Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU33082

Embedded Control Algorithms and Implementation under Hardware Constraints

Zimo Wang

Department of Electronic and Electrical Engineering, University College London (UCL), London, UK
zceezw5@ucl.ac.uk

Abstract. Embedded control lies at the intersection of elegant control design and strict hardware limitations. Robots, motor drivers and the controllers of industrial machines must operate under conditions such as fixed sampling intervals, constrained computing power and memory, limited computing precision, and sometimes restricted communication conditions. As a result, these limitations can subtly influence closed-loop behavior, thus making deployability as important as performance. This paper explores embedded control algorithms, optimization strategies for implementation, and smart control on embedded platforms. Among the common approaches, algorithm families include classical feedback control, state-space methods with estimation, and robust and adaptive methods for handling uncertainties and system drift. In order to make these methods practical in practice, several implementation strategies are proposed, including offline pre-computation, solver selection in optimization control, fixed-point arithmetic and scaling techniques, and execution methods for energy and memory optimization. Meanwhile, real-time scheduling and time predictability are regarded as core design factors, since delays and jitter directly affect system stability. With regard to concerns such as security during deployment, authentication, and worst-case execution time, this study also discusses rule-based control and learning-based intelligent control methods. Besides, it addresses current challenges and future directions, including hybrid model learning, control-computation integration, and verification for safety-critical embedded systems.

Keywords: Embedded control, real-time systems, fixed-point implementation, embedded optimization, intelligent control

1. Introduction

Embedded controllers are the core of many safety and performance-critical systems, including robots, electric drives, unmanned aircraft, and industrial automation. Compared to desktop or cloud platforms, embedded devices must maintain stable control despite limited resources such as processing power, memory, energy, and communication. Besides, embedded systems usually need to meet strict real-time requirements, such as sensing, computing, and execution must be completed within a fixed sampling period. When delays and jitter occur, they can severely influence closed-loop performance, making methods that are theoretically viable ineffective in practical applications,

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU33082

particularly if they cannot operate reliably and predictably on the target hardware. The control design has evolved from classical feedback control (such as PID) to modern state-space methods, and further developed into robust and adaptive techniques capable of handling uncertainties and time-varying dynamics. At the same time, with the increasing demand for higher autonomy, the interest in intelligent control technologies has gradually risen, which has driven the shift from interpretable rule-based control schemes to learning-based methods that use data for modeling and making decisions. However, when these methods are applied to embedded platforms, they often introduce challenges that are easy to overlook. For instance, issues such as limited computational precision, fixed-point scaling, numerical range limitations, solver complexity, and the definition of the worst-case execution time. And these limitations can change the closed-loop dynamics and must be considered in the control design process. This paper reviews embedded control algorithms and optimization techniques to improve system efficiency and real-time performance. It covers algorithm types, predictable execution methods, and intelligent control for reliable, transparent, and secure embedded systems. Furthermore, key challenges and future directions are discussed, focusing on control-computation co-design, edge AI acceleration, and secure intelligent controllers.

2. Overview of embedded control algorithms

2.1. Classical control methods

Classical control remains the default choice in many embedded products due to its simplicity, strong engineering intuition, and low computational cost. The most widely used structure is the PID controller, which combines proportional, integral, and derivative actions to balance tracking, disturbance rejection, and transient performance [1]. In embedded implementations, PID is favored in embedded systems for its low computation and minimal memory, thus making it ideal for low-cost microcontrollers. Practical embedded PID designs typically incorporate derivative filtering to reduce noise amplification, anti-windup mechanisms for actuator saturation, and feedforward paths for measurable disturbances or reference dynamics [1,2].

A common theoretical lens for classical design is the frequency-domain view. By shaping loop gain and phase using Bode or Nyquist methods, designers can reason about robustness margins and the influence of unmodeled dynamics [2]. And this is especially important for embedded control because sampling and computation delay introduce additional phase lag, which reduces phase margin and can destabilize otherwise well-tuned continuous-time designs. Consequently, classical controllers are often tuned in discrete time or validated with discrete-time models to ensure adequate stability margins after discretization [2].

2.2. Modern control methods

Modern control methods are built on state-space modelling and systematic synthesis procedures. A central idea is representing the plant with a set of state variables and designing feedback laws to place closed-loop dynamics in desired regions, often with explicit consideration of multivariable interactions [2]. On embedded platforms, state feedback can be computationally light once the feedback gain is computed offline; the online computation is typically a matrix-vector multiplication per sample.

