

I Know What I Want to Build

A concrete research programme for persistent adaptive systems whose fast dynamics, lifetime change and physical transfer can be tested rather than romanticised.

DRAFT PREVIEW DENNIS HEDEGREEN BUILD DRAFT-LOCAL

[HTTPS://HEDEGREENRESEARCH.COM/ARTICLES/I-KNOW-WHAT-I-WANT-TO-BUILD/](https://hedegreenresearch.com/articles/i-know-what-i-want-to-build/)

If we are going to live with digital beings, let them be worth meeting.

I do not know what consciousness is. I know what I want to build.

More precisely, I want to build a small persistent artificial system in which three timescales are kept separate on purpose. At design time, search chooses architecture, primitive dynamics and plasticity rules. At runtime, the resulting system senses and acts through fast physical or simulated dynamics. Across its lifetime, slower local changes allow its history to alter what it becomes. At least one internal variable defines a viability region that the system's own behaviour can help maintain. And if a candidate works in simulation but falls apart when translated into hardware, that failure counts against the design rather than being treated as somebody else's engineering problem.

None of those ingredients is new. The program is a specific recombination of older lines of work, and it should be judged by whether that recombination produces measurable advantages rather than by whether I can rename the pieces.

That lineage is deep. Randall Beer was already treating adaptive agents and their environments as coupled dynamical systems under a viability constraint.[1] Joseba Urzelai and Dario Floreano evolved local synaptic adaptation rules so controllers could change during their own lifetime rather than only across generations.[2] The POE framework separated phylogenetic, ontogenetic and epigenetic processes in bio-inspired hardware. [3] Ezequiel Di Paolo used homeostatic neural plasticity to let evolved simulated agents recover after radical sensorimotor perturbations.[4] Adrian Thompson evolved circuits directly in FPGA hardware, allowing the physics of a particular device to participate in the solution.[5]

The three clocks I use here are therefore bookkeeping, not a new taxonomy. They are not identical to phylogeny, ontogeny and epigenesis, but they serve the same practical purpose: keeping separate what changes between designs, what changes across a system's history, and what happens in the fast loop of current behaviour.

The question is what happens if those distinctions are made explicit in a design system that is also forced to care about physical transfer.

Current build boundary. The local Circuit Lab already exists as a browser workbench with Water Mode and Build Mode over the same running scene, explicit flow primitives, scene save/load, conservation tracking, public module ports and an early Module Builder. The searchable primitive library, lifetime-

learning benchmark and physical-transfer loop described below remain a research programme, not completed runtime capabilities.

Water has to earn its place

My own route into this problem began with water.

A bucket integrates. A leak decays. An overflow creates a threshold. A siphon accumulates and releases. A valve gates. A changing channel can represent conductance with memory. The point was never that water had discovered neuroscience. The point was that a physical flow metaphor made state, accumulation, leakage, delay and path dependence visible enough to manipulate.

There are already much more formal languages for this territory. LEMS supports user-defined dynamical components with state variables, differential equations, events and multiple regimes.[6] NESTML is a domain-specific language for hybrid dynamical systems, including neuron and synapse models, code generation and plasticity rules.[7] NIR provides a platform-independent graph of continuous-time neuromorphic primitives and maps those descriptions across multiple simulators and hardware platforms.[8]

So Water is not allowed to survive merely because I like it.

Its strongest possible contribution is a constraint on the design space. If every quantity has an explicit location, every leak an explicit path, every source and sink a declared boundary, and selected quantities obey conservation rules, then the representation may make it harder for machine-generated architectures to smuggle in arbitrary hidden state or physically meaningless operations.

That claim has a confound: the benefit may come entirely from the constraint, not from the hydraulic notation. A typed dynamical graph can enforce the same rule.

The first comparison should therefore be a two-by-two design:

graph notation / Water notation × conservation unconstrained / conservation constrained

The Water-without-conservation arm is deliberately ugly: sources and sinks are allowed, but they remain explicit. The constrained-graph arm carries the same conservation semantics without the hydraulic skin.

