

```
> SYSTEM_TARGET: WVS_TELEMETRY  
> ARCHITECTURE: STM32 + LABVIEW  
> STATUS: INITIATING_CASE_STUDY
```

Engineering Reliability: The WVS Modernization

Transforming a Failing Prototype into an
Enterprise-Grade Telemetry System



MISSION-CRITICAL TELEMETRY ACHIEVED

Control Stack Engineering was tasked with rescuing the Wireless Vibration Analysis System (WVS)—a prototype paralyzed by extreme technical debt, memory leaks, and fatal hardware reboots during field testing.

LATENCY

<200ms

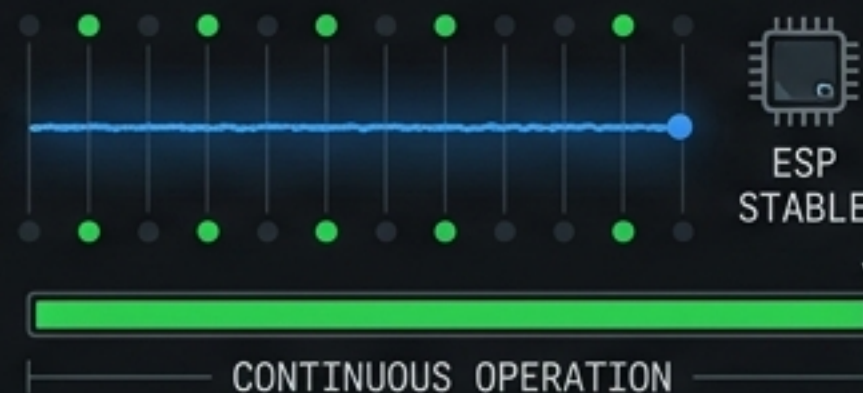
Real-time graphical response
to physical sensor taps



