Constraint-Aware Software-Defined I/O for Reconfigurable Smart Environments

Mohammad Ghavidel Vahid, Giuseppe Tricomi, Luca D'Agati, Francesco Longo,
Antonio Puliafito, Guido Di Bella, Giovanni Merlino
Department of Engineering (DI), University of Messina, Messina, Italy
{mohammad.ghavidelvahid@studenti.unime.it; ldagati, gtricomi, flongo, apuliafito, gudibella, gmerlino}@unime.it

***Abstract*—Smart environments rely on cyber-physical services combining sensing, edge computation, actuation, and reconfiguration. This paper presents a Software-Defined Input/Output (SDI/O) architecture that exposes plant-facing resources as programmable entities while preserving the timing and ownership constraints required by feedback control. The architecture separates environment-level coordination from low-level enforcement, mapping policies to physical devices without assuming that orchestration alone guarantees safe actuation. Proportional–Integral (PI), Proportional–Integral–Derivative (PID), Model Predictive Control (MPC), and Distributed Model Predictive Control (DMPC) modules are deployable only when explicit admissibility conditions on input/output (I/O) compatibility, placement, state age bounds, latency, command ownership, and fallback behavior are satisfied. Prototype evidence on an embedded edge node with WebAssembly execution and Robot Operating System for microcontrollers (micro-ROS) telemetry shows that native, ahead-of-time (AOT), and interpreted PID controllers can run on the same physical loop, while telemetry-driven models support preliminary embedded MPC. The result is a constraint-aware infrastructure for deciding when control services can be rebound, upgraded, or distributed over software-defined resources.**

***Index Terms*—Software-Defined Input/Output, constraint-aware deployment, smart environments, cyber-physical systems, edge control, reconfigurable services, model predictive control.**

I. Motivation and Contribution

Smart environments are not only data-collection platforms. Many priority services require physical action across ventilation, lighting, charging, traffic devices, environmental monitoring, energy equipment, and mobile or robotic assets. They are context-dependent and multidisciplinary, but their Information and Communication Technology (ICT) substrate must still determine when a digital policy can safely become a physical command.

Software-defined city and building infrastructures show how shared physical resources can become programmable and rewireable [1]. SDI/O-based interaction services similarly show how heterogeneous devices can be configured through software-defined abstractions [2]. However, a resource participating in a feedback loop cannot be managed as a generic service endpoint. A command applied to a motor, valve, fan, light, or converter is valid only if measurements are time-valid, deadlines are met, the controller owns the actuator, and local enforcement remains deterministic. This makes the approach relevant to ICT infrastructures and software/services for smart cities, where flexibility must coexist with reliable cyber-physical behavior.

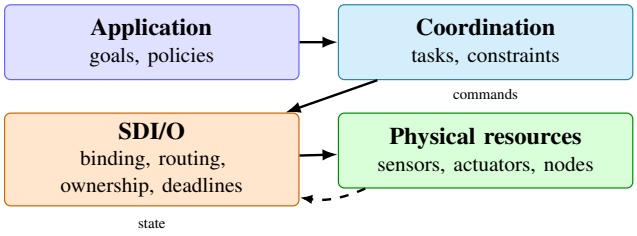

Figure 1. Layered SDI/O stack.

We therefore frame SDI/O as the plant-facing layer of a smart-environment ICT stack. The central claim is that advanced control is an architectural capability rather than an algorithmic assumption. Constraint awareness means checking physical, timing, state-age, ownership, and fallback constraints before accepting a controller-resource binding. Classical PI/PID, MPC, and DMPC are deployable over shared infrastructure only when the architecture preserves their timing, state, communication, and actuation assumptions [3], [4]. The contribution is threefold:

- a layered SDI/O architecture separating environment-level coordination from deterministic low-level actuation;
- a compact set of deployment criteria for binding controller modules to sensing, computation, and actuation resources;
- prototype evidence showing how runtime control, telemetry-driven models, and preliminary embedded MPC can share one reconfigurable input/output path.

II. SDI/O Architecture

Fig. 1 summarizes the four-role stack. The application layer captures goals and policies; coordination maps them to tasks, references, priorities, and constraints; SDI/O exposes sensing, computation, and actuation as typed resources; and the physical layer contains embedded controllers, buses, sensors, actuators, power electronics, and mobile units.

The key separation is between deciding *what* should happen and validating *how* commands reach physical devices. An energy policy may request demand reduction, but a fan or valve command still needs time-valid measurements, a feasible deadline, valid limits, and exclusive actuator ownership. SDI/O does not replace local control; it makes sensing path, control module, placement, route, endpoint, and authority auditable.

Let a controller C be bound to sensing resources S, computation placement P, and actuation resources A. The predicate $\mathcal{D}$ is a Boolean constraint gate:

$$\mathcal{D}(C, S, P, A) = 1 \tag{1}$$

means that the binding (C, S, P, A) may drive the actuator; otherwise $\mathcal{D} = 0$ and SDI/O keeps the previous binding or triggers fallback. The timing constraint is the sensing-to-enforcement budget: T_{sense} is acquisition/timestamping, $T_{exec}(P)$ execution at placement P, T_{comm} routing delay, J jitter margin, and D_C the controller deadline. Thus $T_{sense} + T_{exec}(P) + T_{comm} + J \leq D_C$ must hold, together with I/O type compatibility, state age bounds, actuator ownership, command limits, and fallback availability. In practice, this gate turns reconfiguration into an explicit pass/fail decision rather than a best-effort orchestration action.