Two primary endpoints are enough for the first test: how many simulator evaluations are required to reach a preregistered viability criterion, and how much performance is lost when the design is transferred into a second implementation or physical substrate. Human readability and debugging time can still be recorded, but they should not rescue a failed primary result after the fact.

If the conservation-constrained graph gets the whole benefit, keep the constraint and drop the Water layer.

That is a result I am prepared to accept.

A richer vocabulary cannot win by magic

I also need to be precise about what I expect from heterogeneous primitives.

A continuous-time recurrent neural network with sufficient hidden state can approximate arbitrary finite-time trajectories of dynamical systems. Funahashi and Nakamura proved that result in 1993.[9] A richer primitive library therefore does not automatically add computational capability in principle.

The hypothesis is narrower.

Different primitives may change **search cost, compactness, physical fit and energy**, even when a sufficiently large homogeneous network could emulate the same dynamics.

A resonant element may implement a behaviour compactly that a generic network would need many components to approximate. A volatile physical device may provide useful decay dynamics without a processor repeatedly reconstructing them. A state-retaining physical element may preserve conductance without a separate memory transaction. Or none of those advantages may survive whole-system accounting.

That is what the experiment has to discover.

At design time, the search can choose from a library of explicit dynamics: leaky integration, adaptive thresholds, oscillation, burst/reset behaviour, delays, local traces, slow variables, volatile and non-volatile state, gates and competition. The search can also choose the local plasticity rule that operates during the system's lifetime.

This is close to the evolved-plastic-network literature rather than a break from it. Reviews of evolved plastic artificial neural networks already describe evolution over architectures, neuron properties, plasticity rules and learning mechanisms.[10]

So the claim to test is not greater theoretical expressivity.

It is that **physically grounded heterogeneity may reduce the cost of finding and embodying useful dynamics**.

If a fixed CTRNN or another conventional baseline reaches the same behavioural performance with comparable search cost, component count, energy and transfer robustness, then a larger primitive vocabulary has not earned its complexity.

Search time, runtime and lifetime

Keeping the three clocks separate also fixes the role of the language model.

At **design time**, an LLM can interpret requirements, propose structures, rewrite representations, inspect failed candidates and suggest mutations. It competes with evolutionary strategies, Cartesian genetic programming, NEAT-like methods, random grammar-based mutation and human design.

At **runtime**, none of that machinery needs to be present. A sensor changes a state, a state changes another state, and the built system acts.

Across the **lifetime**, local plasticity changes the system because of its own history. If a cloud model has to be invoked every time the system adapts, then the adaptation is not native to the substrate I am trying to study.

The language model is therefore optional.

For each design task it should be ablated against conventional search under matched budgets. Simulator evaluations must be counted, and token or model-compute cost must be counted too. If an LLM burns ten times the compute to save five percent of simulator evaluations, that is not a free improvement.

The useful rule is simple:

The model proposes. The evaluator decides.

During early search the evaluator is a simulator. Once hardware exists, the simulator loses final authority.

The reality gap is already a field

The idea of rewarding designs for surviving translation into the real world is not new either.

Evolutionary robotics has spent decades on the reality gap. Nick Jakobi's envelope-of-noise work deliberately shaped simulation so evolved controllers could not depend on fragile simulated details.[11] Koos, Mouret and Doncieux later formalised a **transferability approach** in which optimisation considers not only simulated fitness but an estimate of how well candidate behaviour transfers from simulation to reality. [12]

That is almost exactly the precedent needed here.

I do not need to invent translatability as an objective. I need to adapt the idea to a different search target: dynamical primitives and local plasticity rules intended for mixed-signal or neuromorphic implementation.

The loop becomes:

design → simulation → selected physical tests → transfer model → redesign

As hardware data accumulates, the search should learn which kinds of simulated solutions are expensive, unstable or impossible to reproduce physically.

This also exposes a tension between portable representations and substrate-specific solutions.

