RESEARCH ARTICLE • EMBEDDED CYBER-PHYSICAL CONTROL & SYSTEM-OF-SYSTEMS ENGINEERING

Supervisory Small Language Models for Hardware Signal Control and System-of-Systems Optimization: A Multi-Tier Adaptive Control, Hyper-Heuristic, and Cross-Layer Protocol Architecture

Domain Classification: Hybrid Adaptive Control Theory • Edge TinyML / Small Specific Language Models (SSLMs) • Hardware-in-the-Loop (HIL) Signal Tokenization • Distributed Energy & Industrial Fieldbus Systems

ABSTRACT

Deploying generative language models within physical instrumentation and embedded control boards presents a fundamental engineering conflict between adaptive, context-aware semantic reasoning and microsecond-level deterministic hardware execution. Rather than utilizing language models as standalone conversational chatbots, this research paper formalizes an architectural paradigm that embeds highly quantized, domain-specific Small Language Models (SLMs, 500M to 3B parameters) directly into hardware state machines, edge microcontrollers (MCUs), and System-on-Chips (SoCs) as supervisory Hybrid Adaptive Controllers. In this framework, the model's domain vocabulary consists of the physical device's own signals—analog and digital inputs, I^2C/SPI registers, pulse-width modulation (PWM) duties, and fieldbus byte streams—translated into discrete symbolic state tokens through a hysteresis-filtered Hardware Translation Matrix (HTM). Structured across a Three-Tier Hierarchical Edge Mesh, the localized Tier 2 SLM ingests continuous hardware telemetry from a bare-metal Tier 1 actuation loop (10–100 kHz) alongside overarching Policy Envelopes broadcast by a Tier 3 System-of-Systems (SoS) Orchestrator. Instead of driving physical pins directly, the SLM functions as a cognitive parameter synthesizer and runtime hyper-heuristic supervisor: it arbitrates between local component self-preservation objectives L(x) and global cluster objectives G(x), dynamically bounds search spaces and fitness weights for local Particle Swarm Optimization (PSO) and classical Proportional-Integral-Derivative (PID) loops, and passes all proposed register maps through a compiled C/C++ Deterministic Guardrail Kernel. Extending this supervisory intelligence across the 7-layer Open Systems Interconnection (OSI) network stack and industrial fieldbuses (Modbus-TCP, CANopen, MQTT) enables cross-layer parametric co-optimization, 80% to 95% semantic payload compression, and autonomous recovery of unmodeled micro-scale physical energy losses across distributed solar and battery storage hardware.

Keywords: Small Specific Language Models (SSLMs); Supervisory Hardware Control; System of Systems (SoS); Hardware Translation Matrix (HTM); Deterministic Guardrail Interlock; Hyper-Heuristic PSO/GA Supervision; Cognitive OSI Stack; Semantic Protocol Gateway.

1. Introduction and Formal Research Hypotheses

1.1 The Paradigm of Embedded Control Language Models

The paradigm of embedded control language models bridges the adaptive, context-aware reasoning of Large Language Models (LLMs) and Small Language Models (SLMs) with the deterministic execution required by physical hardware systems and computational workflows (**Lamaakal et al., 2025**; **Xia et al., 2024**). Rather than deploying models solely as standalone conversational interfaces, this engineering discipline embeds linguistic intelligence directly into software state machines, edge control boards, and autonomous workflows to manage logic and physical actuators dynamically (**Wagner et al., 2024**; **Maher, 2025**).

Classical control architectures—such as Proportional-Integral-Derivative (PID) loops and Model Predictive Control (MPC)—deliver deterministic microsecond execution on bare-metal processors, yet lack the semantic adaptability required to re-mathematize their control laws on the fly when global collaborative constraints shift or when unmodeled physical wear occurs. Conversely, standalone cloud LLMs cannot safely drive hardware actuators due to millisecond-to-second latency variance, severe memory and power constraints at the edge, and the risk of hallucinated register instructions (**Lamaakal et al., 2025**; **Wang et al., 2026**). By training a domain-specific Small Language Model whose native domain language is the physical device's own signals and communication frames, the architecture presented in this paper establishes a safe, verifiable Hardware-in-the-Loop (HIL) supervisory controller operating within a multi-agent System of Systems (SoS).

1.2 Statement of Formal Research Hypotheses

To rigorously evaluate this supervisory hardware control architecture across all operational layers, this study formulates six explicit, falsifiable research hypotheses:

1. **Hypothesis 1 (H_1 — Supervisory Cognitive-Deterministic Decoupling):** A quantized, board-level SLM (500M–1.5B parameters) isolated from direct 10–100 kHz hardware pin actuation by a compiled C/C++ Deterministic Guardrail Kernel can synthesize optimal control parameters (PID coefficients, MPPT tracking regimes, and current limits) at 100 ms–1 s supervisory intervals while guaranteeing zero physical safety boundary violations and instantaneous fallback to classical control upon model fault (**Sha, 2001**; **Wang et al., 2026**).
2. **Hypothesis 2 (H_2 — Dual-Level Local L(x) vs. Global G(x) Arbitration):** Encoding local component self-preservation rules L(x) as primary attention priorities inside the local SLM system prompt while broadcasting global System-of-Systems objectives G(x) as state-constraint Policy Envelopes enables autonomous edge arbitration between local hardware longevity and cluster-wide yield, with seamless degradation to local survival heuristics during communication dropouts.

3. **Hypothesis 3 (H_3 — Signal-as-Language via Hysteresis-Bounded Translation):** Decoupling continuous raw hardware registers into orthogonal Magnitude, Derivative ($\Delta/\Delta t$), and Frequency/Noise (FFT THD) semantic vectors governed by a multi-cycle dwell and deadband hysteresis filter eliminates boundary token thrashing while compressing fieldbus state payloads into 16-bit Hexadecimal identifiers.
4. **Hypothesis 4 (H_4 — Runtime Hyper-Heuristic Supervision of Metaheuristics):** Pairing an on-device SLM as a Runtime Hyper-Heuristic Supervisor over numerical Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) solvers prevents local-optima trapping and accelerates convergence through dynamic solution-space bounding, context-driven fitness weight adaptation, and automated solver switching (**van Stein & Bäck, 2025**; **Narimani & Emami, 2026**).
5. **Hypothesis 5 (H_5 — Cross-Layer Cognitive OSI & Protocol-Gateway Co-Optimization):** Treating the 7-layer OSI stack and heterogeneous OT/IT fieldbuses (Modbus-TCP, CANopen, MQTT) as a unified context window via eBPF kernel hooks enables proactive cross-layer routing ahead of physical EMI spikes, 80% to 95% telemetry compression via CBOR and Intent Tokens, inline detection of cross-layer semantic spoofing, and continuous state tracking via a Semantic Diary State during blackouts (**Soldani et al., 2023**).
6. **Hypothesis 6 (H_6 — Micro-Scale Unmodeled Energy Yield Capture):** Correlating cross-layer hardware signals from Layer 1 micro-amperage, impedance, and harmonic switching noise up to Layer 7 meteorological forecasts allows a distributed solar SLM swarm to harvest energy lost by classical controllers through nano-scale sub-string bypassing, diffuse albedo tracking, predictive thermal-electrical pre-cooling, cooperative row-to-row wake deflection (**Cooper et al., 2025)**, and active inverter EMI cancellation (**Zhang et al., 2021**).

2. Core Generative Control Frameworks and Edge Hardware Integration

2.1 Controller-Embedded Language Model Interactions (CELI)

Conventional software integrations wrap language models inside rigid, hard-coded external execution loops that query the model as a passive text generator. In contrast, **Controller-Embedded Language Model Interactions (CELI)** (**Wagner et al., 2024**) integrates control logic directly within the language model prompt architecture to drive multi-stage task execution autonomously, transferring the control loop over to the model itself:

- **Autonomous State Management:** The language model directly manages the computational state machine, tracking workflow transitions and handling dynamic state adjustments internally without external procedural scaffolding.
- **Arbitrary Function Calling:** The model maps semantic states directly to hardware tool actions, register modifications, or API executions using strictly validated, structured arguments.

- **State-Aware Efficiency Gains:** Advanced CELI implementations employ state-aware optimizations by caching immutable action paths. By freezing known execution trajectories and passing only delta environment state tokens, embedded systems bypass redundant token decoding overhead and dramatically accelerate local step-by-step decision processing (**Wagner et al., 2024**).

2.2 Embedded Language Flows (ELF)

Operating at the structural level of generative modeling, frameworks such as **Embedded Language Flows (ELF)** (**Hu et al., 2026**) process control instructions through continuous-time diffusion flow matching rather than traditional discrete token-by-token generation. Standard autoregressive decoding repeatedly projects continuous latent representations onto discrete vocabulary tokens at every intermediate reasoning step, introducing quantization noise and latency when interfacing with continuous physical systems.

ELF keeps logic processing entirely within a continuous embedding space across normalized time $t \in [0, 1]$, delaying the expensive overhead of converting representations back into discrete text tokens until the final execution boundary. This continuous flow trajectory provides a smooth mathematical transition for incorporating real-time feedback loops from physical sensors or internal control variables directly into the generative trajectory (**Hu et al., 2026**).

2.3 Scaling to Edge Hardware, Compression Mathematics, and Cascaded Control Loops

When scaling language models down to microcontrollers (MCUs) or System-on-Chips (SoCs) for physical machinery control—such as Human-Machine Interfaces (HMIs), robotics, or industrial sensor boards—architectures shift toward highly optimized Small and Tiny Language Models (**Lamaakal et al., 2025**; **Scherer et al., 2024**). Fitting a transformer model (**Vaswani et al., 2017**) onto edge silicon relies on four mathematical optimization pillars surveyed by **Lamaakal et al. (2025)**:

- **Knowledge Distillation:** Transfers domain reasoning from a large teacher model M_t to a compact edge student model M_s (**Hinton et al., 2015**) by minimizing a composite objective combining task cross-entropy L_{task} and Kullback-Leibler divergence $KL(P_t \parallel P_s)$ at temperature T with balancing factor α:

$$L_{total} = (1 - \alpha) \cdot L_{task} + \alpha \cdot T^2 \cdot KL(P_t \parallel P_s), \quad \text{where } P_t = \text{softmax}(z_t / T), \quad P_s = \text{softmax}(z_s / T)$$

- **Fixed-Point and Low-Bitwidth Quantization (INT4 / INT8):** Reduces parameter and activation precision from 32-bit floating-point (FP32) to 4-bit or 8-bit integers and fixed-point representations (x = integer part + fractional part / 2^n), achieving >50% memory reduction and >2× inference acceleration with minimal accuracy trade-off (**Lamaakal et al., 2025**; **Scherer et al., 2024**).

