

ACTIONGRAPH

A Complete, Cost-Sensitive Motion Control Platform

AI-Ready

MCU to Embedded PC

Cost-Effective Motor Control

Innovative Input Language

THREE COMPETITIVE ADVANTAGES



AI-READY ARCHITECTURE

Declarative YAML language designed for both human and LLM code generation.
Documentation-as-API.



SCALES ACROSS HARDWARE

Same program runs on a \$5 MCU or an embedded Linux PC. No code changes, no recompilation of logic.



COST-EFFECTIVE MOTOR CONTROL

Industrial-grade trajectory planning on commodity hardware. Steppers, servos, DC motors — all supported.

Together: a **complete system** — motion controller + input language + motor control — at a fraction of industrial controller costs.

1. AI-READY BY DESIGN

ActionGraph is not retrofitted for AI — it was built for it:

DECLARATIVE YAML

LLMs excel at generating structured data. AG programs are YAML — a format every LLM knows well.

CONSTRAINED DSL

Limited feature set = fewer ways to go wrong. AI produces more reliable output in constrained domains.

DOCUMENTATION AS API

Same docs serve humans and AI agents. Structured, prescriptive, unambiguous. Single source of truth.

VERIFIABLE OUTPUT

AI-generated code compiles and runs in simulator. Iterate until all tests pass. No blind deployment.

What this means: An LLM agent can read AG documentation, generate a complete robot program, compile it, simulate it, and verify it — autonomously. This turns hours of specialist work into minutes.

AI ADVANTAGE VS. ALTERNATIVES

Platform	AI Code Generation	Verification
ActionGraph	Declarative YAML, constrained domain, structured docs	Built-in simulator, compile-time checks
PLC (Ladder / ST)	Visual format, proprietary IDEs, no text API	Vendor-specific, expensive simulation
ROS2 / C++	General-purpose, infinite code space	Complex build system, runtime crashes
G-Code	Simple but no concurrency, no state machines	No built-in simulation
Arduino / MicroPython	Known formats, but dynamic memory, no RT	No simulation, test on real hardware only

2. SCALES FROM MCU TO EMBEDDED PC

The same AG program runs unmodified across the full hardware spectrum:



WHAT CHANGES

- HAL implementation (platform-specific I/O)
- Transport layer (UART vs. UDP vs. EtherCAT)
- Available memory (limits program size)

WHAT STAYS THE SAME

- AG program (YAML source & bytecode)
- VM execution engine
- Trajectory planning & kinematics
- Communication protocol (RPC)

WHY IT SCALES: ZERO-COST ABSTRACTION

NO DYNAMIC MEMORY

All memory allocated at compile time. No heap, no fragmentation, no GC. Predictable footprint on any platform.

NO OS REQUIRED

Runs bare-metal on MCUs. No RTOS licensing. Single HAL struct of C function pointers is the entire platform interface.

MINIMAL FOOTPRINT

VM core: ~50–200 KB. Per-actuator overhead: ~1–2 KB. Fits in STM32F4 with room to spare.

PORTABLE C++20

Compiles for ARM Cortex-M, ARM Cortex-A, x86. Same codebase, same behavior. No platform `#ifdefs` in VM.

Key insight: The HAL boundary is the *only* platform-specific code. Everything above it — VM, trajectory planner, kinematics, RPC — is identical across all platforms.

3. COST-EFFECTIVE MOTOR CONTROL

Industrial-grade motion control on commodity hardware and motors:

STEPPERS

Step/dir control with PID speed loop, microstep resolution, encoder feedback, homing.

\$2–15 per motor

SERVOS

Position-commanded drives via analog, CAN, or EtherCAT. Works with any servo that accepts position commands.

\$20–200 per motor

DC / BLDC

PWM or analog speed control via generic signal outputs. External driver does commutation.

\$1–30 per motor

Motor-agnostic abstraction: The AG program doesn't know what motor type is attached. It works with *signals* (analog/digital I/O). Stepper vs. servo vs. DC is a HAL configuration choice, not a code change.

BUILT-IN TRAJECTORY PLANNING

Features typically found only in expensive industrial controllers — included in the VM:

JERK-LIMITED S-CURVE PROFILES

Smooth acceleration with bounded jerk. Reduces mechanical wear, vibration, and overshoot. Configurable speed, acceleration, and jerk limits.

MOTION JOINING

Seamless transitions between movements with non-zero endpoint velocity. No stop-and-go between segments.

MULTI-DOF KINEMATICS

Delta, SCARA, 6-DOF articulated arms, gantries. Forward and inverse kinematics computed in the VM tick loop.

PARAMETRIC MOTION

Expression-based custom trajectories. Arbitrary motion profiles from mathematical expressions.

All of this runs on a \$10 STM32 at microsecond timing. No external motion controller needed.

