuning dengan beberapa iterasi, menggunakan tiga tingkatan parameter:

- Kp: Proporsional (mempercepat koreksi *error*).
- Ki: Integral (mengurangi *steady-state error*).
- Kd: Derivatif (mengurangi *overshoot*).

Langkah tuning :

- Awalnya, nilai Kp ditingkatkan bertahap untuk mempercepat respon robot.
- Kemudian Ki dimasukkan secara perlahan untuk mengurangi *error* tetap.
- Terakhir, Kd disesuaikan untuk mengurangi osilasi.

Nilai-nilai awal diambil berdasarkan literatur, kemudian disesuaikan berdasar uji empiris. Contoh konfigurasi parameter ada di Tabel 4–6 pada hasil dan pembahasan.

3.7 Tuning Fuzzy Logic

Fuzzy *controller* dirancang menggunakan metode Mamdani dengan dua input:

- *Error* sudut (θ error).
- Perubahan *error* (Δθ).

Himpunan linguistik:

- Input: Negative, Zero, Positive.
- ΔError: Decreasing, Stable, Increasing.
- Output: Low, Medium, High.

Metode tuning fuzzy:

- Aturan fuzzy ditentukan berdasarkan logika kendali sederhana (lihat Bagian 3.4).
- Fungsi keanggotaan berbentuk segitiga (triangular).
- Defuzzifikasi dilakukan dengan metode centroid.

Setelah mendapatkan output fuzzy, nilai ini digunakan untuk memodifikasi output PID agar lebih adaptif terhadap perubahan cepat atau gangguan kecil.

3.8 Evaluasi Kinerja Sistem

Data dari setiap percobaan dicatat dalam bentuk log sudut dan jarak, lalu dianalisis menggunakan:

- Error rata-rata sudut kemiringan (°): mengukur akurasi keseimbangan.
- Waktu stabilisasi (s): waktu untuk mencapai sudut ±1°.

- Error jarak ke dinding (cm): mengukur keakuratan *wall-following*.
- Grafik respons sistem: untuk membandingkan sebelum dan sesudah tuning.

3.9 Repositori Kode Program

Untuk mendukung replikasi dan pengembangan lebih lanjut, kode program sistem kendali gabungan PID dan fuzzy *logic* Mamdani yang digunakan dalam penelitian ini telah disediakan secara terbuka dan dapat diakses melalui repositori GitHub berikut: https://github.com/dy44h/robot-pid-fuzzy-wallfollowing.git

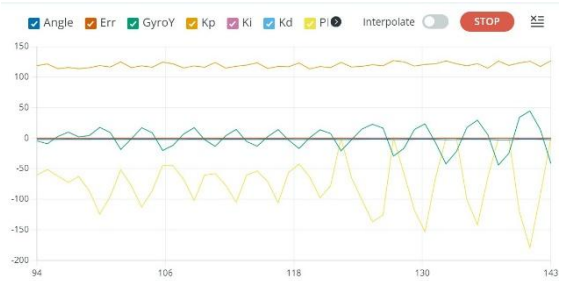
4. HASIL DAN PEMBAHASAN

4.1 Percobaan Pertama

Tabel 4. Tuning Percobaan Pertama.

