hypothesis

WHITE PAPER 03 · AUGUST 2026

TECHNOLOGY ARCHITECTURE

# Factory Floor Up, Not Model Down

*Architecting Resilient, Real-Time AI Infrastructure for Robotics, Industrial Automation, and Life Sciences*

An architectural and operational blueprint for engineering leaders, AI infrastructure architects, robotics teams, and life sciences technologists building physical-first AI platforms.

AI INFRASTRUCTURE · ROBOTICS · INDUSTRIAL AI · LIFE SCIENCES

hypothesisconsultancy.com
hello@hypothesisconsultancy.com

CONTENTS


— Executive summary
1 The paradigm mismatch
2 Sector deep dives
3 The physical-first AI stack
4 Data engineering and telemetry
5 Verification, validation, and safety-critical execution
6 Technical artifacts and deployment blueprint
7 Recommendations for engineering leaders
— Conclusion · About Hypothesis


EXECUTIVE SUMMARY

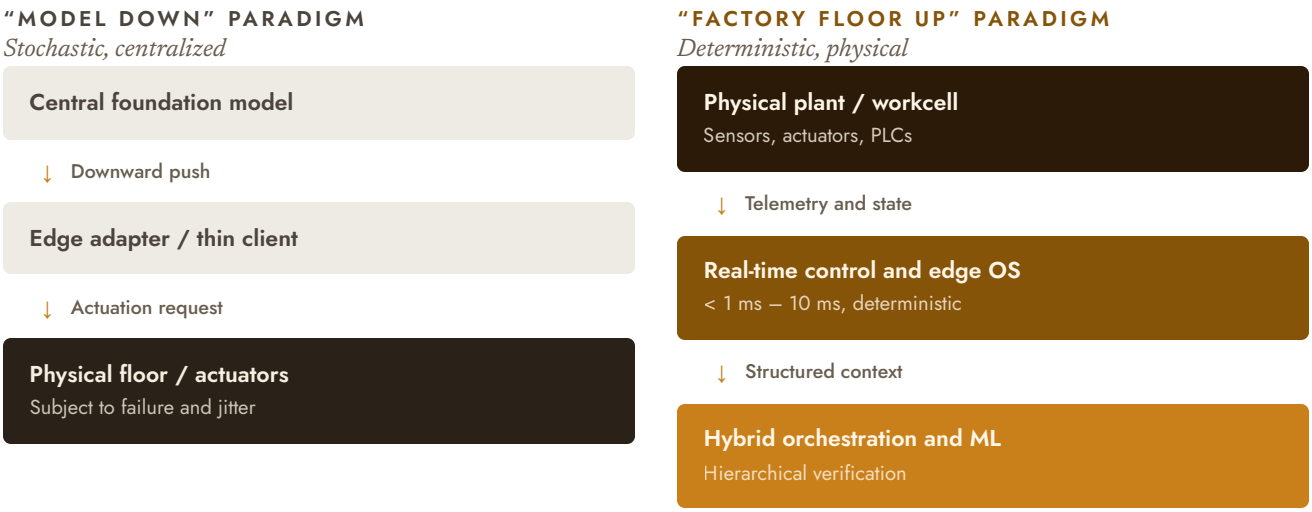
# In physical systems, the model-down paradigm breaks down.

Over the past decade, enterprise artificial intelligence has been dominated by a **"Model Down"** paradigm.

Born in hyperscale cloud environments, this philosophy treats intelligence as a centralized, top-down capability: massive foundation models are trained on unstructured internet-scale data, hosted in centralized data centers, and pushed downward toward end-user endpoints. In digital domains (search, content generation, customer support, and code synthesis) this approach has delivered unprecedented capabilities because these domains tolerate stochastic variance, variable latency, and abstracted compute topologies. When applied to physical systems, however, the "Model Down" paradigm breaks down.

Industrial manufacturing lines, autonomous robotic fleets, high-throughput biopharma synthesis platforms, and safety-critical process industries operate under physical laws, deterministic timing constraints, hard real-time safety envelopes, and stringent regulatory frameworks. Treating the physical environment as merely an ingestion pipe for sensor telemetry and a passive recipient of stochastic model inference leads to catastrophic failure modes: non-deterministic latency spikes, protocol incompatibility, safety boundary violations, unvalidated state drift, and massive edge-to-cloud bandwidth saturation.

**FIGURE 1** TWO WAYS TO PUT INTELLIGENCE INTO PHYSICAL OPERATIONS



To deploy artificial intelligence reliably within physical operations, engineering and product leaders must invert the architectural hierarchy. We must build **"Factory Floor Up."**

A "Factory Floor Up" architecture begins at the physical interface: the microsecond-level determinism of Programmable Logic Controllers (PLCs), the kinematic limits of six-axis robotic manipulators, the fluidic dynamics of automated liquid handlers, and the stringent validation requirements of ISO 13849, IEC 61508, and GAMP 5. Intelligence is layered hierarchically above these deterministic baselines, establishing clear boundaries between stochastic reasoning, deterministic control loops, and hardware-enforced safety interlocks.

This white paper provides an architectural and operational blueprint for engineering leaders, AI infrastructure architects, robotics teams, and life sciences technologists building physical-first AI platforms. It outlines the core architectural dichotomies, details sector-specific implementations across manufacturing, robotics, and wet labs, defines the unified physical AI technology stack, and provides concrete deployment artifacts and organizational methodologies.

SECTION 1

# The paradigm mismatch: "Model Down" vs. "Factory Floor Up"

The fundamental tension between digital-first AI and physical operations stems from mismatched architectural assumptions across latency, determinism, network topology, failure modes, and lifecycle governance.

TABLE 1 TWO SETS OF ARCHITECTURAL ASSUMPTIONS

| ARCHITECTURAL VECTOR | "MODEL DOWN" (CLOUD-NATIVE AI) | "FACTORY FLOOR UP" (PHYSICAL-FIRST) |
|---|---|---|
| **Primary failure mode** | Semantic inaccuracy / hallucination | Kinetic damage / biological spoilage |
| **Latency budget** | 200 ms – 5,000 ms ($P_{99}$ best effort) | 250 µs – 10 ms (hard deterministic) |
| **Compute topology** | Centralized cloud / regional clusters | Heterogeneous edge + on-prem micro data center |
| **Execution nature** | Stochastic, probabilistic inference | Hybrid: deterministic control + ML |
| **Protocol ecosystem** | HTTP/3, gRPC, REST, JSON, Arrow | EtherCAT, PROFINET, OPC UA, ROS 2, CAN |
| **State truth boundary** | Distributed database / eventual sync | Physical reality / sensor fusion bus |
| **Validation standard** | A/B testing, evals, MT-Bench, ROUGE | ISO 13849 PL d/e, IEC 61508, GAMP 5 |
| **Deployment cadence** | Continuous deployment, hot-patch | Validated change control / planned outages |

### 1.1 The pitfalls of model-down implementations

When engineering teams attempt to retrofit centralized, model-centric architectures onto physical infrastructure, they encounter four structural anti-patterns:

#### 1. The latency-jitter paradox

Cloud-hosted vision-language-action (VLA) models or centralized inference endpoints introduce variable network latency (typically 50–300 ms, with standard deviations exceeding 100 ms). That is acceptable for human conversational interfaces. On a high-speed packaging line running at 1,200 units per minute, however, a single 50 ms spike means a full unit has passed the station before the command arrives.