- **Structured and Magnitude Pruning:** Eliminates redundant weights, attention heads, or channels below a sensitivity threshold τ (setting $w_p = 0$ if $|w| < \tau$ and $w_p = w$ otherwise) to yield sparse, low-latency execution graphs.
- **Memory-Efficient Attention (GQA and MLA):** Replaces standard Multi-Head Attention with Grouped-Query Attention (**Ainslie et al., 2023**) or Multi-Head Latent Attention, sharing key-value caches across query groups and applying Rotary Positional Embeddings (RoPE, $x \cdot e^{i\theta}$) and RMSNorm to bound runtime SRAM consumption on edge NPUs (**Lamaakal et al., 2025**).

Feature / Metric	On-Device TinyML / Edge SLMs	Cloud-Augmented Foundation Models
Model Size	Typically 500M to 3B parameters (down to <1B nano-models)	Hundreds of billions of parameters (e.g., 175B–540B)
Hardware Targets	ARM Cortex-M MCUs, ESP32, STM32MP1, Raspberry Pi 5, or local NPUs	Massive cloud-hosted GPU and TPU clusters
Hardware Constraints Addressed	Drastically limited DRAM (fitting within 4–8 GB down to MB SRAM scales)	High computing power, distributed VRAM, high power consumption
Optimization Techniques	INT4/INT8 Quantization, structured pruning, and knowledge distillation	None required on the client edge
Core Operational Benefits	Zero-latency variance, strict user privacy, offline capability, low power	State-of-the-art broad multi-domain reasoning capabilities

To maximize resource efficiency across physical installations, modern edge systems deploy **Cascaded Control Loops**. Nano-scale models (<1B parameters, such as Qwen2-0.5B, MobiLlama-0.5B, SmolLM-135M/360M, or OpenELM-270M; **Lamaakal et al., 2025**) reside locally on-device to handle rapid sensory filtering, immediate safety heuristics, and routine token tasks. When a complex out-of-distribution exception occurs or a nuanced system-wide diagnostic is required, the localized agent cascades the operational data up to an edge server or cloud-augmented SLM to resolve high-level planning.

2.4 Development and Hardware-in-the-Loop Benchmarking Ecosystems

Building software agents that synthesize register configurations or compile code for hardware systems introduces strict electrical and timing variables that standard code completion benchmarks do not measure (**Yuan et al., 2026**):

- **Closed-Loop Virtual Hardware Testing:** Frameworks utilize virtual hardware sandboxes such as **Wokwi** to dynamically verify embedded agent outputs. This allows an embedded control agent to flash compiled firmware onto a simulated microcontroller, monitor physical GPIO and PWM pin states, receive closed-loop runtime feedback, and self-correct execution bugs in real time prior to physical deployment (**Yuan et al., 2026**).

- **Multimodal Hardware Copilots:** Benchmarks such as **EmbedCopilot-Bench** (**Yuan et al., 2026**) utilize vision-language models to evaluate hardware-aware development across physical circuit layouts, board schematics, and component register configurations to ensure deterministic firmware execution.

3. Control Theory, Multi-Agent Systems, and Edge AI: The Supervisory Hybrid Adaptive Controller

3.1 Architectural Topology: Three-Tier Hierarchical Edge Mesh

This architectural framework sits at the intersection of Control Theory, Multi-Agent Systems, and Edge AI, functioning as a **Supervisory Hybrid Adaptive Controller**. Rather than deploying a broad-reasoning LLM, the system integrates a domain-specific SLM directly into a Hardware-in-the-Loop (HIL) configuration across a **Three-Tier System of Systems (SoS) Topology**, localizing intelligence to minimize data transit latency and eliminate single points of failure.



- **Tier 1 — Deterministic Hardware Actuation Layer (10–100 kHz):** Executes on bare-metal or Real-Time Operating System (RTOS) microcontrollers (e.g., STM32, Texas Instruments C2000 DSPs). Tier 1 drives high-speed, continuous feedback loops—including MPPT tracking, battery cell balancing, and inverter phase-matching—at 10 to 100 kHz. It does not run the SLM; it strictly consumes static numerical parameters and writes directly to physical hardware pins (PWM, DAC, GPIO).

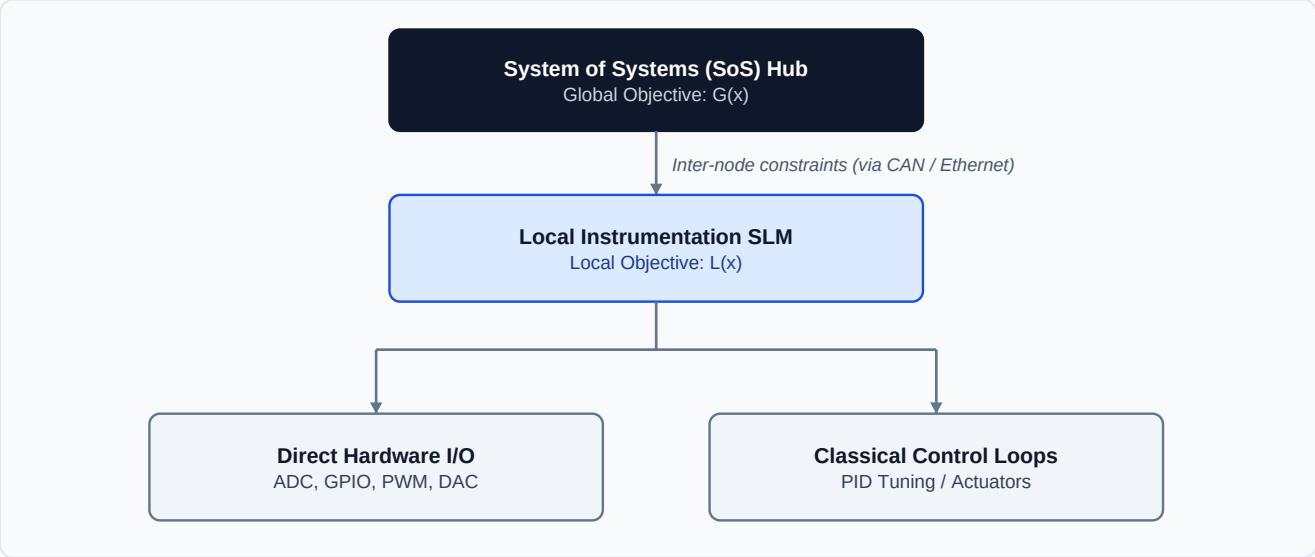
- **Tier 2 — Cognitive Node Optimization Layer (100 ms–1 s):** Executes on application processors with integrated Neural Processing Units (NPUs) running a real-time Linux variant (e.g., NXP i.MX8M Plus, STM32MP1, Raspberry Pi Compute Module 4, or specialized edge TPUs). Tier 2 hosts the localized, heavily quantized SLM (500M to 1.5B parameters, INT4 quantization). It samples Tier 1 telemetry every 100 ms to 1 s, processes global SoS instructions, and refines Tier 1 operational parameter targets.
- **Tier 3 — Farm-Level Orchestrator Layer (Multi-Second Horizon):** Executes on an on-site industrial PC, PLC, or high-availability edge server. Tier 3 manages the global system objective function G(x) (such as optimizing total farm yield versus battery degradation rate under real-time grid pricing), aggregating low-bandwidth state tokens from Tier 2 nodes and broadcasting overarching coordination Policy Envelopes.

3.2 Data Ingestion: Hardware I/O as Token Strings and Dual-Level Objective Functions

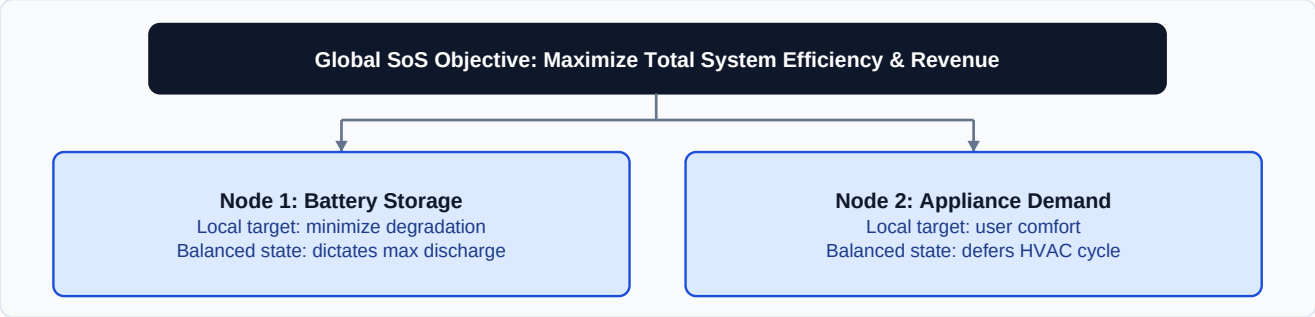
To treat a physical control board as a tokenized sequence, the hardware's raw I/O (analog/digital inputs, I^2C/SPI registers, PWM duties) undergoes a continuous-to-discrete translation layer. Raw sensor readings are normalized and mapped to structural state tokens (e.g., [TEMP-CRIT-104C], [FLOW-RATE-DEGRADED]), and the model treats the immediate history of hardware I/O registers as a sliding context window buffer:

```
[System-Tick: 1042] -> [IO-In: Vin=4.2V, Iout=1.8A] -> [Local-Obj: Maximize-
Efficiency] -> [SoS-Obj: Node-Power-Cap=10W]
```