	S	M	L
Kp	110	140	200
Ki	0.07	0.07	0.07
Kd	0.03	0.6	1.2

Pada proses tuning PID dan fuzzy ini, pemilihan parameter awal. Parameter proportional (Kp) dibagi menjadi tiga tingkatan, yaitu Kp_S = 110.0 untuk error kecil, Kp_M = 140.0 untuk error sedang, dan Kp_L = 200.0 untuk error besar. Pemilihan nilai ini digunakan agar sistem dapat memberikan koreksi yang semakin kuat seiring bertambahnya error, sehingga robot dapat dengan cepat kembali ke posisi seimbang ketika terjadi gangguan. Untuk parameter derivative (Kd), mengunakkan Kd_S = 0.03, Kd_M = 0.6, dan Kd_L = 1.2. Pemilihan awal ini digunakan agar efek peredaman meningkat sesuai besarnya error. Namun, pemilihan parameter Kd tersebut tidak sesuai, karena nilai Kd_M lebih besar dari Kd_L, padahal, koreksi derivatif seharusnya semakin besar pada error yang lebih besar untuk mengurangi osilasi secara efektif. Sementara itu, parameter integral (Ki) dipilih sebesar 0.07 dengan harapan dapat membantu mengoreksi error sisa (*steady-state error*) secara perlahan. Namun, nilai ini cukup besar dan berisiko menyebabkan penumpukan integral (*integral wind-up*), yaitu akumulasi error yang terlalu cepat sehingga memicu *overshoot* dan osilasi yang sulit diredam. Pemilihan parameter awal ini dilakukan untuk memberikan dasar dalam pengujian sistem, sehingga kekurangan pada pemilihan parameter dapat diperbaiki untuk meningkatkan kestabilan dan respons sistem.



Gambar 14. Hasil Pembacaan Plotter dari Tuning Pertama.

Setelah pemilihan parameter awal, sistem diuji dan hasilnya yang dilihat pada Serial Plotter. Pada grafik yang ditampilkan, pada bentuk grafik untuk meluhat sudut (Angle), error (Err), kecepatan sudut (GyroY), dan output PID. Dari hasil percobaan, terlihat bahwa PID output berfluktuasi dengan amplitudo yang cukup besar, yang menandakan sistem belum stabil. Selain itu, GyroY menunjukkan osilasi yang signifikan, dan error tidak langsung menurun ke nol, sehingga koreksi sistem tidak berjalan efektif. Grafik ini menunjukkan bahwa sistem bereaksi terlalu agresif pada error sedang dan tidak cukup responsif pada koreksi kecil. Hal ini menunjukkan bahwa pemilihan parameter, khususnya pada Kd, perlu diperbaiki.

```
Angle:-1.72,Err:-0.74,GyroY:9.43,Kp:116.45,Ki:0.07,Kd:0.26,PID:-88.65
Angle:-1.85,Err:-0.60,GyroY:0.37,Kp:113.72,Ki:0.07,Kd:0.11,PID:-68.68
Angle:-2.07,Err:-0.38,GyroY:-14.21,Kp:122.56,Ki:0.07,Kd:0.23,PID:-43.41
Angle:-2.00,Err:-0.45,GyroY:-8.24,Kp:121.80,Ki:0.07,Kd:0.21,PID:-53.24
Angle:-1.92,Err:-0.53,GyroY:2.88,Kp:113.98,Ki:0.07,Kd:0.14,PID:-60.88
Angle:-1.77,Err:-0.68,GyroY:8.84,Kp:116.06,Ki:0.07,Kd:0.25,PID:-81.25
Angle:-1.93,Err:-0.52,GyroY:1.14,Kp:113.45,Ki:0.07,Kd:0.11,PID:-59.15
Angle:-1.98,Err:-0.47,GyroY:-10.44,Kp:122.81,Ki:0.07,Kd:0.23,PID:-55.52
Angle:-1.84,Err:-0.61,GyroY:0.94,Kp:113.92,Ki:0.07,Kd:0.12,PID:-69.64
Angle:-1.73,Err:-0.73,GyroY:8.20,Kp:116.18,Ki:0.07,Kd:0.25,PID:-86.79
Angle:-1.87,Err:-0.59,GyroY:-3.37,Kp:117.93,Ki:0.07,Kd:0.16,PID:-68.99
Angle:-1.97,Err:-0.48,GyroY:-11.41,Kp:123.28,Ki:0.07,Kd:0.24,PID:-56.66
```

Gambar 15. Hasil Pembacaan Monitor dari Tuning Pertama.

