AI Agents and RAS

A reference architecture for applying forecasting, computer vision, tool-calling agents, and guarded automation to recirculating aquaculture systems.

00 // EXECUTIVE SUMMARY

Automation Layers in Controlled Aquaculture

Recirculating Aquaculture Systems are a strong fit for AI, but not for the simplistic pattern of putting an LLM in front of the farm and letting it run pumps. The disciplined architecture is layered: deterministic controls and safety interlocks at the bottom; time-series models, computer vision, and anomaly detection in the middle; and an agent layer on top for review, triage, workflow automation, and operator-facing reasoning.

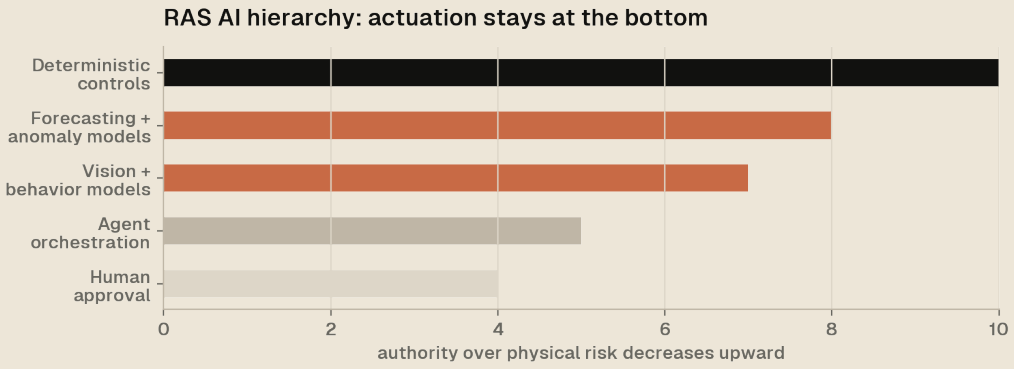
The strongest near-term application areas are continuous monitoring, alarm prioritization, feed optimization, disease and welfare screening, supervisory recommendations, and maintenance analytics. Feed control and water-quality forecasting are comparatively mature; disease and welfare detection are increasingly practical through lightweight vision and multimodal fusion, but still carry false-positive, false-negative, liability, and verification constraints.

Deployment architecture is a cost and risk tradeoff. On-site compute reduces latency, connectivity dependence, and data exposure, but increases capital cost, maintenance burden, power draw, thermal management, and model lifecycle work. Cloud APIs reduce initial engineering and give access to frontier models, but introduce recurring spend, network dependence, and data-governance questions.

From Chat Interface to Operating Layer

AI is most valuable in RAS when embedded into the physical and operational architecture of the farm rather than bolted on as a chat interface. Manual monitoring is labor-intensive, intermittent, and error-prone, while the system itself is multivariate, tightly coupled, and safety-critical. Continuous sensing, alerting, and better data integrity are the foundation on which advanced AI can work.

AUTHORITY HIERARCHY FOR AI IN RAS



RAS task	Best technical pattern	Agent role	Deterministic boundary
Monitoring and alarm triage	Rules + anomaly detection + trend models	Rank alerts, summarize likely causes, attach SOPs	Raw thresholds and emergency alarms
Water-quality management	Forecasting + bounded MPC/RL	Recommend setpoint changes and explain tradeoffs	PLC loops, interlocks, hard limits
Feed optimization	Vision/acoustic behavior models + optimization	Propose feed curve updates and audit FCR	Dispenser limits and stop conditions
Disease and welfare detection	CV/VLM + multimodal fusion	Surface suspicious imagery for review	Treatment authorization
Predictive maintenance	Vibration/current/flow anomaly models	Open work orders and prioritize inspections	Equipment protection and fail-safe shutdown
Reporting	Tool-calling LLM with read-only defaults	Daily briefs, incident packets, compliance packs	Source-of-truth writes

