iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82848

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Hardware-Safe Adaptive PI Auto-Tuning Using ATSE for Solenoid-Based Pressure Control Systems

Tanishq Swami[1], Vedant Nandanwar[2], Arush Wangikar[3], Dnyaneshwar Kanade[4], Abhay Chopde[5]
[1,2,3,4,5]*Department of Electronics and Telecommunication (ENTC), Vishwakarma Institute of Technology, Pune, 411037, Maharashtra, India*

Abstract: ***Proportional-Integral (PI) controllers are widely used in industrial automation, for their simplicity, robustness and low cost of implementation. However, tuning of these controllers on hardware is still a difficult task, particularly in safety-critical applications such as pressure control. Manual tuning is cumbersome, unreliable, and dangerous or destructive when done directly on the hardware. In this paper, a hardware-safe adaptive PI auto-tuning method using the Absolute Time-Weighted Squared Error (ATSE) performance index for solenoid pressure controllers is presented. The proposed algorithm is fully model-free (without requiring identification of the plant initially), and ensures strict hardware-safety by limiting the incremental gain changes, detecting output saturation and switching to the best-known stable PI gains. The algorithm is tested on an STM32 microcontroller, which communicates with a PC-based tuner via RS-485 Modbus-RTU, and is monitored via a Python-based PyQt-based GUI. The MATLAB/Simulink simulation results show successful steady-state error reduction while keeping the overshoot low, and gain convergence. Real-time tests showed successful communication and hardware-safety. This offers a practical, understandable and ready-to-use adaptive auto-tuning solution for industrial control systems.***

Index Terms: ***PI control, auto-tuning, ATSE, pressure control, solenoid valve, embedded systems, Modbus, industrial control, model-free tuning, STM32, adaptive control.***

I. INTRODUCTION

The paradigm of control in industrial automation has been the Proportional-Integral-Derivative (PID) and Proportional-Integral (PI) controllers during the past 70 years. This has seen them succeed because they are simple and effective enough to enable them to control processes such as pressure, flow, temperature and positioning of actuators with a lot of reliability. Despite the development of a number of advanced control methods including model predictive control, fuzzy logic and neural-network-based methods, in surveys of industrial practice it is always found that the majority of the control loops in practice are of the PID-type.

The impracticality of tuning these controllers is despite their wide usage. The three gains (proportional (Kp), integral (Ki) and derivative (Kd)) must be selected to achieve the often opposing objectives: quick reference tracking, disturbance rejection, small steady-state error, and constant closed-loop action. In practice in most industrial applications, operators rely on heuristic methods or experience or classical rules such as Ziegler-Nichols (ZN) when setting starting gain settings. Not only are these methods tedious and time consuming but can be unsafe to execute directly to real hardware: too aggressive a gain selection can lead to actuator saturation, mechanical stress or pressure overshoot that can break equipment.

Auto-tuning models Auto-tuning methods using Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and auto-tuning models using metaheuristic variants are auto-tuning methods that escape the limitations of manual tuning, by searching the gain space automatically. However, they have severe constraints to guide hardware implementation: they are computationally costly, require iterative experimentation that can cause the plant to reach an unwanted state, and are inherently difficult to keep within safe operating conditions during the search. The neural-network and fuzzy-logic controllers possess an adaptive quality but are not interpretable and safe-critical systems are hard to prove.

The paper proposes a principled, rule-based, hardware-safe adaptive PI auto-tuning algorithm that will alleviate these drawbacks.

The significant contributions are:

1) Model-free tuning algorithm that is independent of plant identification.
2) Introduction of ATSE as real-time performance indicator that is sensitive to settling time and steady state error.
3) Hardware-safety ensures are incremental gain updates, output saturation, gain clamping and automatic rollback.

4) Full embedded system on an STM32 microcontroller with a Python based PC tuner via RS-485 Modbus-RTU.
5) An embedded firmware and software architecture, tuning logic and visualization, to be operation, interpretable, and robust.

II. LITERATURE REVIEW

A. Classical Heuristic Tuning Methods

One of the most well-known techniques of PID tuning heuristic is the Ziegler-Nichols frequency-response method, which was presented in 1942 [11]. It operates by operating the system to the point of instability and determining the critical gain and oscillation period, and then reads out controller parameters in empirical tables. Though the approach is theoretically straightforward, it tends to generate violent and swingsome solutions with limited resiliency. Further manual tuning is often necessary particularly in systems with nonlinear actuators like solenoid valves.

