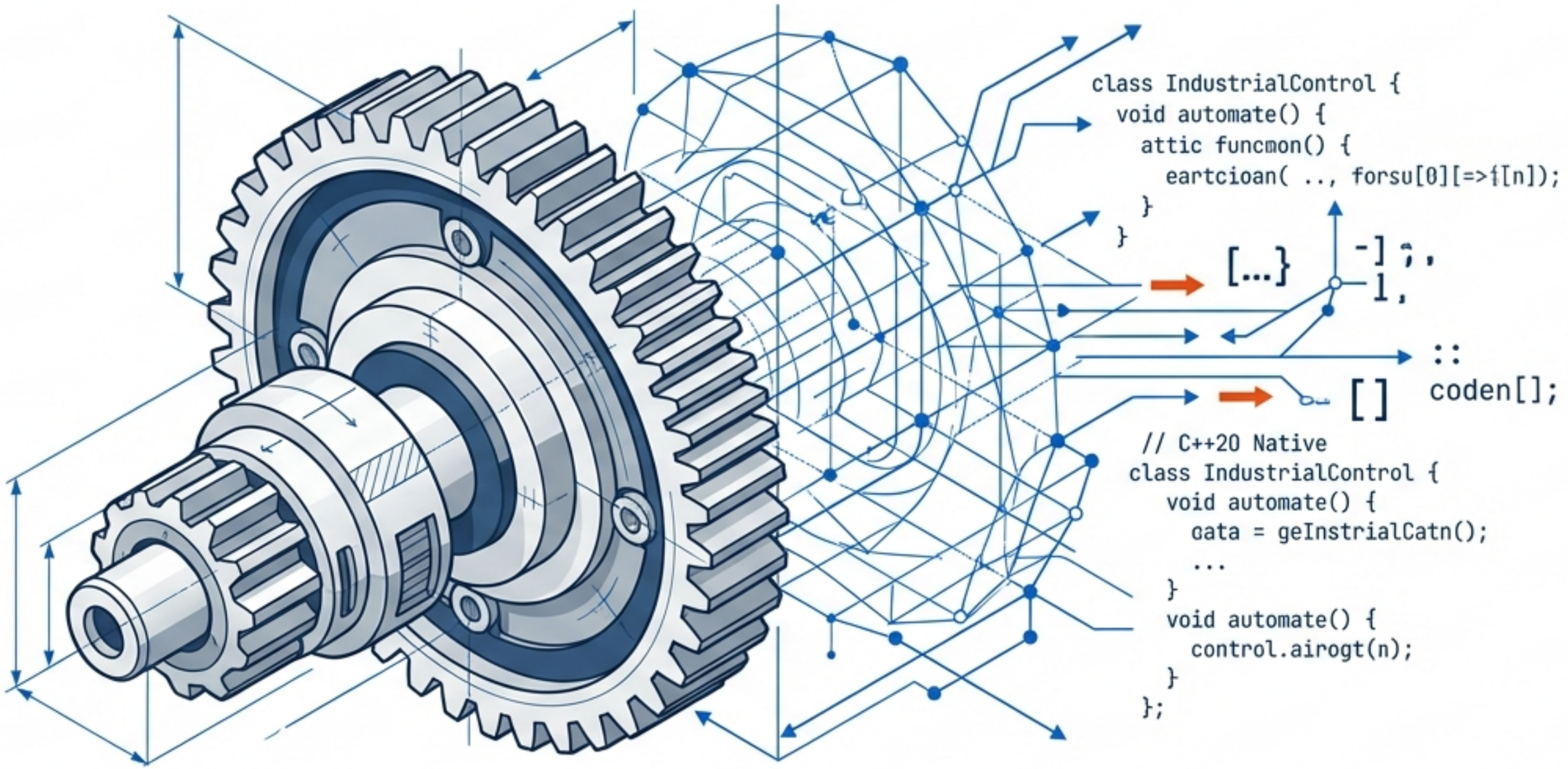
Object-Component Automation (OCA)

The Next-Generation Industrial Control Kernel

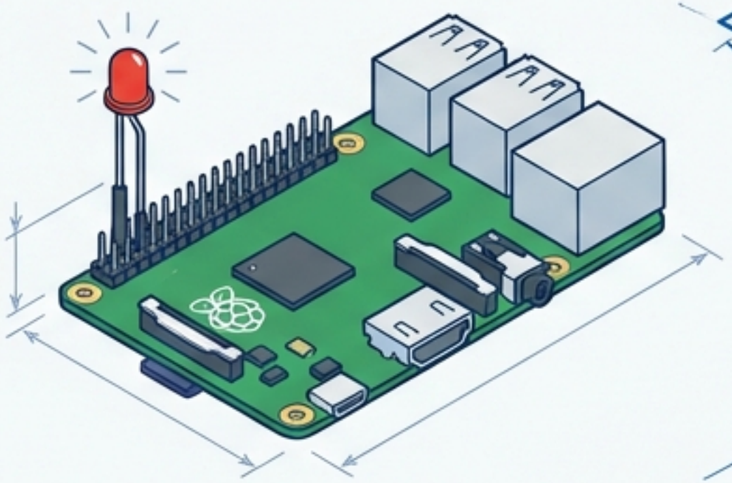


C++20 NATIVE | DETERMINISTIC | CROSS-PLATFORM

The Industry Paradox

Powerful, low-cost computing hardware is everywhere. Yet, ‘PC Control” remains a niche hobbyist endeavor. Why hasn’t the revolution happened?