#### 2. Protocol disconnect and serialization bloat

Translating deterministic fieldbus protocols (such as EtherCAT cyclic synchronous position frames running at 1 kHz) into JSON payloads or standard gRPC streams overburdens embedded processors and destroys microsecond-level synchronization.

#### 3. Stochastic actuation without hard failsafes

Direct-to-actuator model outputs (e.g., end-to-end neural networks predicting raw joint torques or motor current commands) lack deterministic mathematical bounds. Without an intermediate kinematic/dynamic safety filter, an out-of-distribution observation can command an impossible or destructive physical state.

#### 4. Data gravity and network partition vulnerability

Uploading raw, uncompressed multi-modal telemetry (e.g., 8-channel 4K video at 60 FPS, 50 kHz high-frequency acoustic emissions, and multi-axis vibrational accelerometry) saturates uplink bandwidth. If the wide-area network drops, a model-down architecture halts the production cell and breaks the line's availability targets.

SECTION 2

## Sector deep dives

The operational realities of physical environments vary by domain, yet all require a floor-up approach to guarantee reliability, safety, and regulatory compliance.

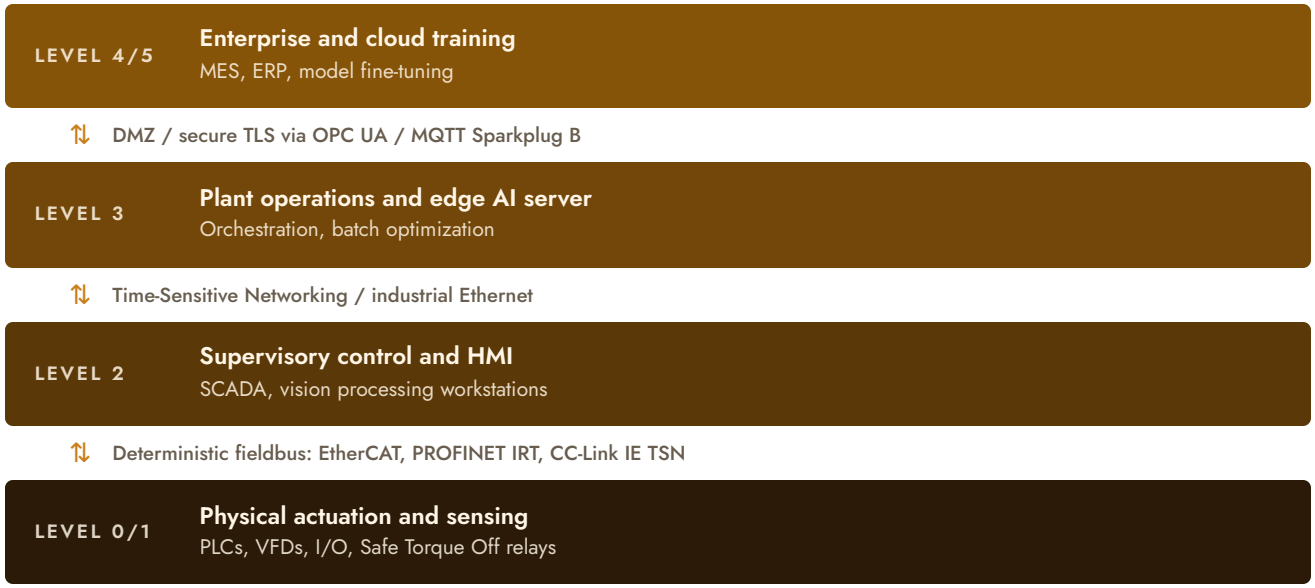
**FIGURE 2** SECTOR ARCHITECTURES

| INDUSTRIAL MANUFACTURING | AUTONOMOUS ROBOTICS & FLEET LOGISTICS | LIFE SCIENCES & AUTOMATED WET LABS |
|---|---|---|
| • Deterministic PLC isolation<br>• Sub-millisecond fieldbuses<br>• Acoustic and vibration edge inference<br>• ISO 13849 PL e safety relays | • Dynamic edge SLAM<br>• Kinematic boundary enforcement<br>• Intermittent mesh synchronization<br>• VLA strategic planning separated from deterministic control | • Closed-loop assay optimization<br>• GAMP 5 and 21 CFR Part 11 compliance<br>• Reagent tracking and process analytics<br>• Environmental drift compensation |

## 2.1 Industrial automation and discrete/process manufacturing

In modern manufacturing, the fundamental unit of trust is the industrial controller (PLC / PAC / IPC). Factory-floor architectures operate on the classic **Purdue Model (ISA-95)**, which enforces strict network and functional segmentation between physical processes (Level 0/1), supervisory control (Level 2), manufacturing operations management (Level 3), and enterprise business networks (Level 4/5).

FIGURE 3 THE PURDUE MODEL (ISA-95), WITH AI PLACED AT LEVELS 3 AND ABOVE




### ARCHITECTURAL IMPERATIVES

- **The deterministic air gap.** Artificial intelligence models must never directly override Level 0/1 safety loops. The safety system (e.g., emergency stops, light curtains, Safe Torque Off (STO) relays) must remain hardwired or executed over verified functional safety protocols (e.g., Safety over EtherCAT (FSoE) or PROFIsafe) meeting ISO 13849 (Performance Level d/e, Category 3/4) or IEC 62061 (SIL 2/3).
- **High-frequency edge telemetry.** Condition monitoring and predictive quality require edge-located sampling. Acoustic emission sensors (10 kHz–100 kHz) and triaxial high-bandwidth vibration accelerometers capture sub-surface tool wear or bearing spalling. Signal processing must occur at the edge via specialized DSPs or local Tensor Processing Units (TPUs), extracting fast Fourier transform (FFT) features or spectral density embeddings before transmitting reduced-dimensionality metrics upstream.
- **Closed-loop setpoint optimization.** AI should act as a supervisory setpoint generator (Level 2/3), outputting bounded optimization envelopes (e.g., adjusting thermal profiles, feed rates, or injection pressures) that the PLC evaluates against internal safety tables before execution.