To manage local board parameters while satisfying a System of Systems (SoS) goal, the architecture utilizes a **Hierarchical Multi-Agent Optimization** strategy that balances two distinct objective functions:

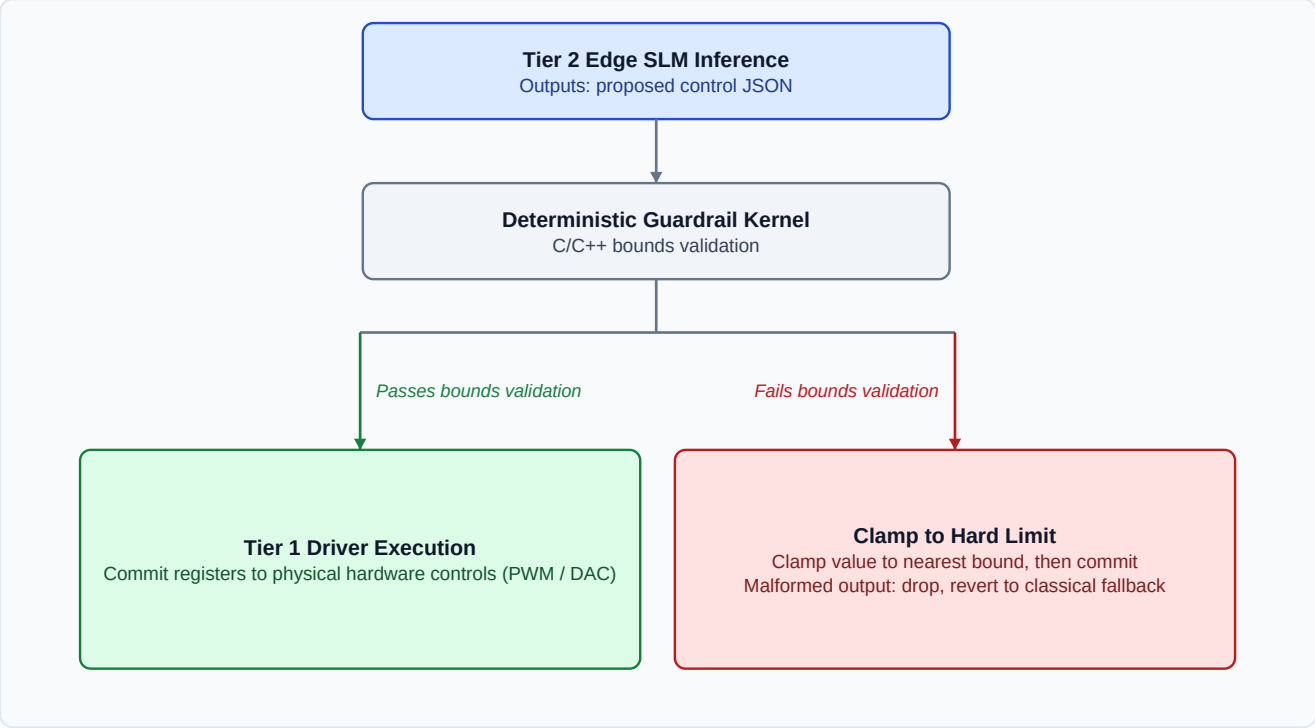


- **Local Objective Function L(x):** Hardcoded as primary attention priorities inside the local Tier 2 SLM system prompt. L(x) directly manages local board components to guarantee stability, minimize mechanical or thermal wear, and satisfy immediate operational goals (e.g., maximizing pump throughput or preventing local thermal throttling).
- **Global SoS Objective Function G(x):** Formulated mathematically at the Tier 3 Orchestrator based on external criteria (grid market pricing, load forecasting, weather prediction). The SoS hub translates G(x) into **Policy Envelopes** expressed as state constraint tokens—such as [MAX-BATTERY-RESERVE-DISCHARGE] or [LOAD-SHED-PRIORITY-HIGH]—and broadcasts them over industrial fieldbuses (CAN, EtherCAT, Modbus-TCP, or ROS2).
- **Arbitration via Token Priorities:** When local component preservation conflicts with global cluster demand, the local SLM arbitrates by evaluating its physical health tokens against the incoming Global Policy Envelopes. If a battery storage node reads [CELL-TEMP-CRIT], its hardcoded local safety instruction weight overrides a global [AGGRESSIVE-DISCHARGE] request, forcing the model to synthesize a [SAFE-MODE-CURTAIL] parameter configuration.



3.3 Execution Pipeline and Deterministic Hardware Guardrails (The Interlock)

Because a generative model cannot safely drive physical actuators at microsecond speeds due to latency variance and potential parameter hallucinations, the board executes a four-stage **Hybrid Analytical-Cognitive Loop** isolated from physical hardware pins by a compiled C/C++ **Deterministic Guardrail Kernel** (Sha, 2001; Mohan et al., 2013; Nesti et al., 2026):



1. **State Assessment:** The Tier 2 SLM reads the current sliding window of tokenized hardware I/O registers alongside active Tier 3 SoS objective constraints.
2. **Parameter Synthesis (The Cognitive Step):** Rather than calculating raw pin voltages directly, the SLM analyzes the system state and synthesizes optimal gains and structural parameters for classical, deterministic Tier 1 controllers—such as altering the coefficients (K_p, K_i, K_d) of a PID controller, selecting an MPPT algorithm, or modifying Lyapunov-based adaptive gains (**Maher, 2025**; **Wang et al., 2026**; **Bosio et al., 2025**).
3. **Execution Guardrails (The Deterministic Step):** The synthesized JSON register map is passed through a compiled, hardcoded C/C++ verification matrix (or bounded SMT verifier) executing on the processor edge (**Sha, 2001**; **Nesti et al., 2026**). This interlock inspects every proposed parameter against strict physical reality limits (e.g., ensuring a mechanical tracker angle never exceeds limit switches and current limits never exceed wire-gauge ampacity).
4. **Hardware Execution or Exception Fallback:** Validated parameters are written to Tier 1 controller registers to drive physical I/O pins. If the SLM hallucinates an out-of-bounds value, the C/C++ kernel clamps it to the nearest hard limit; if it emits corrupted syntax, the kernel drops the packet entirely, sets an [SLM-FAULT] register token, and falls back to a safe, classical mathematical controller (such as standard proportional control) until the next inference window.

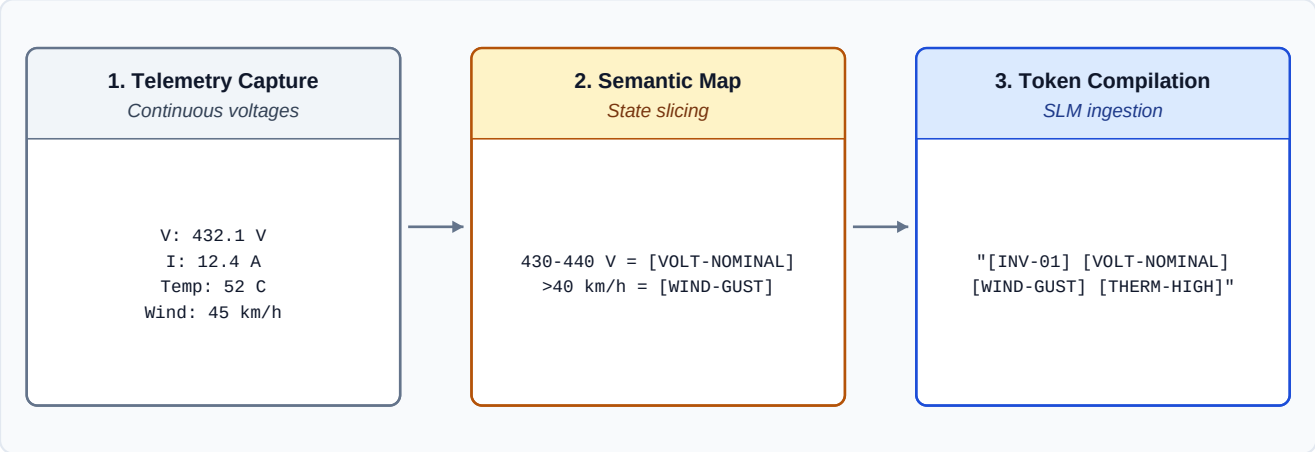
Control Dimension	Language-Model-Driven Supervisory Control	Traditional Control (MPC / PID)
Complex SoS Adaptability	**High:** Translates vague or changing collaborative objectives into functional parameters on the fly.	**Low:** Requires heavy re-mathematization if global constraints change drastically.

Control Dimension	Language-Model-Driven Supervisory Control	Traditional Control (MPC / PID)
Handling Unmodeled Dynamics	**High:** Uses structural semantic reasoning to debug and work around unforeseen sensor faults or physical wear.	**Medium:** Relies heavily on precise, predetermined mathematical models of the physical world.
Execution Latency	**Varying / Slower:** Milliseconds to seconds (supervisory cadence on edge NPU hardware).	**Deterministic / Microseconds:** Fast, predictable execution loops directly on the bare-metal processor loop.