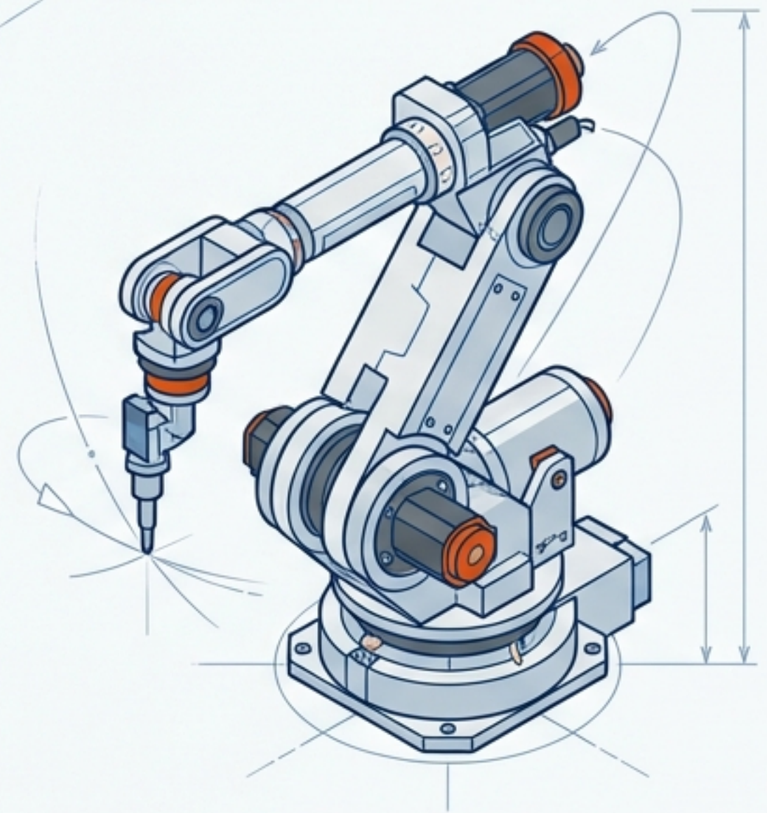
The Hobbyist Trap



Blinking an LED (Python)

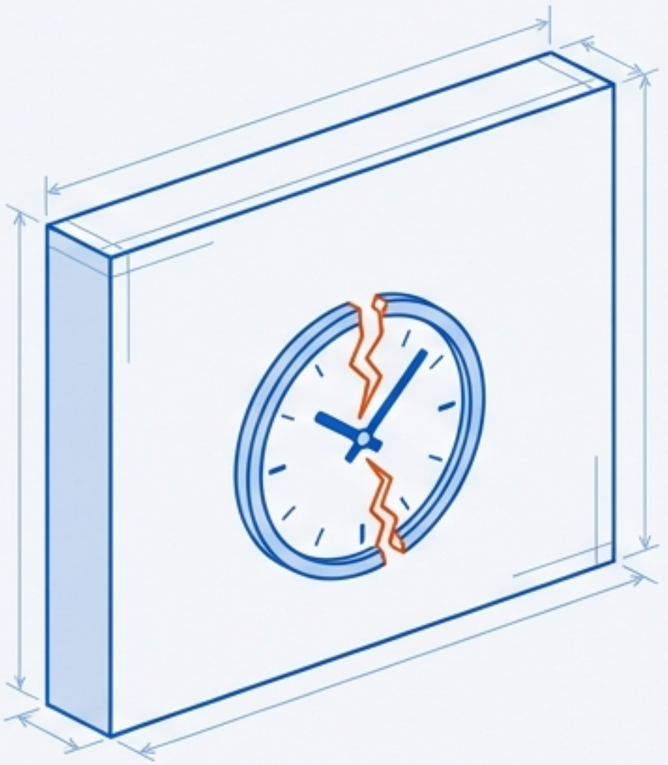
The Architecture Gap

Industrial Reality



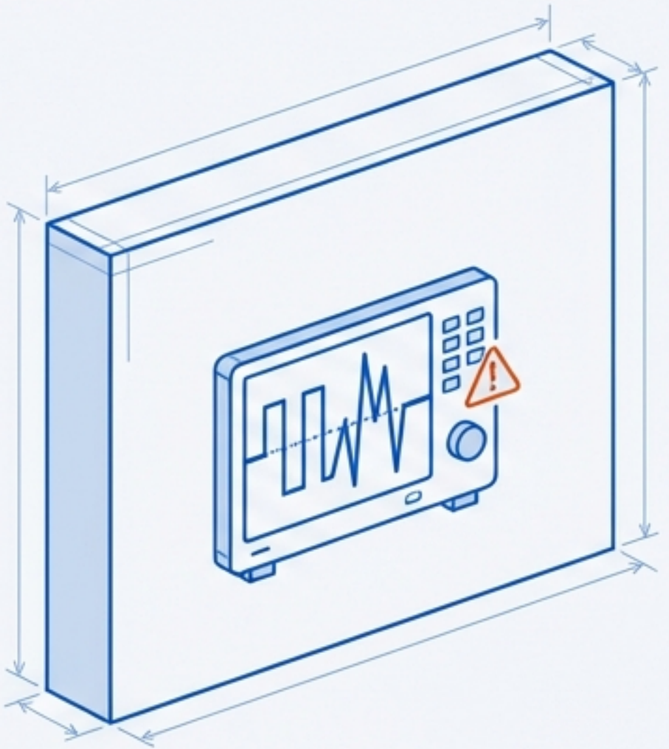
10µs Jitter-Free Control

The Three Walls of Failure



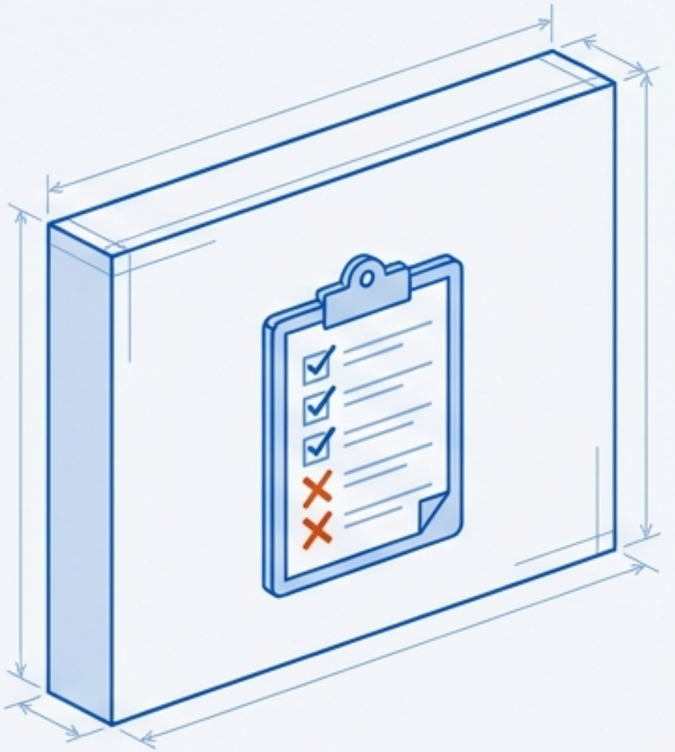
The IT/OT Divide

IT uses **Event-Driven code** (Wait –> React). **OT** requires **Scan-Based** strict timing. Mixing them causes **race conditions**.