## 2.2 Autonomous robotics and fleet operations

Robotics platforms bridge software reasoning and physical motion. Modern robotics architectures often suffer from an over-reliance on monolithic, end-to-end models that attempt to map camera pixels directly to actuator torques.

### ARCHITECTURAL IMPERATIVES

- **Separation of semantic planning and dynamic execution.** A floor-up robotics architecture bifurcates intelligence into asynchronous, high-level semantic reasoning (1 Hz–10 Hz) and deterministic, low-level dynamic control (200 Hz–1,000 Hz). High-level Vision-Language-Action (VLA) models or topological path planners generate task graphs and

collision-free spatial waypoints. Low-level Model Predictive Control (MPC) and Whole-Body Control (WBC) algorithms execute trajectory tracking while enforcing strict kinematic, torque, and obstacle-avoidance boundaries.

**FIGURE 4** THREE CONTROL RATES IN A FLOOR-UP ROBOT




- **Dynamic edge SLAM and state estimation.** Simultaneous Localization and Mapping (SLAM) cannot rely on continuous cloud telemetry. Odometry must be computed via hardware-synchronized multi-modal sensor fusion (combining wheel encoders, high-frequency IMUs, LiDAR point clouds, and stereo event cameras) using Extended Kalman Filters (EKF) or factor-graph optimizers running locally on an edge compute board.
- **Degraded-mode resiliency.** In the event of compute throttling, camera occlusion, or network disconnection, the robotic platform must drop into a deterministic, fail-safe state: decelerating along a verified stopping trajectory, holding position via mechanical brakes, or operating in a degraded baseline mode without cloud dependency.

### 2.3 Life sciences, automated wet labs, and biomanufacturing

Life science environments, including high-throughput screening (HTS) platforms, automated cell therapy bioreactors, and clinical drug formulation pipelines, combine robotic transport, automated microfluidics, and dynamic biochemical reactions.

#### ARCHITECTURAL IMPERATIVES

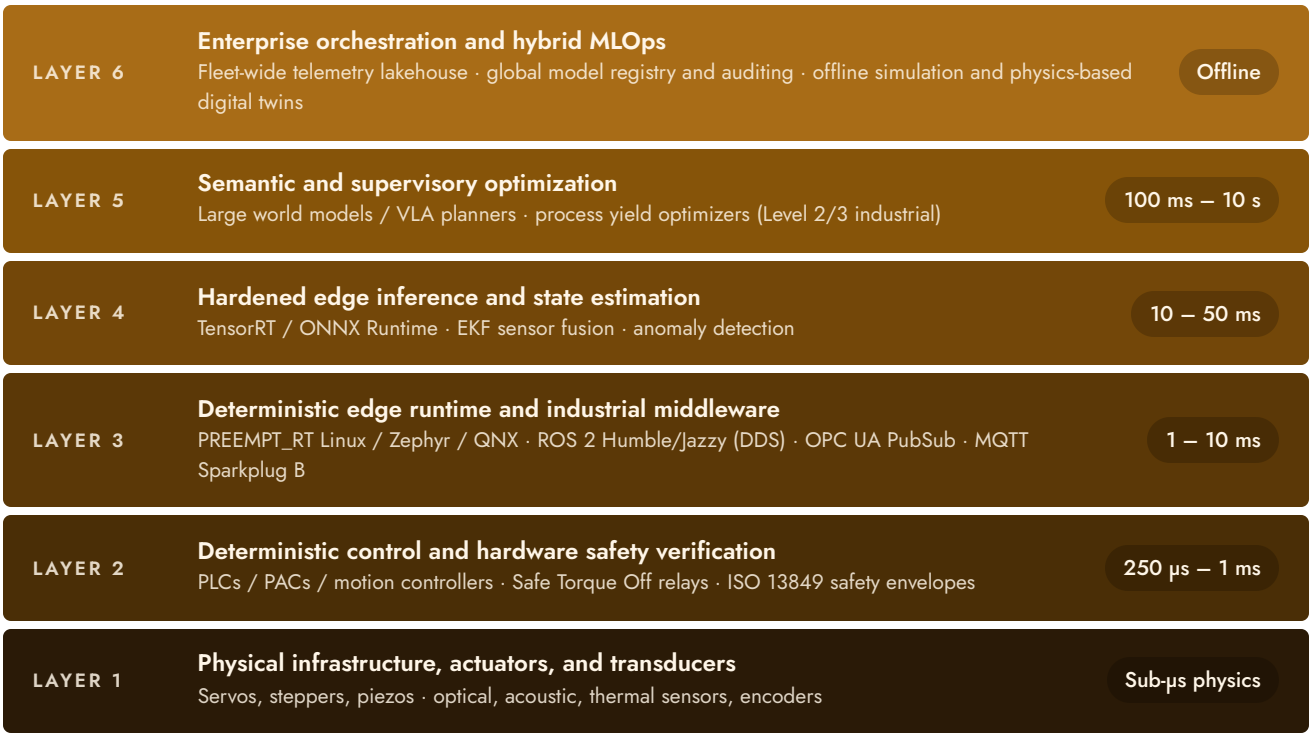
- **GAMP 5 and 21 CFR Part 11 compliance.** Every parameter generated by an AI model that influences an experimental or therapeutic batch must be fully traceable, versioned, immutable, and explainable. Black-box parameter adjustments violate regulatory standards. Model pipelines must feature complete data provenance, model lineage, and cryptographic audit trails.
- **Physical assay constraints and viscosity drift.** Automated liquid handlers (pipetting workstations) fail when fluidic physical properties deviate from calibration. Changes in ambient humidity, temperature, or reagent protein concentrations alter fluid surface tension and viscosity, causing volumetric aspiration errors. A floor-up system couples real-time optical coherence or pressure-transducer monitoring at the pipette tip with edge ML inference to detect droplet formation anomalies, automatically triggering closed-loop volumetric recalibration.
- **Hybrid mechanistic and data-driven models.** Pure statistical deep learning often violates fundamental biophysical conservation laws (mass, energy, stoichiometry). Life sciences infrastructure demands Physics-Informed Neural Networks (PINNs) or hybrid mechanistic models, where mathematical models of enzymatic kinetics or fluid dynamics bound the state space explored by predictive models.