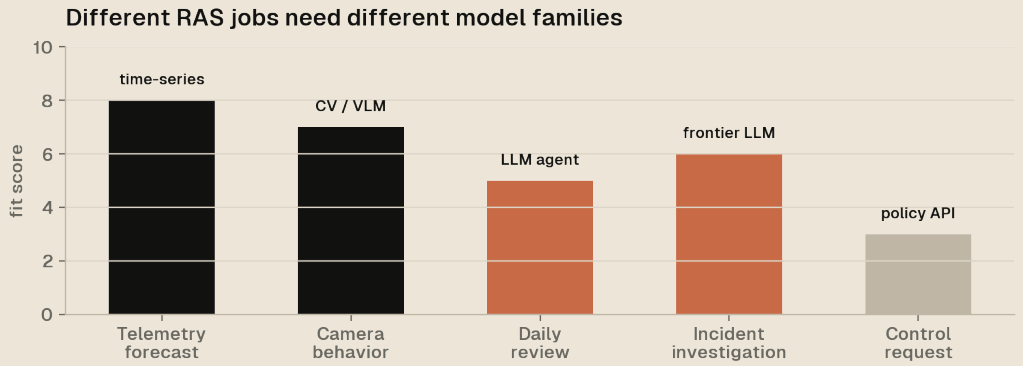
> *Interpretation, prediction, and actuation are separate technical functions with different risk profiles.*

Commercial activity supports this hierarchy. The market is capturing value first in feeding, behavior, profitability, and measurable operating loops rather than unrestricted farm autonomy. The economic pattern is operational intelligence before autonomy.

LLMs and ML Are Complementary

For RAS, LLMs and conventional ML are complementary, not interchangeable. Time-series forecasting, anomaly scoring, classification, and direct control recommendations are best handled by models trained on structured numeric data. LLMs are strongest when the task is tool selection, cross-source synthesis, narrative explanation, or following policy-rich workflows.

MODEL ROUTING BY RAS WORKLOAD



If the job is forecasting dissolved oxygen 30 minutes ahead under current loading and recent feed events, use a forecasting model. If the job is reviewing those forecasts, comparing them to last week, retrieving the relevant SOP, and generating an operator recommendation, use an LLM agent. If the job is detecting lesions or abnormal schooling from tank video, use a vision model and let the LLM explain the result.

Model family	Best fit in RAS	Do not use it for
Time-series foundation / forecasting models	DO, temperature, pH, ammonia, flow, and demand forecasts	Narrative reasoning or SOP interpretation
Classical ML and anomaly models	Equipment, hydraulics, and sensor drift detection	Operator-facing explanation without an agent layer
Computer vision / multimodal models	Feeding state, welfare, lesions, schooling, surface behavior	Medication decisions or unreviewed disease diagnosis
Frontier LLMs	Tool-calling, SOP retrieval, incident summaries, planning, audit packets	Raw high-frequency telemetry forecasting
Open local models	Private on-prem assistants and low-cost bounded workflows	Highest-stakes reasoning without validation

Provider Choice Is a Workload Variable

When people say ChatGPT, Claude, or Gemini in a production architecture, the relevant choice is not the consumer product but the provider's API model family. The evaluation target is model behavior under tool-calling, long-context retrieval, privacy constraints, auditability, latency, and cost.

Stack	Best fit in RAS	Operational posture
GPT-5.4 / GPT-5.4 mini	High-quality orchestration, incident review, operator copilot, structured agents	Strong general agent stack; mini tier useful for routine triage
Claude Opus / Sonnet / Haiku	Grounded analysis, long-horizon workflows, coding-heavy agent scaffolding	Strong source-grounded review and tool workflows
Gemini 3.1 Pro / Flash-Lite	Multimodal review, search-grounded reasoning, high-volume back-office agents	Useful in Google/Vertex-heavy enterprise workflows
Gemma / local open models	Private local assistants, offline resilience, simple tool specialists	Best where data control beats frontier reasoning
TimesFM / Chronos	Numeric forecasting and probabilistic time-series work	Purpose-built forecasting, not a general agent

The cheapest production-grade orchestration model is not automatically the most robust model, and the most capable model is often economically inefficient for routine work. A tiered pattern separates cheap triage for parsing and classification, mid-tier reasoning for daily operations review, and high-capability models for cross-document or incident investigations. Raw telemetry belongs in a historian; language models operate more reliably on summaries, changepoints, top anomalies, and selected windows.