UPTIME

100%

Zero ESP/hardware reboots
during continuous streaming



EFFICIENCY

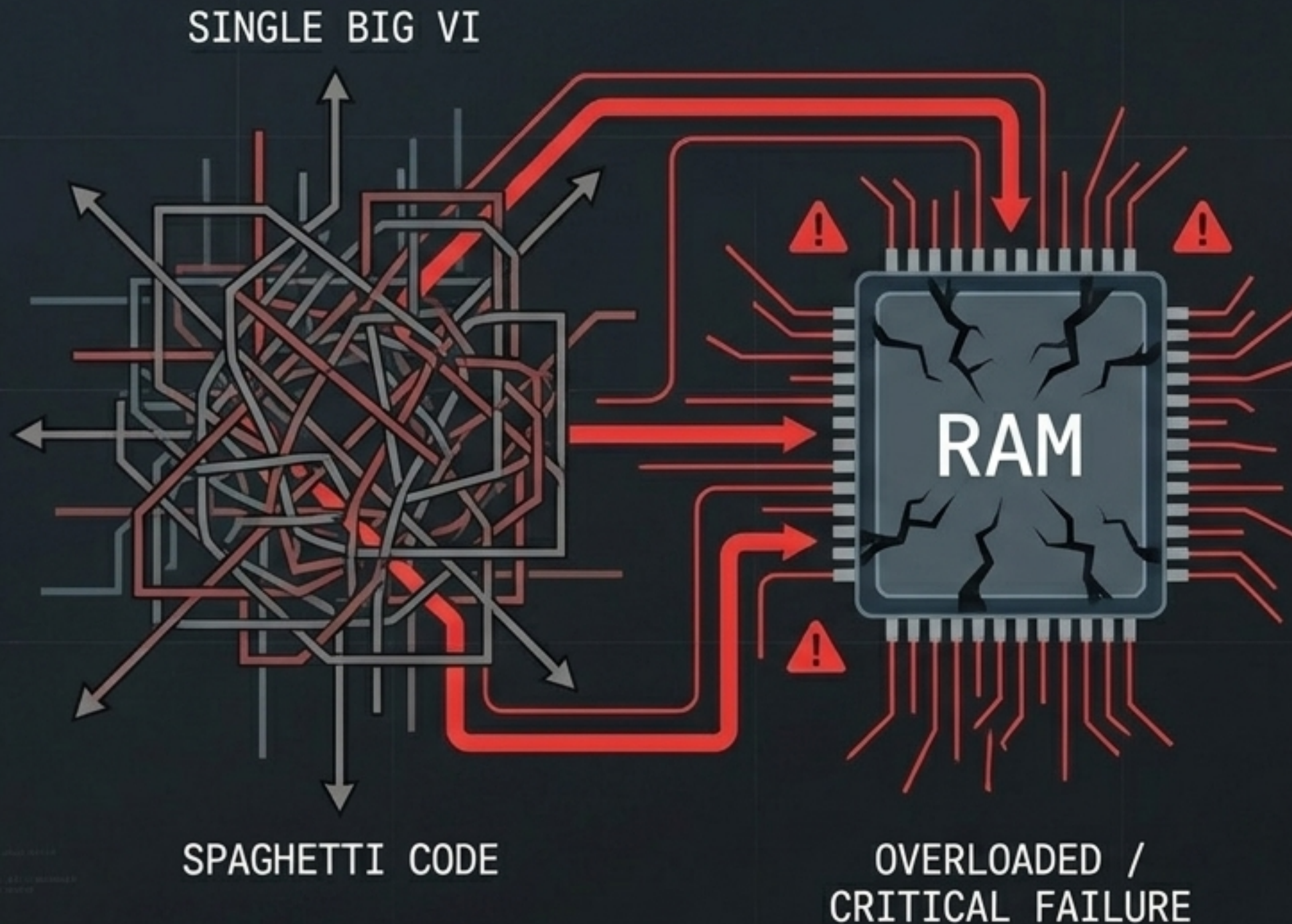
DYNAMIC

Resource-light architecture
tailored for tablet deployments



RESOLUTION: FULL-STACK ARCHITECTURAL OVERHAUL

THE STARTING POINT: ANATOMY OF TECHNICAL DEBT



THE MONOLITH

A rigid Single Big VI housing all logic. Parallel loops with independent event structures resulted in catastrophic race conditions.

MEMORY ASPHYXIATION

Live field tests forced the system to cache massive vibration datasets directly into RAM, instantly crashing resource-limited field tablets.

HARDWARE REBOOTS

Null parameters passed to inefficient firmware queues caused recurring fatal errors, forcing the ESP microcontrollers into continuous reboot cycles.

