PID vs. Fuzzy Logic Control for Robotic DC Drives

Submitted by:

Hamza Saleh Saad   120210223

Course:	MTE 435: Electric Drives
Instructor:	Dr. Tamer Megahed - Eng. Rowan Esam
Date:	Spring 2025

Abstract

This project focuses on controlling the speed of a DC motor to a target of 200 RPM using two control strategies: a Proportional-Integral-Derivative (PID) controller and a Fuzzy Logic Controller (FLC), implemented on an Arduino platform with encoder feedback. The PID controller employs adaptive gain scheduling to dynamically adjust PWM signals, while the FLC uses fuzzy sets and rules, enhanced by an integral term, to handle non-linearities. This report details the algorithms, tuning methods, equations, and implementation of both controllers, comparing their performance based on experimental data. Placeholders are provided for a Tinkercad circuit diagram and MATLAB plots to visualize system behavior.

1 Introduction

The objective of this project is to control the speed of a DC motor to maintain a target RPM of 200 using two distinct control strategies: a PID controller and a Fuzzy Logic Controller (FLC). Both controllers were implemented on an Arduino platform, utilizing an encoder to measure motor speed and adjust the PWM signal to the motor driver.

2 PID Controller

2.1 Algorithm

The continuous-time PID equation is:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

In discrete form:

$$u(n) = K_p e(n) + K_i \sum_{k=0}^{n} e(k)\Delta t + K_d \frac{e(n) - e(n-1)}{\Delta t}$$

2.2 Ziegler-Nichols Tuning Method

The Ziegler-Nichols table:

Table 1: Ziegler-Nichols Tuning Parameters

Controller	K_p	T_i	T_d
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

2.3 Adaptive PID

Gain adjustments:

- If errorRatio > 0.5:

$$K_p = \text{MIN\_KP} + (\text{MAX\_KP} - \text{MIN\_KP}) \times 0.8$$

$$K_i = \text{MIN\_KI}, \quad K_d = \text{MIN\_KD}$$

- If $0.2 < \text{errorRatio} \leq 0.5$:

$$K_p = \text{MIN\_KP} + (\text{MAX\_KP} - \text{MIN\_KP}) \times 0.6$$
$$K_i = \text{MIN\_KI} + (\text{MAX\_KI} - \text{MIN\_KI}) \times 0.5$$
$$K_d = \text{MIN\_KD} + (\text{MAX\_KD} - \text{MIN\_KD}) \times 0.7$$

- If $\text{errorRatio} \leq 0.2$:

$$K_p = \text{MIN\_KP}, \quad K_i = \text{MAX\_KI}, \quad K_d = \text{MAX\_KD}$$

2.4 Implementation

- Anti-windup: limit integral term to $\pm\frac{100}{K_i}$
- Derivative filtering:

$$\text{filteredDerivative} = 0.3 \times \text{rawDerivative} + 0.7 \times \text{previous}$$

- 5-sample moving average for RPM
- PWM constrained to [70, 220]

3 Fuzzy Logic Controller

3.1 Concept

Fuzzy logic controllers use rules and membership functions to process non-linear or uncertain inputs.

3.2 Algorithm

- Inputs: $e = \text{targetRPM} - \text{measuredRPM}$, $\Delta e = e - \text{lastError}$
- Fuzzification, Rule Evaluation, Defuzzification

3.3 Equations

- Membership Function:

$$\mu(x) = \begin{cases} 0 & x \leq a \text{ or } x \geq c \\ \frac{x-a}{b-a} & a < x < b \\ \frac{c-x}{c-b} & b \leq x < c \end{cases}$$

- Rule Strength:

$$\text{ruleStrength} = \min(\mu_{\text{error}}(X), \mu_{\Delta\text{Error}}(Y))$$

- Defuzzification:

$$\Delta\text{PWM} = \frac{\sum(\text{ruleStrength} \times \text{outputValue})}{\sum \text{ruleStrength}}$$

- Final Output:

$$\Delta\text{PWM} = \Delta\text{PWM} + K_i \times \text{integral}$$

$$\text{output} = \begin{cases} \text{bestPWM} + \Delta\text{PWM} & \text{if measuredRPM} < 240 \\ 125 + \Delta\text{PWM} & \text{otherwise} \end{cases}$$

3.4 Implementation

- 5x5 rule base with triangular membership
- Integral term active within ±50 RPM, constrained to ±30
- Feedforward term (`bestPWM`) and noise filtering

4 Comparison

Table 2: Performance Comparison

Metric	PID Controller	Fuzzy Logic Controller
Accuracy	Moderate (80–112 RPM)	Better (136–168 RPM)
Stability	Oscillations (150–415 RPM)	Reduced (100–384 RPM)
Response Time	Fast start, slow settle	Slower start, better settle
Complexity	Simple	More complex
Non-linearity	Poor handling	Handles well

Table 3: Advantages

Controller	Advantages
PID	• Simple and easy to tune • Well-established method • Good for linear systems
Fuzzy Logic	• Handles non-linearity • No model required • Based on expert rules