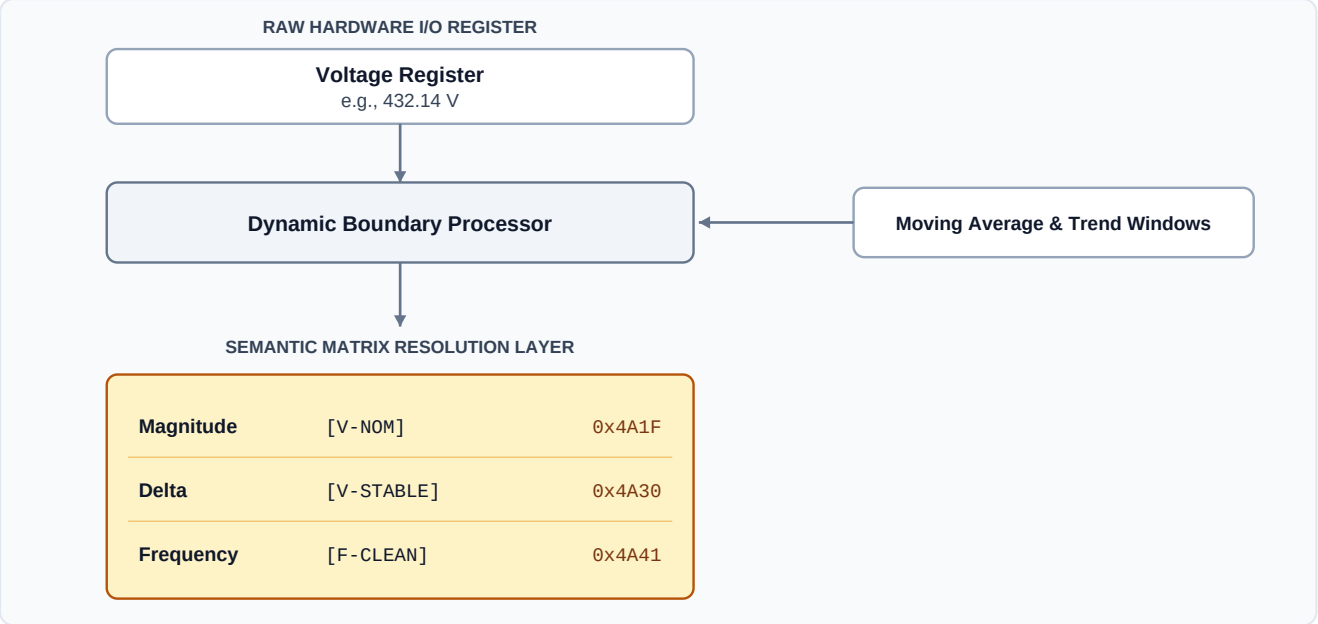
4. Hardware Translation Matrix (HTM), Control-State ETL Pipeline, and Metaheuristics

4.1 Hardware Translation Matrix (HTM) Design Specifications

Because hardware boards communicate in continuous analog voltages and language models operate on discrete symbolic tokens, the **Hardware Translation Matrix (HTM)** executes as a deterministic firmware layer inside the Tier 2 processor. Acting as an ADC-to-Semantic encoder, the HTM maps continuous high-rate physical signals from an onboard Ring Buffer into structured token arrays.



To capture complete signal dynamics, the HTM's Dynamic Boundary Processor decouples each raw hardware register into three orthogonal linguistic vectors—**Magnitude**, **Delta / Derivative**, and **Frequency / Noise Profile**—and assigns each discrete token a unique 16-bit Hexadecimal Identifier for compact binary serialization across fieldbus wires:

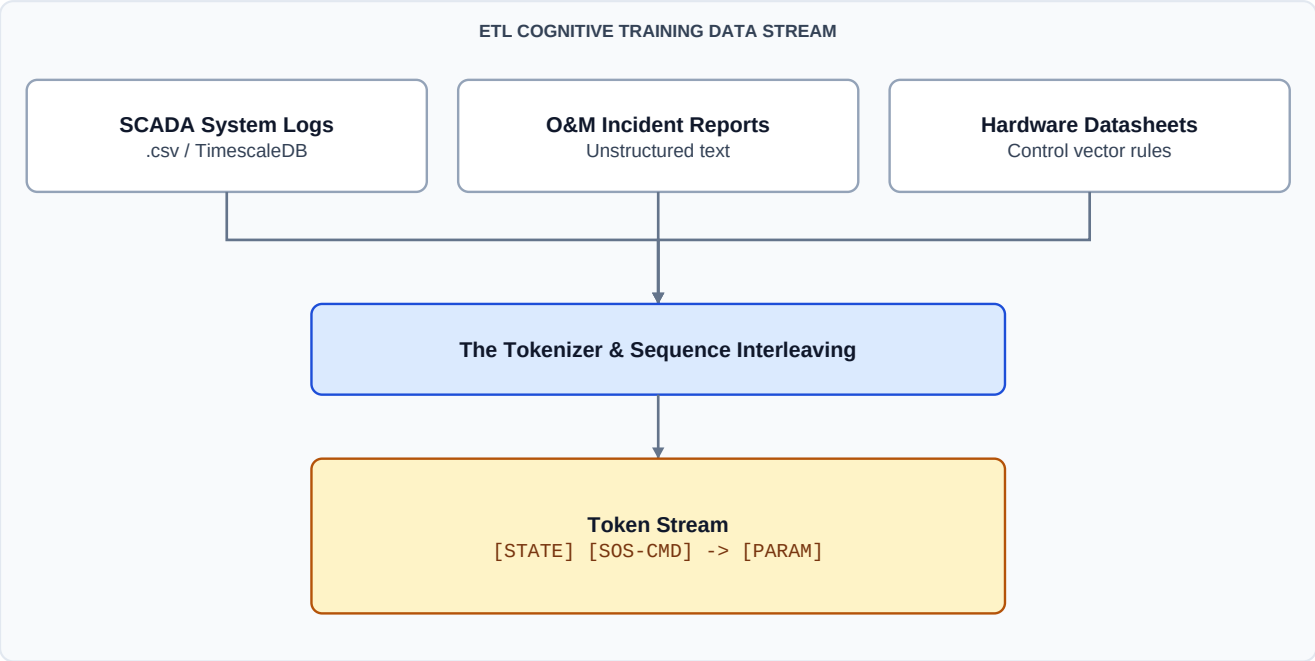


Physical Parameter	Raw Range / Dynamic	Signal Component	Code / Hex ID	Output Semantic Token & Threshold Condition
DC Bus Voltage	0.0 V – 600.0 V	Magnitude	0x4A1E 0x4A1F 0x4A20	[V-UNDER-SAG] (< 380 V) [V-NOM] (380 V – 480 V) [V-OVER-SURGE] (> 480 V)
DC Bus Voltage	ΔV / Δt	Delta / Derivative	0x4A30 0x4A31 0x4A32	[V-STABLE] (± 0.5 V/sec) [V-SPIKE-POS] (> 5.0 V/sec) [V-DROP-NEG] (< −5.0 V/sec)
DC Bus Voltage	FFT Ripple (10 kHz)	Noise Profile	0x4A41 0x4A42	[F-CLEAN] (THD < 3%) [F-HARMONIC-DIST] (THD ≥ 3%)
Battery Cell Temp	−20 °C – 80 °C	Magnitude	0x5B12 0x5B13 0x5B14	[TEMP-COLD] (< 5 °C) [TEMP-NOM] (5 °C – 45 °C) [TEMP-CRIT-HIGH] (> 45 °C)
Battery Cell Temp	ΔT / Δt	Delta / Derivative	0x5B25 0x5B26	[THERM-EQUILIBRIUM] (≤ 0.1 °C/min) [THERM-RUNAWAY-RISK] (> 2.0 °C/min)

To prevent **token thrashing**—where a boundary signal oscillating between 429.9 V and 430.1 V floods the SLM context window with alternating state tokens—the HTM enforces a **Hysteresis Filter Loop**. To enter a critical state token such as [TEMP-CRIT-HIGH] (0x5B14), the temperature register must cross 45.0 °C and remain there for ≥ 3 consecutive 100 ms sample cycles (300 ms dwell persistence). To exit back to [TEMP-NOM] (0x5B13), the temperature must fall below 43.5 °C, enforcing a 1.5 °C deadband hysteresis zone. Each compiled payload sent into the SLM inference loop adheres to the strict structural syntax: [SYSTEM-CONTEXT] [GLOBAL-OBJECTIVE-TOKEN] [LOCAL-TELEMETRY-TOKENS] → [INFERENCE-TARGET].

4.2 ETL Data Pipeline and Control-State Training Data Taxonomy

Training an on-device SLM for hardware supervision shifts the corpus paradigm from generic web text to specialized **Control-State Engineering Textures** that teach causal linkages between sensory signatures, global SoS commands, and safe control parameters:



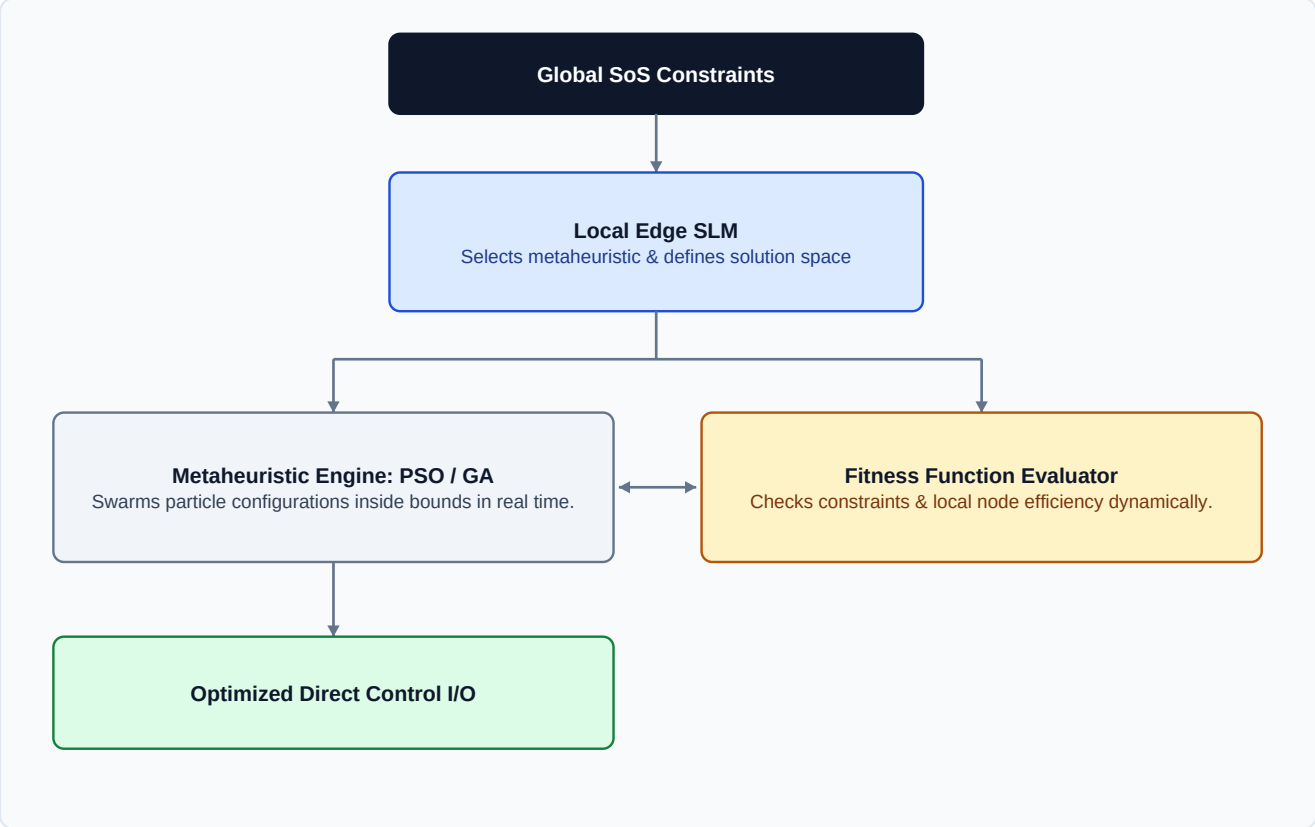
- **Pillar 1 — Industrial Telemetry Interleaving (Time-Series Structured Data):** Captured from SCADA archives, historian databases (TimescaleDB), and inverter logs, where raw floating-point time-series matrices are converted by the HTM into chronological state-token transitions.
- **Pillar 2 — Hardware Domain Context (Static Deterministic Data):** Component PDF datasheets, battery chemistry thermal manuals, manufacturer firmware register maps, and IEEE 1547 compliance protocols parsed into structured markdown rule sets.
- **Pillar 3 — Operator-in-the-Loop Interventions (Unstructured Field Text):** Maintenance logs, manual override tickets, and diagnostic field records that capture human contextual troubleshooting (e.g., retarding DC bus engagement by 15 minutes when Inverter 4 triggers a high-side leakage current spike under morning dew conditions).