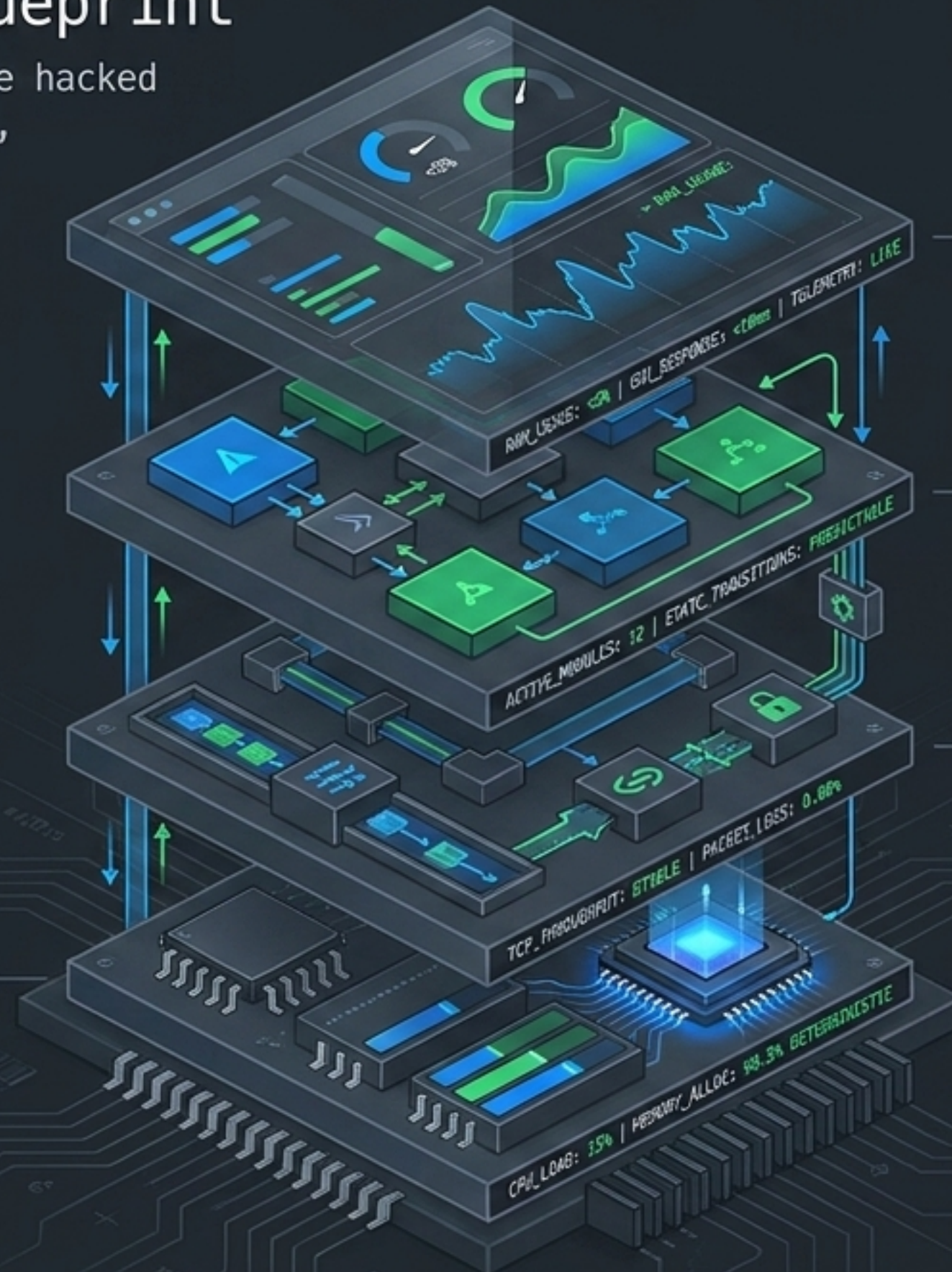
RIGID UX

Hardcoded, fixed-resolution interfaces that shattered when deployed across varying field tablet screens.

The Modernization Blueprint

True industrial reliability cannot be hacked at the surface. It requires systemic, full-stack engineering discipline.