5 Experimental Results

5.1 Circuit Diagram

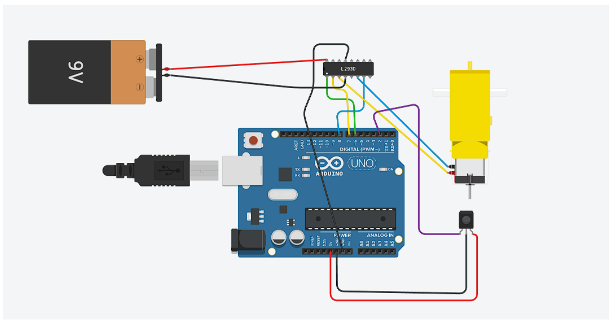
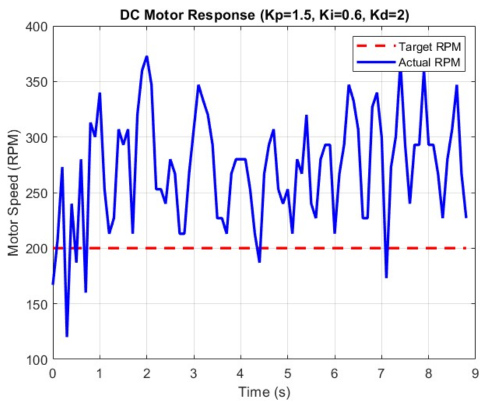
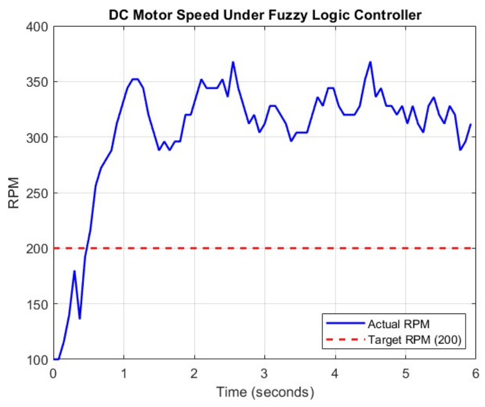


Figure 1: Circuit diagram in Tinkercad

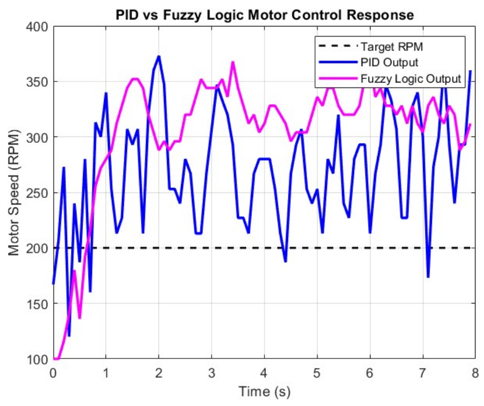
5.2 Plots



(a) PID Controller: RPM vs. Time



(b) Fuzzy Logic Controller: RPM vs. Time



(c) Difference between PID and Fuzzy Logic Controllers

Figure 2: Controller Performance Comparison

6 Conclusion

Both PID and Fuzzy Logic controllers were implemented to regulate DC motor speed at 200 RPM. PID control provided fast reaction but poor handling of non-linearities. The FLC handled non-linearity better and achieved more stable results but required careful tuning. A hybrid strategy could further enhance system performance.

6.1 Problems Faced and Solutions

During the implementation and testing of the control systems, several challenges were encountered:

- **Encoder Noise**: The encoder readings were susceptible to noise, leading to inaccurate RPM calculations. This was mitigated by implementing a 5-sample moving average filter, which smoothed out the RPM data and reduced the impact of transient errors.

- **Overshoot in Fuzzy Logic Controller**: Initially, the FLC caused significant overshoot, with the motor speed exceeding the target RPM before settling. This was addressed by adjusting the fuzzy rules to apply stronger negative PWM adjustments when the error was negative (RPM too high) and by fine-tuning the membership functions for more precise control.

- **Tuning Difficulties**: Both controllers required careful tuning. For the PID controller, the Ziegler-Nichols method provided a starting point, but manual adjustments were necessary to achieve better performance. The FLC required iterative testing to optimize the rule base and membership functions.

- **Hardware Limitations**: The Arduino's processing capabilities and the motor driver's response time posed constraints on the control precision. Code optimization and proper configuration of the motor driver helped alleviate some of these issues.

7 Future Improvements

To enhance the accuracy and robustness of the DC motor speed control system, several avenues for future work are proposed:

- **Advanced Hardware**: Utilizing a more powerful microcontroller or a dedicated motor control board could enable faster processing and support more sophisticated control algorithms.

- **Improved Filtering Techniques**: Implementing advanced filters, such as a Kalman filter, could further reduce noise in the RPM measurements, leading to more stable control.

- **Hybrid Control Strategies**: Combining the strengths of PID and Fuzzy Logic in a hybrid controller could provide better handling of both linear and non-linear system dynamics.

- **Auto-Tuning Mechanisms**: Developing an auto-tuning feature would allow the controllers to adapt to different motors or changing operational conditions without manual intervention.

- **Load Compensation**: Incorporating sensors to detect load variations and adjusting the control strategy accordingly could improve performance under varying conditions.

- **Higher Resolution Encoders**: Using encoders with more ticks per revolution would provide finer speed measurements, enhancing control accuracy.

- **Code Optimization**: Further refining the Arduino code to minimize loop times and maximize responsiveness could lead to better real-time performance.

These improvements could significantly enhance the system's ability to maintain the target RPM with higher precision and stability.