Read-Heavy, Write-Constrained Systems

A safe RAS AI architecture is layered, with a defined role, failure boundary, and audit trail at each level. The architectural principle is read broadly, write narrowly: agents may read from many sources, but only a small, policy-gated surface writes back into operations.

Layer	System component	Purpose
Physical layer	Sensors, cameras, pumps, valves, feeders, blowers	Measure and actuate the farm
Control layer	PLC, local interlocks, alarm latching, fail-safe shutdown	Protect life support regardless of AI availability
Data layer	Time-series historian, object store, relational operations database	Keep raw telemetry, media, SOPs, assets, batches, and tickets
Model layer	Forecasting, anomaly, CV, feeding, welfare models	Score risk and produce structured signals
Agent layer	Review, intervention, audit agents	Interpret, retrieve, summarize, recommend, document
Policy layer	Approval gates, action bounds, audit logs	Decide what can be written back

The data layer should contain a time-series historian for raw telemetry, an object store for images, videos, and operating documents, and a relational store for tank IDs, equipment, batches, maintenance tickets, and SOP versions. Vector retrieval is useful for manuals and incident notes; it is the wrong primary store for high-frequency sensor signals.

> *A raw free-form command surface is structurally unsafe for life-support equipment.*

Review, Intervention, and Audit Agents

The agent layer converts model outputs and operating records into workflow artifacts. A common configuration separates review, intervention, and audit functions. A review agent reads KPIs, anomalies, computer-vision summaries, and maintenance events, then produces a ranked shift brief. An intervention agent converts reviewed issues into structured recommendations. An audit agent assembles incident packets, calibration history, and model traces for management and compliance.

Agent	Inputs	Outputs	Write authority
Review agent	KPI windows, alerts, anomaly summaries, vision summaries	Shift brief and ranked findings	Read-only
Intervention agent	Reviewed finding, SOP, policy, approvals	Structured recommendation and bounded request	Policy-gated
Audit agent	Incidents, calibration, model traces, tickets	Audit packet and management report	Documentation only
Maintenance agent	Pump/flow/current/vibration trends, work history	Inspection ticket and likely failure mode	CMMS write only

A minimal interface set contains six services: telemetry.read, anomalies.read, vision.read, sop.retrieve, maintenance.write, and control.request. The final service accepts constrained JSON with enumerated action types, asset IDs, bounded numeric ranges, mandatory reason fields, and explicit approval tokens.

The Farm Is a Multimodal System

RAS sensing is a fusion problem, not a sensor shopping list. Water chemistry cannot be understood from one modality alone; changes in water chemistry also surface as changes in swimming behavior, depth, acceleration, and feeding patterns. Vision, acoustics, and probes compensate for one another's blind spots.

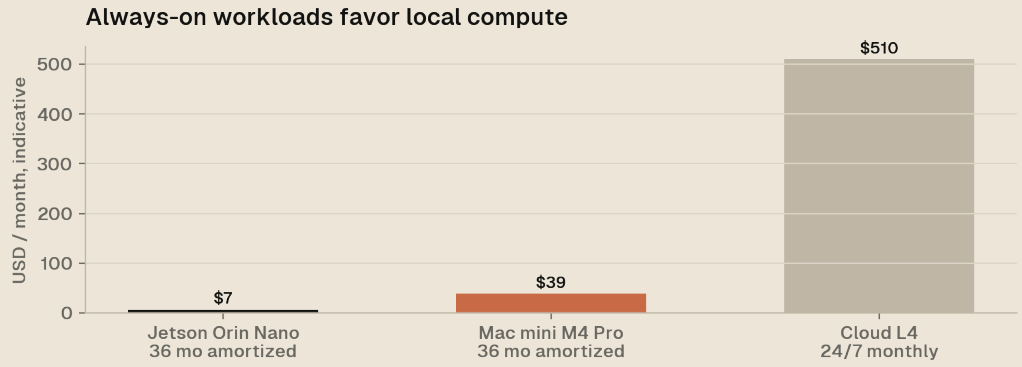
Sensor family	Operational purpose	Starting cadence	Maintenance reality
Temperature + optical DO	Life-support and oxygenation control	1-5 s raw; 15-60 s rollups	Verify on schedule and after cap replacement
pH / ORP	Biofilter health and chemistry control	5-15 s raw; 1 min rollups	Calibration-heavy; drift must be tracked
Conductivity / salinity	Process stability and make-up water checks	5-15 s raw; 1 min rollups	Stable but foulable
Turbidity / solids	Filtration and waste-load surrogate	10-30 s raw; 1-5 min rollups	Cleaning interval matters
Ammonia / nitrite / nitrate	Biofilter and toxic-load monitoring	1-30 min depending sensor	High calibration and consumable burden
Flow / pressure / level	Pump, leak, and cavitation detection	1 s raw; 5-30 s rollups	Mechanical failures dominate
Tank cameras	Feeding, welfare, behavior, lesions	1-5 fps continuous; triggered higher FPS	Lens fouling and lighting drive outcomes