Pada percobaan pertama, parameter yang digunakan adalah Kd_S = 0.03, Kd_M = 0.6, dan Kd_L = 1.2, dengan nilai Ki sebesar 0.07. Berdasarkan hasil pengamatan melalui Serial Monitor, permasalahan utama terletak pada distribusi nilai Kd yang tidak proporsional. Kd_S yang seharusnya memberikan respons derivatif yang cukup untuk koreksi kecil justru terlalu rendah, sehingga sistem menjadi lambat

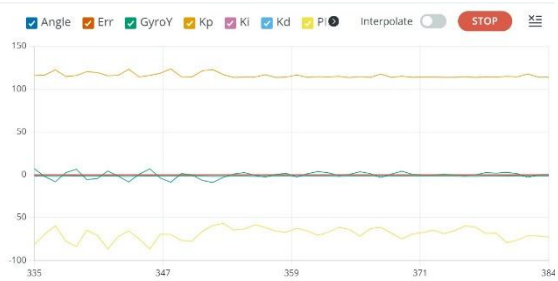
dalam merespons gangguan kecil. Selain itu, distribusi parameter Kd tidak logis karena Kd_M lebih besar dari Kd_L. Seharusnya, pada error besar, Kd_L perlu memiliki nilai yang paling tinggi agar sistem mendapatkan peredaman yang optimal untuk menekan osilasi besar. Ketidakseimbangan pemilihan parameter ini menyebabkan robot bereaksi terlalu agresif pada error sedang, sementara pada error besar, sistem kurang mampu meredam osilasi dengan baik. Selain itu, nilai Ki yang relatif besar mempercepat akumulasi error, yang berisiko menimbulkan integral *wind-up*. Hasilnya, sistem mengalami *overshoot* dan osilasi yang berulang. Kondisi ini menyebabkan robot menjadi tidak stabil, khususnya saat melakukan koreksi kecil dan saat menghadapi gangguan sedang hingga besar. Dari hasil ini, dapat disimpulkan bahwa pemilihan parameter memerlukan penyesuaian agar sistem dapat memberikan respons yang lebih proporsional dan stabil sesuai dengan skala error yang dihadapi.

4.2 Percobaan Kedua

Tabel 5. Tuning Percobaan Kedua.

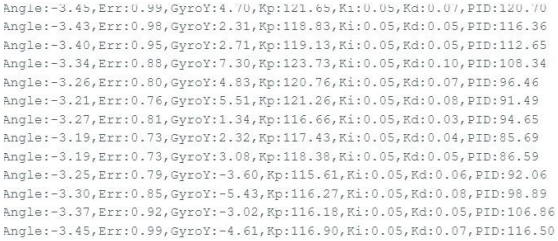
	S	M	L
Kp	110	140	200
Ki	0.05	0.05	0.05
Kd	0.01	0.05	1.0

Pada percobaan kedua, proses tuning parameter PID dan fuzzy dilakukan dengan pendekatan yang lebih baik dibandingkan percobaan sebelumnya. Parameter proportional (Kp) ditetapkan pada tiga tingkatan, yaitu Kp_S = 110.0, Kp_M = 140.0, dan Kp_L = 200.0. Pemilihan nilai ini dipilih agar sistem dapat memberikan koreksi yang proporsional dan sesuai dengan besar kecilnya error. Semakin besar deviasi sudut yang terjadi, semakin besar juga koreksi yang diberikan agar robot dapat dengan cepat kembali ke posisi seimbang. Parameter derivatif (Kd) ditentukan dengan distribusi yang lebih progresif, yaitu Kd_S = 0.01, Kd_M = 0.05, dan Kd_L = 1.0. Nilai Kd ini diberikan agar sistem memberikan respons derivatif yang baik untuk gangguan kecil dan respons yang lebih kuat untuk gangguan besar. Hal ini digunakan agar sistem dapat meredam osilasi dengan baik ketika terjadi deviasi sudut yang besar, tetapi tetap halus saat melakukan koreksi kecil. Sementara itu, nilai integral (Ki) diturunkan menjadi 0.05 untuk mengurangi risiko terjadinya integral *wind-up*, yaitu kondisi di mana akumulasi error berlebihan dapat menyebabkan *overshoot* dan ketidakstabilan sistem



Gambar 16. Hasil Pembacaan Plotter dari Tuning Kedua.