Layer 1: Firmware & RTOS
Bare-metal memory and CPU
task determinism



Layer 4: Data & UX
Weightless RAM and responsive,
color-coded telemetry

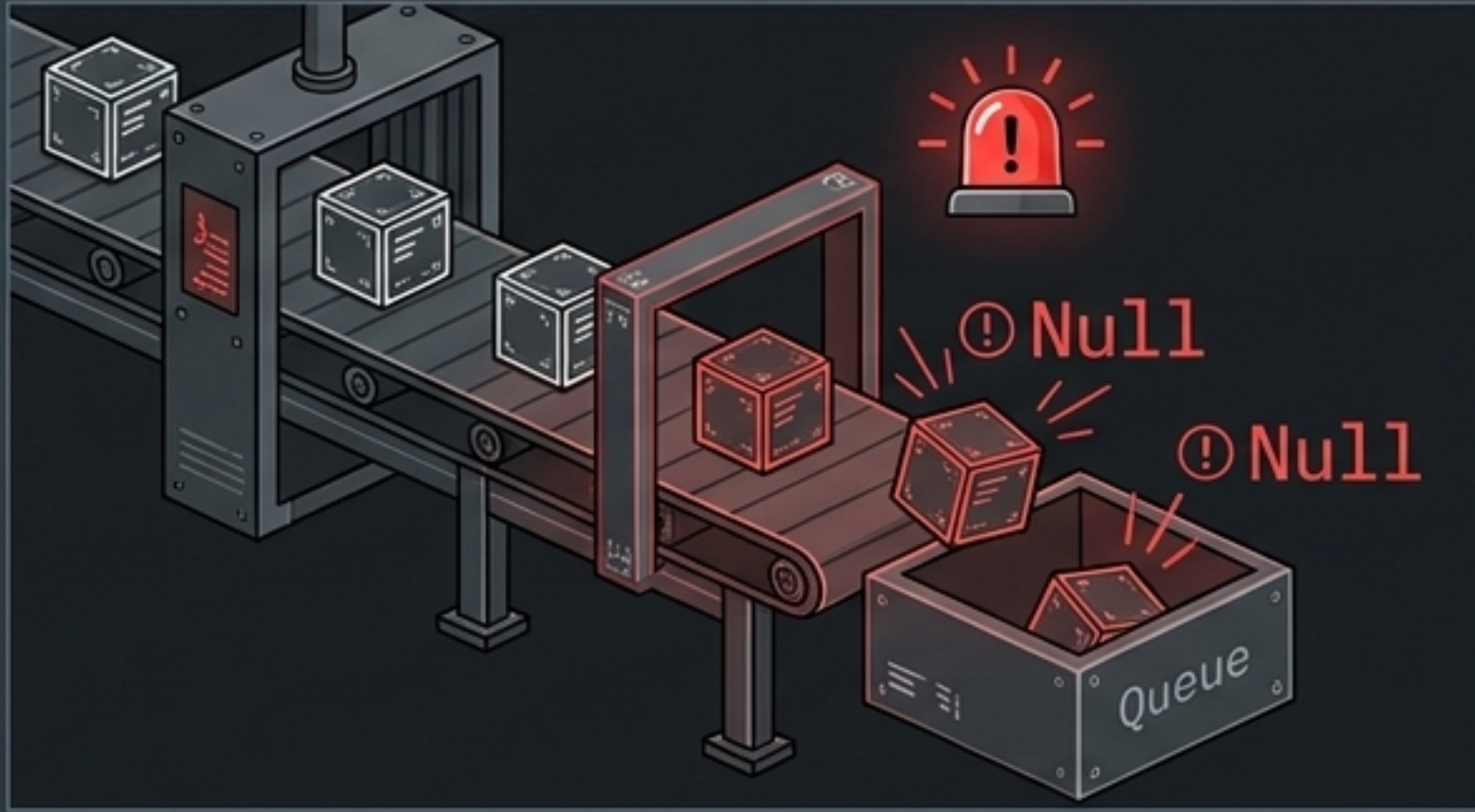
Layer 3: Software Architecture
Modular, dynamically loaded
state machines

Layer 2: Communication Pipeline
Secure, leak-proof TCP/HTTP sockets

> DEPLOYING_MODULAR_FRAMEWORK...

