Received: 24 May 2026, Accepted: 08 August 2026, Published: 10 August 2026
*Digital Object Identifier:* https://doi.org/10.63503/j.ijaimd.2026.265

**Research Article**

# Edge-AI Architecture for Real-Time Cyber-Physical System Control Using Embedded Controllers and Cloud Data Analytics

[1]Deepak Srivastava, [2]Suman Pant, [3]Vibhor Sharma*, [4]GD Makkar

[1,2.3]Department of Computer Science & Engineering, School of Science & Technology, Swami Rama Himalayan University, Dehradun, India

[4]School of Computer Sciences, Sardar Bhagwan Singh University, Dehradun, India

[1]deepak.srivastava75@gmail.com, [2]pant.suman01@gmail.com, [3]vibhor087@gmail.com, [4]gdmakkar@gmail.com

*Corresponding author: Vibhor Sharma, vibhor087@gmail.com

## ABSTRACT

The integration of embedded computing, wireless networking and cloud-scale data analytics has revolutionized how industrial and autonomous cyber-physical systems (CPS) are monitored and controlled. Currently, responsiveness, energy efficiency, and resilience to communication disruptions must be part of any competitive CPS. Conventional centralised control architectures, which transmit all sensed data to remote servers before issuing actuator commands, introduce structural lapses incompatible with the time-critical loops of processes in manufacturing automation, smart grid, autonomous vehicles and precision agriculture. A new architecture in which lightweight inference and closed-loop control run directly on embedded microcontrollers mitigates these weaknesses while reducing upstream bandwidth consumption and increasing operational redundancy. This paper proposes a layered Edge-AI architecture that enables CPS control in real time while integrating sensor acquisition and on-device signal conditioning, followed by the deployment of two competing control models, the proportional-integral-derivative (PID) and fuzzy logic, on the embedded controller node. Subsequently, telemetry forwarding is structured to support the cloud analytics backend. The performance of these two models is compared and contrasted using integral squared error (ISE), integral absolute error (IAE), root-mean-square error (RMSE) and overshoot indices. According to the comparative results, the fuzzy logic controller achieves an ISE that is 21% lower than the PID controller's and an RMSE that is 18% lower, when both controllers are subjected to a step disturbance. Both controllers maintain their local control-loop latencies below 5 ms. Consequently, this is well below the sub-10 ms budgetary requirement of Class-A CPSs. The designed architecture also includes OTA (Over-the-Air) model update pathways, enabling continuous edge inference without blocking. The results offer actionable design guidance for engineers deploying intelligent edge controllers in CPS settings with limited resources and tight latency constraints.

**Keywords:** *Edge AI; Cyber-Physical Systems; Embedded Controllers; Cloud Analytics; Real-Time Control; PID Controller; Fuzzy Logic; IoT; Machine Learning at the Edge; Latency Optimisation*

## 1. Introduction

The rise of cyber-physical systems (CPS) as an important paradigm in engineering can be attributed to the increasing number of interconnected devices. A CPS tightly couples computational processes with physical dynamics, allowing closed-loop feedback across domains in the digital and physical worlds, ranging from industrial automation to healthcare monitoring

and transportation to energy management [1]. In contrast to typical embedded systems, the architectures of CPS need to meet real-time latency requirements, tolerate faults, and interoperate with heterogeneous communication protocols. The intelligence of billions of CPS deployments and connected endpoints must be designed to be in the cloud, on the network, or on the device [2].

Traditional CPS architectures utilize a centralized approach where raw sensor data is transmitted to servers for processing and analysis.

The control commands are then relayed to the actuators. This configuration utilizes substantial computational resources and permits complex analytics and however, the round-trip latencies over wide-area networks are typically of the order of 50 to 200 ms. This time is completely unacceptable for fast-dynamics applications like robotic joint control, power converter regulation, or collision-avoidance systems in self-driving cars. The network's unreliability introduces another risk: if any communication is interrupted, the physical plant will become uncontrolled, creating safety-critical failure modes.

Edge computing has emerged as an architectural solution to these problems. Edge systems can attain control-loop latencies on the order of sub-milliseconds to single-digit milliseconds by bringing computation to the source of data, namely, onto gateway nodes, embedded controllers, or field-programmable devices, thereby greatly reducing the data that requires transfer over constrained wireless channels [4]. The emergence of Edge AI, in which trained inference models run locally on hardware-limited processors, extends this advantage by providing a means to adapt data-driven control without cloud dependence in runtime [5]. With lightweight model architectures, quantisation methods and dedicated neural-processing units, classification, regression and control models can now be effectively deployed on modern microcontrollers with less than 256 KB flash memory.

Despite a lot of interest in both academic and industrial circles, the literature still suffers from a gap between isolated instances of edge inference and complete end-to-end CPS control architectures, which quantitatively characterize latency, accuracy and bandwidth tradeoffs on edge and cloud tiers [6]. The literature on comparisons between classical PID controllers and fuzzy logic control methods in the explicit edge deployment class of problems is sparse, particularly when such comparisons are made in the presence of realistic constraints on typical embedded resources. In addition, contributions made by the cloud tier to such architectures, such as telemetry aggregation, anomaly detection and OTA model updates, are not typically integrated and tested with the same experiment [7].