Dari grafik serial plotter, terlihat bahwa kurva sudut (Angle) cukup stabil dan tidak menunjukkan fluktuasi yang tinggi, menunjukkan bahwa sistem dapat menjaga keseimbangan dengan baik. Error yang terlihat lebih konstan dan kecil, menunjukkan sistem dapat mengoreksi kesalahan dengan cukup cepat. Kurva GyroY menunjukkan fluktuasi kecil di sekitar nol, yang menunjukkan bahwa respons sistem terhadap gangguan sudah cukup halus dan tidak menghasilkan getaran berlebih. Selain itu, kurva PID output tampak konsisten dan tidak mengalami lonjakan besar, menunjukkan bahwa sistem tidak mengalami koreksi berlebihan dan dapat menjaga kestabilan robot. Namun, kelemahan kecil yang terlihat dari grafik adalah kurang responsifnya sistem terhadap perubahan minor, yang dapat dilihat dari lambatnya perubahan PID output terhadap error kecil. Hal ini konsisten dengan hasil analisis serial monitor yang menunjukkan bahwa nilai Kd_S yang terlalu kecil mempengaruhi kecepatan sistem dalam melakukan koreksi halus.



Gambar 17. Hasil Pembacaan Monitor dari Tuning Pertama.

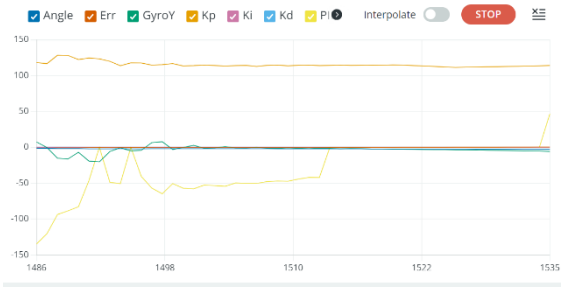
Berdasrkan hasil dari serial monitor, didapatkan beberapa parameter penting yang menunjukkan kinerja sistem secara *real-time*. Sudut (Angle) terlihat lebih stabil di kisaran -3.19° hingga -3.45°, menunjukkan bahwa sistem dapat menjaga posisi robot dalam rentang kesetimbangan yang baik. Nilai error (Err) juga kecil, berkisar antara 0.73 hingga 0.99, yang menunjukkan deviasi dari setpoint sudah minimal. Data PID output berada di rentang 85 hingga 120, yang menunjukkan bahwa koreksi yang diberikan oleh sistem terjaga dalam batas yang cukup normal dan tidak terlalu berlebih. Selain itu, nilai kecepatan sudut (GyroY) juga cukup baik, meskipun masih menunjukkan fluktuasi, nilainya relatif kecil dan cenderung kembali ke nol dengan cepat setelah koreksi dilakukan. Tetapi, dari data ini juga dapat dilihat bahwa sistem memerlukan waktu untuk menyesuaikan koreksi kecil karena nilai Kd_S yang terlalu rendah (0.01), sehingga kecepatan respons untuk error kecil masih cukup lambat. Hal ini menyebabkan sistem tidak terlalu sensitif terhadap gangguan kecil, meskipun dalam skala keseluruhan sistem tetap stabil. Selain itu, variasi PID output yang tidak terlalu fluktuatif menunjukkan bahwa sistem sudah dapat menahan *overshoot* dan osilasi secara signifikan, yang lebih baik perbaikannya dibandingkan percobaan sebelumnya.

4.3 Percobaan Ketiga

Tabel 6. Tuning Percobaan Kedua.

