Architectural Design Patterns for Baseline Regulation in Embodied Systems

Author: Tyson Jeffreys **Date:** December 30, 2025 **Scope:** conceptual supplement (not empirical).

This section presents concrete, implementable architectural patterns that instantiate the paper's core idea: robustness in embodied systems can emerge from **baseline regulation and slow, global constraint signals**, rather than from centralized correction alone.

These examples are not presented as algorithms or proofs, but as **design patterns** that can be layered onto existing robotics and embodied AI stacks.

Example 1: A "Baseline Regulator" Layer Beneath RL / MPC

Goal
Maintain a low-correction, low-energy *stable manifold* during nominal operation so that perturbations do not cascade into instability.

Design
Introduce a slow control loop (e.g., 1–5 Hz) that continuously estimates a **regulatory cost** using signals already present in most systems, such as:

- total actuator effort (e.g., $torque^2$)
- jerk penalties
- contact slip rate
- state-estimation uncertainty (e.g., covariance)
- prediction error between expected IMU / foot forces and observed signals

The baseline regulator does **not** command actions or trajectories.

Instead, it **biases parameters** in the existing control stack:

- adjusts impedance / stiffness gains
- adjusts MPC cost weights (e.g., stability vs. speed)
- adjusts allowed step height and speed envelopes

Why It Matches the Paper
Stability emerges from **distributed regulation rather than centralized command**.
The planner and learned policies remain intact; the baseline regulator functions as an under-layer that prevents overcorrection spirals.

Example 2: Global Constraint Signals as Broadcast Budgets

Concept
Replace task-local optimization with a small set of slow, shared variables that every subsystem can "feel."

Candidate Global Signals

- **Energy budget**: battery SoC, instantaneous power draw, remaining mission energy
- **Thermal budget**: motor and driver temperatures, predicted thermal headroom
- **Traction budget**: global slip likelihood (from fused foot force, IMU, and terrain classification)
- **Compute / latency budget**: control-loop latency, dropped frames

How to Use Them
Broadcast these signals to all layers:

- state estimator
- gait generator
- manipulation controller
- RL policy

Each layer treats them as **boundary conditions**, not commands:

- the estimator increases smoothing when compute budget is tight
- locomotion reduces aggressive maneuvers when traction budget is low
- manipulation reduces acceleration when thermal headroom is limited

This resembles whole-body control's "respect constraints" principle, but with a critical difference: the signals are **slow, shared, and always present**, synchronizing subsystems continuously rather than reacting locally.

Example 3: Predictive Coupling for Active Sensing

Concept
Misalignment between sensing and motion increases prediction error and corrective effort.

Concrete Robot Scenario
A mobile robot with a camera mounted on a pan–tilt unit:

- when the camera is aimed far off the velocity vector, optical flow predictions and inertial predictions diverge
- estimator strain increases

- control corrections increase downstream

Design
Introduce a coupling objective:

- add a cost that gently biases sensor-frame alignment toward expected motion
- allow deviation only when an explicit task requires it (e.g., search or inspection)

Engineering Metric
Minimize the discrepancy between predicted optical flow (from the motion model) and observed flow. Feed this prediction-error term into the baseline regulator as part of the regulatory cost.

This aligns with forward-model and predictive-processing traditions, but reframed as **regulatory cost minimization**, not task optimization.

Example 4: Reinforcement Learning with Global Constraints as Shared Latent State

Concept
Make global constraints first-class citizens in learning-based systems.

Design
Introduce a small set of slow-evolving latent variables:

- energy headroom
- thermal headroom
- traction confidence
- estimator confidence

Then:

- feed these variables into the policy so action selection respects global conditions
- add auxiliary losses that encourage the policy's internal representations to predict or align with these variables

This makes global constraints **accessible across control layers**, rather than implicit or emergent.

Example 5: Distributed Biasing in a Typical ROS Stack

Minimal Intervention Pattern
Without rewriting existing systems:

1. Add a **Constraint Signal Node** that publishes `/global_constraints`
    - energy
    - thermal
    - traction
    - latency
2. Modify existing controllers to subscribe and modulate parameters such as:
    - maximum velocity
    - maximum joint acceleration
    - impedance stiffness
    - planner horizon
3. Log a single **regulatory cost** scalar to tune for low-demand stability.

This pattern directly translates the paper's design principles into a deployable architecture with minimal disruption.

Summary

Across these examples, the common move is architectural rather than algorithmic:

- introduce slow, shared constraint signals
- bias existing controllers instead of commanding them
- maintain a low-correction operating manifold during nominal conditions

Planning and learning then operate **on top of coherence**, rather than being responsible for creating it.