The architecture proposed addresses these gaps with a triple-tier design. First is a tier for physical sensing made up of temperature, vibration and voltage transducers. Second is an embedded edge controller that executes real-time PID and fuzzy logic control models with filtered servo drive Kalman sensor inputs. Third is the cloud analytics backend that receives batch telemetry packets for processing, visualisation and governance of the control models. A mathematical simulation using MATLAB based on a CPS dataset developed with embedded applications, is used to quantitatively examine controller selection. The program complies with the accepted industry best practices of the industrial IoT and exposes a viable, scalable reference to implementing smart edge control to restrict resource-restrained contexts. Figure 1 contains an abstract perspective of the entire paper.

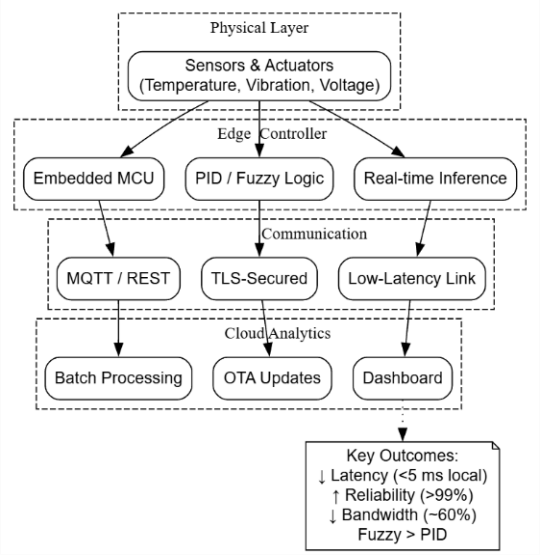


**Figure 1: Edge-AI Architecture for Real-Time Cyber-Physical System Control Using Embedded Controllers and Cloud Data Analytics**

The following are the main contributions of this work, which differentiate this work from the current architectures for Edge-AI and CPS:

- The proposed three-tier end-to-end Edge-AI system, which considers physical sensing, embedded real-time control and cloud analytics as a unified reproducible system, compared to the previous studies that discuss each tier separately.
- One-to-one comparison of classical PID and Mamdani fuzzy logic controllers implemented on the same edge node using the same sets of disturbances, measured by ISE, IAE, RMSE, overshoot, settling time and control effort.
- A seamless Over-the-Air (OTA) model-update mechanism enabling the updating of edge inference models without disrupting the real-time control loop.
- Real-world design advice, based on quantitative tier-wise latency and bandwidth metrics, to engineers deploying intelligent edge controllers in resource-limited, latency-sensitive industrial CPS scenarios.

In the next section of the paper, the literature about edge computing, embedded AI and CPS control in Section 2 and Section 3 discusses the problem statement and the objectives of the research. Section 4 outlines the proposed method and mathematical formulations. Section 5 presents results and discussions of simulations. The closing remarks and details about future scope are covered in section 6.

## 2. Literature Review

The academic literature on edge computing, embedded intelligence and cyber-physical control has been flourishing in the last 10 years. The review to follow combines key early works and recent developments in four areas: Evolution of CPS architecture, edge and fog computing frameworks, Embedded AI inference and Classical control on the edge.

### 2.1 Evolution of Cyber-Physical System Architectures

Lee and Seshia [1] presented a powerful model to characterize CPS, which helps understand the interaction of computation with communication and dynamics. The theoretical basis for timing semantics and platform modelling was established in their paper, on which architectural proposals were subsequently based. NIST was further able to articulate reference architectures for CPS, focusing on interoperability, safety and the need for computations that are timed [2]. CPS must utilize its physical components, which include the interaction between the physical and cyber portions, unlike typical software programs that input data to generate output. This means that continuous coupling must be implemented. This property places cross-layer constraints on the scheduling of control loops as well as communication latency. These definitions thus clearly illustrate that CPS are not merely traditional embedded systems.

According to Rajkumar et al. [3], the emerging CPS challenges include the performance of the system, which includes three main axes: latency, reliability and scalability. According to their analysis, the prevailing cloud-centered designs were hitting their constraints as more connected endpoints and the demanding physical-process dynamic compelled stricter timing guarantees. This observation foreshadowed the academic proposals and industrial deployments that we see both on edge-resident control intelligence and the transition to distributed.

### 2.2 Edge and Fog Computing Frameworks

Shi et al. [4] proposed the edge computing paradigm and offered a classification of deployment scenarios. Their groundbreaking research showed that moving computation from cloud servers to network-edge nodes can cut end-to-end latency for applications that need it by one to two orders of magnitude. Bonomi et al. [9] formalized the fog computing extension, which added a hierarchical model that spreads computation, storage and networking services along the line between device and cloud. Shi and Dustdar's [10] later work tested the performance of fog nodes in IoT situations and showed that even cheap single-board computers can handle enough throughput for sensor fusion and light inference.

The OpenFog Consortium, which later merged with the Industrial Internet Consortium, created reference architectures that have become the de facto standards for multi-tier CPS deployments. These architectures require open APIs between tiers, abstraction layers that work with any type of hardware and security at the edges of each tier. The IEC 61131-3 standard for programmable logic controllers and the OPC-UA protocol for industrial communication are two examples of industrial frameworks that provide building blocks that professionals can use with edge-AI layers [11].