	S	M	L
Kp	110	140	200
Ki	0.06	0.06	0.06
Kd	0.087	0.05	1.0

Pada percobaan ketiga, proses tuning dilakukan dengan pendekatan yang lebih baik berdasarkan hasil evaluasi dari percobaan sebelumnya. Fokus utama tuning pada percobaan ini adalah meningkatkan respons sistem terhadap gangguan kecil tanpa mengurangi stabilitas keseluruhan. Nilai. Pada nilai proporsional (Kp), nilai Kp_S ditetapkan sebesar 110.0 untuk error kecil, Kp_M sebesar 140.0 untuk error sedang, dan Kp_L sebesar 200.0 untuk error besar. Untuk parameter derivatif (Kd), dilakukan peningkatan pada nilai Kd_S menjadi 0.087, sedangkan Kd_M tetap di 0.05, dan Kd_L tetap di 1.0. Peningkatan Kd_S ini digunakan agar sistem lebih cepat merespons perubahan kecil pada sudut dan error, sehingga robot dapat segera melakukan koreksi meskipun gangguan yang terjadi tergolong ringan. Parameter integral (Ki) juga disesuaikan dengan menaikkan nilainya menjadi 0.06. Penambahan ini dilakukan untuk membantu sistem dalam mengatasi error dengan lebih cepat, namun tetap dikontrol agar tidak menghasilkan osilasi atau respon berlebihan yang dapat mengganggu kestabilan robot.



Gambar 18. Hasil Pembacaan Plotter dari Tuning Ketiga.

Berdasarkan hasil pengamatan pada Serial Plotter, terlihat bahwa pada awal pergerakan, sistem memberikan respon koreksi yang cukup agresif dengan perubahan signifikan pada output PID. Garis PID menunjukkan fluktuasi besar di awal yang menunjukkan adanya respon sistem untuk segera mengoreksi posisi robot. Seiring waktu, nilai PID mulai mengecil dan mendekati nol, menunjukkan bahwa sistem mulai mencapai keseimbangan. Grafik error secara bertahap menurun menuju nol, menunjukkan bahwa koreksi sistem baik pada percobaan ketiga ini. Gerakan GyroY yang awalnya cukup aktif juga bergerak menurun, menunjukkan bahwa pergerakan motor semakin stabil dan getaran berkurang setelah sistem mendekati posisi seimbang.

```
Angle:-2.06,Err:-0.39,GyroY:0.47,Kp:112.49,Ki:0.06,Kd:0.09,PID:-44.08
Angle:-2.07,Err:-0.39,GyroY:2.42,Kp:113.05,Ki:0.06,Kd:0.11,PID:-44.20
Angle:-2.25,Err:-0.21,GyroY:-4.76,Kp:116.29,Ki:0.06,Kd:0.12,PID:0.00
Angle:-2.40,Err:-0.06,GyroY:-11.98,Kp:115.69,Ki:0.06,Kd:0.09,PID:0.00
Angle:-2.44,Err:-0.02,GyroY:-8.36,Kp:113.58,Ki:0.06,Kd:0.09,PID:0.00
Angle:-2.51,Err:0.06,GyroY:-7.20,Kp:112.97,Ki:0.06,Kd:0.09,PID:0.00
Angle:-2.58,Err:0.12,GyroY:-7.92,Kp:113.54,Ki:0.06,Kd:0.10,PID:0.00
Angle:-2.61,Err:0.16,GyroY:-5.56,Kp:112.85,Ki:0.06,Kd:0.11,PID:0.00
Angle:-2.73,Err:0.27,GyroY:-7.26,Kp:113.94,Ki:0.06,Kd:0.13,PID:0.00
Angle:-2.82,Err:0.37,GyroY:-9.16,Kp:114.92,Ki:0.06,Kd:0.14,PID:43.69
Angle:-2.90,Err:0.45,GyroY:-8.29,Kp:114.92,Ki:0.06,Kd:0.15,PID:52.73
Angle:-2.98,Err:0.52,GyroY:-7.92,Kp:115.10,Ki:0.06,Kd:0.16,PID:61.42
Angle:-3.04,Err:0.58,GyroY:-7.55,Kp:115.33,Ki:0.06,Kd:0.15,PID:68.42
Angle:-3.13,Err:0.67,GyroY:-8.31,Kp:115.93,Ki:0.06,Kd:0.16,PID:79.56
```

Gambar 19. Hasil Pembacaan Monitor dari Tuning Pertama.