The Reliability Gap

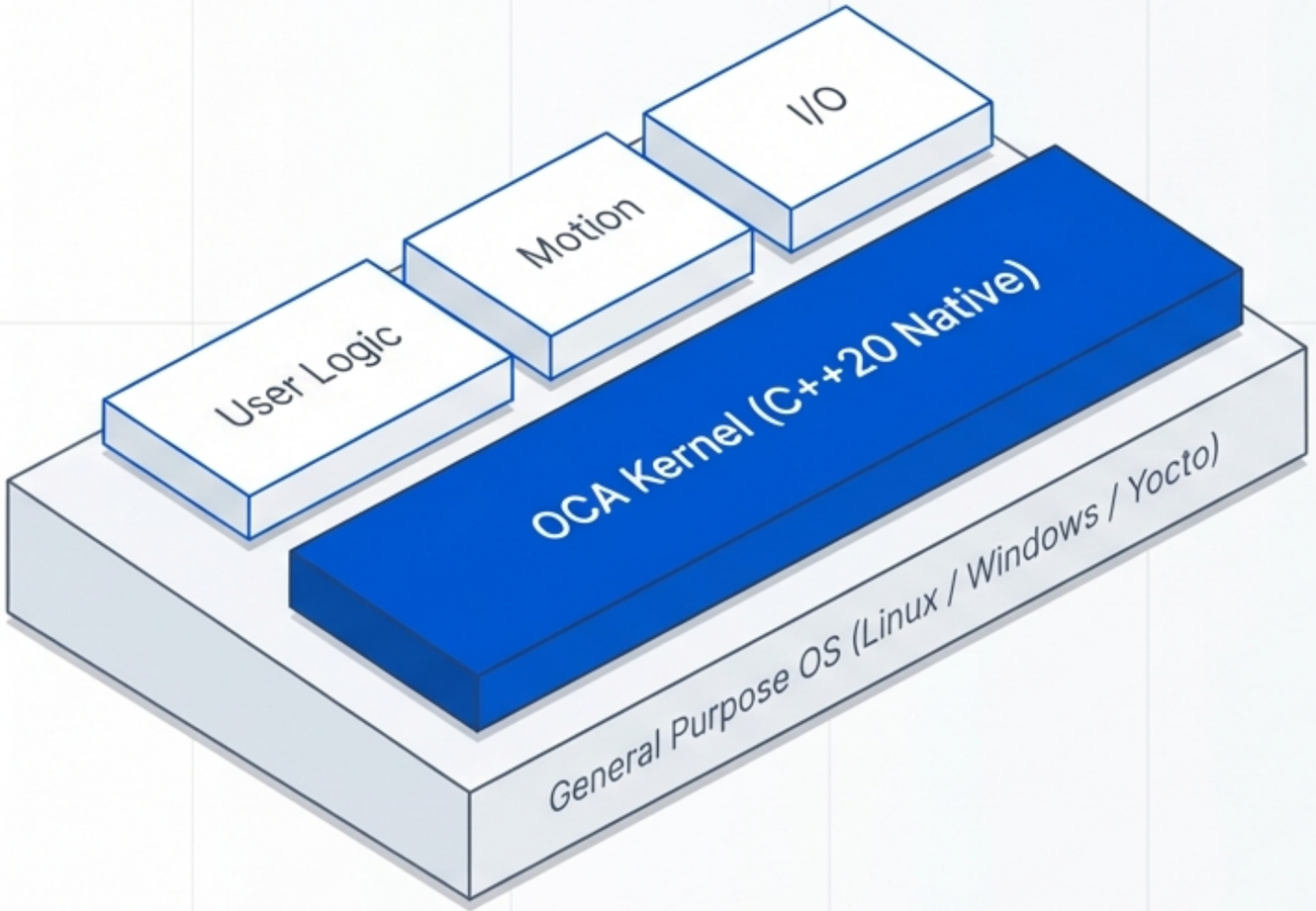
Running ≠ Production Ready. **Garbage Collection** (Java/Python) causes 10ms+ pauses. Control requires **determinism**.



The ‘Boring’ Essentials

DIY projects ignore the unglamorous necessities: **Protocol compatibility** (S7/EtherCAT), **Watchdogs**, and **Safety mechanisms**.

Introducing OCA: A Native Control Kernel



OCA is a **C++20 Native Control Framework**. **It combines the flexibility of IT with the determinism of OT.**

What It Is Not:

- NOT a runtime wrapper (like CODESYS VM)
- NOT a Python script or Node-RED flow
- NOT an emulator

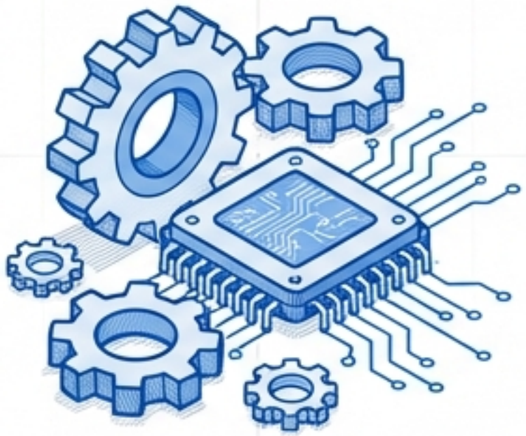
Engineering Determinism: The Process Image

Structurally eliminating race conditions within C++.

Snapshot (Input)

Atomic Capture.
Values Locked.

Compute (Logic)



Read-After-Write
Consistency.