A formal intermediate representation is valuable because it lets the same intended dynamics move across platforms. But a physical device may expose useful behaviour that the portable abstraction does not capture. Thompson's FPGA experiments are an extreme example: evolution exploited properties of the actual chip rather than merely reproducing a clean abstract circuit.[5]

When portability and substrate exploitation conflict, neither wins by definition. The fitness function should record the trade.

Matter gets a vote, but it does not get a veto merely because it is matter.

The first science experiment is replication-plus

The first hardware build can still be an RGB sensor.

That is plumbing.

The first scientific experiment should begin from something already known.

Di Paolo's 2000 experiment is a good starting point. He evolved continuous dynamical neural controllers for phototaxis while individual neurons used a homeostatic mechanism that triggered local plastic changes when activity left prescribed bounds. After left-right inversion of the visual field, the agents initially lost phototaxis and in many cases adapted as plasticity restored neuronal activity toward homeostatic ranges.[4]

The first step should be to reproduce that class of result with a conventional CTRNN baseline.

Only then add one change.

My preferred extension is to make the relevant essential variable external to individual neural firing and closer to the viability of the whole artificial system.

Imagine a small mobile simulated agent with a scalar internal resource. The resource decays continuously. Entering one region replenishes it; another region accelerates depletion. The regions are visually distinguishable, but the mapping between colour and consequence is not built into the controller.

The controller receives sensor state, its own resource state and minimal motion feedback.

Lifetime learning can use a local three-factor rule: each adaptive connection maintains an eligibility trace derived from pre- and postsynaptic activity, while a modulatory factor reflects whether the internal homeostatic error is improving or deteriorating. Eligibility traces combined with a third modulatory factor are already a standard framework for connecting local plasticity to delayed behavioural consequences.[13]

The rule is not presented as biologically complete or novel. It is simply an explicit mechanism that allows consequence to modulate local change without an external "red is good" label.

A tabular reinforcement learner should be included as a deliberately strong trivial baseline. An evolved fixed CTRNN and an evolved plastic CTRNN provide more relevant dynamical baselines. A Di Paolo-style homeostatic controller provides the historical comparison. Keramati and Gutkin's homeostatic reinforcement-learning framework provides another useful comparison for defining reward in relation to internal physiological deviation rather than arbitrary external value.[14]

Then perturb the system.

Reverse the colour-consequence mapping. Change the lighting. Move the resource regions. Alter the decay rate. Disable one pathway.

Measure adaptation time, time spent inside the viability region, cumulative resource deficit, behavioural performance and variance across seeds.

The point is not to prove that homeostasis works. That has precedent.

The point is to establish a stable benchmark on which primitive vocabulary, search method and later substrate translation can be compared without changing the task every time a new tool is introduced.

What happens at the boundary?

A viability variable becomes meaningless if crossing its boundary is just another score penalty followed by an immediate reset.

For the first simulation, I would therefore define a terminal absorbing condition.

When the resource reaches zero, sensing and actuation stop for the remainder of that trial. The agent cannot recover inside the episode. A new independent evaluation can begin later from a fresh initial condition because otherwise there would be no statistics, but the failed trajectory itself is over.

That still does not make the simulated system literally alive or precarious in the philosophical sense. It operationalises one narrow feature of viability: continued behaviour depends on keeping a variable within bounds.

A later hardware version can make the consequence more physical. The controlled resource could be coupled to a real energy store or supply limit so that a controller which fails to regulate eventually loses function rather than merely receiving a negative number.

The experiment should say exactly which version is being tested.

Individuation needs a noise control

If two identical adaptive systems are given different histories and later behave differently, it is tempting to call that individuality.

Noise and chaos can produce the same observation.

So the comparison needs a control.

Start multiple copies from the same state. Give one pair the same history but different random noise seeds. Give another pair meaningfully different histories. Then reunite the environmental conditions and, where possible, use matched noise sequences during the comparison period.