Layer 1: Bare-Metal Memory Optimization

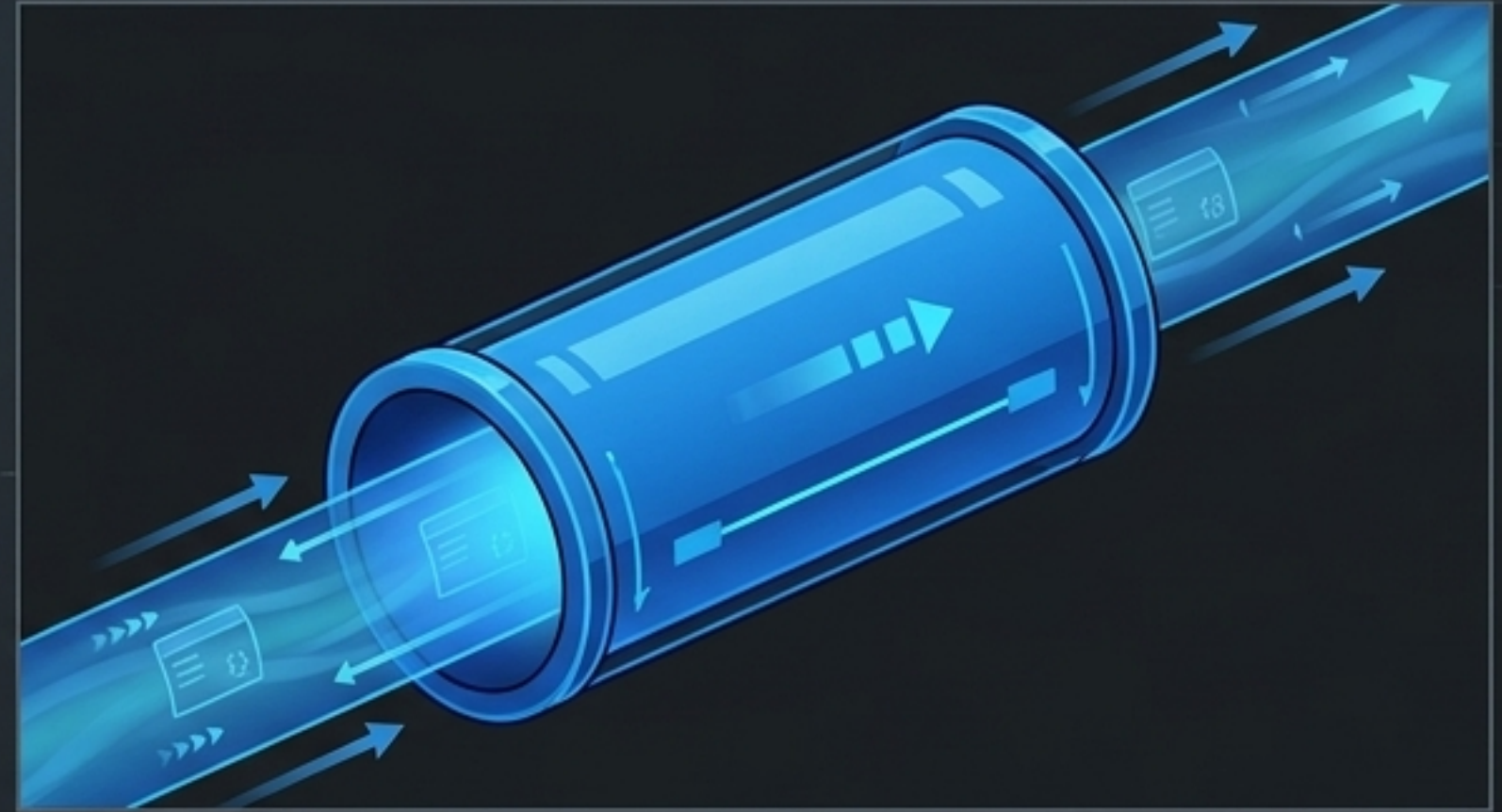
Legacy Data Queues



Deprecating Legacy Queues

Removed the error-prone queue architecture that was passing null parameters and triggering hardware crashes.