SUPPORTED KINEMATICS

Robot Type	Module	DOF
Linear Delta	LINEAR_DELTA_ROBOT	3
Rotating Delta	ROTATING_DELTA_ROBOT	3
SCARA	SCARA_4_DOF_ROBOT_MOVABLE	4
Articulated Arm (DH)	DH_ARM_4/5/6_DoF_MOVABLE	4 / 5 / 6
Cartesian / Gantry	PERPENDICULAR_AXES_ROBOT	1-3
Custom	MOVABLE_BINDING + expressions	Any

Coordinate transforms, master-follower bindings, and joint limit enforcement included.

COST COMPARISON: COMPLETE SYSTEM

TRADITIONAL INDUSTRIAL

PLC + Motion Module	\$500–5,000
Engineering Software	\$1,000–10,000
Servo Drives (per axis)	\$200–2,000
Servo Motors (per axis)	\$100–1,000
Training / Integration	\$5,000+

Vendor lock-in. Proprietary everything. Weeks of integration.

ACTIONGRAPH SYSTEM

MCU Board (STM32/RP2040)	\$5–25
AG Runtime + Tools	—
Stepper Drivers (per axis)	\$2–15
Stepper Motors (per axis)	\$5–30
AI-Assisted Programming	minutes

Open platform. Standard hardware. AI writes the program.

EXAMPLE BOM: 3-AXIS DELTA ROBOT

Component	Part	Cost
Controller	STM32F4 board	\$12
Motor Drivers (x3)	TMC2209 stepper drivers	\$9
Motors (x3)	NEMA17 steppers	\$24
End Switches (x3)	Mechanical microswitches	\$3
Power Supply	24V 5A	\$15
Electronics Total		~\$63

What you get for \$63 in electronics: Jerk-limited delta kinematics, auto-homing, Python integration, simulator, AI-programmable. Features that typically require a \$2,000+ controller stack.

THE LANGUAGE: AN UNFAIR ADVANTAGE

ActionGraph's process-oriented YAML is not just "another way to program" — it changes economics:

AI-GENERATABLE

LLMs reliably produce valid AG programs. Reduces specialist labor from days to minutes for standard tasks.

NO SPECIALIST REQUIRED

Process-oriented thinking is natural for domain experts. No PLC certification, no C++ expertise needed.

REUSABLE COMPONENTS

Standard library (actuators, calibration, pick-and-place). Repeater macro for N-similar patterns. Build once, reuse everywhere.

SIMULATION-FIRST

Built-in Python simulator. Test logic without hardware. Catch bugs before they damage actuators.

The language itself is a cost reducer: less engineering time, fewer bugs, faster iteration.

A COMPLETE SYSTEM, NOT JUST A CONTROLLER

AI Agent — Reads docs, generates programs, verifies via simulation

ActionGraph Language + Compiler — YAML source → VM bytecode

AG Virtual Machine — Deterministic execution, trajectory planning,
kinematics

HAL + Motor Control — Steppers, servos, DC/BLDC via standard signals

Python Client + Simulator — UI, analytics, testing, cloud integration

Every layer is included. No third-party motion controller. No proprietary IDE. No expensive simulation software.

SCALING SCENARIOS

Scenario	Hardware	Cost	What AG Provides
Single Stepper Conveyor belt, turntable	RP2040 + TMC driver	~\$10	PID speed loop, homing, position control
3-Axis Delta Pick-and-place, 3D printer	STM32F4 + 3x stepper	~\$60	Delta IK/FK, jerk-limited motion, auto-homing
6-DOF Servo Arm Assembly, inspection	STM32F7 + CAN drives	~\$300	DH kinematics, coordinated motion, Python UI
Multi-Robot Cell Production line	RT Linux + EtherCAT	~\$1,000	Multi-VM coordination, HMI, full simulation

Same language, same tools, same AI assistant — at every scale.

COMPETITIVE POSITIONING

	ActionGraph	Industrial PLC	Arduino/Maker	ROS2
Hard Realtime	Yes	Yes	No	Partial
AI-Programmable	Native	No	Partial	Partial
Runs on \$5 MCU	Yes	No	Yes	No
Trajectory Planning	Jerk-limited	Yes	Basic	MoveIt
Built-in Simulation	Yes	Extra \$\$	No	Gazebo
BOM for 3-axis	~\$60	\$2,000+	~\$50	\$200+
Multi-Robot	VM-to-VM	Proprietary	Manual	Native

KEY TAKEAWAYS

1. AI-ready architecture reduces engineering cost

LLMs generate valid AG programs. Documentation-as-API. Simulation verifies output. Specialist labor → AI minutes.

2. One platform from \$5 MCU to embedded PC

No dynamic memory, no OS required, portable C++20. Same program, same tools, any hardware.

3. Industrial motion control at commodity prices

Jerk-limited trajectories, multi-DOF kinematics, PID loops — all running on a \$10 chip with cheap stepper motors.

4. Motor-agnostic: steppers, servos, DC, BLDC

The HAL abstraction makes motor type a configuration choice, not an architecture decision. Swap hardware without code changes.

5. Complete system, not just a component

Language + compiler + VM + trajectory planner + simulator + Python client + AI integration. Nothing else to buy.

ACTIONGRAPH

AI-Ready. Hardware-Scalable. Cost-Effective.

Questions?