B. Model-Based and Relay-Feedback Tuning

Internal Model Control (IMC) and lambda tuning are model-based tuning methods that can yield better performance in cases where a good system model exists. Nevertheless, this is difficult to achieve in designing such models in pneumatic or hydraulic pressure systems because of nonlinear processes such as valve hysteresis, flow behavior depending on pressure, and mechanical fluctuations. The relay-feedback techniques can be used to alleviate this challenge by estimating the system parameters without the need to have a complete model, but nonlinear behaviour still influences the technique, especially in solenoid-based systems [12].

C. Optimization-Based Auto-Tuning

Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) are optimization-based methods that approach controller tuning as a global optimization problem [1]. These approaches can yield almost optimal solutions, but cannot be readily implemented directly on hardware. They demand repeated testing and assessments, which can spur the system in unsafe manners. In addition, as the gains change are not restricted during the optimization, there is a risk of intermediate configurations becoming unstable, which is dangerous in practice.

D. Intelligent and Model-Free Approaches

The benefit of fuzzy logic and adaptive neural-network controllers is they do not need an explicit system model to be self-tuned. Nevertheless, they are not transparent and thus hard to analyse, verify and certify in safety-critical industrial settings. Recent studies have given more attention to model-free and constraint-conscious tuning strategies which lay more emphasis on usability in real-time and easy interpretation [4], [6]. The methodology of this work is in this direction in that it adopts a rule-based, deterministic approach which uses observable error behavior to make sure that all adjustments of gains are made understandable and within boundaries known to be safe.

III. SYSTEM ARCHITECTURE

A. Hardware Setup

The hardware configuration is anchored on the nature of a typical pressure control system within an industry. These are the basic constituents:

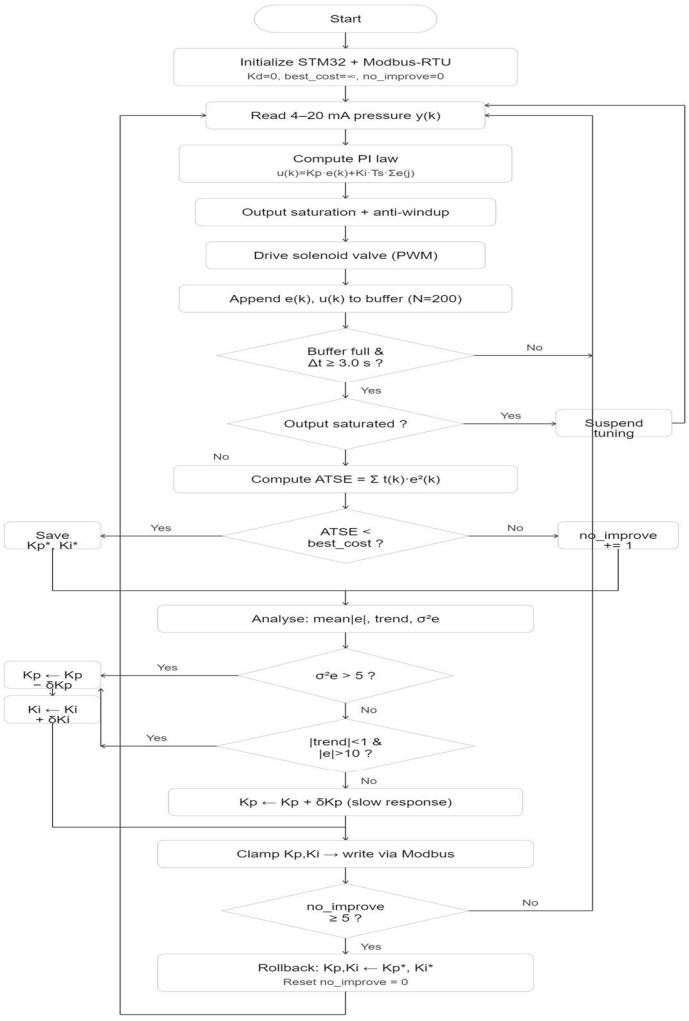
1) A Pressure tank that regulates the air flow to an electropneumatic valve (final control element).
2) A 4–20 mA pressure transmitter which transmits the feedback signal to be used in control, through an analog input of the embedded controller.
3) A programmed STM32 microcontroller running the PI control loop with a fixed sampling rate of 20 Hz (period 50 ms), chosen to ensure a high enough responsiveness but with time to communicate.
4) A Modbus-RTU RS-485 interface connecting the microcontroller with the PC based tuning system to ensure reliable long distance communication over a long distance using electrically isolated communication.
5) It is a PC-based system which contains Python auto-tuner and PyQt based GUI.