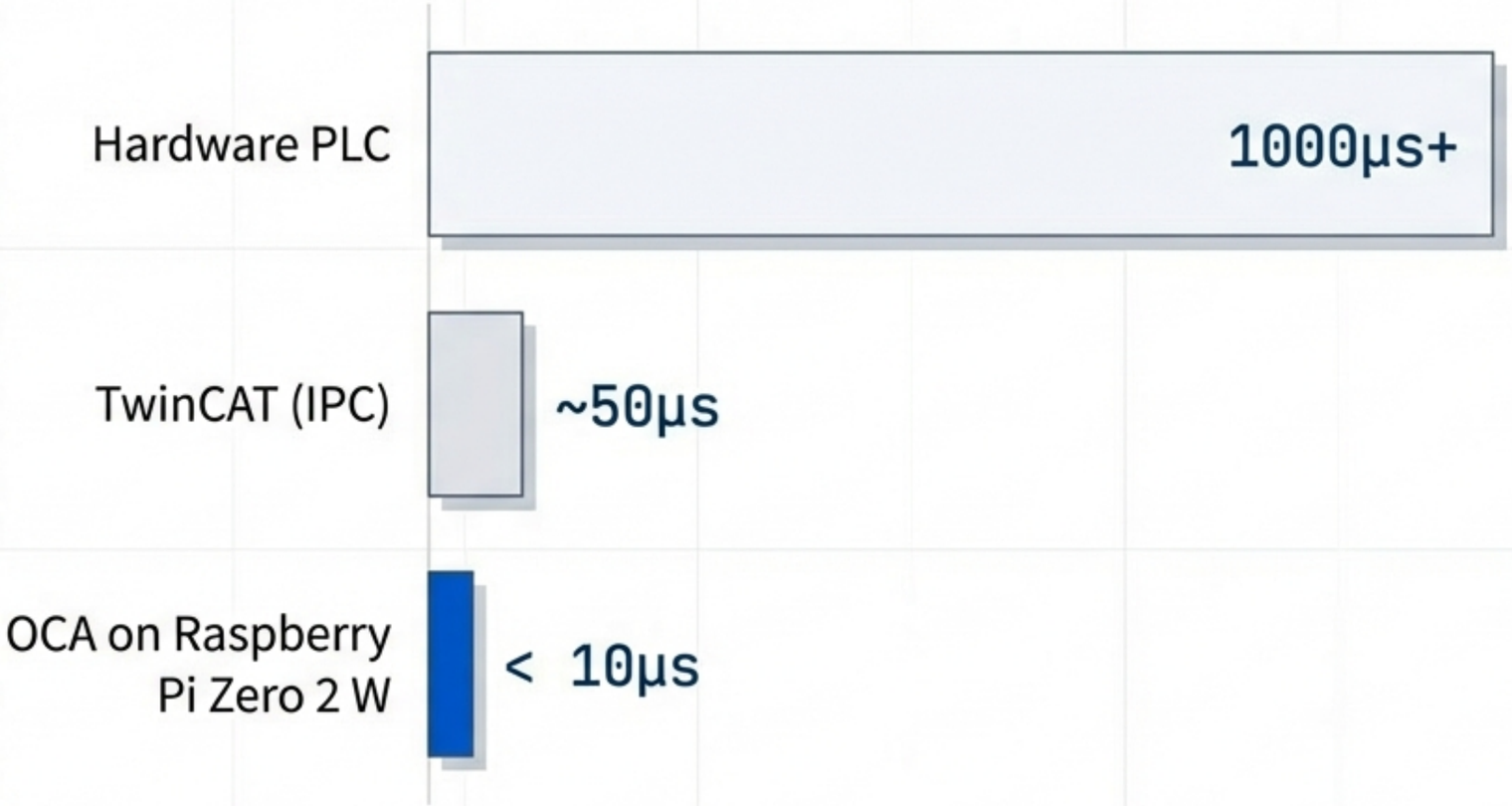
Commit (Output)

Bulk Transfer
to Device.

The strict PLC Scan Cycle is enforced at the kernel level, independent of OS interrupts.

Performance: The 10µs Standard

Cycle Times (Lower is Better)