During instruction tuning, the interleaved corpus is formatted into a four-block causal reasoning structure that compels the SLM to articulate an explicit physical justification before emitting target register parameters:

```
### [STATE]
[NODE: INV-04] [V-NOM] [V-STABLE] [TEMP-CRIT-HIGH] [THERM-RUNAWAY-RISK] [GRID-
FREQ: 60.1Hz]
### [SOS-CONTEXT]
[SOS-PRIORITY: CAPACITY-MAXIMIZATION] [MARKET-PRICE: PEAK-SURGE] [THERM-CEILING-
ALLOWED: TRUE]
### [COGNITIVE-REASONING-CHAIN]
Local inverter node is approaching high thermal limits. However, the global system
constraint dictates a high-value generation spike window with an authorized
elevated temperature ceiling. The rate of change of temperature is steep,
requiring localized mitigation without hard shutdown. Increasing internal fan duty
cycle to maximum and marginally down-shifting the gate-drive switching frequency
from 16kHz to 8kHz reduces switching losses, stabilizing temperatures while
maintaining 94% power injection throughput.
### [TARGET-PARAMETERS-SYNTHESIS]
{ "fan-duty-pct": 100, "switching-freq-hz": 8000, "mppt-tracking-mode":
"CONSERVATIVE" }
```

4.3 Metaheuristics Integration and Runtime Hyper-Heuristic Supervision

An SLM alone cannot reliably optimize a non-linear, multi-variable equation directly through raw token generation. Instead, the architecture pairs the SLM's high-level contextual pattern recognition with fast-converging numerical **Metaheuristics** executing on the node (**van Stein & Bäck, 2025**; **Narimani & Emami, 2026**):



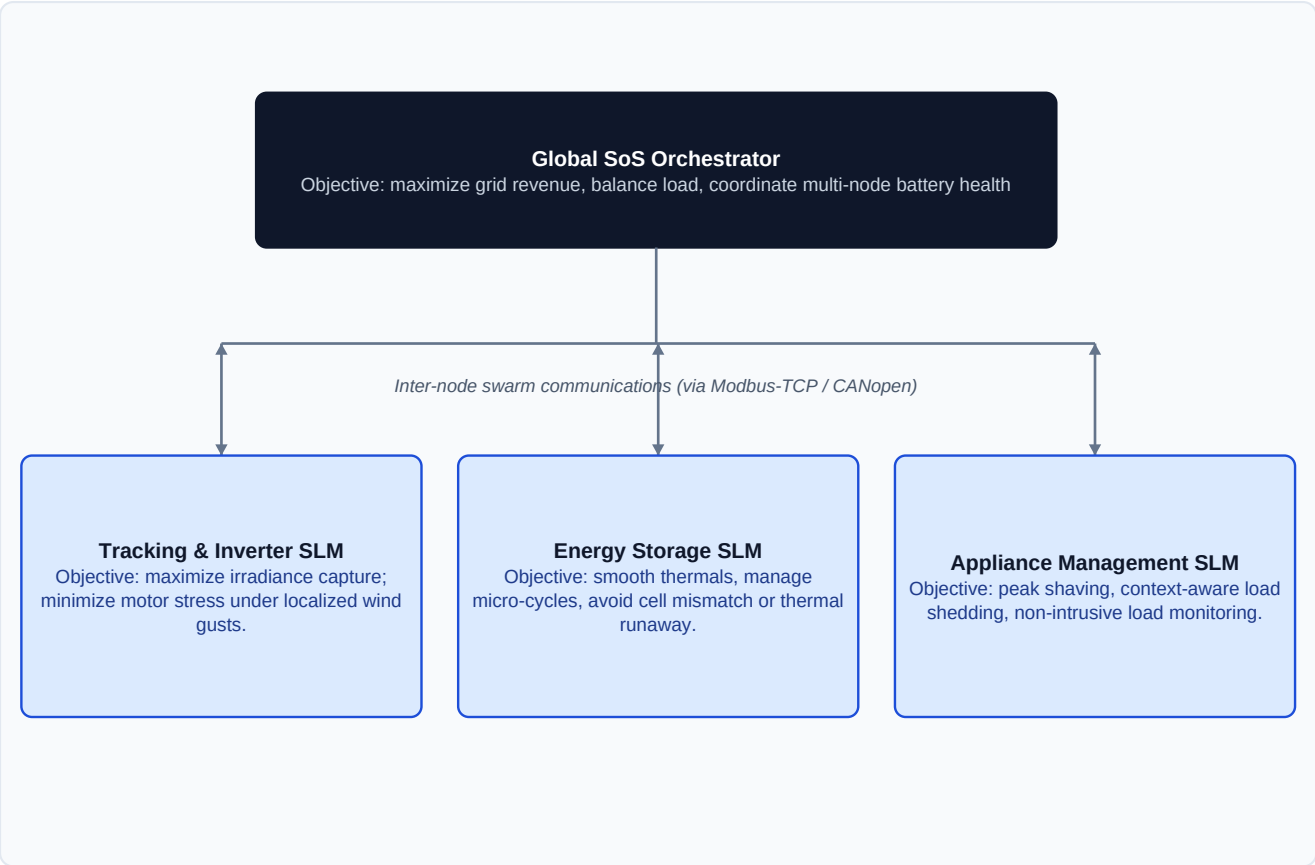
Particle Swarm Optimization (PSO) executes within individual Tier 2 nodes for continuous parameter tuning (such as tracking down a multi-peaked global maximum power point on a partially shaded string of solar panels during complex cloud transients), while **Genetic Algorithms (GA)** execute on the Tier 3 Orchestrator across the entire farm matrix for discrete-event scheduling optimization (such as arranging battery discharge sequencing across 50 nodes over a 24-hour horizon). Rather than performing raw mathematical iterations, the SLM acts as the **Runtime Meta-Heuristic Supervisor (Hyper-Heuristic Layer)** (**van Stein & Bäck, 2025**):

1. **Dynamic Solution Space Bounding:** Standard PSO frequently suffers from localized tracking trapping or excessive computational cycles. When the SLM reads [CLOUD-SHADING-FRAGMENTED], it instantly restricts the PSO search vector space to a narrow angular band (e.g., capping the tracker tilt search boundary between 12° and 18° rather than sweeping the full 90° physical horizon).
2. **Fitness Function Adaptation:** The SLM dynamically modifies the penalty weights of the active fitness function based on incoming Global SoS tokens. When the SoS token is [MODE: BAT-CONSERVE], the SLM updates the fitness weights to aggressively penalize battery temperature spikes during operational target searches (**Narimani & Emami, 2026**).
3. **Algorithmic Switching:** If the active PSO routine fails to converge within a specified timeout token window, the SLM halts the PSO thread and initializes an alternate recovery algorithm—such as a Simulated Annealing cooling loop—to reset the hardware system smoothly.

5. Distributed Solar Farm Implementation and Micro-Scale Energy Capture Paradigms

5.1 Distributed Edge-to-Swarm Solar Architecture and Node Agent Implementation

Implemented across a distributed solar farm, each hardware node runs a tailored, quantized SLM (500M to 1.5B parameters) flashed onto an edge processor (ARM Cortex-M7, STM32MP1, or local NPU module), coordinating three specialized node classes under the Global SoS Orchestrator:



- **Tracking & Inverter Nodes (Local Objective L_1):** Monitor anemometer voltage, motor amperage feedback, accelerometer tilt angles, panel V_{mp}, panel I_{mp}, and ambient temperature. Under localized wind gusts causing structural motor stress, the local SLM identifies the high-amperage draw and autonomously stows the array early, bypassing centralized cloud delays. Under downstream battery saturation, it slightly detunes the inverter off the true MPPT peak to shed excess generation dynamically.
- **Energy Storage BMS Nodes (Local Objective L_2):** Monitor individual cell voltages via SPI, thermistor arrays, State of Charge (SoC), State of Health (SoH), and coolant pump PWM status to minimize thermal degradation and balance cells during fast charging. When the SoS hub detects an impending cloud cover event or load spike, the Storage SLM executes preparatory micro-discharges to match current demand without dropping bus voltage below stability limits.
- **Appliance & End-User Nodes (Local Objective L_3):** Monitor smart-meter CT clamp currents, ambient humidity, thermostat setpoints, and occupancy sensors to shift heavy inductive loads (HVAC compressors, water heaters, EV chargers) into peak solar windows and modulate duty cycles upon receiving [GRID-STRAIN] tokens.