SECTION 3

# Architectural blueprint: the physical-first AI stack

Building from the floor up requires a purpose-built, six-layer architectural stack. Each layer possesses clear boundaries, verified interfaces, and defined latency and safety guarantees.

**FIGURE 5** THE SIX-LAYER PHYSICAL-FIRST AI STACK, WITH LATENCY BUDGETS




### Layer 1: Physical infrastructure, actuators, and transducers

The ground truth of the system. Includes multi-axis brushless DC motors, pneumatic linear slides, microfluidic piezo-pumps, thermocouple arrays, optical time-of-flight sensors, and high-resolution industrial cameras. Signal characteristics are analog or high-frequency digital, constrained by thermal limits, mechanical backlash, friction, and fluid dynamics.

### Layer 2: Deterministic control and hardware safety verification

Executes hard real-time cyclic control (e.g., 1 kHz cyclic synchronous torque or position modes). Built on ruggedized hardware (Beckhoff, Siemens, Rockwell, B&R, or custom microcontrollers) running bare-metal, certified RTOS, or deterministic PLC runtimes (IEC 61131-3). Enforces physical safety interlocks independently of any upstream AI software.

### Layer 3: Deterministic edge runtime and industrial middleware

Bridges the fieldbus and compute platforms. Operates on real-time operating systems (Linux with PREEMPT_RT, which has been part of the mainline kernel since version 6.12, QNX, or VxWorks). Utilizes deterministic, industrial-grade data distribution layers:

- **OPC UA / OPC UA PubSub over TSN** (Time-Sensitive Networking) for structured, standardized telemetry exchange.
- **ROS 2** (Robot Operating System) utilizing real-time Data Distribution Service (DDS) implementations (e.g., CycloneDDS, FastDDS) configured with strict Quality of Service (QoS) profiles (transient local durability, reliable reliability, deadline-driven liveliness).

- **MQTT with Sparkplug B payloads** for stateful, bandwidth-efficient OT-to-IT translation.

### Layer 4: Hardened edge inference and state estimation engine

Executes optimized machine learning models locally on accelerated industrial edge compute hardware (e.g., NVIDIA Jetson Thor/Orin, Intel Xeon with integrated NPU, Hailo-8, or dedicated edge FPGAs). Features model runtimes compiled for deterministic execution (TensorRT, ONNX Runtime with direct memory pinning, OpenVINO). Performs continuous state estimation via sensor fusion, point-cloud segmentation, edge visual inspection, and high-speed vibration analysis.

### Layer 5: Semantic and supervisory optimization

Performs complex tactical decisions, schedule optimization, and multi-step visual reasoning. Evaluates batch parameters, orchestrates multi-robot routing, and updates supervisory control setpoints. This layer operates asynchronously without risking kinetic destabilization if an inference step experiences latency jitter.

### Layer 6: Enterprise orchestration and hybrid MLOps

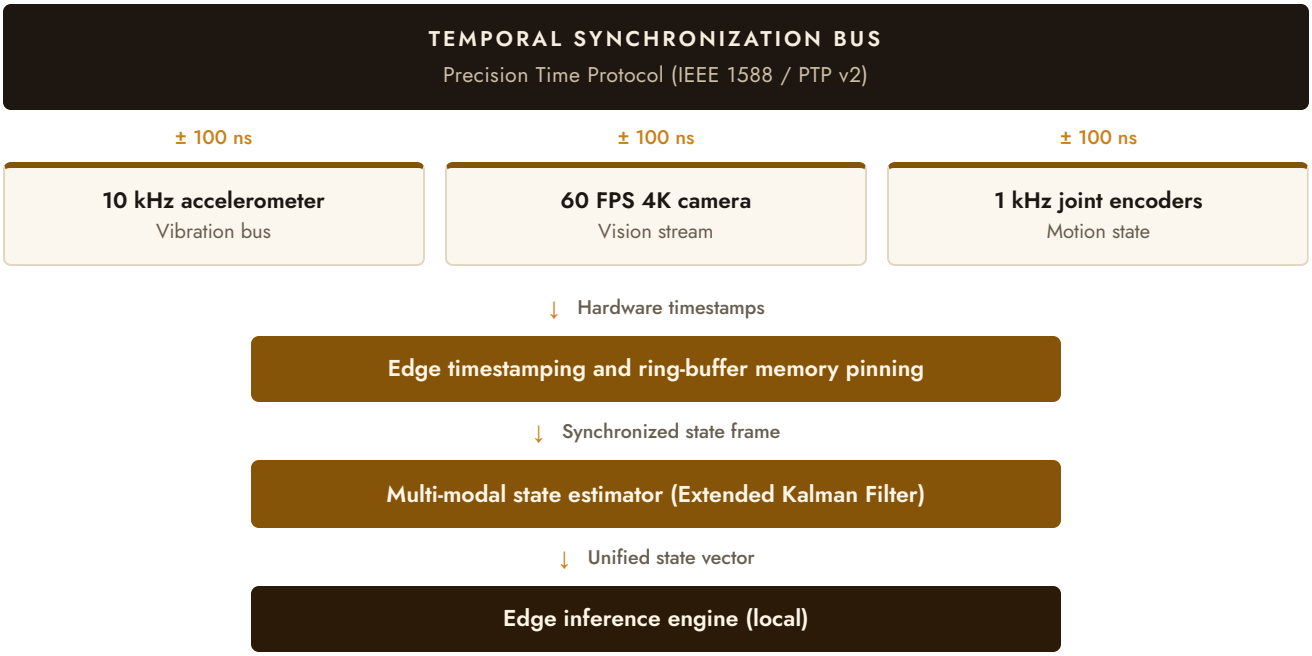
Maintains fleet-wide model lineage, continuous evaluation pipelines, digital twin simulations (e.g., Isaac Sim, Omniverse, Gazebo), and historical data lakehouses. Gathers curated edge-case data, manages active learning loops, and packages signed, immutable deployment containers destined for the physical fleet.

SECTION 4

## Data engineering and telemetry: ingestion, synchronization, and state estimation

Data engineering for physical systems presents distinct challenges compared to traditional enterprise data pipelines. The primary failure point in industrial AI systems is not model capacity; it is **temporal misalignment** and **spatial distortion** across multi-modal sensor streams.