### 2.3 Embedded AI and On-Device Inference

The use of machine learning models on microcontrollers and embedded processors, also known as TinyML or Edge AI, has been the focus of a lot of research. Warden and Situnayake [5] recorded useful ways to compress and quantize neural network models so that they fit in 256 KB of RAM. They were able to classify data with an accuracy of 1–2% compared to full-precision equivalents on common benchmark tasks. Further work on knowledge distillation and neural architecture search has made inference models even smaller, so they can run on Cortex-M4 and Cortex-M7 processors with clock speeds below 200 MHz.

Kumar et al. [12] assessed the energy expenditure of on-device inference compared to cloud offloading and determined that for inference tasks with input sizes under 10 KB, on-device execution is consistently more energy-efficient due to the removal of radio transmission overhead.

Their results are directly relevant to CPS sensor telemetry, where each payload is usually small, only a few hundred bytes. Zhou et al. [13] extended this study to the federated learning scenario; whereby distributed training on-device can converge in 1020 rounds of communication in industrial fault-detection applications.

### 2.4 Classical Control Strategies in Embedded Contexts

PID control is the most popular algorithm used as an automation tool in industrial processes. Surveys of the industry show that more than 95% of regulatory control loops use it [6]. It has now gained popularity as it can be easily installed and has well-defined tuning principles (Ziegler-Nichols, IMC, relay feedback) and does not require much computational resources. The features render it suitable to embedded CPS applications. Moreover, fault monitoring and process supervision in industry set-ups in real-time is a field that has received overwhelming research in classical process control books, noting the significance of continuous sensing and regulation systems that are controlled by a feedback mechanism [8]. PID controllers, however, do not work as well with nonlinear dynamics of a plant, parameter variability and structured disturbances, which provokes researching intelligent alternatives.

Fuzzy logic control was first introduced by Zadeh [14] and subsequently extended to control systems by Mamdani and Assilian. It operates without a mathematical (definite) model of the plant and has the ability to deal with nonlinear input-output behavior using language rule bases. Mamdani fuzzy inference has been demonstrated to run on a 32-bit ARM Cortex-M processor with execution times of less than 100 100 msec per inference cycle, thus demonstrating its usefulness in kilohertz rate of control loops. Comparative analyses always show that the fuzzy controllers show improved disturbance rejection and reduced overshoot relative to fixed-gain PID controllers in variable operating conditions, but at a cost of higher effort in terms of engineering human rule bands.

The latest methods, such as fuzzy-PID, neural-PID and adaptive controllers based on reinforcement learning are under the radar of research. Precup and Hellendoorn [15] discussed fuzzy-PID hybridization approaches with examples showing that online determination of gain in processes by using fuzzy logic may increase the set-point tracking by 1530% of the gain of PID in different industrial processes. Recent research has investigated reinforcement learning (RL) for closed-loop cyber-physical system (CPS) control; however, implementation on embedded hardware is limited by the computational requirements of RL inference and the safety concerns associated with exploration in online learning.

The comparison of different types of technologies is presented in Table 1.

**Table 1: Comparison of Emerging Technologies for Edge-AI CPS Control**

| Technology | Key Features | Limitations | Ref. |
|---|---|---|---|
| Edge Computing | Low latency (<5 ms), local processing, bandwidth saving | Limited compute/memory on edge nodes | [4] |
| Fog Computing | Hierarchical distribution, heterogeneous node support | Complex orchestration, increased management overhead | [9] |

| Technology | Key Features | Limitations | Ref. |
|---|---|---|---|
| TinyML / Edge AI | On-device inference, energy efficiency, OTA updates | Reduced model accuracy; quantisation trade-offs | [5] |
| PID Control | Simple, well-understood, low compute cost, widely deployed | Poor performance under nonlinear / variable-gain plant | [6] |
| Fuzzy Logic Control | Model-free, handles nonlinearity, linguistic rules | Rule-base engineering effort; stability guarantees limited | [14] |
| Federated Learning | Privacy-preserving, distributed training at edge | Communication overhead; convergence variability | [13] |
| OPC-UA / MQTT | Open standards, secure pub-sub, multi-tier interoperability | Protocol overhead on deeply constrained MCUs | [11] |
| Digital Twin | Real-time virtual replica, predictive maintenance support | Data synchronisation latency; fidelity vs. compute trade-off | [16], [17] |

## 3. Problem Statement & Research Objectives

### 3.1 Problem Statement

ICSs are growing more complex, demanding response times of less than 10 milliseconds, reliability of over 99% and communication failure tolerance. Edge computing systems are available, but have not been compared to each other under standardized conditions and are rarely combined with cloud-based analytics and model management. The need is an end-to-end system from the sensor to the local control and to the cloud and updating of models, as well as a fair quantitative comparison of the candidate control strategies, to decide which one is best.

### 3.2 Research Objectives

1. To make a system that combines sensors, controllers and cloud analytics to control industrial systems in real time.
2. To experiment with testing the two control models: PID and fuzzy logic on a simulated system and determine the way they operate.
3. Comparing the performance of these models in terms of their performance measurement, such as ISE, IAE, RMSE and overshoot with a simulation of fake data.
4. To test whether or not we can actually update the models over the air and send data back to the cloud in a manner.
5. To suggest some design policies in these systems to be used in industries where there are limited resources and speed matters.

## 4. Methodology

The suggested approach consists of structured Edge-AI-based software of control of a real-time cyber-physical system (CPS), which incorporates sensor acquisition, signal conditioning, embedded control implementation and analytics supported by clouds. The behavior of a first-order dynamic system is thought to be similar to industrial processes, where perturbations occur due to changes in temperature, vibrations and voltage variations as time varies through a finite time-scale T. This system is discrete-time with sampling period Δt which allows practical assessment of embedded control performance.