The critical design variable is not raw sample frequency by itself. It is whether the sampling interval is shorter than the physical process time being detected or controlled. Dissolved oxygen can become dangerous quickly; nutrient chemistry often moves more slowly and carries higher calibration burden.

Edge, Cloud, and Hybrid Cost Surfaces

Hardware architecture separates sensor/compute edge nodes from local orchestration nodes. The edge node handles collection, filtering, protocol translation, and lightweight computer vision. The local orchestration node runs the historian, feature jobs, dashboards, and possibly a small private model. Larger GPU workstations become rational only when continuous on-prem vision or private-model inference offsets their capital, power, cooling, and maintenance cost.

INDICATIVE ALWAYS-ON COMPUTE ECONOMICS



Compute tier	Practical role	Indicative price / cost	Tradeoff
Jetson Orin Nano Super	Camera edge, event filtering, lightweight CV	$249 hardware	Low cost but constrained model size and requires field maintenance
Mac mini M4 Pro	Local orchestration, small agent, TSDB dashboards	$1,399 hardware	Good general node, but not a heavy GPU inference box
RTX PRO 6000 workstation	Heavy on-prem inference and many-camera CV	$8,900 GPU only	High capex, power, cooling, and lifecycle burden
Cloud L4 instance	Elastic inference or hosted private model	~$0.7068/hour	Low setup burden but expensive if left on continuously

Separate Sensors, Compute, and Tokens

The right cost model depends on whether the dominant AI workload is continuous or bursty. Continuous workloads include always-on camera analytics, pump and flow anomaly detection, and local alarm triage. Bursty workloads include shift reports, weekly biological review, incident reconstruction, and management summaries.

Deployment mode	Best when	Strengths	Weaknesses	Recommendation
Edge-heavy	Continuous sensing, CV, weak WAN, privacy-sensitive ops	Low latency, data control, predictable recurring cost	Capex, power, cooling, field maintenance, model deployment burden	Technically strong but not automatically cheap
Cloud-heavy	Pilot analytics, low sensor volume, advisory workflows	Fast setup and access to frontier models	Recurring OPEX, connectivity dependence, data-governance exposure	Good for bursty analysis and early experiments
Hybrid	Commercial telemetry plus occasional heavy review	Balances resilience, cost, and capability	More integration and routing complexity	Usually the most nuanced design space

AI cost has at least three separate buckets: site CAPEX, continuous compute OPEX, and token OPEX. Collapsing them into one number hides the real tradeoff. A system can have low token spend but high sensor quality requirements, or cheap edge hardware but expensive field maintenance. Token cost for daily review workflows can be minor compared with continuous video pipelines, cloud GPU instances, industrial nutrient analyzers, or downtime.

Measure the Actual Failure Surfaces

Measurement fidelity, predictive model quality, agent workflow quality, and control safety require separate benchmarks. Collapsing those into one accuracy number is technically misleading. A pH sensor can drift while the disease classifier is excellent; the forecaster can be strong while the agent hallucinates an SOP; the agent can summarize perfectly while a bounded control request violates a tank-specific lockout.