FIGURE 6 ONE CLOCK FOR EVERY SENSOR



### 4.1 Temporal synchronization via Precision Time Protocol (IEEE 1588 / PTP)

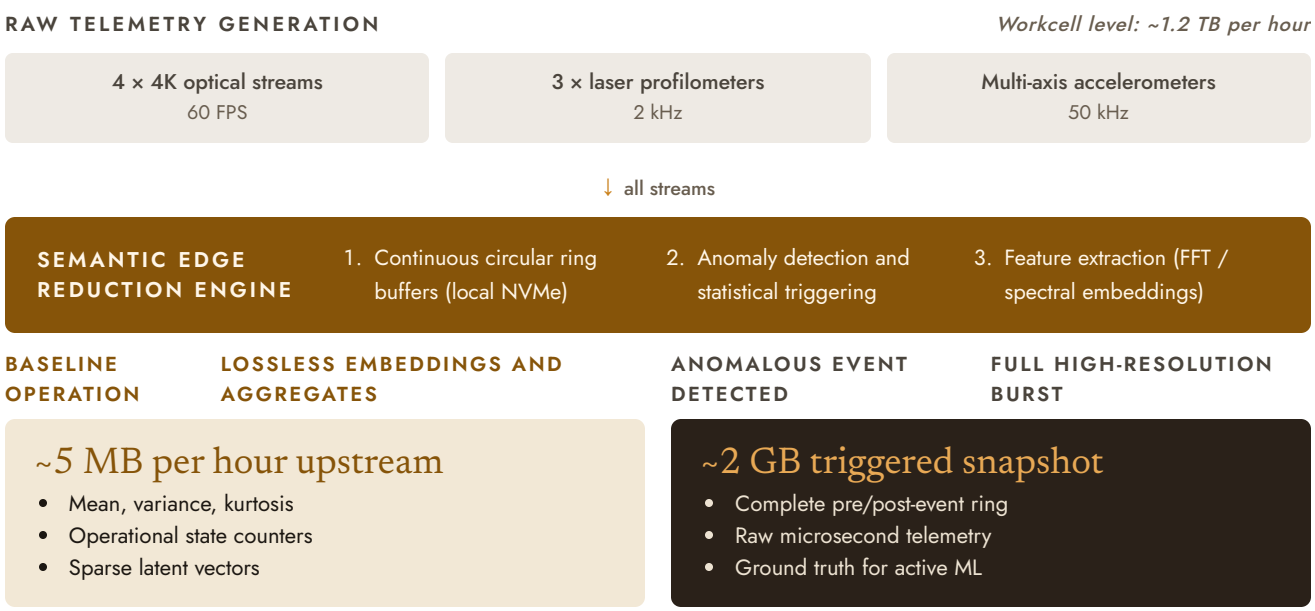
Standard network protocols like NTP provide millisecond-level time synchronization, which is insufficient for high-speed multi-sensor fusion. When an autonomous mobile robot maneuvers at 2 meters per second, a 20 ms timing discrepancy between a LiDAR sweep and an IMU reading produces a 40 mm spatial localization error, enough to cause navigation failures in tight aisles.

- **Implementation standard.** Implement IEEE 1588v2 (PTP) or IEEE 802.1AS (gPTP for Time-Sensitive Networking) across all edge sensors, network switches, and compute nodes.
- **Hardware timestamping.** Ingest cards (NICs, frame grabbers, and fieldbus master interfaces) must timestamp incoming frames at the PHY/MAC hardware layer upon arrival, bypassing operating system interrupt handling delays.
- **Deterministic interpolation.** Implement edge ring buffers that align heterogeneous telemetry rates (e.g., IMU at 1 kHz, joint torques at 500 Hz, depth cameras at 30 Hz, spectrometers at 5 Hz) into unified state frames using cubic spline or SLERP (spherical linear interpolation) methods anchored to absolute PTP timestamps.

### 4.2 Bandwidth optimization: semantic edge filtering vs. raw ingestion

Streaming uncompressed raw telemetry to centralized repositories is economically and technically unfeasible. A single production workcell equipped with four 4K cameras, three high-speed profilometers, and multi-axis vibrational arrays generates over 1.2 TB of raw data per hour.

FIGURE 7 REDUCE AT THE EDGE, BURST ON ANOMALY




- **Circular ring buffering with anomaly triggering.** Maintain high-resolution telemetry in rolling local NVMe ring buffers (e.g., retaining the last 30 minutes of raw data).
- **Continuous edge feature extraction.** Extract and transmit low-bandwidth statistical metrics (mean, RMS, kurtosis, peak frequency components) or latent-space vector embeddings during standard operating conditions (reducing uplink bandwidth from gigabits to kilobits per second).
- **Automated failure snapshotting.** When an edge anomaly detector (e.g., isolation forest, autoencoder reconstruction error, or physical threshold breach) identifies an out-of-family operational state, the edge runtime automatically freezes and tags the preceding and succeeding 60-second raw telemetry window, securely routing this rich data artifact to the central MLOps lakehouse for root-cause analysis and active model fine-tuning.

SECTION 5

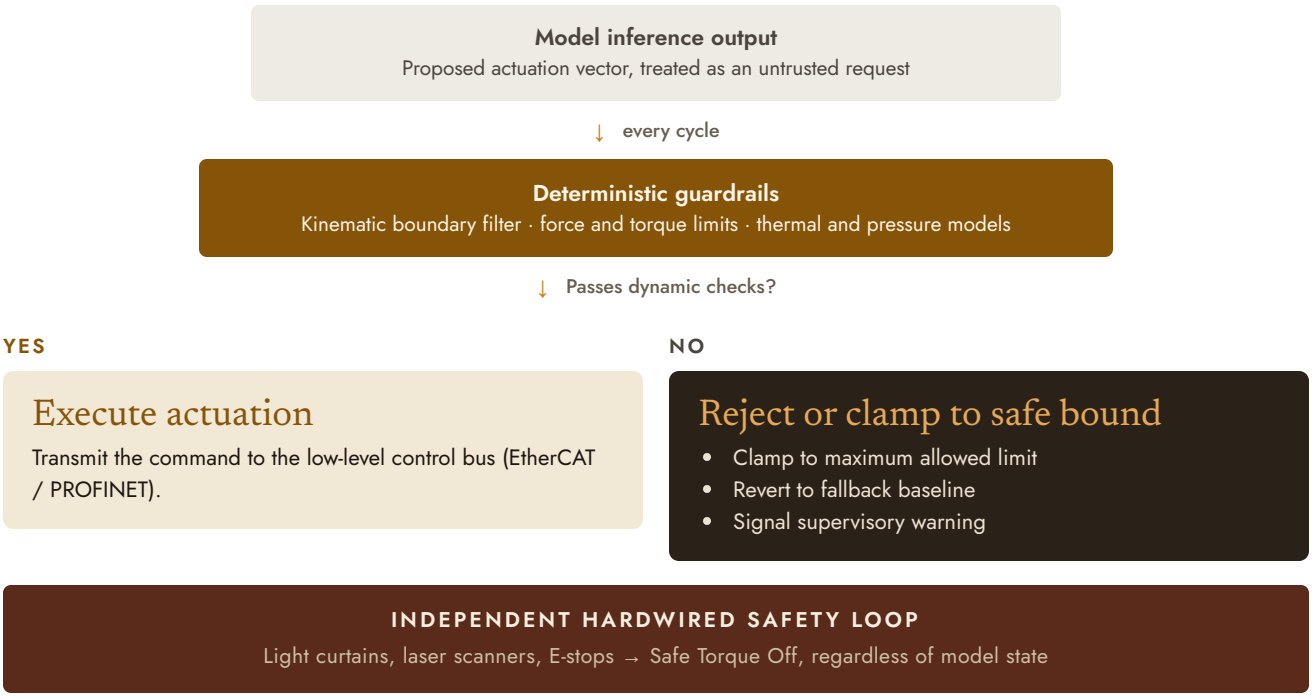
# Verification, validation, and safety-critical execution