Pada Serial Monitor, hasil yang ditunjukkan menunjukkan kondisi robot. Di awal percobaan, sistem mendeteksi sudut kemiringan sekitar -2 derajat dengan error yang cukup berlebih, sehingga menghasilkan output PID negatif yang besar untuk segera mengembalikan posisi robot. Seiring waktu, nilai error secara bertahap menurun hingga mendekati nol, yang ditunjukkan dengan penurunan output PID. Hal ini menunjukkan bahwa sistem dapat memperbaiki posisi secara bertahap dan akurat. Selain itu, perubahan nilai GyroY yang berfluktuasi pada awal kemudian menurun dan stabil, memperlihatkan bahwa sistem dapat mengontrol pergerakan motor dengan baik. Respon cepat yang dihasilkan, terutama dalam mengoreksi error kecil, menunjukkan bahwa peningkatan Kd_S memberikan hasil yang baik dalam mempercepat koreksi tanpa menimbulkan osilasi yang berlebihan.

4.4 Perbandingan Ketiga Percobaan

Pada percobaan pertama, parameter PID menggunakan nilai dasar dengan kombinasi Kp yang rendah dan Kd yang belum optimal. Respon sistem pada tahap ini masih cenderung lambat dalam mengoreksi error, terlihat dari grafik Serial Plotter yang menunjukkan pergerakan sudut (Angle) dan error yang memerlukan waktu cukup lama untuk stabil. Output PID di awal juga kecil, sehingga koreksi awal robot kurang agresif. Nilai Kd yang terlalu rendah menyebabkan sistem lambat dalam merespons perubahan sudut secara cepat. Hal ini berdampak pada keseimbangan robot yang masih goyang dan respon yang cenderung telat dalam mengatasi gangguan.

Pada percobaan kedua, dilakukan peningkatan pada nilai Kp dan sedikit penyesuaian pada Kd. Sistem mulai menunjukkan perbaikan dalam hal respon koreksi. Dari Serial Plotter, terlihat bahwa error mulai menurun lebih cepat dibandingkan percobaan pertama. Respon motor menjadi lebih tegas, dan GyroY menunjukkan kestabilan yang lebih baik. Namun, dalam percobaan ini, sistem masih membutuhkan waktu untuk mencapai setpoint dengan akurasi yang baik. Perbaikan yang terjadi sudah cukup signifikan dibandingkan percobaan awal, tetapi koreksi terhadap gangguan kecil masih kurang cepat. Parameter Kd yang belum cukup besar masih menjadi faktor yang membatasi kecepatan respon derivatif sistem.

Percobaan ketiga adalah tuning terbaik dari seluruh percobaan. Pada tahap ini, Kd_S ditingkatkan menjadi 0.087 dan Ki dinaikkan menjadi 0.06. Hasilnya, sistem menjadi jauh lebih responsif dan stabil. Dari Serial Plotter, terlihat bahwa error menurun drastis dalam waktu yang lebih singkat. Output PID menunjukkan koreksi yang cepat namun tetap terkontrol. Sudut (Angle) stabil dengan getaran minimal, dan nilai GyroY menjadi relatif kecil setelah koreksi dilakukan. Dari Serial Monitor, sistem mampu mengurangi error secara progresif dan output PID menunjukkan penyesuaian yang tepat sesuai kondisi sudut. Peningkatan Kd_S sangat efektif dalam mempercepat respon koreksi terhadap gangguan kecil. Tambahan Ki juga membantu menghilangkan error secara perlahan tanpa menimbulkan osilasi berlebihan.