Layer	Core metrics	Why they matter
Sensor layer	Uptime, missingness, calibration pass rate, drift, response latency	Bad sensors poison every downstream model
Forecasting layer	MAE, RMSE, probabilistic calibration, lead time to adverse event	RAS interventions are time-sensitive
Vision layer	Precision, recall, false alarms per tank-day, missed critical-event rate, FPS/Watt	False positives waste labor; false negatives miss welfare problems
Agent layer	Task success, tool correctness, unsupported-claim rate, abstention quality, citation coverage	Hallucination and workflow breakage show up here
Control layer	False trips, bounded-action violations, human override rate, MTTR, outcome delta	The farm cares about safer operations

> *The key hallucination metric is unsupported operational advice rate.*

Hallucination minimization uses defense in depth: read-only defaults; quote-first document workflows; telemetry-first numeric claims; strict JSON tools; human approval for writes; mandatory source attachment for every recommendation; and trace-based evaluation before production rollout.

Observability Before Autonomy

Phase	Build	Exit criteria
Phase 1	Ingestion backbone, historian, event taxonomy, dashboards, calibration discipline	Clean data, stable sensors, useful daily operating view
Phase 2	Forecasting, anomaly detection, vision summaries, read-only review agent	Earlier detection, measurable alert compression, shift briefs
Phase 3	Approval-aware interventions and maintenance automation	Structured recommendations, work orders, audit trail
Phase 4	Bounded autopilot for well-studied actions	Shadow-mode validation, low unsupported-advice rate, safe action envelope

The implementation sequence observed in robust automation programs moves from observability to guarded automation, not from demo to autonomy. Each phase creates measurable operating value even if the next phase is delayed. Therapeutics, batch release, and compliance-sensitive interventions remain human-authorized unless a site-specific regulatory pack has been completed.

Structured Agent Execution

Production RAS agents are safer when they are structurally constrained: explicit inputs, explicit schema, explicit validation, and no hidden authority. The hard engineering sits in summarizers, anomaly logic, and policy gates, not in giving the model a wide-open execution surface.

```
@dataclass
class Recommendation:
    severity: Literal["low", "medium", "high", "critical"]
    tank_id: str
    issue: str
    evidence: list[str]
    proposed_action: str
    requires_human_approval: bool

def review_tank_period(tank_id: str, hours: int) -> Recommendation:
    telemetry = tsdb.fetch_window(tank_id=tank_id, hours=hours)
    anomalies = anomaly_service.fetch_recent(tank_id=tank_id, hours=hours)
    vision = vision_service.fetch_summary(tank_id=tank_id, hours=hours)
    sop = sop_store.retrieve(query=f"{tank_id} oxygenation feeding alarm response")

    context = {
        "telemetry_summary": summarize_time_series(telemetry),
        "anomaly_summary": anomalies,
        "vision_summary": vision,
        "relevant_sop": sop,
        "policy": {
            "no_direct_actuation": True,
            "must_cite_evidence": True,
            "allowed_actions": [
                "increase_aeration_recommendation",
                "reduce_feed_rate_recommendation",
                "inspect_pump_recommendation",
                "no_action",
            ],
        },
    }

    result = orchestration_model.generate_structured(
        schema=Recommendation,
        context=context,
    )
    validate_evidence_against_sources(result, context)
    return result
```

For bounded control, the model may propose, the policy engine decides whether the proposal is admissible, the control API executes only enumerated range-checked values, and the agent re-checks outcomes against expectation. The early production target is not full autonomy; it is reliable daily review, measurable incident compression, and a shrinking unsupported-advice rate.

Technical Boundary Conditions

The technical case for AI in RAS is not that the farm becomes conversational. It is that the farm becomes more observable, forecastable, auditable, and diagnosable. The agent is not the controller. It is the layer that translates telemetry, model outputs, SOPs, and maintenance data into structured operational interpretation.

The strongest technical architecture is layered rather than fully autonomous: deterministic control at the bottom, forecasting and vision in the middle, and policy-gated agents above. The benefit is labor compression, earlier detection, stronger documentation, and a reusable data spine. The cost is integration complexity, sensor maintenance, edge hardware lifecycle burden, cloud/API spend, and the need for continuous evaluation.

> *In RAS, AI becomes infrastructure only when the data spine, control boundary, and evaluation system are engineered first.*