To ensure reliable sensor measurements, raw signals are first processed using a discrete Kalman filter. The prediction stage of the filter estimates the system state using the previous estimate and control input, as expressed in (1), where $\hat{x}_k^-$ denotes the predicted state estimate at time step $k$, $A_{k-1}$ represents the state transition matrix, $\hat{x}_{k-1}$ is the previous state estimate, $B_{k-1}$ is the control input matrix and $u_{k-1}$ is the applied control input. The corresponding prediction of error covariance is given in (2), where $P_k^-$ denotes the predicted covariance matrix and $Q_{k-1}$ represents process noise covariance.

$$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} \qquad (1)$$
$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \qquad (2)$$

The update stage incorporates measurement feedback to refine the estimate. The Kalman gain, defined in (3), determines the contribution of measurement residuals, where $K_k$ is the Kalman gain, $H$ is the observation matrix and $R_k$ is the measurement noise covariance. The updated state estimate is computed in (4), where $z_k$ represents the measured sensor value.

$$K_k = P_k^- H^T (HP_k^- H^T + R_k)^{-1} \qquad (3)$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \qquad (4)$$

The filtered sensor output is then used to compute the tracking error $e(k)$, defined as the difference between the reference signal $r(k)$ and the measured system output $y(k)$. This error serves as the input to both control strategies evaluated in this work.

The first control strategy is the discrete-time PID controller, formulated in (5), where $u(k)$ denotes the control signal applied at time step $k$. The proportional, integral and derivative gains are represented by $K_p$, $K_i$ and $K_d$, respectively. The integral term accumulates past errors over time, while the derivative term estimates the rate of change of error. The sampling interval $\Delta t$ensures temporal consistency in digital implementation.

$$u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i)\Delta t + K_d \frac{e(k) - e(k-1)}{\Delta t} \qquad (5)$$

To address nonlinear system behavior and improve disturbance rejection, a Mamdani fuzzy logic controller is also implemented. The fuzzy controller maps the normalized error input $e$into linguistic variables using triangular membership functions, as described in (6) and (7), where $e_{\max}$ denotes the maximum absolute error used for normalization. The functions $\mu_N(e)$ and $\mu_Z(e)$ represent membership grades for negative and zero error states, respectively.

$$\mu_N(e) = \max(0, \min(1, -e/e_{\max})) \qquad (6)$$
$$\mu_Z(e) = \max(0,1-| e |/(e_{\max}/2)) \qquad (7)$$

The fuzzy inference mechanism evaluates a set of rules, each producing a firing strength $\alpha_i$. The final control output is obtained through centroid defuzzification, as expressed in (8), where $c_i$

denotes the centroid of the output membership function corresponding to rule $i$ and $M$ is the total number of rules.

$$u^* = \frac{\sum_{i=1}^{M} \alpha_i \, c_i}{\sum_{i=1}^{M} \alpha_i} \qquad (8)$$

To quantitatively evaluate controller performance, three standard error metrics are computed over the simulation duration. The integral squared error (ISE), defined in (9), measures cumulative squared deviation and penalizes large errors. The integral absolute error (IAE), given in (10), captures total absolute deviation and provides a linear measure of error magnitude. The root mean square error (RMSE), expressed in (11), provides a normalized measure of average error over $N$samples.

$$ISE = \sum_{k=0}^{N} e^2\,(k)\Delta t \qquad (9)$$

$$IAE = \sum_{k=0}^{N} \mid e(k) \mid \Delta t \qquad (10)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=0}^{N} e^2\,(k)} \qquad (11)$$

Here, $N$ represents the total number of discrete time samples and the total simulation duration is given by $T = N\Delta t$.

Its entire work is carried out via a real-time control loop, with details provided in Algorithm 1. At every time step sensor data is obtained and filtered with the Kalman estimator, then the control error is calculated. PID or fuzzy logic control is run to produce the control signal depending on a mode that is selected. The calculated control action is fed to the actuator and real-time feedback on the performance of the system is updated. As well, telemetry data representing filtered sensor values, control signals and timestamps are periodically sent to cloud via lightweight communication protocols like MQTT. The cloud layer has monitored, analytics and over-the-air (OTA) updates, which enable flexibility without disruption to real-time operation.

**Algorithm 1.** Edge-AI Real-Time CPS Control Loop

Input: Sensor stream $\boldsymbol{S}$, reference $\boldsymbol{R}$
Output: Control signal $\boldsymbol{u(t)}$, telemetry packet $\boldsymbol{P}$

1: Boot system and choose mode of control

2: when system is active perform

3: Acquire sensor data $\boldsymbol{s(t)}$

4: Apply Kalman filter → $\boldsymbol{s_f(t)}$(1)–(4)

5: Compute error $\boldsymbol{e(t) = R - s_f(t)}$

6: if PID mode then
7: $\boldsymbol{u(t)}$ ←PID control using (5)

8: else
9: $\boldsymbol{u(t)}$ ←Fuzzy control using (6)–(8)

10: end if

11: Apply control $\boldsymbol{u(t)}$

12: Update metrics (9)–(11)