Deploying AI models to physical systems requires moving beyond traditional software metrics (e.g., accuracy, $F_1$-score, perplexity) to rigorous engineering validation.

## 5.1 The dual-track safety architecture

To satisfy international functional safety standards, machine learning components must be insulated from direct, unconstrained physical actuation.

FIGURE 8 THE DUAL-TRACK SAFETY ARCHITECTURE




1. **Deterministic guardrails.** The output of any model (e.g., an inference predicting robotic arm joint positions or an algorithm setting chemical reagent dosing rates) is treated as an *untrusted request*. It must pass through an analytically verified mathematical filter:
    - **Robotics:** enforces Cartesian operational workspace limits, maximum joint velocities, jerk limits, torque saturation boundaries, and self-collision constraints.
    - **Industrial process:** enforces thermal slew rate ceilings, maximum allowable pressure thresholds, and stoichiometric bounds.
2. **Hardwired functional safety loop.** Completely separate from the operating system hosting the AI models, a dedicated safety controller continuously evaluates independent safety inputs (e.g., light curtains, safety laser scanners, emergency stop circuits, over-travel switches). If a breach occurs, the safety controller asserts Safe Torque Off (STO) or Safe Stop 1/2 (SS1/SS2), cutting actuator power at the hardware level regardless of model state.

## 5.2 Regulatory and validation crosswalk

Deployments in regulated sectors require aligning the software development lifecycle with established compliance frameworks:

**TABLE 2** REGULATORY AND VALIDATION CROSSWALK

| REGULATORY STANDARD | DOMAIN | ARCHITECTURAL COMPLIANCE MANDATE |
|---|---|---|
| **ISO 13849-1 / IEC 62061** | Machinery safety (manufacturing) | Deterministic safety architecture; Performance Level (PL d / PL e); separation of safety and control bus |
| **IEC 61508 (SIL 1 – SIL 4)** | Functional safety (E/E/PE systems) | Formal software verification; deterministic fault behavior; bounded execution runtime memory |
| **GAMP 5 (Category 4/5 software)** | Life sciences / pharmaceuticals | Validated lifecycle documentation; traceable model requirements; strict audit trails for parameters |
| **FDA 21 CFR Part 11 / EU Annex 11** | Electronic records / biopharma | Cryptographic signatures on model artifacts, immutable event logs, role-based execution control |

SECTION 6

# Technical artifacts and deployment blueprint

To bridge theory and execution, this section provides reference deployment artifacts illustrating the "Factory Floor Up" methodology.

*These listings are illustrative patterns for discussion. They are not certified safety functions and must be verified independently before use on physical equipment.*

## 6.1 Real-time deterministic guardrail (Python/C++ integration pattern)

The following example demonstrates a low-latency, deterministic safety wrapper designed to sit between a high-level model inference engine and a low-level industrial controller. It validates model recommendations against kinematic and thermal bounds before issuing commands to the hardware abstraction layer.

**LISTING 1** DETERMINISTIC SAFETY GOVERNOR `deterministic_guardrail.py`

```
"""
deterministic_guardrail.py
Reference safety wrapper for physical AI actuation (illustrative).
Validates model-generated setpoints against hard physical envelopes.
"""

from dataclasses import dataclass
from typing import Tuple, Optional
import numpy as np
import time
import logging

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger("SafetyGuardrail")

@dataclass(frozen=True)
class PhysicalConstraints:
    max_joint_velocities: np.ndarray  # rad/s
    max_joint_accelerations: np.ndarray  # rad/s^2
```

```
    joint_position_limits_lower: np.ndarray  # rad
    joint_position_limits_upper: np.ndarray  # rad
    max_temperature_celsius: float
    max_emergency_stop_deceleration: float  # rad/s^2

class DynamicSafetyGovernor:
    def __init__(self, constraints: PhysicalConstraints, loop_frequency_hz: float = 250.0):
        self.constraints = constraints
        self.dt = 1.0 / loop_frequency_hz
        self.last_valid_positions: Optional[np.ndarray] = None
        self.last_valid_velocities: Optional[np.ndarray] = None

    def evaluate_and_clamp_setpoint(
        self,
        target_positions: np.ndarray,
        current_positions: np.ndarray,
        measured_temperature: float
    ) -> Tuple[np.ndarray, bool]:
        """
        Evaluates proposed target positions against hard physical constraints.
        Returns:
            Tuple[np.ndarray, bool]: (Safe positions to execute, Was_Clamped_Flag)
        """
        # 1. Thermal Emergency Check
        if measured_temperature > self.constraints.max_temperature_celsius:
            logger.critical(f"Thermal boundary violated: {measured_temperature}°C. Commanding controlled stop.")
            return self._compute_emergency_stop(current_positions), True

        # Initialize internal state tracking on first cycle
        if self.last_valid_positions is None:
            self.last_valid_positions = np.copy(current_positions)
            self.last_valid_velocities = np.zeros_like(current_positions)

        was_clamped = False
        safe_positions = np.copy(target_positions)

        # 2. Hard Kinematic Absolute Position Limits
        clamped_positions = np.clip(
            safe_positions,
            self.constraints.joint_position_limits_lower,
            self.constraints.joint_position_limits_upper
        )
        if not np.array_equal(clamped_positions, safe_positions):
            was_clamped = True
            safe_positions = clamped_positions

        # 3. Dynamic Velocity Limits (Derived over discrete time step)
        implied_velocity = (safe_positions - self.last_valid_positions) / self.dt
        velocity_scaling_factors = np.abs(implied_velocity) / self.constraints.max_joint_velocities
        max_v_scale = np.max(velocity_scaling_factors)

        if max_v_scale > 1.0:
            was_clamped = True
            # Scale displacement proportionally across all joints to preserve trajectory vector
            scaled_velocity = implied_velocity / max_v_scale
            safe_positions = self.last_valid_positions + (scaled_velocity * self.dt)
            implied_velocity = scaled_velocity

        # 4. Dynamic Acceleration Limits
        implied_acceleration = (implied_velocity - self.last_valid_velocities) / self.dt
        accel_scaling_factors = np.abs(implied_acceleration) / self.constraints.max_joint_accelerations
        max_a_scale = np.max(accel_scaling_factors)

        if max_a_scale > 1.0:
            was_clamped = True
```

```
            scaled_acceleration = implied_acceleration / max_a_scale
            effective_velocity = self.last_valid_velocities + (scaled_acceleration * self.dt)
            safe_positions = self.last_valid_positions + (effective_velocity * self.dt)
            implied_velocity = effective_velocity

        # Update state history
        self.last_valid_positions = np.copy(safe_positions)
        self.last_valid_velocities = np.copy(implied_velocity)

        return safe_positions, was_clamped

    def _compute_emergency_stop(self, current_positions: np.ndarray) -> np.ndarray:
        """Computes deterministic deceleration ramp down to zero velocity."""
        if self.last_valid_velocities is None or np.all(self.last_valid_velocities == 0):
            return current_positions

        decel = np.sign(self.last_valid_velocities) * self.constraints.max_emergency_stop_deceleration * self.dt
        new_velocities = np.where(
            np.abs(self.last_valid_velocities) <= np.abs(decel),
            0.0,
            self.last_valid_velocities - decel
        )
        self.last_valid_velocities = new_velocities
        safe_positions = current_positions + (new_velocities * self.dt)
        self.last_valid_positions = safe_positions
        return safe_positions
```

