MAVERICK: Hardware-Free Validation for AV Control Systems

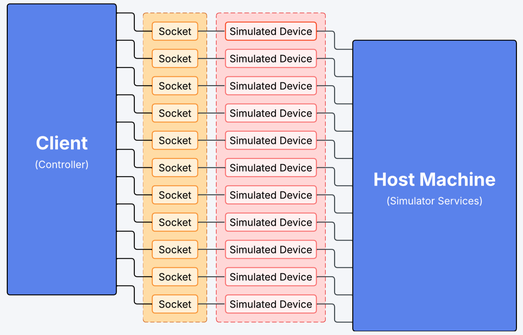
Pershing Technologies Senior Design Project

Kenneth Blake, Linh Cao, Rolana Duran, Daniel Epstein, Noor e Hira, Joelyne Jacob
Stakeholder: Matt Jackson, Pershing Technologies | Drexel University CCI

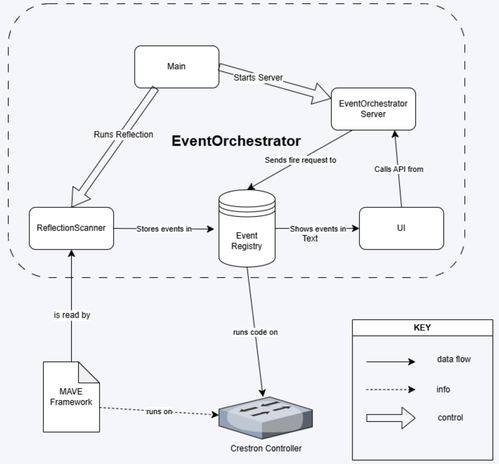
Problem Statement

Pershing Technologies develops enterprise AV control systems for complex environments, including for the Department of Defense. Their codebase, MAVE, is divided across four interconnected developer pillars running on a Crestron controller with limited debugging capability. Developers have no way to test their code without physical access to expensive hardware at secure facilities — creating bottlenecks, reducing test coverage, and increasing the risk of defects in environments where failure is not acceptable. MAVERICK eliminates this dependency through a 100% software-based validation suite that runs entirely on a local Windows workstation.

Objective 1: Socket Server – System Architecture



Objective 2: Event Orchestrator – System Architecture



Objective 1: Socket Server

A TCP-based server that emulates 11 physical AV devices, allowing Pershing's control code to send commands and receive protocol-accurate responses without any hardware present. All device behavior is defined in a JSON configuration file — no source code changes required to add or modify devices.

Device Type	Key Commands
BluRayPlayer	PLAY, NEXT, LAST
Camera	POWER, PAN, TILT, ZOOM
Tuner	SET, CHANNEL UP, CHANNEL DOWN
Display	POWER, INPUT, VOLUME
DSP	VOLUME, MUTE
Lighting	PRESET
MatrixSwitch	AV ROUTE, VIDEO ROUTE, AUDIO ROUTE
PowerController	OUTLET SET
SinagePlayer	SET, UPDATE
VideoWall	LAYOUT
VideoTeleconferenceCodec	DIAL, HANGUP, SHARE

How it Works

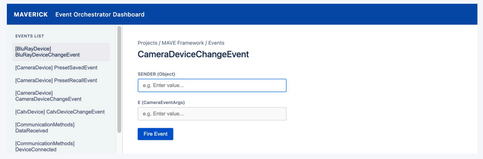
Incoming commands are matched against regex patterns with named capture groups. A mini-scripting engine then executes a sequence of actions — updating state variables and returning a formatted response.

Key Capabilities

- Stateful emulation — maintains power state, volume, input, presets per session
- Protocol-accurate responses matching exact real-hardware formatting
- Supports 5+ concurrent client connections across all 11 devices
- Comprehensive session-level logging via Serilog
- 12 xUnit test suites, 150+ test cases across all device types

Objective 2: Event Orchestrator

A reflection-based tool that scans compiled MAVE assemblies at runtime, discovers every event and its full parameter schema, wires live delegates, and serves a web UI so developers can inject events into an isolated code pillar — without requiring the other three pillars to be present or complete.



How it Works

Scan compiled DLLs → Discover event signatures → Wire live delegates → Serve dynamic web UI → Fire event & capture logs

Key Capabilities

- Automatically adapts as the MAVE codebase evolves — no hardcoded event lists
- Live delegate wiring enables authentic execution, not text-based simulation
- Embedded web UI served directly from the compiled assembly — no external server required
- Single executable deployment, localhost only, zero external dependencies
- Real-time log interception captures full execution trace for debugging

Tech Stack

- .NET 8 – Socket Server
- .NET 10 – Event Orchestrator
- C#
- xUnit + Moq
- TCP Sockets
- System.Reflection
- Serilog
- HTML / JavaScript

Design Decisions & Trade-offs

Data-Driven vs. Hardcoded Logic

- Chose JSON configuration over hardcoded device behavior for the Socket Server
- Trade-off: Added parser complexity vs. operational flexibility for Pershing
- Result: Pershing can add devices without code rebuilds

Reflection vs. Static Analysis

- Chose runtime reflection over Roslyn static analysis for Event Orchestrator
- Trade-off: Runtime overhead vs. ability to wire live delegates
- Result: Authentic execution on Crestron controller, not text-based mocks

Embedded UI vs. External Server

- Chose embedded HttpListener over Node.js/React
- Trade-off: Limited UI capabilities vs. single-executable deployment
- Result: Zero installation overhead, runs immediately on developer workstations

Impact & Key Outcomes

Technical Achievements

- Successfully emulates all 11 AV device types with protocol-accurate behavior
- Handles 5+ concurrent connections with independent session state isolation
- Real-time log capture provides complete execution trace for debugging
- Automatic event discovery adapts as MAVE codebase evolves

Operational Impact

- Eliminates hardware dependency during development
- Enables parallel development across 4 independent pillar teams
- Reproducible test environments across all developer workstations
- Single executable deployment with zero external dependencies

Acknowledgements

Matt Jackson — Stakeholder, *Pershing Technologies*
Dr. Filippos Vokolos — Faculty Advisor, *Drexel University*
Drexel University College of Computing & Informatics