State estimation is frequently required because not all states are directly measurable. The Kalman filter provides a principled framework for optimal linear estimation under stochastic noise assumptions [3]. In embedded applications, Kalman filtering is used for sensor fusion and noise

DOI: 10.54254/2755-2721/2026.GU33082

reduction, and it can be implemented efficiently with careful numerical choices. However, embedded deployment introduces sensitivity to finite precision: covariance updates and matrix operations can suffer from numerical drift or loss of key matrix properties if implemented naively. Thus, practitioners often select numerically stable implementations and tune noise models to balance responsiveness and robustness [2,3].

2.3. Robust control and adaptive control

Robust and adaptive control address uncertainty and variability that are difficult to handle with fixed-gain designs. Robust control methods explicitly model plant uncertainty and synthesize controllers that guarantee performance across an uncertainty set. A representative framework is H∞ and related robust synthesis tools, which formalize worst-case disturbance attenuation and stability margins [4]. From an embedded viewpoint, robust control can improve reliability under changing loads, temperature effects, and component tolerances, but it may produce higher-order controllers that increase memory and computation demands, motivating simplified realizations or controller order reduction [4].

In contrast, adaptive control adjusts controller parameters online to handle time-varying dynamics or uncertainties. Robust adaptive control aims to maintain stability and acceptable performance despite modelling errors and limited excitation [5]. In embedded systems, adaptive controllers can be valuable for plants whose parameters drift over time, but online adaptation introduces risks like noise sensitivity and increased computational cost. Thus, embedded adaptive designs usually use parameter bounding and monitoring to ensure safe operation under limited precision and computation [5].

3. Optimization methods and implementation of control systems

3.1. Numerical algorithms and fixed-point techniques

In many embedded controllers, numerical computation is the main contributor to runtime cost. This is particularly true for designs that involve estimation such as Kalman filtering, multivariable feedback, or optimization-based control like MPC. To minimize the online workload, a common approach is to perform most computations offline, leaving the online loop to consist of simple, repeatable operations like precomputing gains, factorizing matrices, or simplifying the model.

For optimization-based controllers, feasibility is heavily influenced by both the problem formulation and the choice of solver. In the case of embedded MPC, online control is often reduced to a quadratic program (QP), which drives the need for tailored solvers and automatic code generation. For example, CVXGEN generates compact C solvers for small to medium convex programs, enabling low-latency execution on embedded targets [6]. On the other hand, OSQP uses an operator-splitting method and is widely favored for its numerical robustness and warm-start support, reducing average iterations in receding-horizon control [7]. In practice, embedded implementations also take advantage of structural properties and reuse intermediate results between time steps to keep computations bounded.

Finite precision is another key issue, as many microcontrollers lack floating-point units or perform better with fixed-point arithmetic. Fixed-point implementation requires careful scaling, like selecting the right Q-format, along with explicit saturation handling and sensitivity checks to avoid overflow and maintain closed-loop stability. There are some common methods, including range analysis to determine scaling factors for signals and states, saturation arithmetic to avoid wrap-

DOI: 10.54254/2755-2721/2026.GU33082

around, and numerically stable update forms for recursions. In estimation and optimization, limited precision can accumulate into bias or disrupt key matrix properties. To reduce these failure modes, embedded designs commonly rely on numerically stable factorizationbased updates, mild regularization, and explicit bounds that prevent ill-conditioned steps. These choices also affect timing behaviour. Specifically, when numerics become fragile, solvers may require extra iterations, filters may need re-initialization, and protective checks can be triggered, which ultimately leads to irregular execution time and jitter in the control loop.

3.2. Energy and memory efficiency in control

Energy and memory limits are often the main constraints in embedded control, especially for battery-powered robots, wearables, and wireless sensor networks. A recurring observation is that computation and communication can be traded against control performance, which motivates designs that minimize update activity when the system evolves slowly or additional control actions provide little benefit.