STM32 Stream Buffers

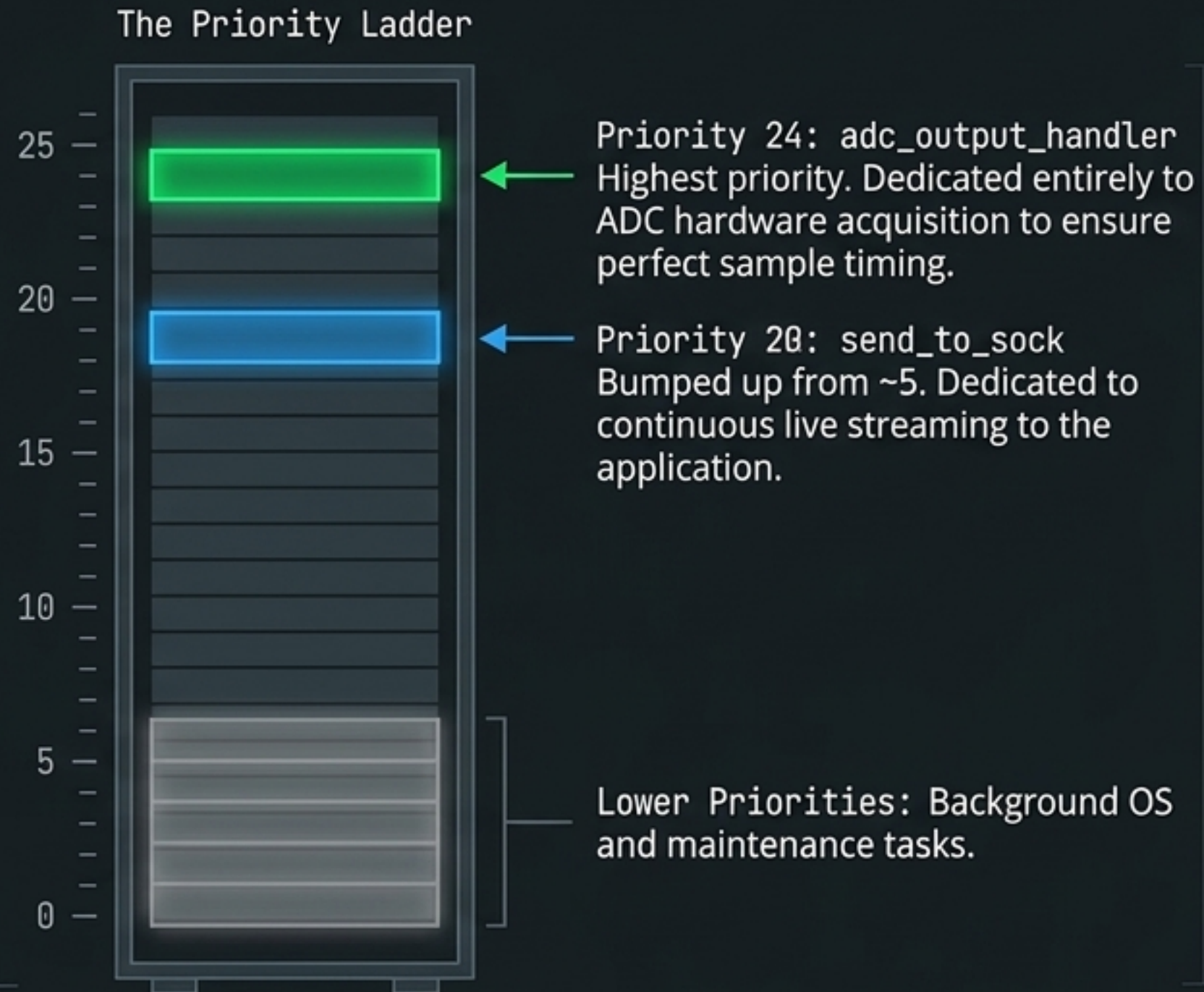


STM32 Stream Buffers

Implemented high-speed stream buffers to handle the continuous influx of accelerometer data, drastically reducing memory overhead and eliminating the root cause of the sensor reboot loop.

Layer 1: Deterministic RTOS Prioritization

To achieve zero packet loss during continuous vibration streaming, we re-architected the FreeRTOS task scheduler.



By isolating and elevating the acquisition and TCP streaming tasks, we eliminated resource starvation and ensured the STM32 firmware never choked during high-frequency field tests.

Layer 2: Pipeline Stabilization & Socket Management

The HTTP Command Layer



The TCP Streaming Layer



Orphaned Socket Eradication

In the legacy system, closing the LabVIEW tablet app left zombie TCP sockets running on the sensor, eventually exhausting memory.