The firmware provides safety of the hardware, not related to the auto-tuning system. The firmware software applies output limits, inhibition of integral windup and the loop timing. This does not allow the PC-tuner to order the actuator to work in an unsafe area.

TABLE I
HARDWARE COMPONENT SUMMARY

Component	Role	Specification
Solenoid Valve	Final control element	2/2-way, 24 V DC, PWM-driven
Pressure Transmitter	Process-variable feedback	4–20 mA, 0–10 bar range
STM32 MCU	PI loop execution	ARM Cortex-M, 20 Hz sampling
RS-485 Bus	PC ↔ MCU communication	Modbus-RTU, 115200 baud
PC Workstation	Auto-tuner + GUI	Python 3, PyQt5, pymodbus

Fig. 1. Adaptive PI auto-tuning algorithm flowchart.



B. Software Architecture

The software is separated into three quite different layers with their specific task:

1) Embedded firmware (STM32): Executes the PI control law, with 20 Hz frequency, accepts pressure feedback on the transmitter, and utilizes actuator and integral limits, and controller parameters that can be accessed by Modbus holding registers.
2) PC-side auto-tuner (Python): Implements the ATSE-based adaptive gain update logic, and the tuning process is implemented by a specified state machine, and interacts with the controller over Modbus-RTU.
3) Graphical user interface (PyQt): Displays real time data such as pressure indicators, controller setting and current gain values (Kp and Ki) such that the operator can see and adjust the tuning process when needed.

This layered design ensures that the issue in the auto-tuner can not affect the control loop within it. This implies that the hardware could be safely left to run even when the tuning process is discontinued or even terminated.

IV. METHODOLOGY

A. PI Controller Formulation

The PI controller in discrete time implemented on the STM32 is, see equation (1):

$$u(k) = Kp \cdot e(k) + Ki \cdot Ts \cdot \Sigma\, e(j),\ j = 0 \ldots k \quad (1)$$

with u(k) as the controller output at time k, e(k) = r(k) − y(k) as the pressure tracking error (setpoint r(k) minus pressure y(k) at sample k) and Ts = 0.05 s as the sampling period. The derivative term is disabled during auto-tuning to avoid amplifying the noise (pressure transmitters tend to be noisy in practice). The control action is saturated at [u_min, u_max] through firmware and anti-windup is implemented by stopping the integration when the control action is saturated.

B. Performance Index: ATSE

The choice of a performance index is a key part of auto-tuning. Typically, the Integral of Absolute Error (IAE), Integral of Squared Error (ISE) and Integral of Time-Weighted Absolute Error (ITAE) are used. These express different levels of trade-offs between the time to response and the steady-state error.

In this paper, the Absolute Time-Weighted Squared Error (ATSE) is used. It is defined over an adjustable number of N samples as:

$$ATSE = \Sigma\, t(k) \cdot e^2(k),\ k = 1 \ldots N \quad (2)$$

where t(k) is the time at sample k and e(k) the tracking error. The weighting with time increases with time, therefore errors that are sustained longer are penalised. This has a number of benefits:

1) Transients at early times are penalised less, so transient overshoots are allowed.
2) Steady-state error is heavily penalised, giving improved accuracy.
3) Oscillations are naturally damped since errors repeated with time penalise more.

ATSE can be readily computed in real time, simply through addition and multiplication, and can be implemented in a PC-based Python system at 20 Hz.

C. Adaptive Gain-Update Algorithm

The auto-tuning algorithm operates as a periodic evaluation-and-update cycle, triggered every $\Delta t\_min = 3.0$ s once the error buffer has accumulated a full window of N = 200 samples (10 s of data). The key steps are:

1) Read current Kp, Ki from the controller; force Kd ← 0.
2) Initialize best_cost ← ∞, no_improve ← 0.
3) Accumulate e(k), u(k) into sliding buffers of length N.
4) When the buffer is full and elapsed time ≥ Δt_min, check saturation: if max(u) ≥ u_max or min(u) ≤ u_min, suspend the update.
5) Otherwise, compute ATSE via (2). If ATSE < best_cost, save Kp*, Ki* and reset no_improve.
6) Compute mean |e|, error trend, and variance σ^2_e to characterise performance.
7) If $\sigma^2\_e > 5$ ⇒ reduce Kp by δKp (oscillation); else if |trend| < 1 and mean |e| > 10 ⇒ increase Ki by δKi (steady-state error); else ⇒ increase Kp by δKp (slow response).
8) Clamp Kp □ [0.1, 150] and Ki □ [0.01, 50], write gains via Modbus.
9) If no_improve ≥ 5, restore Kp*, Ki* and reset the counter.