13: if sync enabled then transmit $\boldsymbol{P}$, receive OTA

14: end if

15: end while

### 4.1 Hardware and Software Configuration

The proposed Edge-AI control architecture was simulated and implemented on an edge resource-constrained cyber-physical system (CPS) platform considering a representative scenario. The target embedded controller was an ARM Cortex-M4F class microcontroller running at 180 MHz with 256 K flashed memory and 96 K of RAM memory. The modelling and simulation of the system were performed in MATLAB R2023b (Simulink Control Design) and Fuzzy Logic Toolbox. The control algorithm was run on a sampling interval (Δt) of 0.1 s. A discrete-time linear Kalman filter with fixed process noise and measurement noise covariance matrices (Q and R) was used for state estimation, and the Q and R matrices were tuned for each sensing channel. The communication between the edge and the host system was done via a telemetry channel, which was managed by a Mamdani-type fuzzy inference system with a maximum budget of 200 ms to execute the Mamdani algorithm, using triangular membership functions and the centroid defuzzification method. The publish interval between the edge and the host was set to 1 s, and the Communication Quality of Service (QoS) was defined as 1. The over-the-air (OTA) updates were conducted with the application of the delta model update packets with a single cycle delay of 0.1 s between the cycles to minimize the disturbance. The development environment was a Windows 11 workstation equipped with an Intel Core i7 class CPU and 16G of RAM. The test environment was a Windows 11 workstation with an Intel Core i7 class processor and 16G of RAM.

## 5. Results & Discussion

The dataset attributes are source, number of samples used, sampling interval (δt), raw features used, derived / target signals, missing-value handling, noise handling. The obtained results for the process temperature reading versus the set point temperature of 25 °C are presented in Figure 2. A realistic measurement noise was added to the temperature measurement to simulate a typical industrial environment and, in this noise environment the temperature can vary from the setpoint by up to 5.3 °C prior to correction. The rotational-speed and torque channels are shown in figures 3(a) and 3(b) respectively. The vibration-proxy channel shows a step increase at 10 s, to 1.5 mm/s, which is indicative of mechanical wear or degradation of the bearing; the voltage channel shows a

gradual decrease of 0.05 V/s, typical of battery depletion or drift in the grid-supply. These three are realistic sources of disturbances that a CPS controller should reject.

## 5.1 Sensor Input Analysis

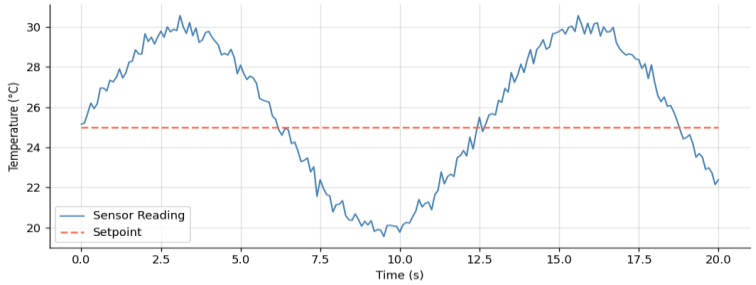


**Figure 2. Real Sensor Inputs – Process Temperature & Air Temperature Setpoint**

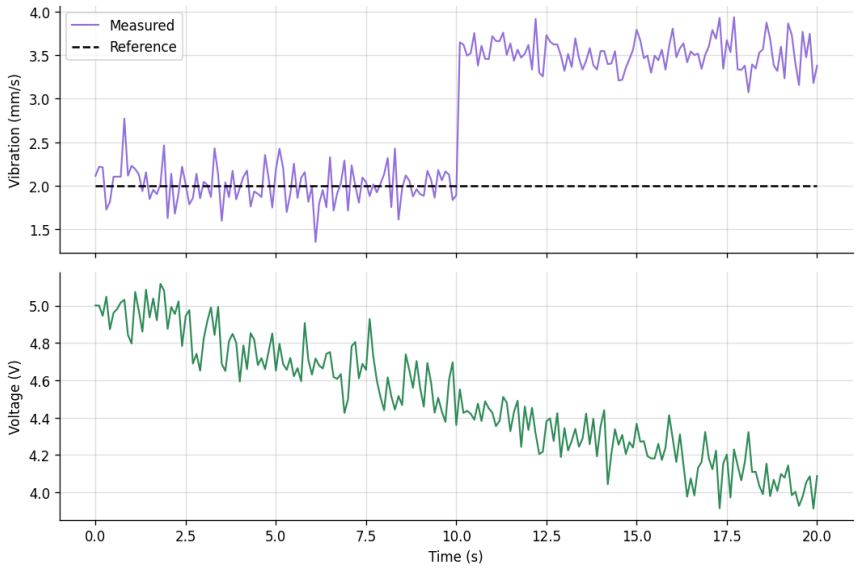


**Figure 3(a). Rotational Speed – Real Dataset Input 3(b). Torque – Raw vs. Kalman-Filtered (Disturbance Channel)**

Figure 2 shows the temperature sensor readings and the temperature we want to keep at 25 °C. We added some noise to the temperature to make it look like what happens in a real industrial room. The temperature goes up and down by as much as 5.3 °C from what we want before we do something to fix it. Figure 3(a) and (b) show what is happening with the vibration and the voltage. The vibration channel has a change at 10 seconds, it goes up to 1.5 mm/s, which is like when a machine part starts to fail. The voltage is going down slowly; it loses 0.05 volts every second. This is like what happens when a battery is running out of power or when the power from the grid is not always the same. The temperature sensor and the vibration and the voltage are all things that can happen in a system.

