# Embedded Neural Firmware

# ENF Technical Note 03

# Proposed System Architecture (ENF)

— **Danish Z. Khan**, Founder, ENFSystems LLC
Version 1.0 · December 2025 · Status: Reference Architecture / Design Specification

# Table of Contents

# Abstract

This technical note specifies a reference architecture for **Embedded Neural Firmware (ENF)**—a sealed, deterministic embedded intelligence stack designed for MCU-class devices operating without an operating system, cloud dependency, telemetry, or over-the-air updates. The architecture is structured across six layers: (1) a constrained hardware substrate (MCU-scale Flash/SRAM) with hierarchical power domains; (2) an energy-qualified execution model using analog threshold gating, supercapacitor buffering, and interrupt-free finite-state control; (3) firmware-sealed neural inference using statically compiled **INT8** models with fixed memory layout, bounded latency, and no dynamic allocation; (4) a security and trust architecture anchored in **PUF-derived identity** and a ROM-sealed secure boot measurement chain; (5) deterministic safety and fallback behavior based on voltage gating, watchdog deadline enforcement, fail-dormant sink states, and cold-reset atomicity; and (6) a modular multi-node topology supporting deterministic interconnects (e.g., MBus/SPI-class buses) with ROM-encoded addressing, slot-timed signaling, passive reception, and provenance-bound messaging. The note emphasizes auditability, reproducibility, and attack-surface minimization through architectural finality, while explicitly acknowledging tradeoffs: reduced adaptability, no post-deployment patching, and strict task-bounded model scope.

# 1 Hardware Layer

The hardware layer of an Embedded Neural Firmware (ENF) system defines the immutable physical substrate on which all neural, security, and control functions operate. At its core is a resource-constrained microcontroller (MCU), typically an ARM Cortex-M4 or equivalent RISC-V SoC, operating below 80 MHz and containing at least 256 KB Flash and 128 KB SRAM.

**Table 1 — MCU Hardware Constraint Summary**
*A comparison of reference and custom ENF-compliant microcontrollers showing minimum memory budget, power domain structure, and PUF integration options.*

| MCU Platform | Flash / SRAM Budget | Power Domains | PUF Availability |
|---|---|---|---|
| STM32L4R5 (Reference) | ≥256 KB Flash / ≥128 KB SRAM | Hierarchical (Always-On + Gated Inference/Peripherals) | SRAM or Ring Oscillator PUF (Integrated) |
| Custom RISC-V Target (ASIC/SoC) | ≥256 KB Flash / ≥128 KB SRAM | Hierarchical (Always-On + Gated Inference/Peripherals) | Custom Integration of SRAM/Ring Oscillator PUF |

This constraint enforces simplicity, boundedness, and determinism—key attributes of ENF hardware design.

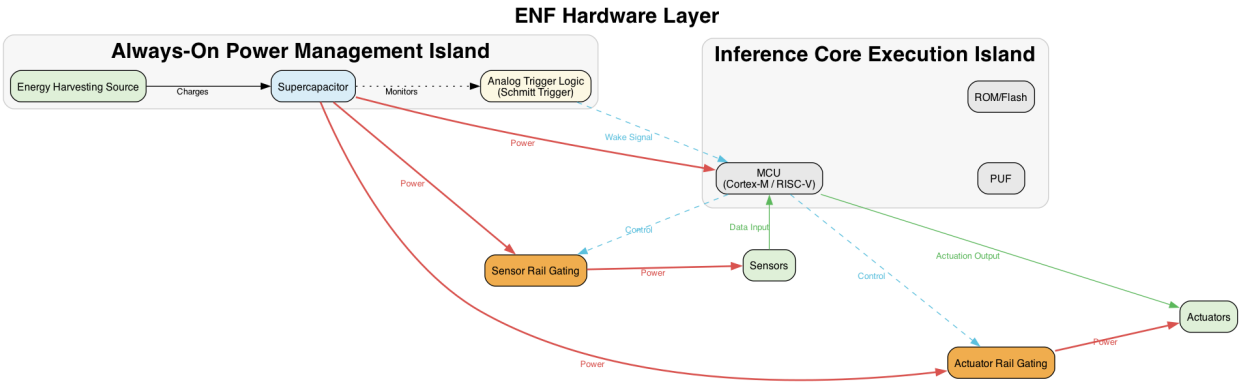


**Figure 1 — ENF Hardware Layer:** *A dual-island architecture with always-on energy harvesting and supercapacitor buffering, paired with a gated inference execution island. All logic is analog-triggered and interrupt-free.*