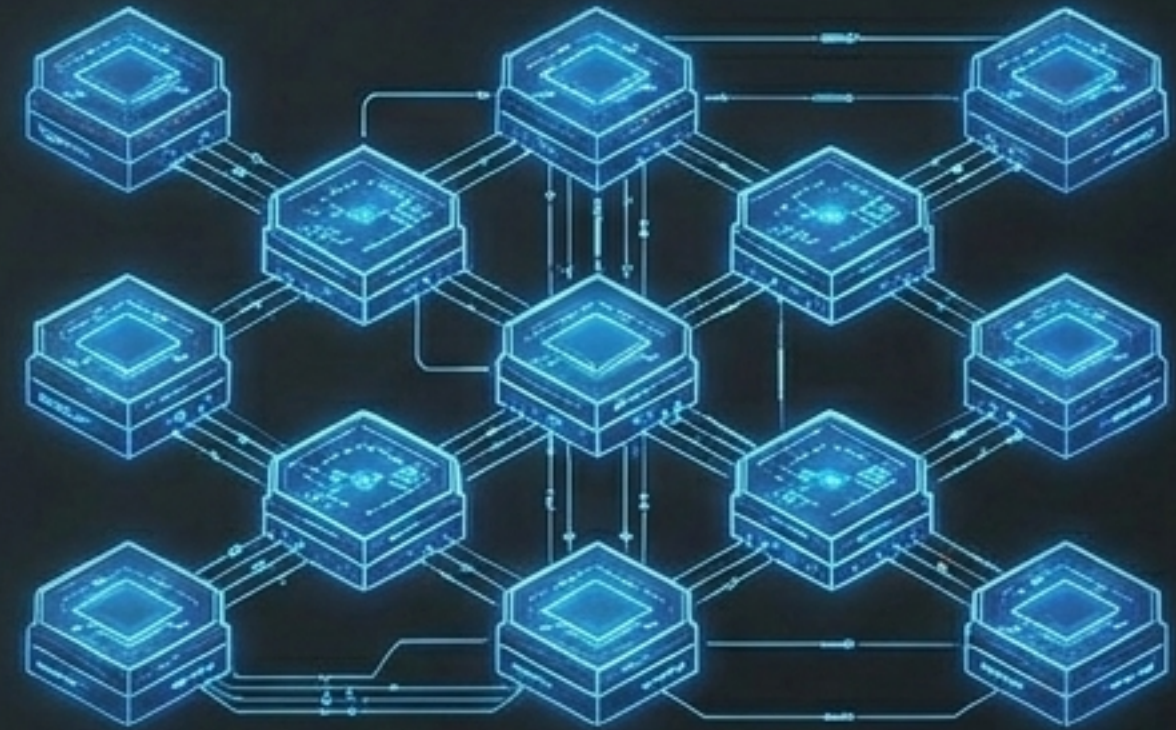
Continuous Polling Loop

Engineered a dedicated firmware loop that aggressively polls the socket state. If the remote application disconnects, the firmware instantly detects the drop, cleanly terminates the socket, and frees the memory block.

Layer 3: Transition to the Queued State Machine



Single Big VI



QSM Architecture

Decoupled Logic

Replaced overlapping event structures with a robust Queued State Machine (QSM).