In addition, a key viewpoint comes from control under communication and data-rate constraints. When the feedback channel is limited, designers must decide what to transmit, how often to transmit it, and how quantization alters stability and performance [8]. This leads to strategies such as quantized feedback, state compression, and event-based communication. Event-triggered and self-triggered control are especially useful, as they update the control input only when a trigger indicates potential performance degradation. By avoiding unnecessary sampling, computation, or transmission, these strategies can improve energy efficiency while maintaining stability under suitable triggering rules [9]. From an implementation standpoint, the fixed-rate loop is replaced by conditional execution. The trigger itself must be simple and robust to noise to prevent excessive switching or "chattering."

Memory efficiency is also crucial, as large buffers raise RAM usage and can cause costly access patterns in long horizons, high-order observers, or large learning models. Common remedies include reduced-order modeling, shorter prediction horizons, and compact parameterizations to save memory and reduce computation. Approximate representations, like fixed-point coefficients, lookup tables for nonlinear functions, and compressed models, can reduce memory usage but introduce approximation error. In a closed loop, this error acts like an additional disturbance or uncertainty and should be analysed in that role rather than treated as a purely numerical artifact.

3.3. Real-time scheduling and timing optimization

Real-time constraints distinguish embedded control from many other software workloads. Each control cycle must complete sensing, computation, and actuation within a fixed sampling period, and variation in completion time (jitter) can be as harmful as a large average delay. Timing is therefore a design variable, not a background implementation detail.

One important aspect of this is schedulability. Given multiple tasks, such as control, estimation, communication, and logging, the system must meet deadlines under the chosen schedule. Classic work in real-time control formalized how scheduling constraints interact with control quality and provided conditions under which task sets remain schedulable [10]. In practice, designers choose task periods, priorities, and execution budgets so that the control loop meets its deadline even under worst-case execution time (WCET) assumptions. When deadlines cannot be met, computation is often restructured by moving heavy work to lower-priority tasks or using multi-rate architectures with a fast inner loop and slower outer loop. Timing optimisation also depends on algorithm choice.

DOI: 10.54254/2755-2721/2026.GU33082

Fixed-gain controllers and linear observers typically have stable execution cost, whereas optimization-based controllers may show input-dependent iteration counts. This motivates bounded-iteration solvers, warmstarting, and early termination rules that guarantee completion within the sampling period while still returning acceptable control actions [7]. In networked settings, timing also includes communication delays and packet losses. Eventtriggered control can reduce network load but must be coordinated with scheduling policies to prevent bursty traffic and deadline misses [9].

4. Intelligent control and future development

4.1. Rule-based intelligent control

Rule-based intelligent control encodes human insights or heuristics in an interpretable form, with fuzzy control as a prime example. By using graded membership functions, it can represent linguistic concepts like "small error" or "large error" and generate smooth nonlinear control actions without requiring an explicit plant model [11]. In addition, many practical designs follow the Mamdani framework, with a defuzzification step producing a crisp control signal [12]. From an embedded viewpoint, rule-based controllers are attractive because the inference pipeline can be implemented as a fixed sequence of membership evaluations and rule aggregations, which makes runtime cost relatively predictable. Their transparency also supports debugging and maintenance, since engineers can inspect rules directly and adjust them in a controlled manner. On the other hand, scalability is considered as the main drawback. As the number of inputs or linguistic partitions grows, the rule base can expand rapidly, increasing both memory usage and computation. Performance is also sensitive to rule design and membership tuning, which may depend on expert knowledge or structured tuning procedures.

4.2. Learning-based intelligent control

Learning-based control uses data to identify dynamics or to directly learn control policies. Neural-network-based control is one of the earliest learning paradigms in this area, where networks act as nonlinear function approximators for system identification or for synthesizing control laws [13].On embedded systems, neural networks are commonly used to compensate unmodeled effects like friction or parameter drift, or to implement policy mappings for higherlevel decisions. However, deployment, requires attention to inference latency, precision, and stability effects of approximation error. In parallel, reinforcement learning (RL) frames control as sequential decision-making, aiming to learn a policy that maximizes long-term reward through interaction with the environment [14]. Despite its successes on complex tasks, embedded RL faces additional constraints. For example, online exploration can be unsafe for physical systems, and training is usually too expensive to run on edge devices. For these reasons, many embedded RL pipelines rely on offline training and on-device inference, often combined with model-based elements or supervisory safety mechanisms. This has driven growing interest in safe RL methods that integrate explicit safety constraints, risk measures, or fallback strategies [15].