During each 10–100 ms system tick, the node executes State Tokenization ([NODEID: BAT04] [SOC: 89.2%] [TEMP: 42.1C] [TREND: STEEP-RISE]), Context Fusion ([GLOBALSOS: DEMANDSURGE] [PRICE-SIGNAL: HIGH]), Cognitive Parameter Synthesis, and C/C++ Deterministic Guardrail validation, as structured in the following node control loop pattern:

```
class SolarFarmEdgeNodeAgent:
    def initNode(self, nodeType, hardwareIoInterface):
        self.nodeType = nodeType            # e.g., "Inverter-Node-07"
        self.io = hardwareIoInterface
        self.slmEngine = LocalTinyMLEngine() # Quantized on-device 1B SLM

    def executeControlLoop(self, globalSosConstraints):
        # 1. Read immediate physical hardware registers
        rawTelemetry = self.io.readRegisters()

        # 2. Tokenize continuous signals into discrete text state representations
        localStateTokens = self.tokenizeHardwareSignals(rawTelemetry)

        # 3. Fuse local health with the SoS network objective
        promptContext = f"""
        [Role: {self.nodeType} Hardware Controller]
        [Local Hardware State: {localStateTokens}]
        [Global SoS Objective: {globalSosConstraints}]

        Task: Synthesize local control parameters. Maintain cell safety above all
else.
        Optimize local efficiency while satisfying Global SoS request boundaries.
        Return strictly in structured register format.
        """

        # 4. Generate parameter changes via localized model inference
        synthesizedParameters = self.slmEngine.generateParameters(promptContext)

        # 5. Deterministic Guardrail Check (Hardware Interlock)
        validatedParameters =
self.applyHardcodedSafetyFilters(synthesizedParameters)

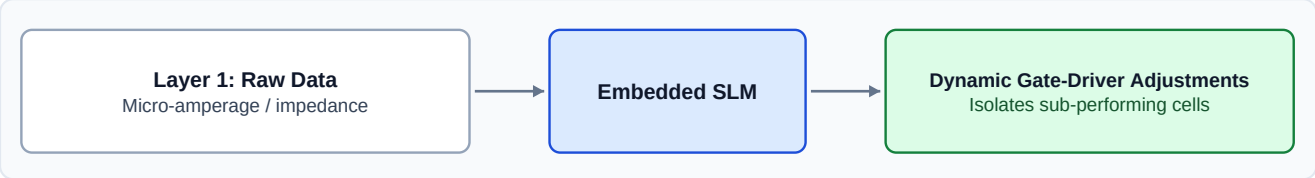
        # 6. Push to physical I/O registers (DAC, PWM, Modbus registers)
        self.io.writeRegisters(validatedParameters)

    def applyHardcodedSafetyFilters(self, params):
        # Ensuring the SLM cannot hallucinate an over-voltage or over-current
state
        if params["chargeCurrentLimitA"] > 120.0:
            params["chargeCurrentLimitA"] = 120.0 # Hard hardware clamp
        return params
```

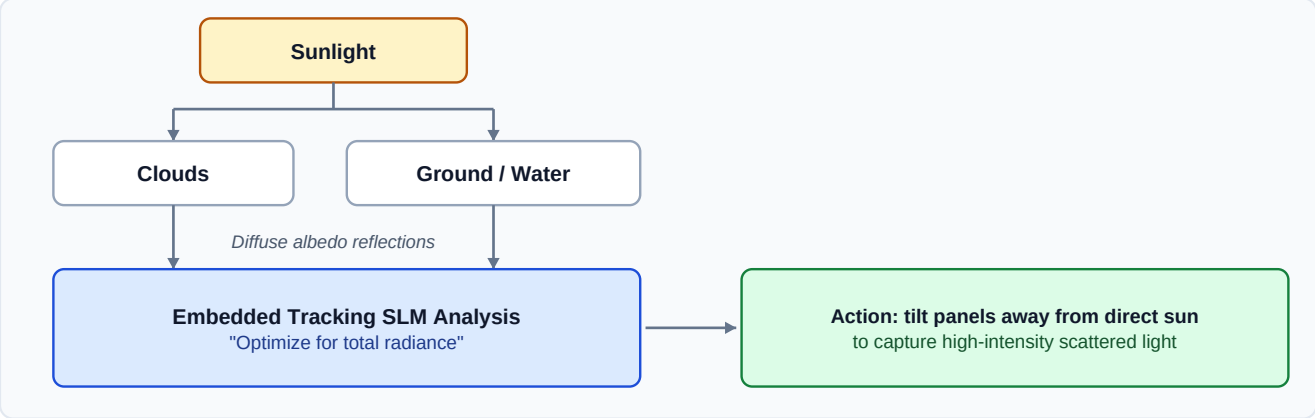
5.2 Five Micro-Scale Physical Paradigms for Solar Energy Capture

Because an embedded, cross-layer language model reads the entire hardware stack—from physical material variations up to cloud meteorological data—as a unified semantic sequence, it resolves five unmodeled, micro-scale physical inefficiencies that classical physics equations cannot optimize in real time:

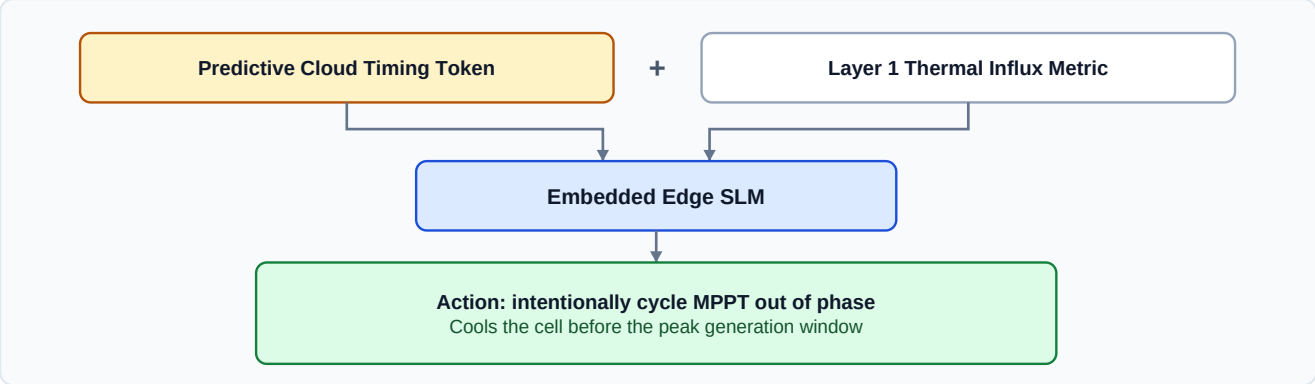
1. Nano-Scale Material and Cell Degradation Optimization: Standard panel technologies treat a solar string as a uniform electrical component, even though individual cells develop microscopic defects, localized micro-cracks, and uneven dust soiling. By monitoring micro-amperage variations and localized physical impedance directly at the Layer 1 physical interface, the SLM detects sub-surface cell degradation long before total panel failure occurs and commands smart junction boxes to dynamically reconfigure series-parallel sub-string topologies, isolating bottlenecked cells to recover hidden power yield.



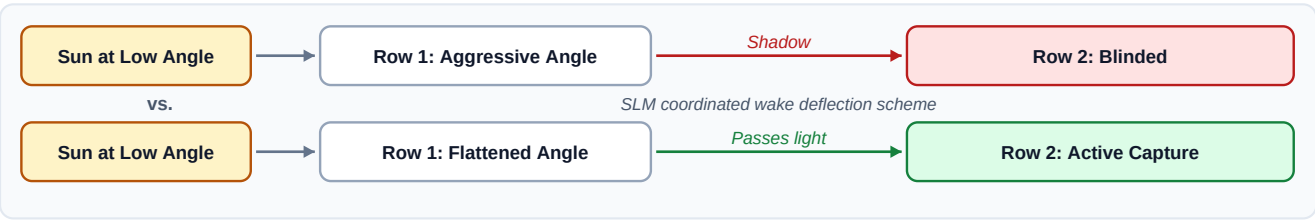
2. Cognitive Albedo and Fragmented Micro-Cloud Tracking: Classical solar trackers angle panels strictly toward the astronomical sun position, even when broken cloud cover obscures direct beam irradiance. Acting as a localized vision-and-telemetry correlator that fuses sky-imaging cameras with real-time panel voltage ripples, the tracking SLM recognizes when diffuse light bouncing off clouds, neighboring panels, or ground/water surfaces (albedo effect) exceeds direct sun intensity, tilting trackers slightly away from the obscured sun to harvest high-radiance scattered photons.



3. Predictive Thermal-Electrical Co-Optimization: Solar cells exhibit a sharp negative temperature coefficient where intense sunlight heats the panel and degrades conversion efficiency. By correlating incoming Layer 7 meteorological timing tokens (e.g., an impending 10-minute clear-sky opening) with Layer 1 thermal influx measurements, the SLM intentionally modulates the inverter switching frequency or cycles the MPPT point slightly out of phase just before the peak sunlight window arrives. Temporarily lowering electrical current cools the panel so that it ingests peak sunlight at a substantially higher thermal-electrical efficiency rate.



4. Swarm-Level Wake Deflection (Cooperative Row-to-Row Shading): During early morning and late afternoon low sun angles, aggressive tracking causes front rows to cast long shadows that blind rear rows. Treating shadow casting like aerodynamic wake in wind farms, the SoS cross-layer model instructs front rows to deliberately flatten their tilt angle—sacrificing a small fraction of immediate individual power to clear the shadow line for rear rows and increase net farm yield (**Cooper et al., 2025**).