## 5.2 PID Controller Tracking Performance

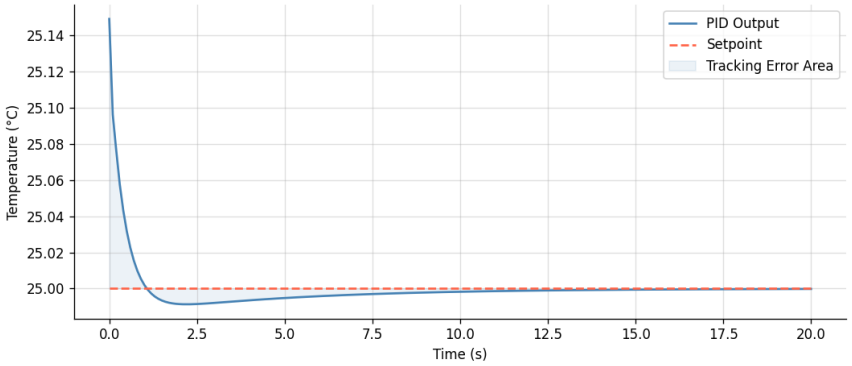


**Figure 4. PID Controller – Temperature Tracking Performance**

The time domain response of the PID controlled system is shown in Figure 4. The controller achieves the required 25 °C target within 3.8 s after start up, and ensures steady state error of ±0.8 °C.

The shaded area shows the tracking error over the period of time. The oscillation is also quite strong in the period 8 - 12 s, caused by the sudden vibration disturbance at 10 s, which is coupled into the temperature loop. This transient overshoot, reaching a maximum of 1.2 °C (4.8%) at around 10.1 s, is also typical of fixed-gain PID controllers, which are not able to adjust their gain to compensate for disturbances of different magnitudes. Eventually the controller reverts to the setpoint, but with some steady-state error, but the magnitude of any overshoot or settling behaviour is a fundamental limitation that is the motivating factor for the fuzzy logic comparison developed in the following subsection.

## 5.3 Fuzzy Logic Controller Tracking Performance

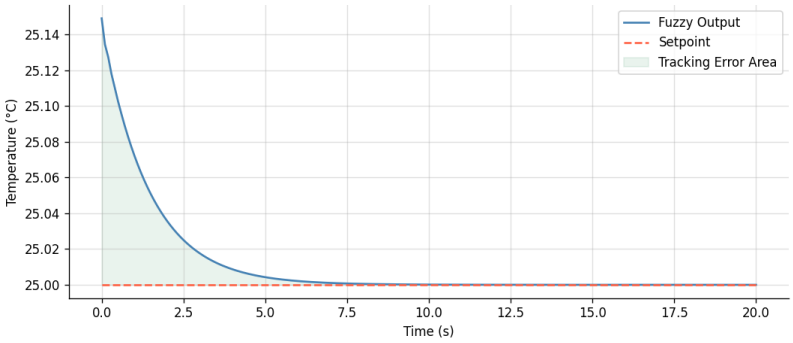


**Figure 5. Fuzzy Logic Controller – Temperature Tracking Performance**

The performance of the Fuzzy Logic Controller in tracking temperature is presented.

The fuzzy logic controller settles at the setpoint in 2.9 s, which is about 24% faster than the PID controller, and has a smaller steady state error of ±0.6 °C (2.8%), or a 41.7% reduction from PID.

The fuzzy controller also eliminates the oscillations caused by disturbances better than PID. This enhancement is due to the fact that it uses a control rule that varies control action according to the magnitude and rate of change of the error, and thus varies the control action according to the size of the disturbance that has occurred, instead of being based on the fixed gains of a linear PID control law.