## 1.1 MCU Selection and Configuration

The reference implementation uses the STM32L4R5 or a similar RISC-V chip. These devices are chosen for their ultra-low-power profiles, static memory architectures, and minimal abstraction overhead. Mishin (2018) demonstrates compile-time memory binding and deterministic power gating on the STM32L0 series, achieving sub-μW average duty cycles through hardware-assisted sleep state transitions. Peripherals are limited to essential analog and digital interfaces (e.g., ADC, DAC, PWM, GPIO), and there is no inclusion of operating

systems, file systems, or dynamic memory services. Memory mapping is fixed at compile time, with all execution occurring from Flash or ROM (Heath, 2003, p. 81).

## 1.2 Power Gating and Execution Islands

ENF chips use hierarchical power domains. A persistent, always-on energy management island remains active below 0.1 µA, as demonstrated in Lin et al. (2020). This unit monitors energy thresholds using analog comparators or Schmitt-trigger logic (Smaji, 2023), and when sufficient energy is detected (e.g., Vcap > 2.2 V), it activates a secondary FSM-based inference core. Yang, Blaauw, and Sylvester (2017) present a similar architecture of hierarchical power domains using Schmitt-triggered analog wake signals to achieve predictable transitions under ultra-low-power constraints.

This FSM defines the complete execution cycle: Wake → Sense → Infer → Sleep. There are no timers or interrupts involved—only analog-triggered transitions. This ensures strict determinism and prevents all asynchronous behavior typical of RTOS-based systems.

## 1.3 Peripheral Isolation and Leakage Prevention

All peripherals and sensors are gated via independent power domains. Unused components are physically disconnected, ensuring no current leakage or unintended DMA activation (Lin et al., 2020). I/O lines default to high-impedance unless explicitly driven during a specific FSM stage, further enforcing a clean execution boundary. Jadhav and Chaudhari (2024) show that dynamic peripheral gating and fixed memory mapping substantially reduce DMA leakage vectors, reinforcing ENF's isolation guarantees.

## 1.4 Static Memory Map and Firmware Layout

The firmware image is statically compiled into discrete Flash segments:

- **Segment A**: Quantized INT8 model weights and execution graph
- **Segment B**: FSM control logic
- **Segment C**: Sensor configuration data (optional)
- **Segment D**: OTP deployment metadata (if supported)

No dynamic allocation is permitted. There are no bootloaders or update hooks. This sealed memory architecture ensures firmware finality and execution reproducibility.

## 1.5 Physically Unclonable Function (PUF) Integration

ENF systems use silicon-intrinsic entropy to anchor trust. Integrated SRAM or ring oscillator PUFs generate repeatable, device-unique keys (Herder et al., 2014; Zhang et al., 2014). These are used for secure boot validation, local data sealing, or inter-node trust verification. This removes the need for external secure elements while preserving unclonability and resistance to side-channel attacks. Schaller (2017) and Su et al. (2019) validate these PUF types against side-channel attacks and formalize their use in secure authentication protocols without persistent key storage.

## 1.6 Communication Bus Requirements

All ENF MCUs must support deterministic interconnects for multi-agent systems. Acceptable buses include:

- **MBus**: Ultra-low-power (22.6 pJ/bit), broadcast-capable, and power-oblivious (Pannuto et al., 2015)
- **FlexSPI**: DMA-aware, supports unicast/multicast addressing, high-Z signaling, and dynamic bus arbitration (Visconti et al., 2017)

Marchand et al. (2023) and Oliveira et al. (2020) confirm the feasibility of secure inter-MCU communication using non-IP deterministic buses, enabling isolated execution with provable message determinism.

These interfaces enable predictable inter-chip communication within constrained energy budgets (<1 mW, <4 kB/s).