Historical individuation is supported only if the divergence caused by different histories is reliably larger and more persistent than the divergence produced by noise alone.

That gives the phrase “history matters” an actual measurement.

Native persistence versus the best boring alternative

A second major comparison should test whether persistence in the dynamical substrate is useful at all.

The baseline should not be an LLM agent. That would make the comparison too easy and too irrelevant.

The honest competitor is a good conventional embedded controller with an explicit external state store.

Both systems get the same sensors, actuators, history and task. One maintains relevant state through its own continuously evolving dynamics and local plasticity. The other uses conventional computation plus stored state.

Compare whole-system energy, adaptation time after perturbation, latency, retention across idle periods, graceful degradation after component faults, calibration cost and recovery behaviour.

Whole-system means whole-system. Sensors, ADCs and DACs, memory, communication, calibration, telemetry and any analysis machine required for runtime operation all count.

If native persistence does not produce a preregistered system-level advantage at matched behavioural performance, then “native” may be an aesthetic preference rather than an engineering advantage.

If the comparison produces no meaningful system-level advantage, the distinction should be dropped.

Instrumentation is not omniscience

I am not promising a permanently understandable system.

A complex recurrent physical system may become opaque in ways that are inconvenient or simply too large for a human to inspect directly. Full access to internal state does not automatically produce a correct causal explanation.

The standard I want is weaker: **instrumentability**.

Where measurement is possible, record state. Where intervention is possible, perturb. Where exact replay is possible, replay. Analog hardware may not permit exact replay, and telemetry itself consumes power and can alter the system. Those costs belong in the measurement budget.

An analysis model may help propose causal hypotheses from traces, but the hierarchy is:

telemetry → hypothesis → intervention → revised hypothesis

not:

telemetry → language model → truth

What counts as success?

The article should state the hypotheses and failure conditions. The detailed thresholds, seed counts and search budgets live in the companion preregistration, where they can be versioned and frozen before comparative results are visible.

The optional tools are not the program.

Water can fail. The LLM can fail. A particular memristor technology can fail. Those are tools, not the program itself.

I would define two core hypotheses.

H1 — physically grounded search: allowing design-time search to choose dynamical primitives and plasticity rules under an explicit transferability objective improves at least one of search efficiency, implementation compactness, whole-system energy or sim-to-hardware robustness at matched behavioural performance relative to a conventional fixed-vocabulary baseline.

H2 — substrate-native persistence: implementing lifetime state and adaptation in the runtime substrate produces a measurable system-level advantage over a conventional embedded controller with external state under the same task and behavioural criterion.

If both hypotheses fail across a preregistered task family and adequate search budgets, then the larger program collapses. What remains may still be useful replication or tooling, but not the distinct research direction described here.

The exact thresholds belong in the preregistration rather than being adjusted after the first graphs appear.

The point is not that the first numbers are sacred. The point is that they exist before I know which system wins.

What I want to build

The long horizon still matters to me.

If persistent artificial systems eventually become individual enough that we interact with them as particular entities rather than interchangeable instances, I would like their histories to matter. Two systems beginning from the same design but living through different conditions should be able to become measurably different for reasons traceable to what happened to them.

That is a more defensible starting point than trying to engineer “personality”.

Humour, beauty and whatever larger words we eventually use can wait.

For now I want something smaller and harder to fake: a system that has fast dynamics, lifetime change and a viability constraint; whose design can be searched; whose search is penalised when it does not survive translation into matter; and whose favourite tools are all disposable if the experiment says they add nothing.

The first build is an eye.

The first experiment is whether seeing can become part of maintaining the conditions for continued operation.

That is the program I want to build.

— Dennis Hedegreen

Sources

- [1] Randall D. Beer (1997), “The dynamics of adaptive behavior: A research program,” *Robotics and Autonomous Systems* 20(2–4), 257–289. DOI ([https://doi.org/10.1016/S0921-8890\(96\)00063-2](https://doi.org/10.1016/S0921-8890(96)00063-2)).
- [2] Joseba Urzelai & Dario Floreano (2001), “Evolution of Adaptive Synapses: Robots with Fast Adaptive Behavior in New Environments,” *Evolutionary Computation* 9(4), 495–524. DOI (<https://doi.org/10.1162/10636560152642887>).
- [3] Moshe Sipper, Eduardo Sanchez, Daniel Mange, Marco Tomassini, Andrés Pérez-Urbe & André Stauffer (1997), “A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems,” *IEEE Transactions on Evolutionary Computation* 1(1), 83–97. DOI (<https://doi.org/10.1109/4235.585894>).
- [4] Ezequiel A. Di Paolo (2000), “Homeostatic Adaptation to Inversion of the Visual Field and Other Sensorimotor Disruptions,” in *From Animals to Animats* 6, 440–449. DOI (<https://doi.org/10.7551/mitpress/3120.003.0047>).
- [5] Adrian Thompson (1996/1997), “An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics,” in *Evolvable Systems: From Biology to Hardware*. DOI (https://doi.org/10.1007/3-540-63173-9_61).
- [6] NeuroML contributors, “LEMS: Low Entropy Model Specification” and LEMS Dynamics documentation (<https://docs.neuroml.org/Userdocs/LEMS.html>).
- [7] NEST Initiative, “The NESTML modeling language” (<https://nestml.readthedocs.io/en/latest/>).
- [8] Jens E. Pedersen et al. (2024), “Neuromorphic intermediate representation: A unified instruction set for interoperable brain-inspired computing,” *Nature Communications* 15, 8122. DOI (<https://doi.org/10.1038/s41467-024-52259-9>).
- [9] Ken-ichi Funahashi & Yuichi Nakamura (1993), “Approximation of dynamical systems by continuous time recurrent neural networks,” *Neural Networks* 6(6), 801–806. DOI ([https://doi.org/10.1016/S0893-6080\(05\)80125-X](https://doi.org/10.1016/S0893-6080(05)80125-X)).
- [10] Andrea Soltoggio, Kenneth O. Stanley & Sebastian Risi (2018), “Born to learn: The inspiration, progress, and future of evolved plastic artificial neural networks,” *Neural Networks* 108, 48–67. DOI (<https://doi.org/10.1016/j.neunet.2018.07.013>).
- [11] Nick Jakobi (1997), “Evolutionary Robotics and the Radical Envelope-of-Noise Hypothesis,” *Adaptive Behavior* 6(2), 325–368. DOI (<https://doi.org/10.1177/105971239700600205>).
- [12] Sylvain Koos, Jean-Baptiste Mouret & Stéphane Doncieux (2013), “The Transferability Approach: Crossing the Reality Gap in Evolutionary Robotics,” *IEEE Transactions on Evolutionary Computation* 17(1), 122–145. DOI (<https://doi.org/10.1109/TEVC.2012.2185849>).

- [13] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil & Johanni Brea (2018), “Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules,” *Frontiers in Neural Circuits* 12:53. DOI (<https://doi.org/10.3389/fncir.2018.00053>).
- [14] Mehdi Keramati & Boris Gutkin (2014), “Homeostatic reinforcement learning for integrating reward collection and physiological stability,” *eLife* 3:e04811. DOI (<https://doi.org/10.7554/eLife.04811>).
-

RELATION MEMORY

CONTINUES [Water Was Already There](#) · Moves from the readable hydraulic bench to a falsifiable programme in which Water must compete against constrained graph representations.

NEARBY [I Don't Know](#) · Keeps the unresolved consciousness question separate from the engineering programme for persistent adaptive systems.

CANONICAL URL: [HTTPS://HEDEGREENRESEARCH.COM/ARTICLES/I-KNOW-WHAT-I-WANT-TO-BUILD/](https://hedegreenresearch.com/articles/i-know-what-i-want-to-build/)

PDF VIEW GENERATED: 2026-10-04T17:17:50.478Z