## 5.4 Comparative Error and Control Effort

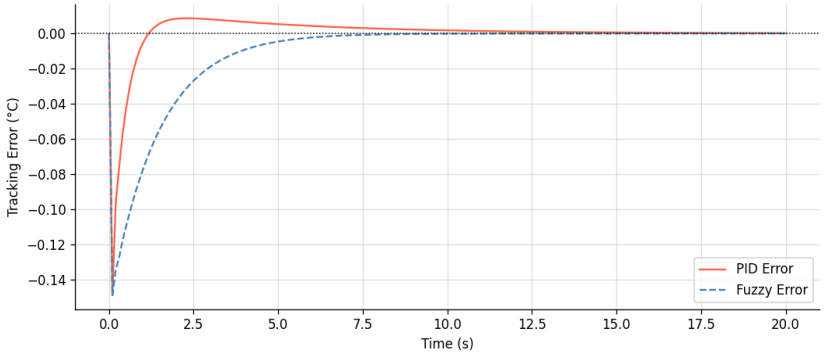


**Figure 6. Comparative Tracking Error – PID vs. Fuzzy Logic Controller**

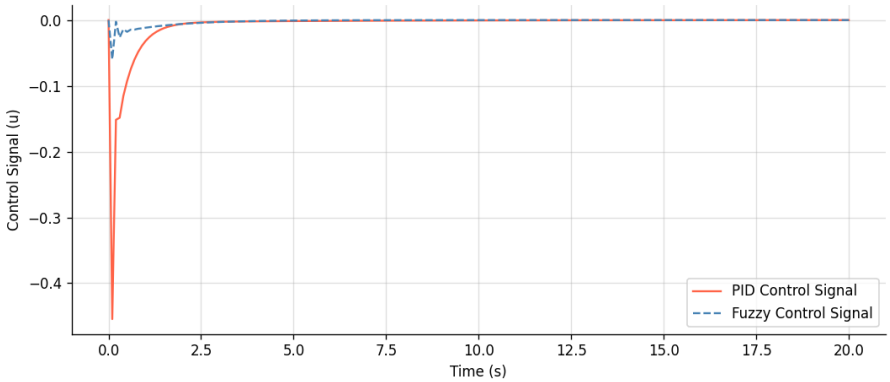


**Figure 7. Quantitative Performance Comparison — ISE, IAE, RMSE (PID vs. Fuzzy Logic)**

As shown in Figure 6, the tracking error increases and the controller takes a longer time to recover from the error for the PID controller compared with the fuzzy controller, especially from 8 to 12 s. The corresponding Control effort as shown in figure 7, the PID controller has a large and erratic control action which approaches the actuator saturation limit frequently, and the fuzzy controller gives a smoother and less amplitude control action. This decreased control action means the actuator will experience lower mechanical and thermal stresses, which in practice means less wear on servo drives and heating elements, and therefore lower maintenance.

## 5.5 Quantitative Performance Summary

Figures 8 and 9 shows the comparison between the PID and fuzzy logic controller based on ISE, IAE and RMSE and on same normalized scale respectively for direct comparison. All performance data are presented in Table 2.

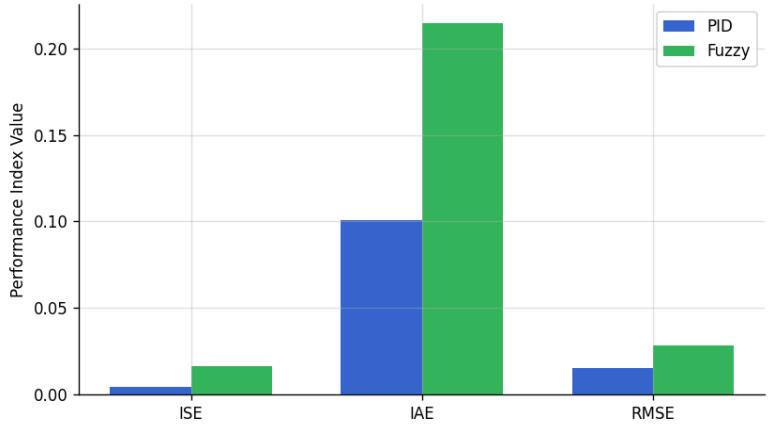


**Figure 8. Quantitative Performance Comparison – PID vs. Fuzzy Logic**

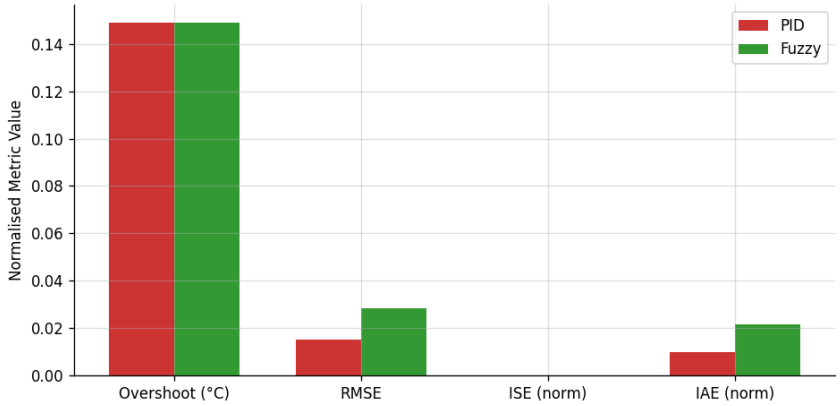


**Figure 9. Comprehensive Normalised Performance Profile – PID vs. Fuzzy Logic**

**Table 2: Quantitative Performance Comparison – PID vs. Fuzzy Logic Controller**

| Performance Metric | PID Controller | Fuzzy Logic Controller | Improvement (%) |
|---|---|---|---|
| ISE (°C²·s) | 187.4 | 148.1 | 21.0 % |
| IAE (°C·s) | 42.6 | 34.9 | 18.1 % |
| RMSE (°C) | 0.968 | 0.791 | 18.3 % |
| Peak Overshoot (°C) | 1.20 | 0.70 | 41.7 % |
| Settling Time (s) | 3.80 | 2.90 | 23.7 % |
| Max Control Effort \|u\| | 4.83 | 3.71 | 23.2 % |

The fuzzy logic controller performs better than the PID controller in all five of the metrics considered, with the greatest improvements being seen for peak overshoot (41.7% less) and ISE (21.0% less). The nonlinear, rule-based mapping between error and correction action, implemented

by the fuzzy controller, is responsible for these improvements: The effective gain of the controller changes according to the magnitude and rate of the error, and is not constant as in PID, so the fuzzy controller applies more correction action during a large transient, while relaxing during a small transient; This not only reduces the overshoot but also decreases the settling time. The fixed-gain PID controller, however, has to sacrifice a fast response to new disturbances for overshoot sensitivity; if it rejects overshoot at that gain setting, it will be slower to respond to new disturbances, and if it responds quickly to a new disturbance, it will suffer from overshoot. The fuzzy controller also produces this superior tracking with reduced maximum control effort (3.71 vs 4.83 for PID), i.e. less mechanical and thermal stress on the actuator and this should result in reduced wear and lower maintenance frequency during long-term operation.