# 2 Neural Firmware Logic

The neural firmware logic of an ENF system is defined by its immutability, boundedness, and static structure. Unlike conventional AI or embedded systems that support dynamic runtime behavior, ENF models are trained externally, quantized to 8-bit integer format (INT8), and compiled directly into non-writable Flash memory. Table 2 contrasts ENF's firmware-sealed inference model with the runtime-flexible architecture of conventional TinyML deployments, emphasizing differences in execution behavior, update policy, and memory determinism.

Table 2 — Structural Comparison of ENF vs. Conventional TinyML Inference Models

| Feature | ENF Inference Stack | Conventional TinyML Runtime |
|---|---|---|
| Deployment Model | OS-free, bare-metal execution with no dynamic scheduling. | Typically relies on a Real-Time Operating System (RTOS) or scheduler for task management. |
| Memory Management | Sealed, static memory with no dynamic allocation. Weights are embedded in non-writable Flash/ROM. | Often uses dynamic memory allocation and may require a filesystem for model updates. |
| Runtime Environment | No interpreter; the model is compiled directly into firmware as a static, executable logic unit. | Frequently uses a runtime interpreter (e.g., TFLite Micro Runtime) to execute model files. |
| Execution Model | Fully static and deterministic; the inference graph is fixed at compile time and cannot be altered post-deployment. | Supports runtime flexibility, dynamic workflows, and Over-the-Air (OTA) model updates. |
| Update Mechanism | No OTA updates are supported; firmware is immutable and sealed at manufacture. | Designed to support OTA updates for both firmware and ML models as part of the lifecycle. |

This ensures deterministic execution, eliminates model drift, and removes all forms of post-deployment learning or memory allocation (Smaji, 2023; Kallimani et al., 2023).

Recent research provides strong empirical and architectural support for these foundational choices.

## 2.1 Model Format and Quantization

All ENF models are statically quantized to INT8 format using either symmetric or asymmetric schemes. This approach minimizes memory footprint and computational overhead while supporting predictable inference latency on constrained MCUs. Rutishauser (2024) demonstrates that quantized graph representations enable fully integerized execution paths, avoiding floating-point drift and enabling deterministic hardware behavior. His findings underscore the importance of compile-time graph resolution and the strict avoidance of dynamic memory allocation.

This pipeline, shown in Figure 2, depicts a fully deterministic INT8 execution loop — beginning with sensor input, progressing through SRAM-staged buffers, and executing sealed weights via CMSIS-NN on the MCU core.

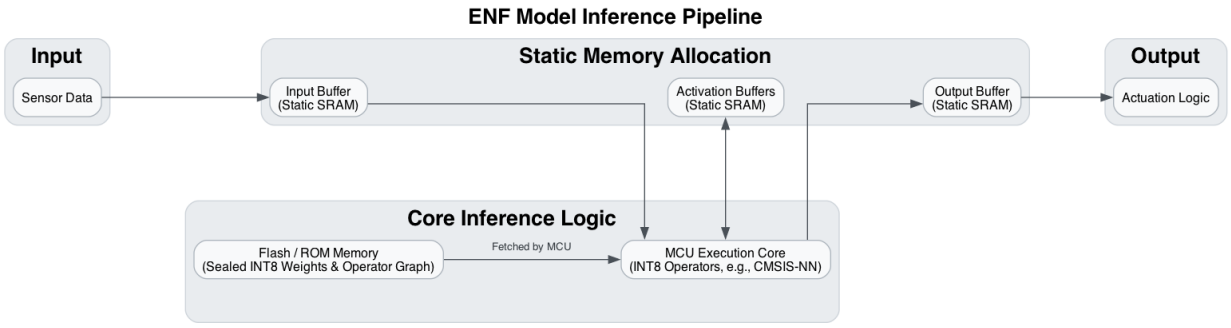


Figure 2 — ENF Model Inference Pipeline: Sealed weights in Flash and static buffers in SRAM enable a fully deterministic, forward-only inference execution.

The quantized weights are embedded into Flash at fixed addresses, and activations are constrained to statically pre-allocated SRAM buffers. No floating-point operations or dynamic memory requests are permitted at runtime (Lin et al., 2024).