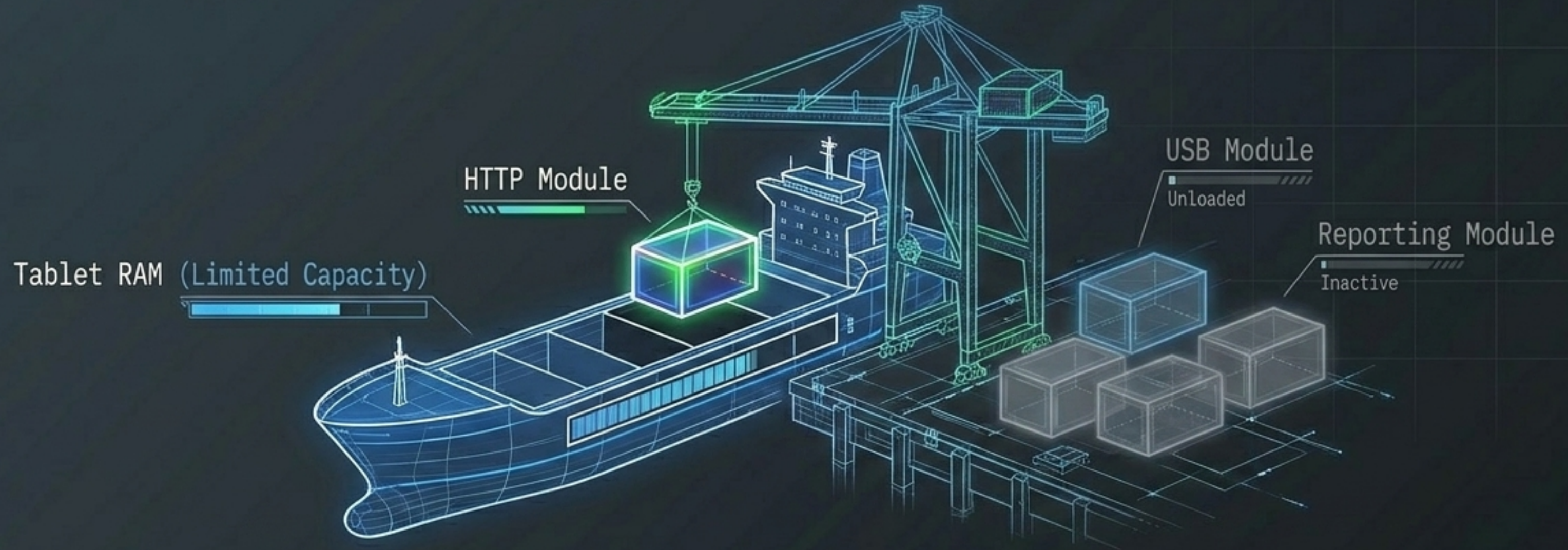
Parallel Processing

Separated the producer loop (handling UI events) from consumer loops (handling HTTP communication and data processing).

Dedicated Stream Handlers

Isolated the intense TCP socket streaming into its own standalone loop to prevent UI freezing during high-speed data acquisition.

Layer 3: Dynamic Memory Footprint



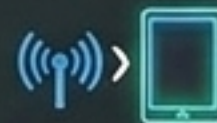
On-Demand Loading

Modules remain locked out of memory until explicitly called.



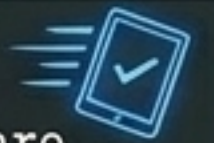
Contextual Resource Allocation

The HTTP Module is only loaded into RAM when wireless sensors are actively selected by the user. If USB sensors are used, the wireless module unloads completely.



Result

A lightweight, agile software footprint that easily operates on highly constrained tablet hardware without risk of crashing.



Layer 4: The 'Weightless RAM' Data Strategy

Field testing requires continuous, hours-long live streaming.
Storing this in RAM was mathematically impossible for the hardware.



The Insight

By streaming data continuously to a **background binary file**, the tablet's disk drive handles the heavy lifting. RAM is strictly reserved for reading the small, active 'sliding window' required to draw the UI graph, rendering memory limits irrelevant.

Layer 4: Responsive & Deterministic Telemetry



Fluid Responsiveness

Eradicated the rigid, hardcoded coordinate system. The UI now auto-scales to any tablet resolution flawlessly.

Cognitive Color-Coding

Sensor data is instantly identifiable.

[NEON BLUE & GREEN]: Standard vibration mapping sensors.

[BLACK]: The Reference Sensor (dedicated to calculating phase difference and time-synchronization across the machine).

Audio Diagnostic Integration

Processed vibration data is converted to audible frequencies, allowing engineers to literally hear the machine's mechanical state through the tablet.

Synthesis: Prototype vs. Enterprise-Grade

Dimension	Legacy Prototype	Production-Ready WVS
Architecture	Monolithic Single Big VI	Modular QSM / Dynamic Loading
Data Handling	RAM-Constrained Bottleneck	Weightless Disk Streaming
Firmware Execution	Rigid Queues / Unprioritized	Stream Buffers / FreeRTOS Priority 24
System Stability	Fatal Memory Leaks / ESP Reboots	100% Deterministic Uptime
User Interface	Fixed Resolution / UI Freezes	Responsive / <200ms Latency

> VALIDATION: SYSTEM PASSES ALL PRODUCTION CHECKS

The Guarantee of Engineering Discipline

Real-Time Feedback Latency: < 200 Milliseconds

The modernized WVS translates physical sensor taps into visual telemetry in under 200ms. By applying rigorous, full-stack engineering discipline—from bare-metal RTOS prioritization to dynamic UI rendering—we proved that true industrial reliability is never a single hack. It is a system-wide architecture.