5. KESIMPULAN

a. Penelitian ini menunjukkan bahwa pemilihan dan distribusi parameter Kd serta nilai Ki berpengaruh signifikan terhadap kestabilan dan kecepatan respons sistem kendali robot. Pada percobaan pertama, ketidaksesuaian distribusi Kd (di mana Kd_M lebih besar dari Kd_L) dan nilai Ki yang relatif tinggi menyebabkan respons sistem menjadi tidak proporsional, cenderung *overshoot*, serta tidak stabil. Percobaan kedua memperbaiki distribusi Kd secara progresif dan menurunkan Ki, sehingga menghasilkan sistem yang lebih stabil dan tidak mudah osilasi, meskipun respons terhadap gangguan kecil masih kurang sensitif akibat nilai Kd_S yang terlalu rendah. Percobaan ketiga

membuktikan bahwa peningkatan Kd_S mampu meningkatkan kepekaan sistem terhadap gangguan minor, tanpa mengorbankan stabilitas saat menghadapi error besar. Perpaduan parameter ini menghasilkan sistem yang responsif, stabil, dan cepat kembali ke setpoint, menunjukkan bahwa tuning derivatif berbasis fuzzy secara progresif efektif dalam sistem penyeimbang berbasis sudut.

b. Namun, penelitian ini memiliki beberapa keterbatasan, seperti tuning parameter yang masih dilakukan secara manual sehingga memerlukan waktu dan pengalaman untuk memperoleh hasil optimal. Selain itu, percobaan dilakukan dalam kondisi lingkungan dan beban yang relatif konstan, sehingga belum menguji adaptabilitas sistem terhadap kondisi dinamis atau perubahan lingkungan. Ke depannya, pengembangan dapat difokuskan pada penerapan algoritma tuning otomatis berbasis kecerdasan buatan, seperti fuzzy-adaptive tuning atau metode optimasi heuristik (misalnya PSO atau GA). Dengan demikian, sistem ini mampu beradaptasi dengan berbagai kondisi operasional, serta dapat diterapkan lebih luas pada robotika dan sistem kendali presisi tinggi lainnya.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

[1] W. Junfeng, Z. Wanying, and W. Shengda, “A two-wheeled self-balancing robot with the fuzzy PD control method,” *Math. Probl. Eng.*, vol. 2012, 2012.

[2] I. D. Rachmawati, P. W. Rusimamto, Endryansyah, and M. S. Zuhrie, “Perancangan dan implementasi fuzzy logic control untuk pengaturan kestabilan gerak pada two wheels self-balancing robot berbasis Arduino Uno,” *J. Tek. Elektro*, vol. 9, no. 3, pp. 717–723, 2020.

[3] Y. Ma, F. Meng, and S. Xiong, “Design and implementation of a two-wheeled self-balancing car using a fuzzy Kalman filter,” *Appl. Sci.*, vol. 14, no. 1, 2024.

[4] C. H. Chen, S. Y. Jeng, and C. J. Lin, “Mobile robot wall-following control using fuzzy logic controller with improved differential search and reinforcement learning,” *Mathematics*, vol. 8, no. 12, 2020.

[5] A. Karambakhsh, M. Y. A. Khanian, and M. R. Meybodi, “Robot navigation algorithm to wall following using fuzzy Kalman filter,” in *Proc. 9th IEEE Int. Conf. Control Autom. (ICCA)*, Santiago, Chile, 2011.

[6] M. R. M. Romlay, M. I. Azhar, S. F. Toha, and M. M. Rashid, “Two-wheel balancing robot: Review on control methods and experiments,” *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 7, no. 6s, pp. 106–112, 2019.

[7] W. Fadlun and A. Sugiharto, “Pemodelan dinamika dan perancangan kendali logika fuzzy untuk robot self-balancing roda dua,” *J. Inform. dan Tek. Elektro Terapan (JITET)*, vol. 13, no. 2, pp. 1088–1097, 2025.

[8] Á. Ordy, R. Fullér, I. J. Rudas, and P. Odry, “Fuzzy control of self-balancing robots: A control laboratory project,” *Comput. Appl. Eng. Educ.*, vol. 28, no. 1, pp. 1–24, 2020.