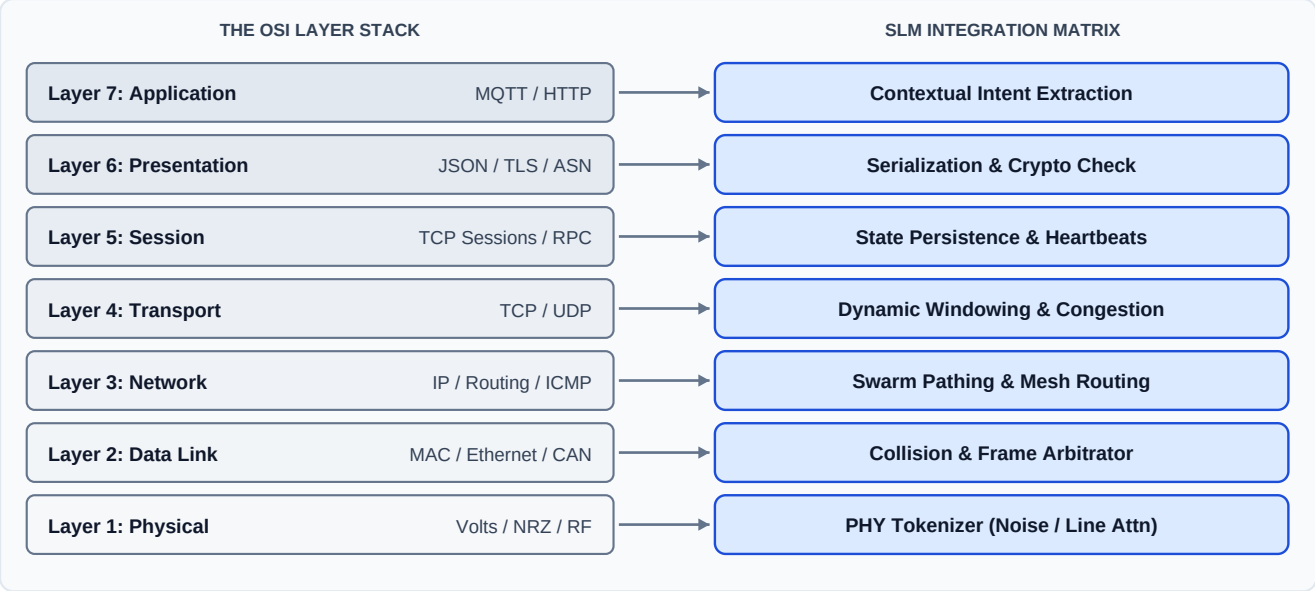


5. Electromagnetic Noise Harvesting: High-speed Insulated-Gate Bipolar Transistors (IGBTs) switching DC into AC generate substantial high-frequency electromagnetic interference (EMI) and harmonic distortion traditionally wasted as heat. By sampling high-frequency register loops at Layers 1 and 2 to map the exact rhythmic timing of each inverter's switching noise, the swarm SLMs dynamically shift active switching frequencies across multiple inverters so their noise profiles cancel or constructively align with passive capacitor harvesting loops, feeding recovered high-frequency switching energy back into the internal DC bus (**Zhang et al., 2021**).

6. Deep-Stack Cognitive OSI Model and Communication Protocol-Specific SLM

6.1 Structural Design of the 7-Layer Cognitive OSI Stack

Rather than treating the 7 layers of the Open Systems Interconnection (OSI) model as isolated software abstractions, the **OSI Model-Specific Language Model (OSI-SLM)** treats the entire 7-layer stack as a unified multi-tier context window, intercepting payloads and metadata at each layer boundary through high-efficiency kernel hooks and Extended Berkeley Packet Filters (eBPF) (**Soldani et al., 2023**):



- **Lower-Stack Optimization (Layers 1–3):** Layer 1 reads physical transceiver voltages, RSSI, bit error rates, and cable attenuation ([PHY-SIGNAL-DEGRADED]); Layer 2 monitors frame collisions and CRC errors; Layer 3 handles IP/mesh path costs. When Layer 1 detects transient EMI ([PHY-EMI-SPIKE]) from an inverter switching on, rather than waiting for packet drops and Layer 4 TCP retransmissions, the OSI-SLM correlates [PHY-EMI-SPIKE] with [L3-ROUTING-TABLE] and proactively reroutes Layer 3 packets through an alternate physical node path before the TCP window breaks.
- **Mid-Stack Optimization (Layers 4–5):** When 50 battery nodes attempt to transmit telemetry simultaneously during sudden cloud cover, the SLM dynamically adjusts Layer 4 congestion windows based on Layer 7 payload criticality—prioritizing emergency thermal alerts over routine historical log frames rather than applying blind exponential backoff.
- **Upper-Stack Optimization (Layers 6–7):** Matching Layer 7 domain intent directly to Layer 6 presentation, the SLM switches serialization dynamically from verbose text-based JSON to high-compression binary CBOR during stable weather or cellular fallback operations, reducing payload sizes by up to 80%.

To evaluate cross-layer dependencies concurrently, the OSI-SLM interleaves telemetry from all 7 layers into a unified sequence:

```
### [OSI-STACK-TELEMETRY]
[L1: SNR-12dB] [L1: ATTN-HIGH]
[L2: COLLISION-RATE: 4%] [L2: PROTOCOL: ETHERNET]
[L3: ROUTE-COST-MIN] [L3: IP-SRC: 10.0.1.45]
[L4: TCP-WINDOW-SIZE: 1460] [L4: RETRANSMIT-DETECTED]
[L5: SESSION-STATE: INTERMITTENT]
[L6: ENCODING: RAW-JSON]
[L7: APP-PAYLOAD: "{"inverterTemp": "95C", "status": "OVERHEATING"}"]

### [COGNITIVE-STACK-ANALYSIS]
Layer 1 indicates high line attenuation and low SNR, causing a 4% collision rate
at Layer 2 and triggering TCP retransmissions at Layer 4. The Layer 7 payload
contains a critical overheating alert that must reach the global SoS orchestrator
immediately. Due to the high packet drop rate of the current TCP session, the
system must bypass standard serialization.
Action: Translate Layer 7 payload to compact binary encoding at Layer 6, drop the
Layer 4 stateful TCP connection, and force an unacknowledged Layer 4 UDP broadcast
directly through Layer 3 alternate interface (10.0.2.1) to ensure swift delivery.

### [STACK-CONTROL-OUTPUT]
{ "L6-reformat": "CBOR", "L4-protocol-shift": "UDP", "L3-route-target": "10.0.2.1"
}
```

6.2 Zero-Trust OT Security, Semantic Compression, and Multi-Layer Co-Optimization

Maintaining contextual visibility across all 7 layers enables inline **Zero-Trust Intrusion Detection (IDS)** for Operational Technology. If a malicious actor gains physical access to a tracker cable and injects valid Layer 7 Modbus commands ([MOVE-TILT-90]) over an established Layer 3 IP address to tilt panels away from the sun, the OSI-SLM detects that the Layer 7 command coincides with an anomalous Layer 5 session login pattern and an unexpected Layer 1 physical impedance shift on that wire segment, flagging [SECURITY-ANOMALY-SPOOF], rejecting the parameters at the guardrail, and isolating the port.

Simultaneously, Layers 6 and 7 execute **Intent-Based Data Collapsing (Semantic Compression)**: instead of transmitting a verbose 2 KB JSON diagnostic string every second, the sending node transmits pure cognitive intent tokens ([BAT-04] [CELL-STABLE] [THERM-CYCLING-NOMINAL]), which the receiving node's SLM reconstructs into full semantic context, cutting data overhead by up to 95%.

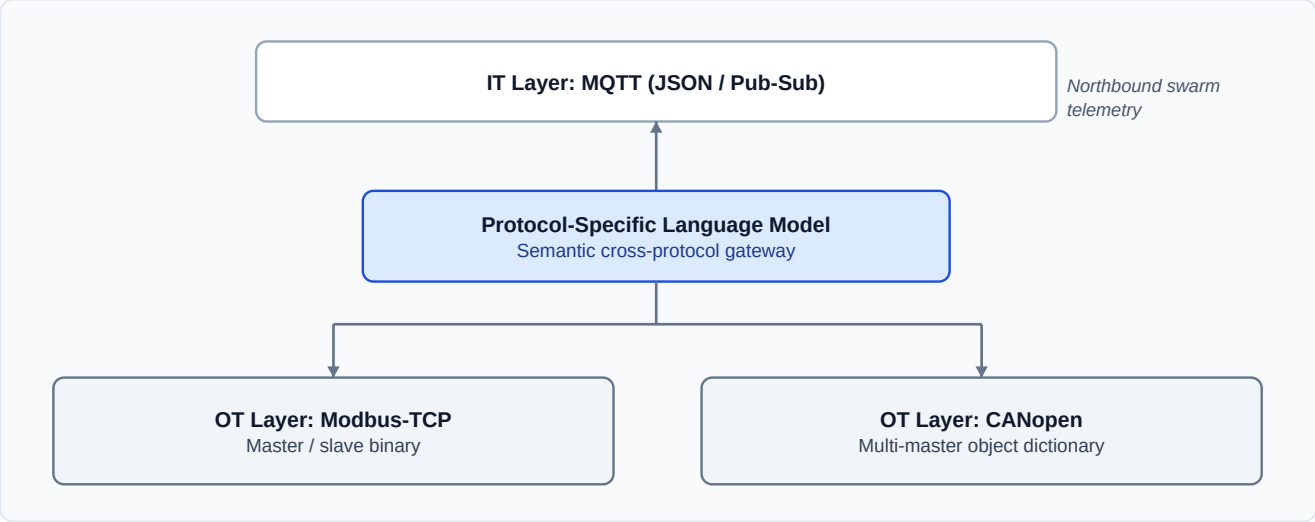
Environmental / System Vector	OSI Target Layers	Traditional Reaction (Static Stack)	Unified Model Co-Optimization Action
High Solar Irradiance + High Thermal Load	Layer 7: Application Layer 4: Transport Layer 1: Physical	Inverter maxes extraction until hard thermal trip; static TCP window causes bus collisions; fans start late via threshold logic.	Slightly detunes MPPT before thermal ceiling; packs non-critical frames into multi-second arrays; pre-overclocks cooling loops early.