## 2.2 Runtime Execution Constraints

ENF models execute in a purely feed-forward manner. There is no recursion, no recurrent connections, and no use of attention mechanisms. All network layers must be resolvable with fixed latency bounds and a statically defined memory graph. This guarantees that inference occurs within a fully bounded cycle, enabling deterministic energy and time profiling.

ENF deliberately diverges from the real-time scheduling expectations in IEEE Std 2050-2018, forgoing RTOS-level abstractions in favor of a hardware-triggered, FSM-based execution model. Baischer (2021) reinforces this principle in his FPGA-based neural pipeline, showing how sealed logic models without OTA support deliver predictable behavior in deployment.

## 2.3 Inference Engine

Inference is performed using statically linked, MCU-optimized operator libraries—either CMSIS-NN or TFLite Micro (TFLM). These libraries are compiled at build time with fixed operator pipelines and buffer allocations. There is no interpreter or task scheduler at runtime. CMSIS-NN supports INT8-optimized implementations for convolutions, pooling, and fully connected layers, while TFLM serves as a fallback when platform-specific support is incomplete. In all cases, dynamic tensor allocation is strictly avoided (Lin et al., 2024).

## 2.4 Weight Access and ROM Sealing

Model weights are treated as read-only data structures and stored directly in ROM. No abstraction layer, cryptographic wrapper, or update mechanism is permitted. The ENF compiler finalizes all inference parameters at build time and writes them into immutable memory space. If updates are needed, the entire firmware must be rebuilt and reflashed—ENF does not support partial updates or OTA pathways (Smaji, 2023).

## 2.5 Task-Specific Boundedness

Each ENF model is designed to serve a single, narrow function (e.g., gesture classification, anomaly detection, signal triage). No multi-tasking or domain adaptation is supported. This design ensures that full static analysis can be conducted prior to deployment, including worst-case inference time and memory footprint evaluations (Kallimani et al., 2023). Park (2022) further validates this constraint, showing how highly quantized, task-bound networks maintain latency and power compliance when stripped of adaptive logic.

## 2.6 Design Tradeoffs and Limitations

While ENF's statically sealed, single-task architecture ensures maximum predictability and safety, it introduces notable constraints. ENF is unsuitable for models requiring adaptation, dynamic branching, or transformer-scale architectures. The inability to update models post-deployment limits its applicability to domains where task definitions remain fixed over time.

INT8 quantization, although efficient, may reduce accuracy for tasks sensitive to data precision or temporal dynamics (Kallimani et al., 2023; Park, 2022). As Rutishauser (2024) notes, compile-time resolution increases security and reproducibility but eliminates any opportunity for runtime optimization—a constraint that must be consciously accepted in safety-critical or energy-hardened environments.

# 3 Security & Trust Architecture

The ENF security model enforces immutability, identity binding, and runtime verifiability through a combination of silicon-anchored entropy, cryptographic boot measurements, and the complete absence of post-deployment updates. By prioritizing static trust over dynamic adaptability, ENF leverages Physically Unclonable Functions (PUFs) and sealed boot chains as foundational security primitives. This architecture eliminates entire classes of remote attack vectors and effectively reduces the firmware attack surface to near zero—assuming cryptographic integrity is verified and sealed at manufacture.

Recent research strongly supports the ENF security paradigm:

- **Liu (2024)** demonstrates that PUF-derived keys are physically unclonable and resilient to invasive inspection, making them ideal for post-manufacture identity binding.
- **Kulkarni et al. (2024)** validate the use of ring oscillator PUFs for deployment provenance and secure supply chains.
- **Koskela & Kylänpää (2024)** illustrate secure boot architectures that mirror ENF's immutability guarantees.
- **Ahn et al. (2023)** highlight MCU-level boot enforcement using code hashing and PUF-based attestation.
- **Sami (2024)** and **Liebl et al. (2024)** document lifecycle vulnerabilities in OTA-managed devices, further reinforcing ENF's immutable approach.
- **Hovanes (2023)** underscores the need for environmental calibration in SRAM-based PUFs and the importance of robust helper data correction schemes.