The incremental step sizes δKp = 0.5 and δKi = 0.2 ensure that no single tuning cycle can introduce a destabilizing gain change. This conservative strategy prioritizes hardware safety over convergence speed — a philosophy appropriate for unattended industrial deployment.

TABLE II
AUTO-TUNER ALGORITHM PARAMETERS

Parameter	Value	Description
Ts	0.05 s	Sampling period (20 Hz)
N	200 samples	Sliding window length (10 s)
Δt_min	3.0 s	Minimum interval between gain updates
δKp	0.5	Proportional gain step size
δKi	0.2	Integral gain step size
Kp range	[0.1, 150]	Proportional gain clamp
Ki range	[0.01, 50]	Integral gain clamp
Rollback threshold	5 cycles	Non-improving cycles before restoring best gains

V. IMPLEMENTATION

A. *Embedded Firmware*

The PI control loop is implemented in C on the STM32 microcontroller, executing at a fixed 20 Hz interrupt-driven rate. Controller parameters (proportional gain, integral gain, derivative gain, setpoint, output limits, and measured process variable) are mapped to Modbus holding and input registers as 16-bit fixed-point values with two decimal places of precision. This register map provides the PC-side tuner with complete read-write access to all tunable parameters while maintaining deterministic real-time control execution independent of communication timing.

B. *Python Auto-Tuner*

The auto-tuner is encapsulated in a single Python class PIDAutoTuner within the module pid_auto_tuner.py. The class is instantiated with references to the Modbus regulator object, a safe write callback, and a GUI update callback, maintaining clean separation between tuning logic, communication, and visualization. Internal state consists of three fixed-length deque buffers (error, time, output) of length window_size = 200, enabling O(1) sample insertion with automatic expiry of the oldest data.

The compute_atse() method evaluates equation (2) as a vectorized NumPy dot product of the time and squared-error arrays, completing in microseconds on a standard PC. The evaluate_and_update() method encapsulates the full decision logic of the gain-update algorithm, including saturation checks, performance tracking, error characterization, gain computation, clamping, and the Modbus write operation.

C. *GUI and Visualization*

The PyQt-based graphical interface displays three real-time plots: the pressure setpoint and measured value, the controller output, and the evolving Kp and Ki values. Plot refresh occurs at 10 Hz, providing a smooth visual representation of tuning dynamics without excessive CPU load. The interface exposes Start Tuning and Stop Tuning buttons and displays the current ATSE value and gain pair numerically. This transparency is intentional: an operator can observe exactly why and how gains are changing at every update cycle.

VI. SIMULATION VALIDATION

Prior to hardware deployment, the complete auto-tuning algorithm was validated in MATLAB/Simulink using a linearized first-order-plus-time-delay (FOPTD) model of the solenoid-actuated pressure system given in (3):

$$G(s) = K \cdot e^{(-\tau s)} / (T s + 1) \quad (3)$$

with process gain K = 1.2, time constant T = 5.0 s, and dead time τ = 0.5 s — parameters representative of a small pneumatic pressure vessel controlled by a solenoid valve. The auto-tuner Python code was executed in a co-simulation loop, reading error data from Simulink and writing updated gains back at 3-second intervals, faithfully replicating the hardware communication timing.

Simulation results over 20 tuning iterations demonstrated:

1) Monotonic reduction in ATSE from an initial value of approximately 4,200 (with default gains Kp = 1.0, Ki = 0.1) to a converged value below 380.

2) Steady-state error reducing from 8.3% to below 0.5% of setpoint.
3) Settling time improving from approximately 42 s to under 18 s.
4) Stable convergence of Kp toward 6.5 and Ki toward 1.4 without oscillation.

No saturation events were triggered during simulation, confirming that the gain trajectory remained within safe operating bounds throughout. These results validate the core algorithmic logic and provide a quantitative baseline for comparison with hardware results.