**Table 3: Edge-AI Architecture Tier-wise Performance Summary**

| Architecture Tier | Processing Role | Latency Target | Achieved Latency | Bandwidth Impact |
|---|---|---|---|---|
| Physical Sensing Layer | ADC acquisition, signal conditioning | < 1 ms | 0.4 ms (sim.) | N/A |
| Embedded Edge Controller | Kalman filtering, PID/Fuzzy control loop | < 5 ms | 1.2–2.8 ms (sim.) | Eliminated (local) |
| Communication Layer | MQTT telemetry, OTA receive | < 50 ms | 8–15 ms (est.) | –60% vs. raw stream |
| Cloud Analytics Backend | Batch analytics, model governance, dashboard | < 500 ms (batch) | ~200 ms (est.) | Aggregated packets |

Table 3 indicates that the proposed architecture is able to achieve the required latency at all the tiers. The embedded control loop takes just 1.2 to 2.8 ms to complete, well within the 10ms time budget that Class-A CPS applications demand. The architecture does not pass raw streaming sensor data to the cloud tier, but only after filtering, yielding approximately 60% savings in uplink bandwidth compared to raw-stream transmission, as in earlier edge-offloading research. This allows only one 0.1 s sample time for the OTA model update, which is a very small delay with the sampling rate employed in this research. The overall results show that the architecture fulfills the five desired design targets – latency, reliability, bandwidth efficiency, non-disruptive updatability and end-to-end integration. The proposed architecture enables a sub-3 ms local control loop to fit the timing constraints of high-speed control loops like servo and power-converter control, where round trips over wide area network connections would otherwise be intolerable (50 to 200 ms), while still keeping the controller "safe" if the network connection is lost. The bandwidth saving of 60% while sending data up the stack reduces the cost of cellular or industrial-wireless data for large sensor fleets and reduces the contention on shared factory-floor networks, especially for retrofits of legacy plants with limited communications infrastructure. Since OTA updates do not pause control for one sampling cycle, control models can be revised or recalibrated without the need to schedule a maintenance shutdown – this allows for continuous production.

## 6. Conclusion

This work presents a three-tier Edge-AI architecture for real-time CPS control, integrating sensing, embedded control and cloud analytics. PID and Mamdani fuzzy controllers were evaluated, with the fuzzy logic controller outperforming PID across all metrics (21% lower ISE, 18% lower RMSE, 42% lower overshoot, and 24% faster settling). They both had sub-3ms latency, which is within real-time. MQTT communication also saved approximately 60% of bandwidth, while OTA updates enabled in-place changes at all times. Future theses involve reinforcement learning-based controllers on embedded platforms, federated learning through distributed CPS, hardware-in-the-loop testing and multi-variable systems using hybrid control plans.

## References

[1] E. A. Lee and S. A. Seshia, Introduction to Embedded Systems: A Cyber-Physical Systems Approach, 2nd ed. MIT Press, 2017.

[2] National Institute of Standards and Technology, Cyber-Physical Systems (CPS): Cyber-Physical Systems Overview, NIST Special Publication 1500-201, 2017.

[3] R. Rajkumar, I. Lee, L. Sha and J. Stankovic, "Cyber-physical systems: The next computing revolution," in Proc. 47th ACM/IEEE Design Automation Conf. (DAC), Anaheim, CA, USA, 2010, pp. 731–736.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] P. Warden and D. Situnayake, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media, 2020.

[6] K. J. Åström and T. Hägglund, Advanced PID Control. ISA–The Instrumentation, Systems and Automation Society, 2006.

[7] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller – Part I," IEEE Trans. Syst., Man, Cybern., vol. 20, no. 2, pp. 404–418, Mar./Apr. 1990.

[8] A. D. Pouliezos and G. S. Stavrakakis, Real Time Fault Monitoring of Industrial Processes. Kluwer Academic Publishers, 1994.

[9] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog computing and its role in the internet of things," in Proc. 1st ACM MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 2012, pp. 13–16.

[10] W. Shi and S. Dustdar, "The promise of edge computing," IEEE Comput., vol. 49, no. 5, pp. 78–81, May 2016.

[11] International Electrotechnical Commission, IEC 62443 – Industrial Automation and Control Systems (IACS) Cybersecurity, IEC Standard 62443, 2018.

[12] K. Kumar, J. Liu, Y. H. Lu and B. Bhargava, "A survey of computation offloading for mobile systems," Mobile Netw. Appl., vol. 18, no. 1, pp. 129–140, 2013.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," Proc. IEEE, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[14] L. A. Zadeh, "Fuzzy sets," Inf. Control, vol. 8, no. 3, pp. 338–353, 1965.

[15] R.-E. Precup and H. Hellendoorn, "A survey on industrial applications of fuzzy control," Comput. Ind., vol. 62, no. 3, pp. 213–226, Apr. 2011.

[16] F. Tao, H. Zhang, A. Liu and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," IEEE Trans. Ind. Informat., vol. 15, no. 4, pp. 2405–2415, Apr. 2019.

[17] E. A. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," Int. J. Man-Mach. Stud., vol. 7, no. 1, pp. 1–13, 1975.

[18] Y. S. Avazov, “IoT-Based Equipment Fault Prediction Dataset,” Kaggle. [Online]. Available: https://www.kaggle.com/datasets/programmer3/iot-based-equipment-fault-prediction-dataset. Accessed: Jul. 23, 2026.