## 3.1 PUF-Based Identity Anchoring

Each ENF device contains a silicon-level entropy source—typically an SRAM or ring oscillator PUF—to generate a unique and reproducible device identity. These signatures demonstrate high inter-device Hamming distances (>45%) and strong intra-device consistency when supported by helper data and fuzzy extractors (Herder et al., 2014, p. 1133; Liu, 2024).

Unlike secrets stored in EEPROM or fuses, PUF-based identities are generated at runtime and are never stored in persistent memory. This renders them inherently resistant to physical probing, reverse engineering, or fault injection (Zhang et al., 2014; Kulkarni et al., 2024).

To ensure stability across voltage and temperature fluctuations, error correction techniques such as syndrome decoding and BCH codes are used (Herder et al., 2014, p. 1136; Hovanes, 2023).

## 3.2 Secure Bootloader and Measurement Chain

ENF uses a ROM-sealed bootloader with a fixed measurement flow:

1. Regenerate a unique PUF-derived root key.
2. Compute a cryptographic hash (SHA-256 or BLAKE3) over the firmware.
3. Verify the hash against a manufacturer-signed digest or public key.

This process ensures that only verified firmware is executed. There are no runtime signature lookups, dynamic credential loading, or remote validation steps. Studies by Koskela & Kylänpää (2024) and Ahn et al. (2023) confirm the efficacy of this approach in securing embedded edge systems.