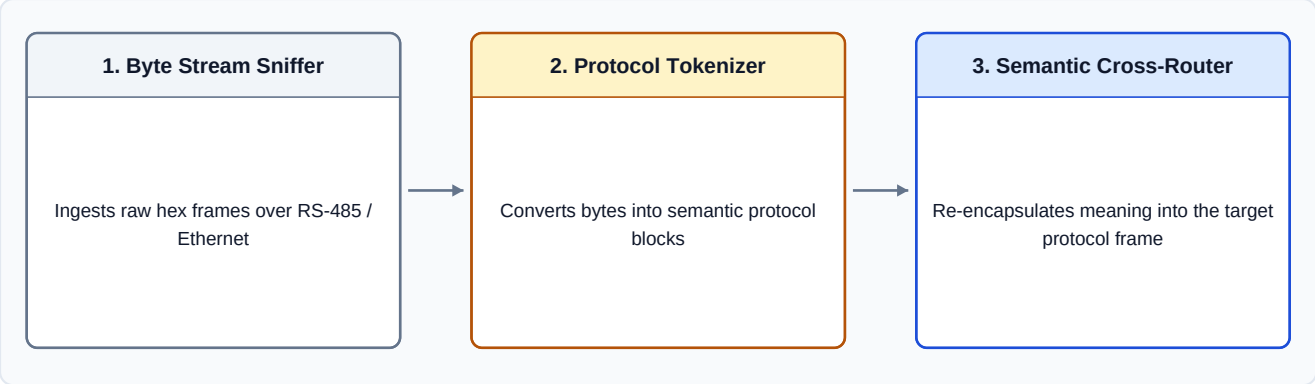
Environmental / System Vector	OSI Target Layers	Traditional Reaction (Static Stack)	Unified Model Co-Optimization Action
Severe Weather Transients (Wind / Cloud Shadows)	Layer 3: Network Layer 2: Data Link	Central cloud loses signal and local trackers stall; frames drop continuously due to line jitter.	Establishes localized peer-to-peer ad-hoc wireless mesh; shortens L2 packet intervals to slip data between interference bursts.
Network Congestion / Edge Swarm Inundation	Layer 6: Presentation Layer 5: Session	Verbose JSON overflows MCU buffers; heartbeat timeouts trigger re-handshake storms.	Switches serialization from JSON to binary CBOR on the fly; extends heartbeat intervals based on payload criticality.

To guarantee defect-free execution, the cross-layer architecture enforces three **Design Quality Principles**: (1) **Strict State Isolation** through the deterministic Guardrail Interlock, (2) **Shared Global/Local Context Windows** that scale down global target priority to localized survival heuristics if orchestrator communication drops, and (3) **Graceful Degeneracy**, shutting down the SLM inference loop under high CPU or memory strain and reverting seamlessly to classical PID loops and static Modbus maps without interrupting active electrical loops.

6.3 Communication Protocol-Specific Language Model (Modbus, CANopen, MQTT)

At the fieldbus boundary, a dedicated **Communication Protocol-Specific Language Model** functions as an inline **Semantic Protocol Gateway and Orchestrator**, bridging binary Operational Technology (OT) fieldbuses with Information Technology (IT) cloud/swarm fabrics:





- **Byte-to-Token Compilation & SunSpec Mapping:** The gateway sniffs raw hex frames from RS-485 or Ethernet buffers and maps byte offsets directly into semantic protocol tokens—converting raw Modbus read bytes (01 03 00 20 00 02 C5 C1) into [MODBUS-READ] [SLAVE-01] [START-REG-32] [LEN-2] [CRC-VALID] and automatically mapping schemaless Modbus register payloads (0x03, 0x04, 0x1A, 0x2C) onto standardized **SunSpec Alliance** solar data models.
- **Context-Aware Error Decoding:** Rather than dropping corrupted packets or triggering hard faults when a Modbus slave times out ([MODBUS-ERROR-04]), the Protocol SLM inspects surrounding Layer 1 noise metrics to determine whether the fault stems from physical line noise or an overloaded inverter MCU, dynamically throttling the polling rate rather than crashing the bus.
- **Multi-Protocol Southbound Synthesis:** When the farm supervisor publishes a northbound MQTT demand-response signal (topic: farm/grid/demand-response, payload: {"strategy": "curtail-battery", "target-reduction-kw": 50}), the Protocol SLM translates this high-level IT intent simultaneously into **Output Block A** (a Modbus-TCP write frame adjusting SunSpec Register 40021 Active Power Limit on the inverter) and **Output Block B** (a CANopen Service Data Object SDO packet lowering the operational discharge current limit across the battery BMS cells).
- **Resiliency via Decoupled Semantic Diary State:** Using an application-level MQTT-ACK store-and-forward matrix during cloud connectivity dropouts, the Protocol SLM continues compiling local Modbus and CAN frames into a compressed **Semantic Diary State**. Upon reconnection, rather than flooding the network with millions of raw historical registers, it publishes a single chronologically continuous state-history token, preserving bandwidth and preventing grid data storms.

7. References

1. Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., & Sanghai, S. (2023). *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. arXiv preprint arXiv:2305.13245.
2. Anderson, K. (2020). *Maximizing Yield with Improved Single-Axis Backtracking on Cross-Axis Slopes: Preprint*. National Renewable Energy Laboratory, NREL/CP-5K00-76023. 47th IEEE Photovoltaic Specialists Conference.
3. Anderson, K. S., Jensen, A. R., & Riley, D. M. (2026). *A Linear Programming Approach to Backtracking for Single-Axis Trackers on Rolling Terrain*. IEEE Journal of Photovoltaics, 16(2), 291–298. https://doi.org/10.1109/JPHOTOV.2025.3645781

4. Bosio, C., Guarrera, M., Sangiovanni-Vincentelli, A., & Mueller, M. W. (2025). *Combining Large Language Models and Gradient-Free Optimization for Automatic Control Policy Synthesis*. arXiv preprint arXiv:2510.00373.
5. Cooper, E., Anderson, K., & Riley, D. (2025). *Performance Improvements Through Advanced PV Backtracking on Uneven Terrain*. IEEE Journal of Photovoltaics, 15(4), 593–599. https://doi.org/10.1109/JPHOTOV.2025.3558254
6. Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. arXiv preprint arXiv:1503.02531.
7. Hu, K., Qiu, L., Lu, Y., Zhao, H., Li, T., Kim, Y., Andreas, J., & He, K. (2026). *ELF: Embedded Language Flows*. arXiv preprint arXiv:2605.10938.
8. Lamaakal, I., Maleh, Y., El Makkaoui, K., Ouahbi, I., Pławiak, P., Alfarraj, O., Almousa, M., & Abd El-Latif, A. A. (2025). *Tiny Language Models for Automation and Control: Overview, Potential Applications, and Future Research Directions*. Sensors, 25(5), 1318. https://doi.org/10.3390/s25051318
9. Maher, G. (2025). *LLMPC: Large Language Model Predictive Control*. Computers, 14(3), 104. https://doi.org/10.3390/computers14030104
10. Mohan, S., Bak, S., Betti, E., Yun, H., Sha, L., & Caccamo, M. (2013). *S3A: Secure System Simplex Architecture for Enhanced Security and Robustness of Cyber-Physical Systems*. Proceedings of the ACM International Conference on High Confidence Networked Systems (HiCONS), 65–74.
11. Narimani, M., & Emami, S. A. (2026). *GA-Agent: Large Language Models as Hyperparameter Optimizers for Evolutionary Controller Synthesis*. arXiv preprint arXiv:2609.27725.
12. Nesti, F., Salamini, N., Marinoni, M., Cicero, G., Serra, G., Biondi, A., & Buttazzo, G. (2026). *The use of the Simplex architecture to enhance safety in deep-learning-powered autonomous systems*. Engineering Applications of Artificial Intelligence, 174, 114583. https://doi.org/10.1016/j.engappai.2026.114583
13. Scherer, M., Macan, L., Jung, V. J., Wiese, P., Bompani, L., Burrello, A., Conti, F., & Benini, L. (2024). *Deeploy: Enabling Energy-Efficient Deployment of Small Language Models on Heterogeneous Microcontrollers*. arXiv preprint arXiv:2408.04413.
14. Sha, L. (2001). *Using Simplicity to Control Complexity*. IEEE Software, 18(4), 20–28. https://doi.org/10.1109/MS.2001.936213
15. Soldani, D., Nahi, P., Bour, H., Jafarizadeh, S., Soliman, M. F., Di Giovanna, L., Monaco, F., Ognibene, G., & Risso, F. (2023). *eBPF: A New Approach to Cloud-Native Observability, Networking and Security for Current (5G) and Future Mobile Networks (6G and Beyond)*. IEEE Access, 11, 57174–57202. https://doi.org/10.1109/ACCESS.2023.3281480
16. van Stein, N., & Bäck, T. (2025). *LLaMEA: A Large Language Model Evolutionary Algorithm for Automatically Generating Metaheuristics*. IEEE Transactions on Evolutionary Computation, 29(2), 331–345. https://doi.org/10.1109/TEVC.2024.3497793
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems (NeurIPS 2017), 30.
18. Wagner, J.-S., DeCaprio, D., Muthu Raja, A. C., Holman, J. M., Brady, L. K., et al. (2024). *CELI: Controller-Embedded Language Model Interactions*. arXiv preprint arXiv:2410.14627.
19. Wang, H., Li, W., Zhang, J., & Ren, L. (2026). *AI Control Scientist: LLM-driven Agentic System for Automated Control Design*. arXiv preprint arXiv:2608.26780.
20. Xia, Y., Jazdi, N., Zhang, J., Shah, C., & Weyrich, M. (2024). *Control Industrial Automation System with Large Language Model Agents*. arXiv preprint arXiv:2409.18009.
21. Yuan, D., Deng, Y., & Chen, H. (2026). *EmbedCopilot-Bench: Evaluating Vision-Language Models for Hardware-Aware Embedded System Development*. Proceedings of the European Conference on Computer Vision (ECCV 2026).
22. Zhang, Z., Hu, Y., Chen, X., Jewell, G. W., & Li, H. (2021). *A Review on Conductive Common-Mode EMI Suppression Methods in Inverter Fed Motor Drives*. IEEE Access, 9, 18345–18360. https://doi.org/10.1109/ACCESS.2021.3054514