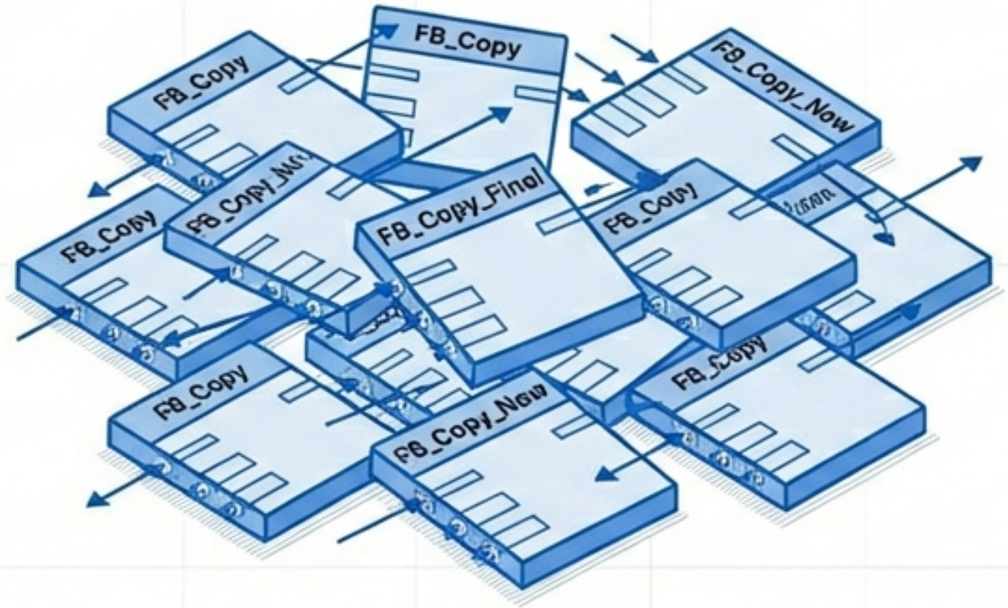


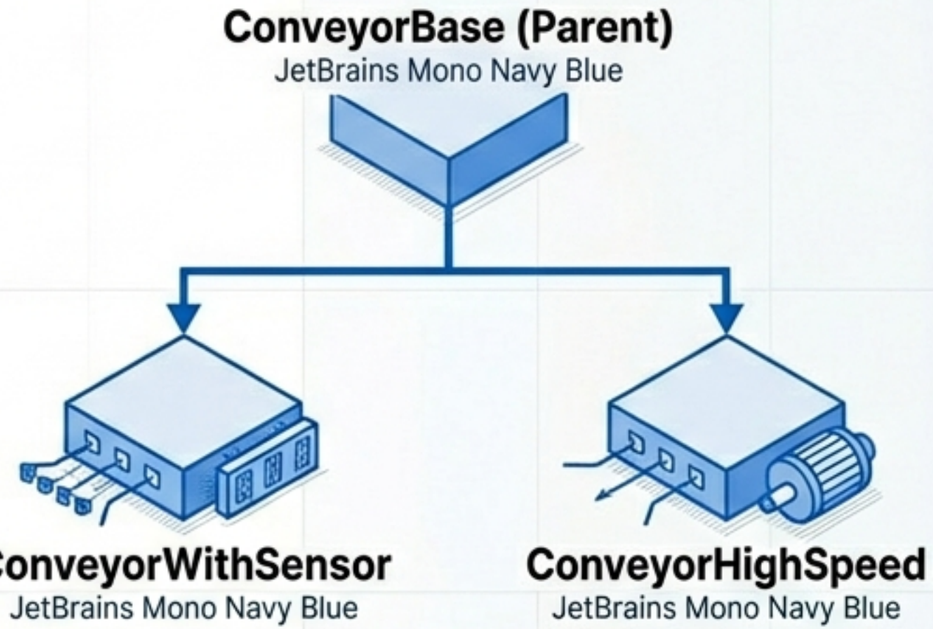
Zero Allocation Strategy:

- All memory (tags, buffers, instances) is pre-allocated at startup.
- No ‘new’ or ‘malloc’ during runtime = Zero Garbage Collection Pauses.

True Object-Oriented Control

The Old Way (Function Blocks)	**The OCA Way** (Inheritance)





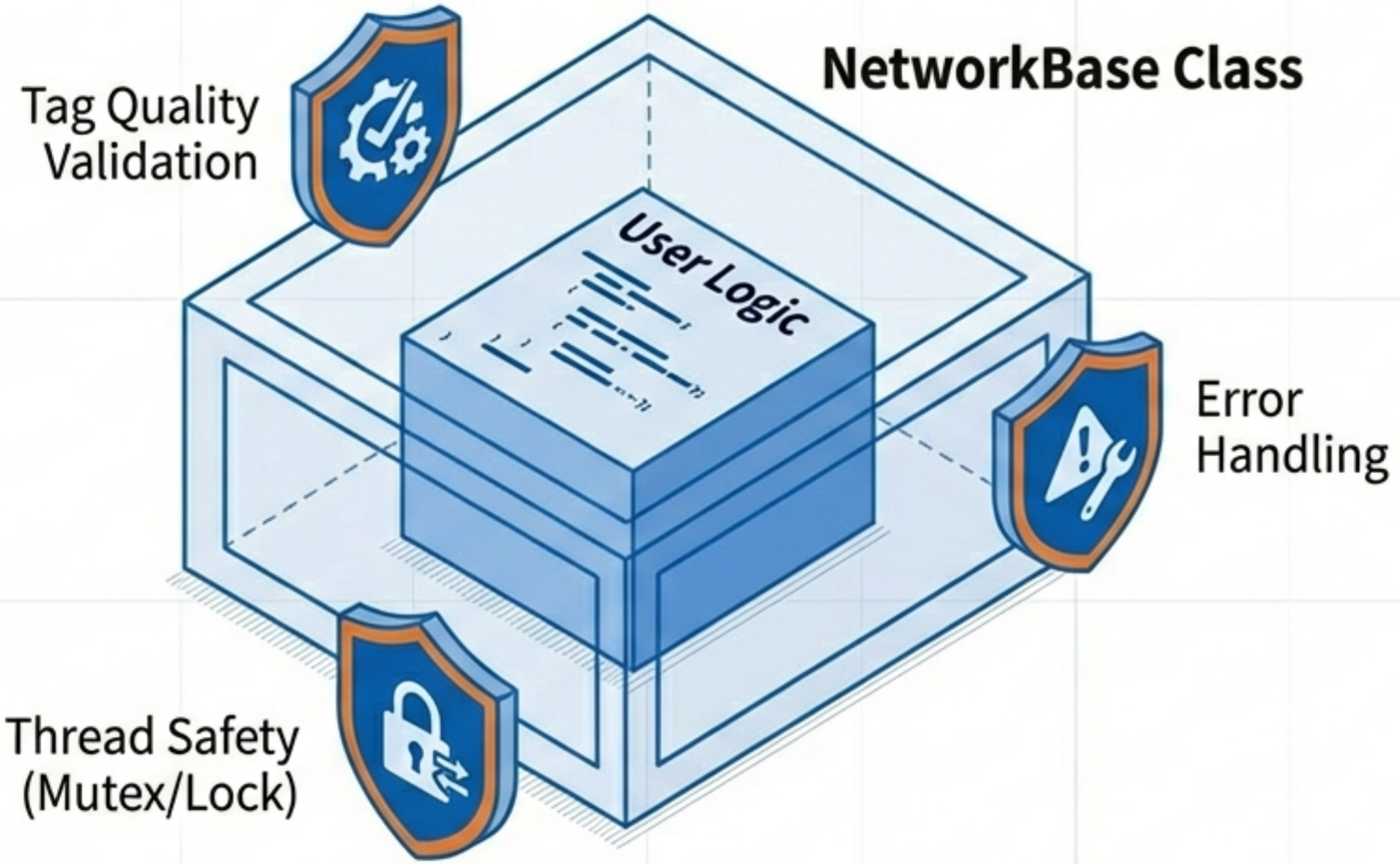
⚠ Code Duplication. Maintenance Nightmare.

Modify the Parent, update 100 children instantly.

Polymorphism: `The main loop simply calls 'conveyor->Run()'.`
The system executes the correct specific logic automatically.

