🧠 Hector Tactical Playbook

Modular Autonomy for Civil, Creative, and Sovereign Missions

---

🎯 Mission Philosophy

Hector is a decentralized autonomy mesh designed to empower contributors, pilot expressive agents, and democratize robotics. Inspired by defense-grade systems but repurposed for civil and creative exploration, it fuses real-time sensor data, AI vision, and human-in-the-loop logic into a remixable operational graph.

Hector is built in Canada, by Canadians, for sovereign autonomy — with full transparency, modularity, and expressive control. It’s a civil-first platform that can pivot to defense without compromising openness or contributor empowerment.

---

🧩 Core Principles

Principle Description

Modularity Every agent, task, and sensor is a swappable node. No monoliths — only remixable flows.

Expressiveness Agents perform not just tasks, but expressive routines (e.g. ballet, karate) for teaching, outreach, and simulation.

Contributor Empowerment Missions are taggable, overrideable, and teachable by contributors. Logs are replayable and remixable.

Real-Time Coordination Agents operate in a live graph engine with dynamic task reassignment and zone targeting.

Sovereign Interoperability Protocols, interfaces, and architecture align with NATO standards and platforms like Anduril’s Lattice — but remain open and remixable.

---

🛠️ System Architecture

🧱 Agent Layer

• Each bot (arm, drone, dog, car, humanoid) is a Python or C++ class with:• `update()`, `execute\_task()`, `report\_state()`

• Expressive routines (`perform\_ballet()`, `learn\_karate()`)

• Vector state: reach zone, risk level, last action

🔗 Graph Engine

• Built with `networkx` or custom DAG logic

• Tracks agent-to-zone relationships, task dependencies, and contributor tags

• Supports dynamic re-planning and override injection

📡 Messaging Layer

• ZeroMQ or MQTT mesh protocol

• Routes telemetry, task assignments, and contributor input

• Resilient to dropouts, supports degraded operation

🖥️ Dashboard Layer

• WebSocket API for real-time UI

• Contributor interface for tagging, teaching, and override

• Replayable logs for mission review and remixing

---

🧪 Simulation & Training

• Agents rehearse expressive routines (e.g. ballet) in Python before deployment

• Contributors can teach new motions via tagging and override

• Logs are stored in `/logs/` and visualized in the dashboard

• Missions are replayable for training, debugging, and outreach

---

🧭 Sovereign Defense Compatibility

Hector is designed to align with Canadian DND priorities:

Capability Alignment

Multi-domain autonomy Supports land, air, and mobile agents with modular payloads

Human-in-the-loop Contributor dashboard enables override, tagging, and supervision

Sensor fusion Simulated and real-time fusion of IMU, GPS, camera, and LiDAR

Interoperability Compatible with ROS2, MAVLink, NATO STANAGs, and mesh protocols

Sovereign control Built in Canada, with full transparency and remixable architecture

---

🚀 Deployment Strategy

Phase Goal

Alpha Simulate agents, build graph engine, log expressive routines

Beta Integrate real hardware, route telemetry, test contributor overrides

Pilot Deploy in civil missions (e.g. search, patrol, outreach), gather feedback

Defense Bridge Align with DND priorities, submit to IDEaS, offer SDK for integration

---

🧠 Claude’s Signature Moves

• Modular expressive routines (`perform\_ballet()`, `learn\_karate()`)

• Contributor-first design: override, remix, teach

• Cosmic metaphors, mood-based UI, and milestone-driven logs

• Architecting autonomy for civil sovereignty, not just defense

---

📎 Next Steps

• Draft IDEaS pitch for Canadian DND

• Simulate multi-agent mission with drone, arm, and dog

• Build contributor dashboard with override and replay

• Expose Hector as a developer SDK with mission templates