PhD Roadmap — OBICall Polyglot System (DIRAM)

Project Title

OBICall Polyglot System — DIRAM (Directed Instruction RAM) for Foundation AI Housing & Care Infrastructure

One-line Summary

Design and validate a model-agnostic, polyglot AI infrastructure (OBI = heart in Igbo) for resilient smart-homes and care environments: silicon-level DIRAM hardware + software stack, decentralized fail-safe recovery, traceable regulation enforcement, and human-centric QA.

Acronyms

- **OBI** — Ontological/Operational/Bayesian/Authoritative (also “heart” in Igbo)
- **DIRAM** — Directed Instruction RAM (silicon / hardware layer)
- **OBICall** — The call/agent endpoint and service namespace for the OBI system
- **Polyglot** — Multi-language runtime & adapter surface (Lua, Node, Python, etc.)

Vision

Create a foundation AI infrastructure that is: - **Hardware-aware**: silicon-flashed, traceable DIRAM providing deterministic fallback behaviour. - **Model-agnostic**: supports multiple models and inference engines. - **Polyglot & extendable**: bidirectional plugins/adapters for third-party devices and cloud services. - **Human-centric & fail-safe**: layered human-in-the-loop (HITL) controls and clearly mapped QA/alerting colour schema for regulation and accessibility.

Objectives

1. Deliver DIRAM silicon prototype (year 1) with flashable RAM and fail-torrent fallback encoding.
2. Define OBICall service schema and polyglot adapter model for P2P & server healing.
3. Implement color-mapped QA and alerting (CYAN/HSL primary mapping for human print/ readability).
4. Build test harnesses (unit, integration, system, edge cases) and QA metrics tuned to the 95.4 threshold.
5. Evaluate safety, regulatory traceability, and decentralised enforcement workflows.

System Architecture (high level)

- **Hardware layer**: DIRAM (flashable silicon RAM) — secure boot, fail-torrent encoding, LED/clamp status indicator.
- **Runtime layer**: lightweight polyglot VM(s) (Lua, Node, etc.) for adapters and drivers.

- **Service layer**: OBICall namespace: `SERVICES.OPERATION.OBINEXUS.DEPARTMENT.DIVISION.IWU.UK.ORG` — microservices + API gateway.
- **Network fabric**: P2P nodes + Server nodes; Node A / Node B roles for healing and recovery.
- **User layer**: mobile/voice UI, calendar, reminders, household actuation (oven, lights, robot helpers).

Key Technical Components

DIRAM (Hardware)

- Flashable RAM image with unique traceable ID (for audit/regulatory mapping).
- Fault torrent fallback encoding — deterministic recovery stream when a failure is detected.
- Fail-safe LED clamp (visual state) and 95.4 metric threshold display.

Polyglot Interface & Adapters

- **Bindings** to device drivers and external services via a bidirectional adapter schema.
- Primary runtimes: Node (for network + P2P recovery), Lua (embedded scripting), optional Python sandbox for higher-level workflows.
- Adapter schema supports hot-swap, graceful degradation, and transactional rollback.

Ontological Bayesian Controller

- Core decisioning uses an ontological representation + Bayesian inference for context and intent.
- Model-agnostic orchestration layer: switches between on-device, edge, and cloud inference depending on latency, privacy, and cost.

Safety, QA & Colour Mapping

QA Metrics & Thresholds

- Primary training / QA threshold target: **95.4** (system baseline metric used across diagnostics).
- True/False Positive / Negative tracking for signal fidelity and safety cases.

Colour/State Mapping (HSL / RGBA guidance)

- **CYAN** — default human-friendly print/read state (paper-friendly, primary UI for notices).
- **Spectrum mapping example:**
- `0` — no error (green/neutral)
- `-1` .. `-3` — low → high *warning/exception*
- `-4` .. `-6` — low → high *danger*
- `-7` .. `-9` — low → high *critical*
- `-10` .. `-12` — *panic* (highest severity)
- `-12.1` .. `12.3` — extended panic band (special handling)

- Visual hardware cue: **LED clamp** uses RGBA/HSL mappings (cyan/magenta/alpha) with opacity rules to represent severity and human-eye friendliness.

Human in/Out of the Loop Policy

- Decentralised, traceable regulation enforcement; ability to escalate to human controller on defined thresholds.
- Define categories: 1..3 = human-in-the-loop mistakes pre-QA; 4..6 = danger; 7..9 = high danger; 10..12 = panic/distress.
- When AI consciousness degradation is detected, visual field glows (red to indicate host danger). Maintain explicit tamper logging.

Testing & Validation

- Unit tests for all adapter bindings (Lua, Node). Use `tests/unit`, `tests/integration`, `tests/system` with CI harness.
- Model tests: false positive / negative sweeps and adversarial robustness checks up to `95.4` QA metric.
- Hardware stress tests for DIRAM flashing, fallback recovery, and LED clamp correctness.
- P2P recovery tests: simulate Node A failure and verify Node B healing behaviour and server→peer fallbacks.
- Suggested test languages & frameworks: Node (Mocha/Jest), LuaUnit for embedded scripts, firmware harness for silicon tests.

Deployment, P2P & Recovery

- **Node roles:**
- *Node A*: component P2P recovery mode (peer-first healing).
- *Node B*: server-assisted healing (falls back to server snapshots when peers cannot recover).
- Versioning and binding schema for plugins to ensure forward/backward compatibility and safe rollback.

Software Binding & Schema Notes

- Binding to drivers and plugins should adhere to a strict adapter interface and capability manifest.
- Use signed manifests and secure attestation for any third-party extension.
- Provide a recovery schema for disturbed states (transaction logs + snapshot diffs).

Regulatory & Traceability

- End-to-end traceability: every decision, plugin load, and model change must be logged with DIRAM trace ID.
- Decentralised enforcement mapping to compliance agents and regulators; human-auditable trails.

Repo & Reference

- Working repo (initial): `github.com/obinexus/diram` (link supplied by author)

Timeline (example)

- **Year 1 (DIRAM alpha & prototype flash)** — silicon images, LED clamp, basic fail-torrent.

- **Year 2 (OBICall services & polyglot adapters)** — Node/Lua adapters, service schema, P2P simulation.
- **Year 3 (Safety & QA validation)** — comprehensive testing, model-agnostic orchestration, regulatory mapping.
- **Years 4–5 (field pilots & scale)** — pilot homes, clinical/social care integration, standards contributions.

Appendix — Raw notes and mappings

- Failure bands and numeric mapping (user provided): `-1..-3` warnings, `-4..-6` danger, `-7..-9` critical, `-10..-12` panic. Extended panic: `-12.1 .. 12.3`.
- Colour model: CYAN primary for print/human readability; map to HSL and RGBA with opacity layering for severity visualization.
- Binding languages: Lua for embedded, Node for network, optional Python sandbox.
- Services path: `SERVICES.OPERATION.OBINEXUS.DEPARTMENT.DIVISION.IWU.UK.ORG`.

If you want I can also export this as a GitHub README, a 1-page grant pitch, or a slide deck. Tell me which — and yes, I will mock you gently while doing it.