Safety by Design: The 'NetworkBase' Class

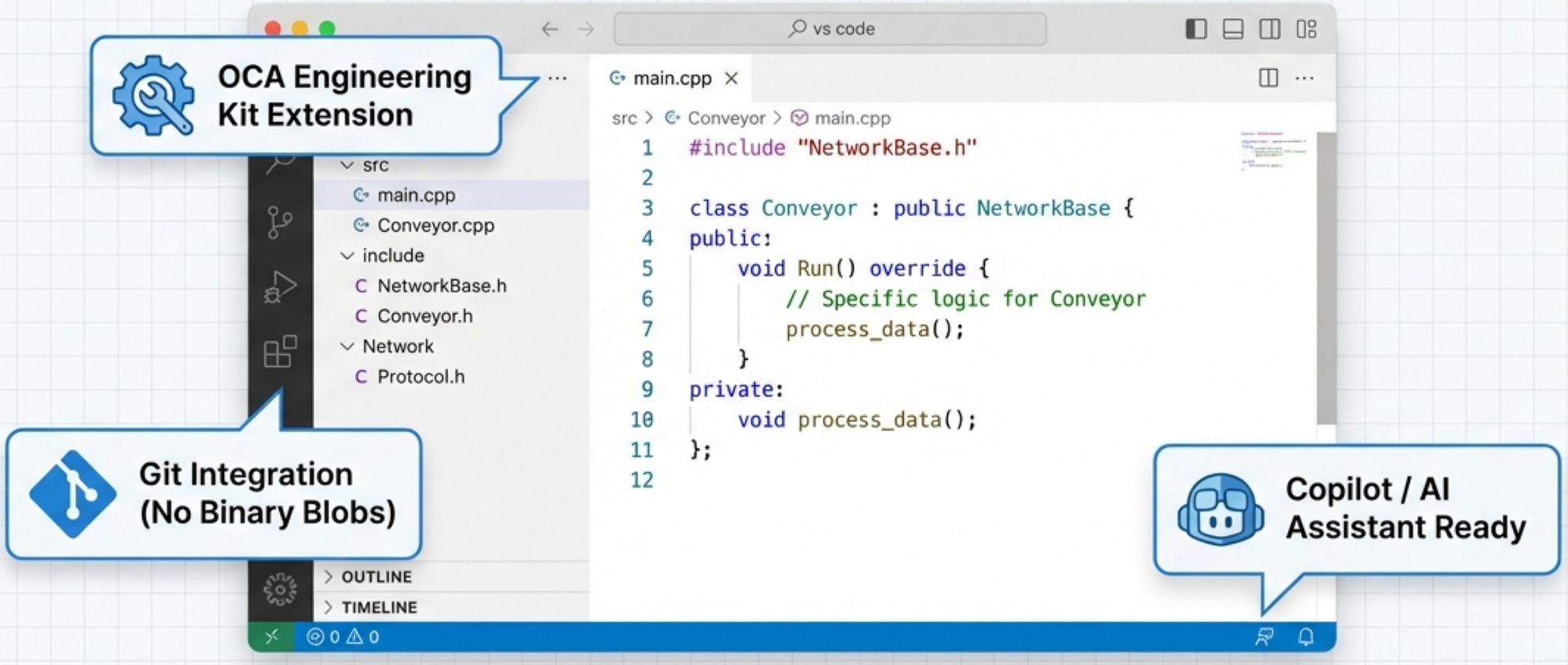
Complexity is hidden. Safety is inherited.



Developers simply inherit from NetworkBase. The framework handles the dangerous ‘plumbing’ in the background, ensuring safe code by default regardless of developer skill.

Inheritance = Automatic Safety.