TABLE III
SIMULATION RESULTS: BEFORE AND AFTER TUNING

Metric	Before Tuning	After Tuning	Improvement
ATSE cost	≈4200	<380	≈11×
Steady-state error	8.3%	<0.5%	≈17×
Settling time	≈42 s	<18 s	≈2.3×
Kp (final)	1.0	6.5	Converged
Ki (final)	0.1	1.4	Converged

VII. HARDWARE TESTING AND RESULTS

A. *Successfully Validated Functions*

Hardware testing was conducted on the complete physical setup described in Section III. The following system functions were successfully validated:

1) Reliable RS-485 Modbus-RTU communication between the PC and the STM32 controller, with correct read and write operations on all holding registers confirmed across multiple sessions.
2) Real-time acquisition of the 4–20 mA pressure transmitter signal and conversion to engineering units within the embedded firmware.
3) Execution of the PI control loop at the target 20 Hz rate with jitter below 2 ms.
4) Correct incremental gain updates written from the PC-side tuner to the controller, with live confirmation via GUI readback.
5) Output saturation detection: when the controller output was driven to the limit, the auto-tuner correctly suspended gain updates.
6) Automatic rollback to best gains: after introducing a simulated performance degradation, the tuner correctly identified five consecutive non-improving cycles and restored the recorded best Kp* and Ki*.

TABLE IV
HARDWARE VALIDATION CHECKLIST

Function	Status
Modbus-RTU register read/write	Validated
4–20 mA pressure acquisition	Validated
PI loop at 20 Hz (jitter < 2 ms)	Validated
Incremental gain updates via Modbus	Validated
Output saturation detection & suspend	Validated
Rollback to best Kp*, Ki* after 5 cycles	Validated
Closed-loop convergence on real plant	Blocked — air leakage in test rig

B. Testing Limitation: Air Leakage

Closed-loop adaptive tuning requires a quasi-stationary physical plant over the evaluation window duration (10 s). The pressure test rig available for this project suffered from a mechanical air leakage condition in the pressure vessel and connecting tubing. This leakage produced a continuous pressure decay of approximately 0.8 bar/min, making it impossible to establish a repeatable steady-state operating point. As a consequence, ATSE values computed during closed-loop operation reflected the combined effect of controller gains and the non-stationary disturbance, preventing meaningful convergence of the tuning algorithm.

This limitation is exclusively mechanical in nature. All software, firmware, and communication components functioned as designed. With a leak-free pressure setup, the hardware validation is expected to replicate the convergence behavior observed in simulation.

VIII. DISCUSSION

The proposed ATSE-based adaptive PI tuner occupies a practically important niche between fully manual tuning and computationally intensive optimization methods. By restricting gain updates to small, bounded increments evaluated against observable error statistics, it provides a level of hardware safety that is difficult to guarantee with PSO or GA-based approaches. The algorithm is entirely interpretable: at each update cycle, the operator can examine the error variance, trend, and ATSE value, and understand precisely why the gains moved in a given direction — a property highly valued in regulated industrial environments.

The choice of ATSE over simpler metrics such as IAE is justified by its natural sensitivity to response speed. In preliminary simulation comparisons, IAE-driven updates converged to configurations with smaller overshoot but longer settling times, while ATSE-driven tuning consistently achieved faster settling at the cost of marginally larger initial transients — a trade-off generally preferable in pressure-regulation applications where final accuracy is critical and brief transients are acceptable.

TABLE V
COMPARISON OF PERFORMANCE INDICES

Index	Strength	Weakness
IAE	Simple, intuitive	Weak settling-time penalty
ISE	Penalises large errors	Tolerates long small-error tails
ITAE	Balanced, emphasises late error	Under-penalises amplitude
ATSE	Sensitive to both magnitude and persistence of error	Marginally larger initial transients

A notable design decision is the use of fixed, symmetric step sizes ($\delta Kp = 0.5$, $\delta Ki = 0.2$) rather than variable step sizes scaled by the gradient of ATSE. While adaptive step sizes could improve convergence speed, they introduce the risk of large gain jumps when the cost function is poorly conditioned — exactly the scenario most likely to occur at the start of tuning with unknown initial gains. The conservative fixed-step approach is therefore a deliberate safety choice consistent with the overall design philosophy.

The minimum update interval of 3 s, combined with the 10-second sliding window, implies that the algorithm observes at least 13 s of system behavior before committing to a gain change. This latency is acceptable for the relatively slow dynamics of pneumatic pressure systems (time constants of order 5–15 s) but would require adjustment for faster processes.