### 6.2 Deterministic middleware configuration (ROS 2 / CycloneDDS QoS)

In robotics and automated workcells, default networking profiles use standard best-effort discovery and non-real-time memory allocations. Deploying on the factory floor requires strict, deterministic DDS configurations with locked memory pages and bounded queuing.

**LISTING 2** REAL-TIME DDS PROFILE — cyclonedds_realtime_profile.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- cyclonedds_realtime_profile.xml -->
<!-- Reference CycloneDDS configuration for real-time robotic workcell -->
<CycloneDDS xmlns="https://cdds.io/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Domain id="any">
        <General>
            <NetworkInterfaceAddress>auto</NetworkInterfaceAddress>
            <AllowMulticast>true</AllowMulticast>
            <EnableMulticastLoopback>false</EnableMulticastLoopback>
        </General>
        <Internal>
            <!-- Lock heap allocations to prevent dynamic paging during execution -->
            <Watermarks>
                <WhcHigh>524288</WhcHigh>
                <WhcLow>131072</WhcLow>
            </Watermarks>
            <!-- Real-time thread priority assignment for receive/delivery loops -->
            <SchedulingPolicy>prio_fifo</SchedulingPolicy>
            <SchedulingPriority>80</SchedulingPriority>
        </Internal>
        <Tracing>
            <Verbosity>warning</Verbosity>
            <OutputFile>/var/log/cyclonedds_rt.log</OutputFile>
        </Tracing>
    </Domain>
</CycloneDDS>
```

### 6.3 Edge container deployment blueprint

Edge nodes must maintain high availability, read-only system configurations, and explicit hardware access without container orchestrator overhead introducing latency jitter.

**LISTING 3** HARDENED EDGE INFERENCE SERVICE `docker-compose.edge-inference.yml`

```yaml
# docker-compose.edge-inference.yml
# Hardened edge deployment profile for industrial AI inference
version: '3.8'