4.3. Challenges and development trends

In embedded systems, safety and verifiability are critical challenges for intelligent control. Compared with classical controllers, learning-based learning-based policies are harder to certify,

DOI: 10.54254/2755-2721/2026.GU33082

especially when model mismatches or rare events occur [15]. Timing predictability is another concern. Even if average inference time is small, worst-case latency and jitter can degrade closed-loop performance. Finite precision and quantization can further introduce subtle failure modes, where small numerical effects accumulate into meaningful deviations in the control action. Moreover, several development directions are emerging. For example, hybrid model-based and learning control improves adaptability while also preserving structure. Besides, control-computation co-design is crucial, as intelligent controllers must fit resource budgets and deployment constraints. Additionally, verification-oriented intelligent control is gaining traction, with methods like constraint-aware learning, runtime monitoring, and fail-safe systems that revert to a certified baseline controller when needed [15]. These trends point to embedded intelligent control integrating learning, safety, and reliable real-time execution

5. Conclusion

This paper reviewed embedded control from three perspectives: algorithm families, implementation optimization, and emerging trends in intelligent control. Classical methods like PID are favored for their simplicity and predictable behavior. Modern state-space and robust control approaches enhance reliability under uncertainty, though they require more effort for synthesis and validation. A key theme is balancing theoretical methods with embedded hardware constraints. From an implementation standpoint, deployability is shaped by system-level details, often reducing online computation by offloading work offline. Finite-precision effects, energy, and memory efficiency also drive design choices, especially in resource-constrained systems. Real-time scheduling and timing optimization highlight the need for integrated control design and software scheduling. Intelligent control offers autonomy and adaptation but raises challenges in safety, interpretability, and timing predictability. Rule-based methods provide transparency, while learning-based methods handle complexity but can be difficult to certify. Future research will likely focus on hybrid designs, control-computation co-design, and verification-oriented architectures to ensure reliable behavior under uncertainty.

References

[1] K. J. Astro¨m, & T. H¨agglund. (1995). PID controllers: Theory, design, and tuning (2nd ed.). Research Triangle Park, NC, USA: ISA.
[2] G. F. Franklin, J. D. Powell, & A. Emami-Naeini. (2015). Feedback control of dynamic systems (7th ed.). Boston, MA, USA: Pearson.
[3] R. E. Kalman. (1960). A new approach to linear filtering and prediction problems. ASME Journal of Basic Engineering, 82(1), 35-45.
[4] K. Zhou, J. C. Doyle, & K. Glover. (1996). Robust and optimal control. Upper Saddle River, NJ, USA: Prentice Hall.
[5] P. A. Ioannou, & J. Sun. (1996). Robust adaptive control. Upper Saddle River, NJ, USA: Prentice Hall.
[6] W. P. M. H. Heemels, K. H. Johansson, & P. Tabuada. (2012). An introduction to event-triggered and self-triggered control. In Proceedings of the 51st IEEE Conference on Decision and Control (CDC), 3270–3285.
[7] G. N. Nair, F. Fagnani, S. Zampieri, & R. Evans. (2007). Feedback control under data rate constraints: An overview. Proceedings of the IEEE, 95(1), 108-137.
[8] D. Seto, J. P. Lehoczky, L. Sha, & K. G. Shin. (1996). On task schedulability in realtime control systems. In Proceedings of the 17th IEEE Real-Time Systems Symposium (RTSS), 13-21.
[9] J. Mattingley, & S. Boyd. (2012). CVXGEN: A code generator for embedded convex optimization. Optimization and Engineering, 13(1), 1-27.
[10] B. Stellato, G. Banjac, P. Goulart, P. Bemporad, & S. Boyd. (2020). OSQP: An operator splitting solver for quadratic programs. Mathematical Programming Computation, 12, 637-672.

DOI: 10.54254/2755-2721/2026.GU33082

[11] L. A. Zadeh. (1965). Fuzzy sets. Information and Control, 8(3), 338-353.
[12] E. H. Mamdani. (1974). Application of fuzzy algorithms for control of simple dynamic plant. Proceedings of the IEE, 121(12), 1585-1588.
[13] K. S. Narendra, & K. Parthasarathy. (1990). Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks, 1(1), 4-27.
[14] R. S. Sutton, & A. G. Barto. (2018). Reinforcement learning: An introduction (2nd ed.). Cambridge, MA, USA: MIT Press.
[15] J. Garc´ıa, & F. Fern´andez. (2015). A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research, 16, 1437-1480.