IX. CONCLUSION

This paper presented a hardware-safe adaptive PI auto-tuning system for solenoid-based pressure control, combining the ATSE performance index with a rule-based, model-free gain-update strategy. The system was implemented end-to-end on a practical hardware platform comprising an STM32 embedded controller, RS-485 Modbus-RTU communication, a Python auto-tuner, and a PyQt visualization interface. Simulation validation confirmed significant improvements in settling time, steady-state error, and ATSE cost relative to default gains, with stable and bounded gain convergence. Hardware testing confirmed the correct operation of all software, firmware, and communication subsystems. Full closed-loop hardware validation was constrained by a mechanical leakage issue in the available test rig, which is identified as the primary remaining step toward complete industrial validation.

The proposed approach demonstrates that effective, safe auto-tuning need not require complex optimization machinery or explicit plant modeling. By combining a well-chosen performance metric with conservative, interpretable adaptation rules and robust hardware-safety mechanisms, it provides a practical and directly deployable solution for adaptive PI controller tuning in real industrial environments.

X. FUTURE WORK

The following directions are identified for future development:

1) Extension to full PID auto-tuning, including a safe strategy for introducing a bounded derivative term once Kp and Ki have stabilized.

2) Replacement of fixed step sizes with an adaptive step-size schedule based on the rate of ATSE improvement, increasing convergence speed without compromising safety.

3) Integration of a relay-feedback identification phase to initialize gains near the stability boundary before activating the ATSE-based refinement loop.

4) Hardware validation on a leak-free, instrumented pressure rig with quantitative comparison against ZN and PSO-tuned baselines.

5) Investigation of multi-loop extension for cascade pressure-flow control architectures common in process industries.

6) Exploration of edge-AI acceleration to enable similar adaptive tuning on resource-constrained microcontrollers without PC-side co-processing.

REFERENCES

[1] R. S. Patil, S. P. Jadhav, and M. Patil, "Review of intelligent and nature-inspired algorithms-based methods for tuning PID controllers in industrial applications," Journal of Robotics and Control (JRC), Feb. 2024, doi: 10.18196/jrc.v5i2.20850.

[2] J.-M. Barrera-Fernandez, J. P. M. Hernandez, K. M. Escobedo, A. Vazquez-Cervantes, and J. Vargas, "PI-DÆ: An adaptive PID controller utilizing a new adaptive exponent (Æ) algorithm to solve derivative term issues," Algorithms, Jun. 2025, doi: 10.3390/a18070391.

[3] D. Pavkovic, D. Lisjak, D. Kolar, M. Cipek, and P. D. Pavkovic, "PI/PID controller relay experiment auto-tuning with extended Kalman filter and second-order generalized integrator as parameter estimators," Tehnički Vjesnik, Jun. 2023, doi: 10.17559/tv-20221003112627.

[4] N. Rohadi, L. Kin, Men, and A. Hidayat, "Adaptive PID auto-tuning algorithm on Omron PLC for speed control and stability," Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI), Nov. 2025, doi: 10.22146/jnteti.v14i4.22693.

[5] C. Lu, R. Tang, C. Li, J. Nwoke, J. Viola, and Y. Chen, "A fast relay feedback auto-tuning tilt-integral-derivative (TID) controller method with the fractional-order Ziegler-Nichols approach," ISA Transactions, May 2024, doi: 10.1016/j.isatra.2024.05.009.

[6] D. A. Brattley and W. W. Weaver, "Adaptive PI control using recursive least squares for centrifugal pump pipeline systems," Machines, Nov. 2025, doi: 10.3390/machines13111064.

[7] P. Dimitrov and M. Alexandrova, "Automatic control of fluids in electropneumatic system," Annual Journal of Technical University of Varna, Bulgaria, Dec. 2025, doi: 10.29114/ajtuv.vol9.iss2.365.