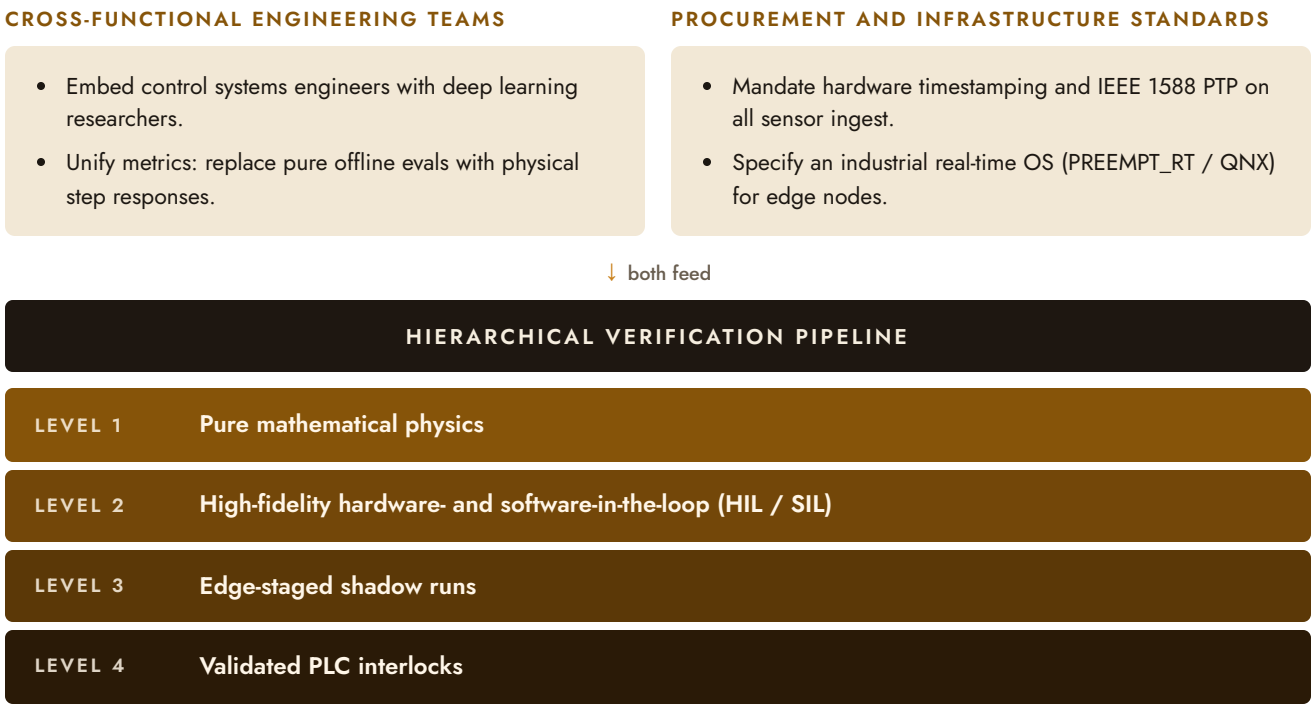
services:
  inference-runtime:
    image: internal-registry.corp/industrial-ai/vision-inference:v2.4.1
    container_name: edge_vision_inference
    restart: always
    network_mode: "host" # Avoid Docker bridge latency jitter
    ipc: "host"          # Enable zero-copy shared memory IPC
    privileged: false
    cap_add:
      - SYS_NICE         # Required to set real-time thread scheduling (SCHED_FIFO)
      - IPC_LOCK         # Lock memory pages to prevent swap access
    devices:
      - "/dev/nvhost-ctrl"
      - "/dev/nvhost-nvdec"
      - "/dev/nvhost-nvenc"
      - "/dev/nvhost-gpu"
    volumes:
      - type: bind
        source: /etc/ptp4l.conf
        target: /etc/ptp4l.conf
        read_only: true
      - type: bind
        source: /opt/industrial/models/pinned_engine.engine
        target: /models/inference.engine
        read_only: true
      - type: tmpfs
        target: /dev/shm
        tmpfs:
          size: 2147483648 # 2GB pinned shared memory buffer
    environment:
      - OMP_NUM_THREADS=4
      - CUDA_DEVICE_MAX_CONNECTIONS=1
      - RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
      - CYCLONEDDS_URI=file:///etc/cyclonedds_realtime_profile.xml
    deploy:
      resources:
        reservations:
          devices:
            - driver: cdi
              device_ids:
                - nvidia.com/gpu=all
```

SECTION 7

# Practical recommendations for engineering and product leaders

Transitioning an organization from a "Model Down" to a "Factory Floor Up" paradigm requires concrete changes across engineering processes, procurement strategies, and team structures.

**FIGURE 9** ORGANIZATIONAL ALIGNMENT BLUEPRINT




## 7.1 Cross-functional team architecture

- **Pair machine learning engineers with OT/control engineers.** Prevent organizational silos by ensuring data scientists building physical models work alongside controls engineers who understand PLC code, fieldbus cycle budgets, and plant-floor hazard protocols.
- **Redefine success metrics.** Discontinue evaluating models exclusively on offline benchmarks (e.g., test-set MSE, validation loss). Establish primary operational metrics: *$P_{99}$ inference latency*, *physical jitter*, *zero-crossing boundary violations*, *tracking error under load*, and *OEE impact*.

## 7.2 Hardware and infrastructure selection

- **Enforce deterministic time protocols across the bill of materials (BOM).** When procuring industrial cameras, LiDARs, encoders, and network interfaces, mandate hardware support for IEEE 1588v2 (PTP) or IEEE 802.1AS. Reject sensors limited to software-level USB polling without hardware trigger capabilities.
- **Prioritize heterogeneous edge hardware.** Select industrial edge compute platforms that feature dedicated hardware engines for signal ingestion (e.g., FPGAs, PCIe frame grabbers with direct DMA memory access) alongside machine learning acceleration hardware.

### 7.3 Testing, simulation, and staged deployment pipelines

- **Hardware-in-the-loop (HIL) and software-in-the-loop (SIL).** Before deploying a model to a physical asset, pass its outputs through automated HIL testing rigs. Run millions of stochastic edge cases in physics simulation platforms (e.g., Isaac Sim, Drake, Gazebo) with simulated latency jitter, sensor dropouts, and physical disturbances.
- **Shadow execution mode.** Deploy new models to production edge systems in "shadow mode." The model consumes live, synchronized sensor telemetry and generates predictions in real time, but its outputs are logged to ring buffers and evaluated against the existing baseline without driving physical actuators. Transition to active actuation only after meeting zero-violation criteria over hundreds of operational hours.

CONCLUSION

## The industrial intelligence imperative

The promise of physical AI cannot be realized through centralized, top-down AI architectures alone.

Autonomous factories that dynamically reconfigure, robotic workcells that manipulate arbitrary objects with superhuman precision, and automated laboratories that accelerate therapeutic discovery all depend on something the model-down paradigm does not provide.

> *The physical world is not an abstraction. It is a domain of mass, inertia, friction, thermal dissipation, and non-negotiable safety boundaries.*

Building **"Factory Floor Up"** is not merely an alternative engineering approach; it is the fundamental prerequisite for deploying artificial intelligence into physical operations. By grounding our systems in deterministic control, sub-millisecond synchronization, hardware-enforced guardrails, and rigorous safety verification, we can construct industrial AI infrastructure that is both technically ambitious and structurally resilient.

Engineering leaders who embrace this shift will move beyond fragile pilot projects to build the future of physical autonomy: robust, scalable systems that operate safely and predictably across the global industrial footprint.

hypothesis

ABOUT HYPOTHESIS

# Independent advisory for frontier technology.

Hypothesis advises founders and corporate innovation teams bringing frontier technology into complex, regulated markets, across AI infrastructure, robotics, industrial AI, and life sciences. We help teams pressure-test foundational assumptions, sharpen their commercial narrative, and turn working prototypes into scalable businesses.

We work from the factory floor up, not the model down: starting with the operators who must buy, deploy, and trust a system, then working backward to the product, pricing architecture, and go-to-market strategy. Many engagements begin with our operator-grade diagnostic, an evaluation of commercial readiness before a company commits capital, headcount, or manufacturing capacity.

| 70+ | 15+ | 3 |
|---|---|---|
| Deep tech startups advised on landing first customers | Years bringing frontier technology to market | Founder exits across the team |

Our work is advisory only. We do not sell to your customers, negotiate your contracts, or broker capital, and we accept no commissions or success fees. Engagements run on fixed terms, agreed in writing, so our advice serves only what is right for your company.

## Building from the factory floor up?

Start with one conversation. No pitch and no obligation, just a candid view of where your product and go-to-market stand.

hello@hypothesisconsultancy.com
hypothesisconsultancy.com
77 Geary Street, 5th Floor
San Francisco, CA 94108

This paper is for general information only. It does not constitute engineering, safety, legal, or regulatory advice. The code listings in Section 6 are illustrative patterns, not certified safety functions, and must be independently verified before use on physical equipment. Standards cited should be confirmed against their current published versions. © 2026 Hypothesis Holding LLC. All rights reserved.