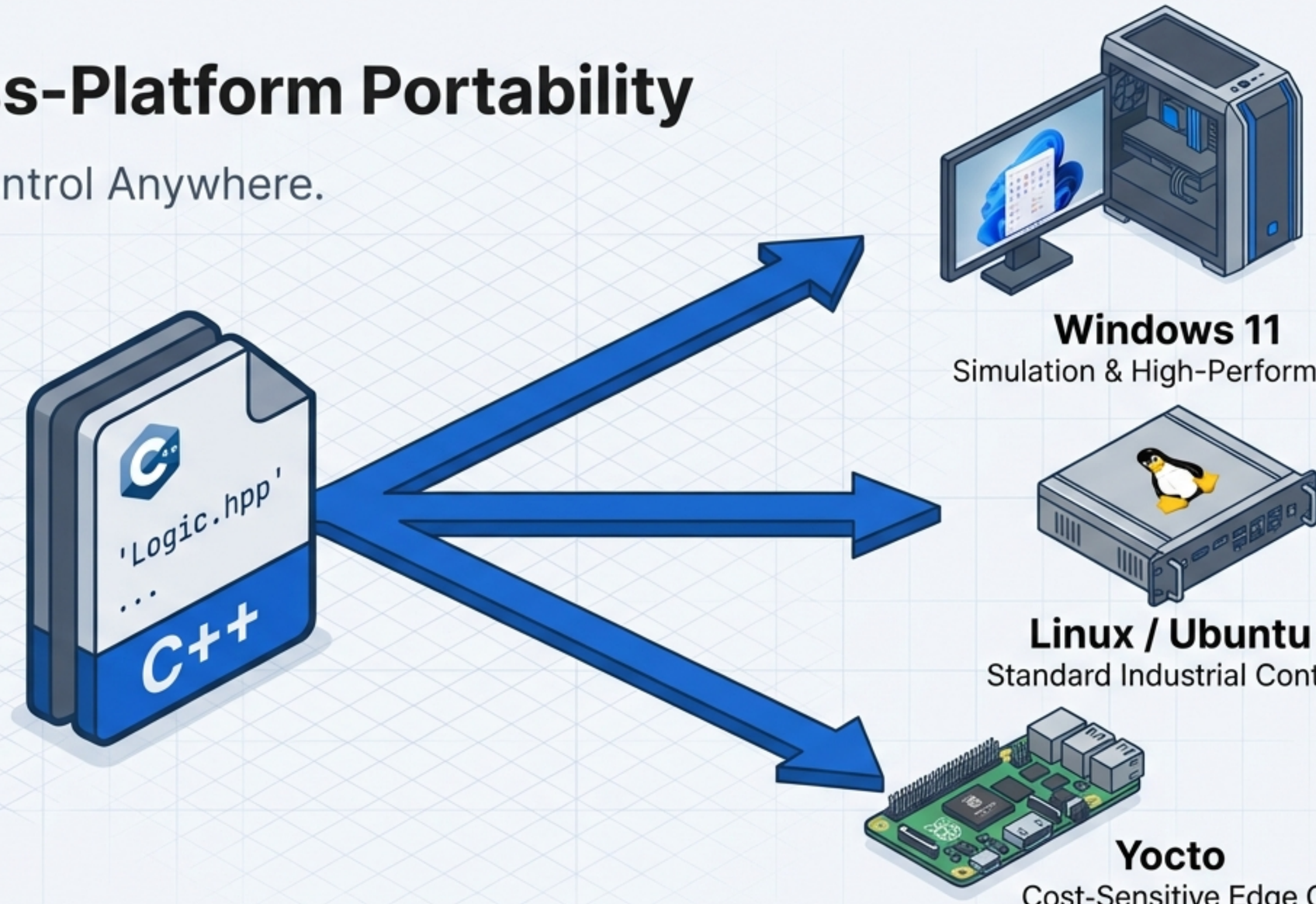
Unlocking the IT Ecosystem



One-click project generation. Header-only development. Modern CI/CD pipelines.

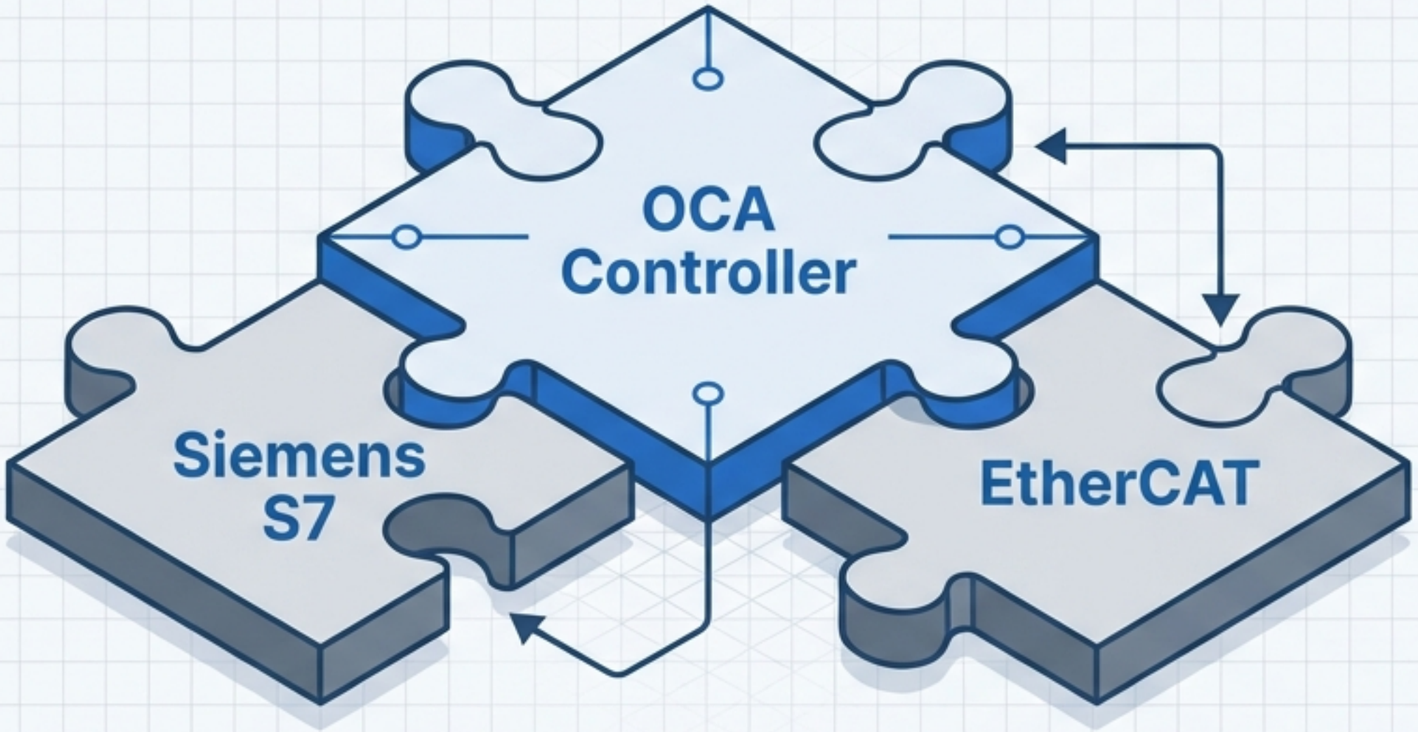
True Cross-Platform Portability

Write Once, Control Anywhere.



Infrastructure as Code: Automated deployment scripts (‘**deploy_oca.sh**’) enable rapid scaling.

Multi-Vendor Interoperability



Native support for S7 and EtherCAT allows mixing devices from different vendors.

The Dialects Library

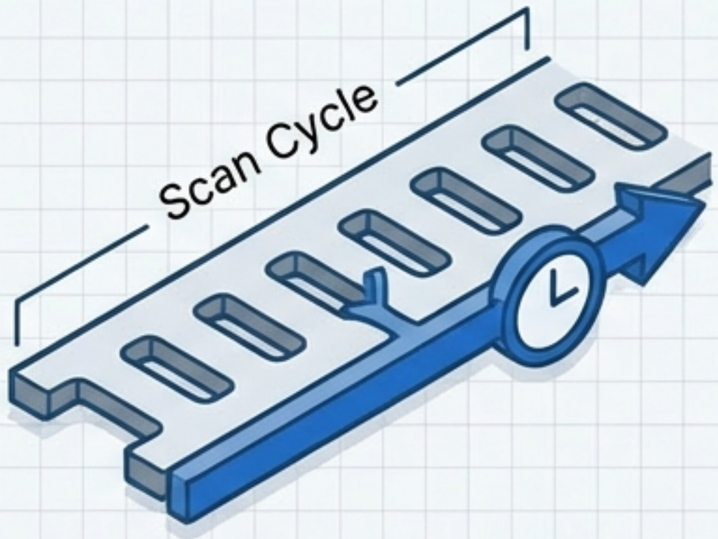
```
// Siemens Style
P_TRIG(this, 'StartBtn');

// Mitsubishi Style
PLS(this, 'StartBtn');

// Beckhoff Style
R_TRIG(this, 'StartBtn');
```

Write C++ using the logic names you already know.