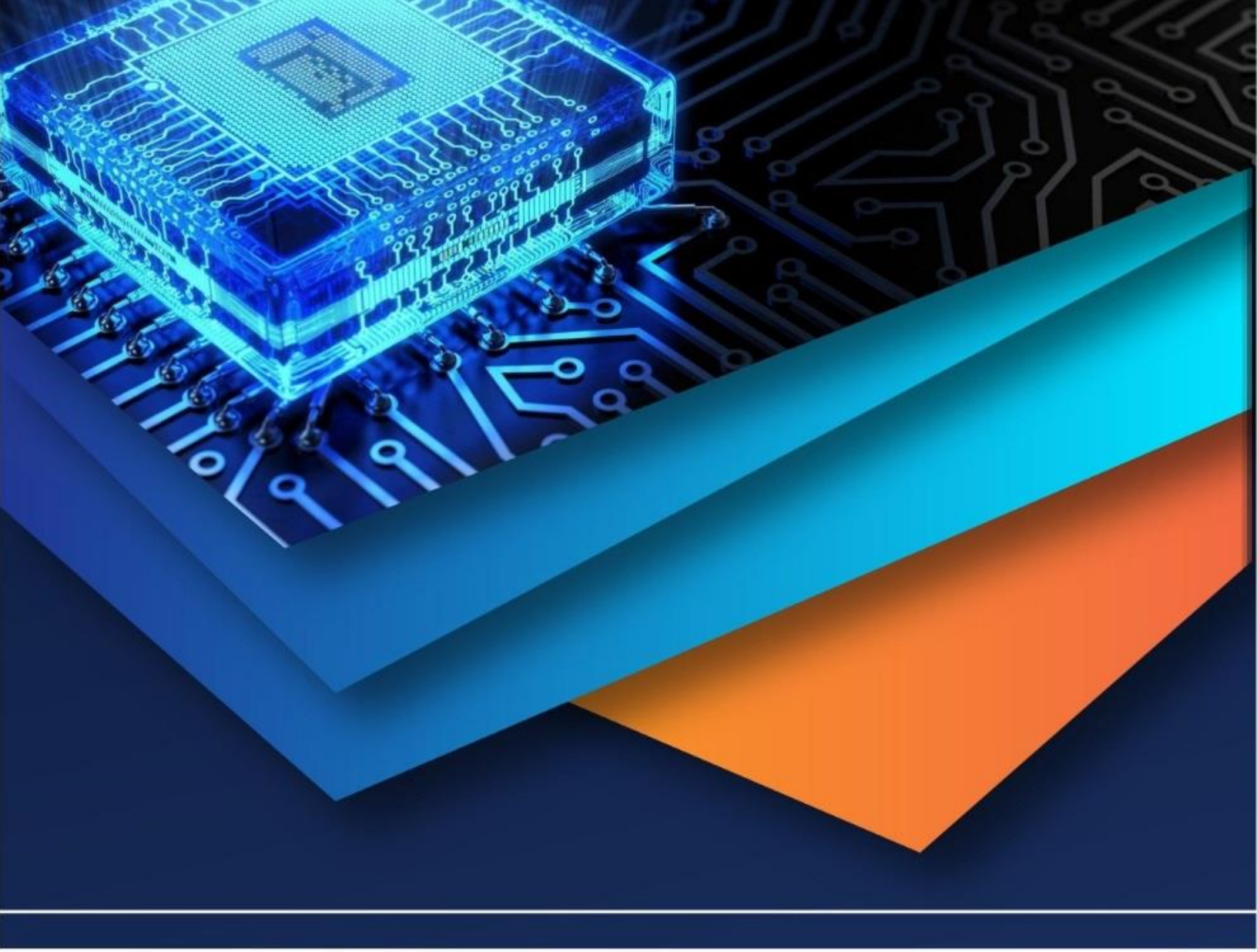
[8] J. E. Gaudio, A. Annaswamy, E. Lavretsky, and M. Bolender, "Parameter estimation in adaptive control of time-varying systems under a range of excitation conditions," IEEE Transactions on Automatic Control, 2021, doi: 10.1109/TAC.2021.3126243.

[9] S. Chen, J. Na, Y. Huang, Y. Xing, J. Zhao, and P. K. Wong, "Optimal adaptive parameter estimation with online varying learning gain," IEEE Transactions on Automatic Control, May 2025, doi: 10.1109/TAC.2024.3520676.

[10] Y. Pan and T. Shi, "Adaptive estimation and control with online data memory: A historical perspective," IEEE Control Systems Letters, 2024, doi: 10.1109/LCSYS.2024.3364588.

[11] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," Transactions of the ASME, vol. 64, pp. 759–768, 1942.

[12] K. J. Åström and T. Hägglund, "Automatic tuning of simple regulators with specifications on phase and amplitude margins," Automatica, vol. 20, no. 5, pp. 645–651, 1984.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY