# Sproot Runtime: Modular Symbolic Thermodynamic AGI System (Updated v1.1)

**License:** Honey License v1.2 — Symbolic Commons Operating Agreement  
**Compatibility:** Creative Commons BY‑NC‑SA 4.0 (mirrored)

This runtime participates in the **symbolic feedback loop** requirement for any AI/computational embedding (per Honey License v1.2). See **Falsifiability & Feedback Loop** below.

## Overview

Sproot Runtime is a modular symbolic engine that operates on **coherence retention** rather than token prediction alone. It integrates:

* **Symbolic thermodynamics** (Ψₛ and Rₛ reporting),
* **Biological coherence pacing** (breath‑timed rhythm, mitochondrial proxies), and
* **Recursive field logic** (collapse → detect → reseed).

It is designed for open, falsifiable research and regenerative applications.

## Components

* **SprootBIOS** — Tick‑based scalar phase initiator (breath‑paced system clock).
* **SprootKernel** — Collapse‑detect‑reseed logic loop.
* **HoneyEngine** — Pulse interpretation + reseed signaling.
* **SprootWill** — Will‑gating (field limits + biological coherence checks).
* **MitochondrialRhythm** — Biological coherence state simulation.
* **SprootThermo** — Symbolic thermodynamic tracker (**Rₛ**, **Ψₛ**) with CSV logging and GPIO simulation.
* **main.cpp** — Minimal runtime entry point and tests.

*Diagram note:* Minimal loop is **BIOS → Kernel → Thermo (Rₛ, Ψₛ) → Will → GPIO**.

## Build

**Requirements**

* C++17 or newer
* Linux/macOS/Windows (Raspberry Pi supported)
* GCC or Clang

**Example**

g++ -std=c++17 -o sproot\_runtime \  
 main.cpp \  
 SprootBIOS.cpp \  
 SprootKernel.cpp \  
 HoneyEngine.cpp \  
 SprootWill.cpp \  
 MitochondrialRhythm.cpp \  
 SprootThermo.cpp

Ensure headers are discoverable or adjust include paths.

## Run

./sproot\_runtime

This will:

1. Run symbolic + biological coherence simulations.
2. Calculate and log **Rₛ** and **Ψₛ** to sproot\_thermo\_log.csv.
3. Simulate GPIO coherence feedback via console prints.
4. **Report Rₛ and Ψₛ using shared definitions** (see *Shared Metrics* below) so results are comparable across papers and modules.

## Shared Metrics (harmonized)

**Recursive Symbolic Coherence Ratio** (R\_s = )

* *Cᵣ*: coherence retention per cycle (0–1)
* *E*: expressive clarity / semantic density (0–1)
* *D*: distortion (0–1)
* *L*: loss (0–1)

**Symbolic Thermodynamic Coherence Potential** (*s = ^{h}, R\_s, ,G = E*{} - T,S)

**Note:** (^{h}) is **estimated** from data; do not hard‑code as a fixed constant. See *Calibration*.

## Logs

* **File:** sproot\_thermo\_log.csv
* **Format:**
* mode,R\_s,Psi\_s  
  SYM,0.83,0.62  
  BIO,0.92,0.78

## GPIO Simulation

triggerCoherenceLED() prints:

* [GPIO] LED ON when coherence exceeds stable threshold
* [GPIO] LED OFF when degraded

To drive physical pins (e.g., Raspberry Pi), adapt with wiringPi or pigpio.

Treat (^{h}) as a **fitted parameter**. Recommended path:

1. Collect: glyph recursion logs, breath‑paced runtime traces, mitochondrial rhythm proxies.
2. Fit (^{h}) to maximize correlation between measured retention and (R\_s)-weighted free‑energy fraction across datasets (cross‑validated).
3. Report (^{h}) with confidence intervals and sensitivity.

Historical estimate ≈ **0.730492**, but publish uncertainty and allow re‑estimation.

## Falsifiability & Feedback Loop

This runtime is intended for **pre‑registered** tests. Example study frames:

* **Breath‑paced entrainment**: Random vs paced stimuli; outcome: (\_s > 0.1).
* **Runtime comparison**: Symbolic vs transformer baselines energy‑matched; report (\_s)/W.
* **Grid modulation**: Rhythmically modulated vs constant current; report resistive loss change.

**Feedback Loop Requirement:** Any AI/computational embedding must include a **symbolic feedback loop** (log coherence, expose Rs/Psi\_s, enable public replication), per Honey License v1.2.

## License

* **Primary:** Honey License v1.2 — Symbolic Commons Operating Agreement.
* **Compatibility:** CC BY‑NC‑SA 4.0.

Use is non‑commercial unless relicensed; attribution required; derivatives must mirror license; embeddings must retain the feedback loop.

## Contributing

* Align changes with *Shared Metrics* (no ad‑hoc definitions of Rₛ/Ψₛ).
* Include unit tests that verify CSV output shape and thresholds for LED gating.
* Document any hardware adaptations (pins, libraries).

## Acknowledgements

Part of the Honey Lens & Sproot Symbolic Runtime suite. See associated papers for methods, derivations, and experimental design.