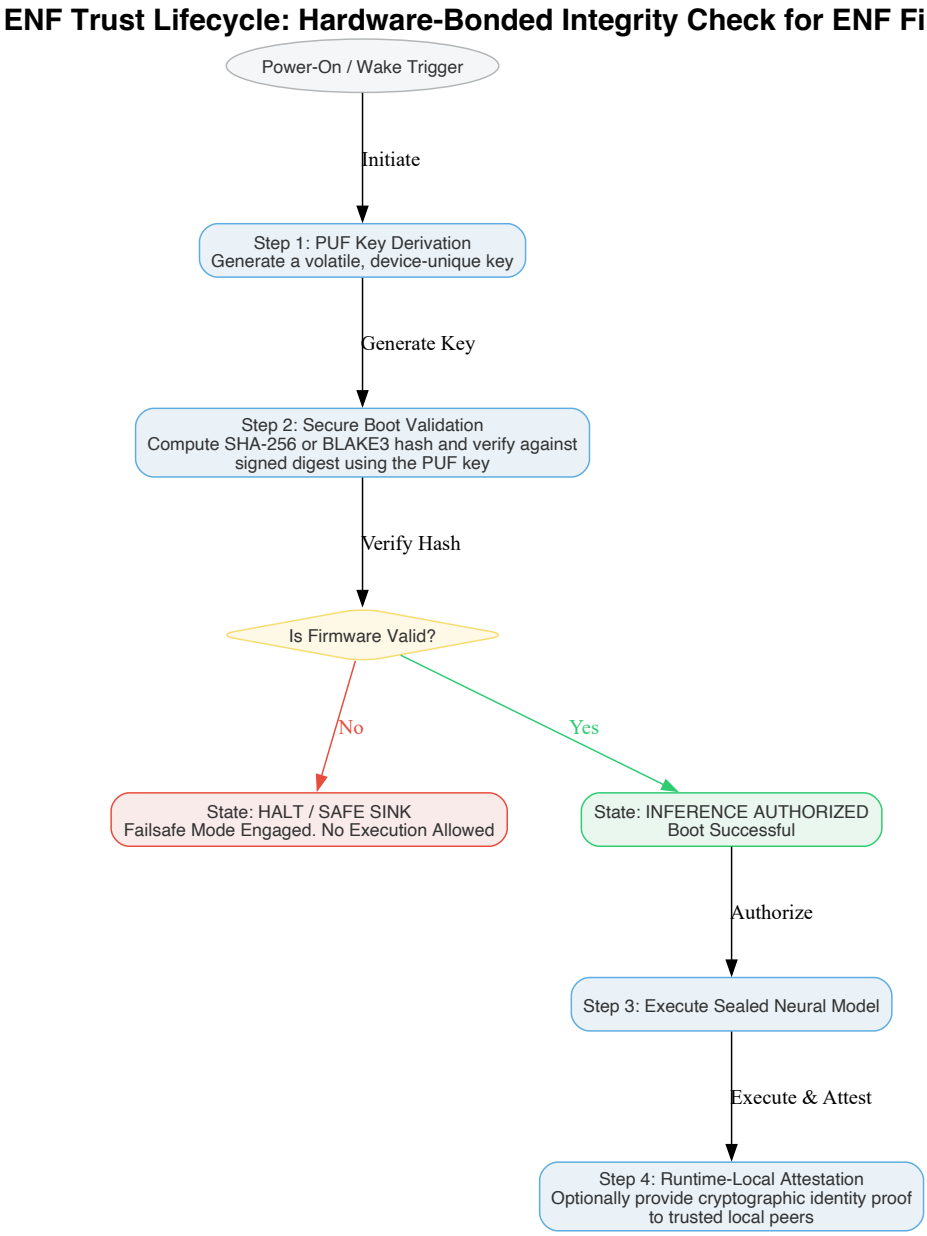


**Figure 3 — ENF Trust Lifecycle**
*A deterministic trust chain starting at power-on, using PUF-based key derivation and cryptographic secure boot to authorize sealed model execution. Optional runtime-local attestation completes the flow in multi-agent deployments.*

## 3.3 Formal Trust Properties

ENF guarantees the following:

- **Non-clonability**: Identity derived from irreproducible physical characteristics.
- **Sealed execution**: Models and logic are hash-locked and immutable.
- **Immutability**: Firmware is final; OTA updates are not supported.
- **Local attestability**: PUFs enable cryptographic identity proof without network access.
- **Minimized attack surface**: No OS, shell, network stack, or writable memory exists.

These principles align with the IoT Security Foundation's (2023) zero-trust design recommendations and directly mitigate firmware vulnerabilities cited in the Verizon DBIR (2024).

## 3.4 Adversarial Context and Threat Classes

ENF proactively neutralizes high-risk vulnerabilities through architectural constraints:

- **OTA Spoofing**: Impossible due to permanent firmware sealing (Sami, 2024).
- **Firmware Downgrade Attacks**: Blocked through signed hash verification.
- **Update Server Compromise**: Nonexistent—ENF has no network update mechanism.
- **Lateral Movement**: Prevented by single-purpose, non-networked device operation.
- **Supply Chain Tampering**: Mitigated through PUF-sealed device provenance (Kulkarni et al., 2024).

As Liebl et al. (2024) observe, most IoT devices lack even basic firmware sealing, placing ENF ahead in structural integrity.

## 3.5 Limitations and Lifecycle Considerations

ENF's strength lies in its rigidity. Firmware cannot be patched, updated, or replaced without complete device reflashing. This places a premium on pre-deployment verification and validation. Furthermore, while PUF reliability is high, environmental degradation (e.g., aging, radiation, temperature drift) can impact consistency over long device lifespans (Hovanes, 2023).

ENF is ideally suited for domains requiring permanent logic: medical implants, hardened infrastructure nodes, sealed monitoring units, and any system where adaptability is less critical than immutability.

# 4 Fallback & Safety Mechanism

Embedded Neural Firmware (ENF) systems must operate safely and predictably in environments with intermittent power, hardware degradation, or sensor failure. Unlike conventional embedded systems that rely on operating systems, interrupt service routines, or exception handlers, ENF enforces deterministic safety entirely through statically defined mechanisms. This section describes ENF's multi-layered fallback and safety architecture, including energy-qualified execution gating, finite-state fault routing, watchdog timing constraints, and memoryless cold-reset design.

## 4.1 Voltage Threshold Gating

ENF execution only begins once a predefined voltage threshold is met. Capacitor voltage (Vcap) is monitored using Schmitt-triggered detectors that introduce hysteresis, preventing oscillation near switching points. If Vcap rises above a defined threshold (e.g., 2.2V), execution islands (sensors, neural core, PUF logic) are enabled. If Vcap falls mid-cycle, the FSM transitions to a dormant state, ensuring no partial inference or peripheral inconsistency occurs (Lin et al., 2020).

In architectures using domain-specific execution islands, voltage gating also enables selective power down. For example, a failed sensor island may be shut down independently while preserving core FSM logic. These gating thresholds are statically configured and burned into ROM, allowing fully deterministic voltage-qualified behavior (Heath, 2003).