Resilience in the Real World

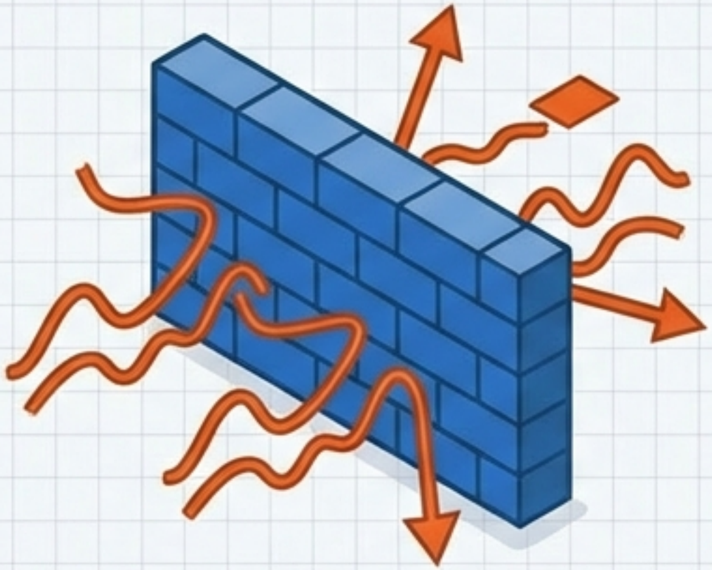
Built for Stability in Industrial Environments.

Cooperative Interrupts



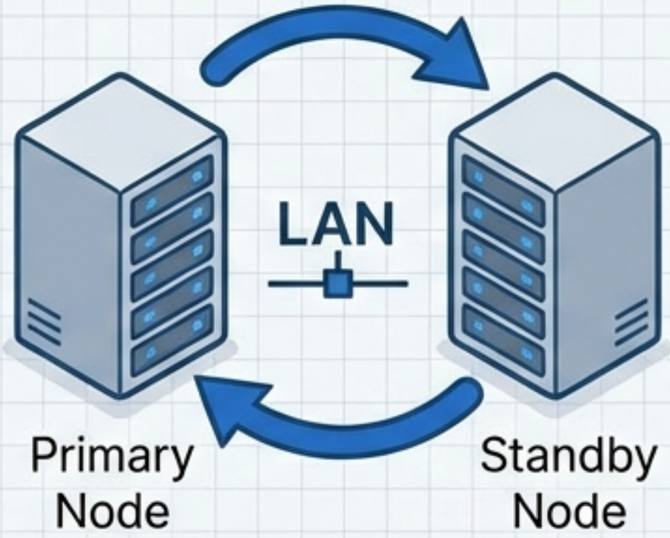
High-precision tasks scheduled at safe boundaries within the scan cycle. Jitter-free.

Network Storm Protection



Hardened against IP floods and invalid packets. Bad data cannot crash the kernel.

N-Redundancy



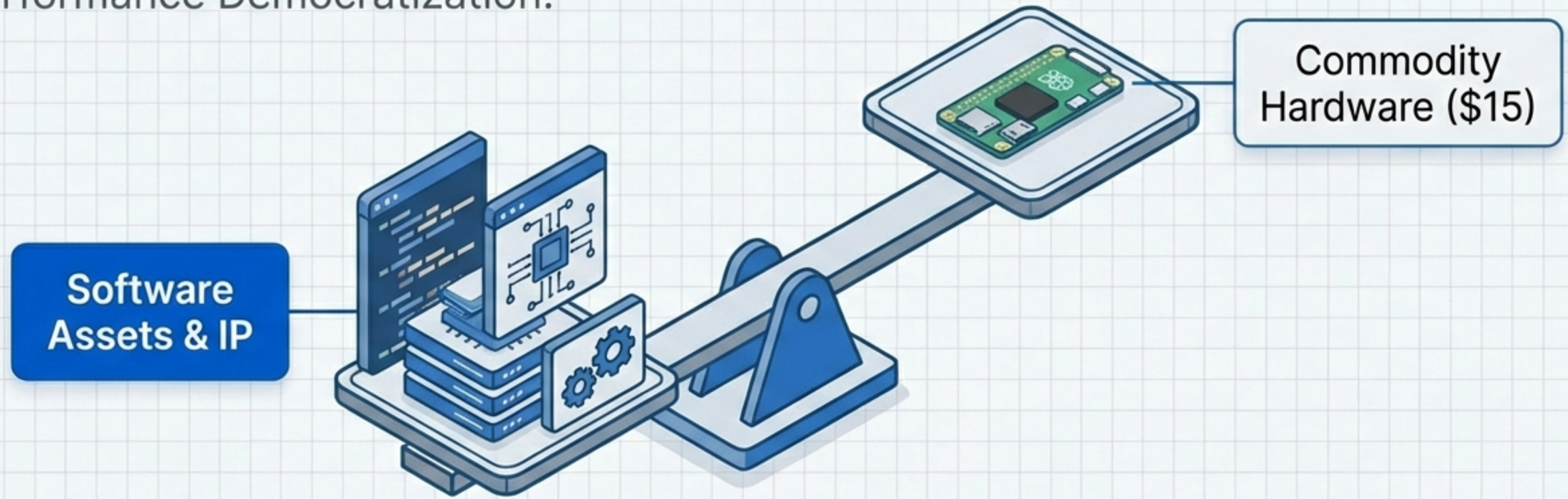
High Availability failover using standard LAN.

The Competitive Landscape

	Hardware PLC	SoftPLC (e.g. CODESYS)	OCA
Cost	High	Medium	Low
Architecture	Proprietary	Black Box Runtime	White Box (Native)
Speed	Standard	Moderate	**Extreme (<10µs)**
Flexibility	Vendor Lock-in	License Bound	Open Ecosystem

The Economic Impact

Performance Democratization.



- Achieving high-end PLC performance on $15 hardware (Raspberry Pi Zero 2 W).
- The Shift: Removing the ‘Performance Tax’ imposed by hardware vendors. Value transfers entirely to your Intellectual Property.

The Industrial Kernel for the Next 40 Years

Source Sans Pro Stop building on legacy paradigms.
Start building on a true Industrial OS.