PI/PID, MPC, and DMPC impose progressively stronger deployment conditions: PI/PID requires local sampling, bounded jitter, time-valid measurements, and exclusive actuator ownership; MPC adds a valid plant model, optimizer deadline, and enforceable first command; DMPC further requires neighbor-state age bounds, synchronization, coupling-variable exchange, and local fallback.

This view also clarifies communication semantics. Upstream telemetry carries measurements, timestamps, controller state, prediction errors, and command logs; downstream directives carry references, constraints, controller selection, and policies; lateral exchange carries neighbor state or coupling variables for distributed control. Delay, ordering, and loss behavior determine whether a smart-environment service can safely close the loop.

III. Prototype Evidence

The prototype instantiates one plant-facing SDI/O node: an STM32 Nucleo-F746ZG/Arm Cortex-M7 running Zephyr, a motor with encoder feedback, an L9110 H-bridge, interrupt counting, software pulse-width modulation (PWM), WebAssembly controller execution, and micro-ROS telemetry. The setup keeps the physical I/O path fixed while the control mechanism changes, making deadline and enforcement effects observable.

Three PID implementations were measured on the same loop. Native C required 1.3 μs on average, WebAssembly AOT required 7.90 μs, and interpreted WebAssembly required 36.5 μs. The AOT controller was measured over 510 samples with a 7.74–14.23 μs range and approximately 0.0052% processor utilization for the observed 152 ms period. Runtime execution therefore introduces measurable overhead, but remains well below this plant's timing budget. The result illustrates the same admissibility rule: execution time and jitter must fit the deadline.

Telemetry converted the node from an actuator endpoint into an observable model-aware resource. micro-ROS logs supported duty-cycle identification, embedded model deployment, and online validation. After exponential filtering with $\alpha = 0.2$, the telemetry was fitted to

$$v[k+1] = 0.664v[k] + 0.333u[k], \tag{2}$$

Equation (2) is a first-order discrete-time predictor: $u[k]$ is the PWM duty command, $v[k]$ the measured speed, and $v[k+1]$ the next predicted speed. The terms 0.664 and 0.333 capture retained speed and the duty-to-speed gain identified from logs rather than analytical assumptions. The model achieved mean absolute error (MAE) ≈ 2.60 and root mean square error (RMSE) ≈ 4.48; during preliminary embedded MPC, online prediction remained observable with MAE ≈ 4.9 and RMSE ≈ 7.1.

For smart-environment deployment, the experiment suggests reporting sensing-to-enforcement latency, deadline-valid commands, stale-command rejection, ownership conflicts, rebinding time, and fallback activations. These metrics expose compatibility with the physical process and support rollback when timing or state-age guarantees degrade.

The most relevant lesson was architectural: an identification-stage scheduler overwrote the optimizer duty output before actuation. Removing the overwrite made the applied command match the MPC prediction. This confirms why SDI/O must track command ownership and enforcement: a correct optimizer is insufficient if another pipeline can modify the physical command. The prototype validates local I/O, runtime PID substitution, telemetry, embedded models, and preliminary MPC; multi-node DMPC, stale-command rejection, rollback, and resource registries remain future validation steps.

IV. Conclusion

This paper positions SDI/O as constraint-aware infrastructure for smart-environment cyber-physical services. The main idea is not to move control to the cloud, but to make sensing, controller binding, placement, and actuation endpoints programmable while keeping deterministic enforcement near the resource. This distinction is essential when environment services must adapt to context without losing safety, accountability, or timing guarantees.

The constraint-based view gives researchers and practitioners a practical deployment question: can this controller, on this placement, with these measurements and this actuator, satisfy the deadline, state-age bounds, ownership, and fallback conditions of the physical process? Prototype results suggest that runtime PID and preliminary embedded MPC are feasible on the considered node, while distributed control requires stronger evidence on synchronization, coupling-variable age bounds, and fallback. Future work will extend telemetry with resource identifiers, command deadlines, ownership events, rebinding time, stale-command rejection, and multi-node measurements.

References

[1] G. Tricomi, G. Merlino, F. Longo, S. Distefano, and A. Puliafito, "Software-defined city infrastructure: a control plane for rewireable smart cities," *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 180–185, 2019.

[2] G. Tricomi, L. D'Agati, Z. Benomar, F. Longo, G. Merlino, and A. Puliafito, "Interfacing intelligent personal assistant to sdi/o with one click," in *2022 International Conference on Computer Communications and Networks (ICCCN)*, 2022, pp. 1–8.

[3] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.

[4] M. Ghavidel Vahid, R. Maamoor, L. D'Agati, A. Puliafito, F. Longo, and G. Merlino, "Reconfigurable distributed model predictive control for decentralized it/ot systems," in *2025 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2025, pp. 384–389.