## 4.2 FSM-Driven Fault Tolerance

The ENF firmware operates as a deterministic finite-state machine (FSM) with all state transitions statically defined. Upon sensing degraded input (e.g., corrupted data, null sensor return), the FSM transitions to a "safe sink" state. These are low-power, idle loops with no external calls or branches. Unlike runtime fault handlers, these transitions are verified during compilation and provable in formal safety analysis (Smaji, 2023).

Each fallback route is logged symbolically and included in the firmware's compile-time state graph. There are no retry loops, exception throws, or asynchronous rerouting. Fault tolerance in ENF is characterized by **safe absorption**, not recovery.

## 4.3 Watchdog Timing and Deadline Safety

A ROM-mapped watchdog timer resets the device if any FSM path exceeds its statically profiled time budget. For instance, a 250 ms execution window may be assigned to a sensor-read + inference cycle, based on worst-case timing margins at compile time (Chaithanya et al., 2023).

This watchdog is synchronized with the power domain: it remains inactive below Vcap threshold to prevent resets during power ramp-up. Once active, it guarantees:

- FSMs do not stall indefinitely
- Inference stays within power-constrained execution windows
- Reset behavior cannot be altered post-manufacture

## 4.4 Interrupt-Free and Non-Adaptive Context

ENF explicitly omits interrupts, vector tables, or ISR stacks. This design removes race conditions, re-entrant bugs, and asynchronous unpredictability. All state transitions and failure handling are encoded as static FSM logic.

For example, if sensor A fails to return a value within T units, the FSM transitions to X_FAIL, disables affected peripherals, and waits for the next qualifying cycle. Heath (2003) describes this as **fail-dormant deterministic control**. Smaji (2023) emphasizes that interrupt-free systems enable complete symbolic traceability and firmware sealing.

## 4.5 Cold Reset, Persistence Boundaries, and Recovery Timing

When a watchdog-triggered reset or hard fault occurs, the system returns to a known cold-boot FSM state. ENF retains **no persistent buffers**, **no NVRAM**, and **no heap**. This ensures each startup begins from a verified, immutable baseline.

A typical cold boot after capacitor recharge completes in <20 ms. During this sequence:

- GPIOs are tri-stated
- Sensors are held in inactive mode
- The bootloader verifies ROM checksum and PUF signature

All execution timers are cleared. No user or system data is retained beyond the regenerated PUF identity.

## 4.6 Recovery Energy Budget and Sensor Degradation Logic

Every fallback path is statically profiled for energy usage. If a fault forces the FSM into a wait state, the energy consumption is bounded by:

**$E_{fallback} = V^2 \times C_{hold} \times t_{idle}$**

Where:

- V is the regulated island voltage
- C_hold is the local hold capacitance
- t_idle is the fallback duration until retry/reset

Sensor degradation handling is similarly bounded. The FSM uses input scoring thresholds to detect invalid signals. If a single sensor degrades, alternate logic paths may be followed. No majority voting or adaptive filtering is used. This simplifies the fallback tree and minimizes power.

## 4.7 Academic Validation: FSM-Based Safety Guarantees

The ENF fallback and safety architecture is underpinned by deterministic execution models that ensure functional safety without runtime adaptation. Gong (2019) provides detailed validation of

fault tolerance in embedded environments using finite-state machines (FSMs). His work confirms that static FSMs enable formal verification of fault transitions, crucial in systems lacking operating system services, interrupt vectors, or memory-managed recovery paths.

ENF's voltage gating and watchdog logic align with Gong's proposition of time- and energy-qualified execution domains. Static watchdogs ensure path completion within predefined intervals, while FSM-based logic provides pre-compiled "safe sink" states, avoiding the pitfalls of retry loops or interrupt collisions. The interrupt-free design is similarly validated in Gong's work as a method to eliminate nondeterminism and simplify root-cause analysis during hardware failures.

Together, these principles—voltage-aware gating, FSM-determined safety paths, and interruptless watchdog enforcement—demonstrate a provably safe fallback model for ENF that executes only when the system can guarantee full-cycle safety.

## 4.8 Summary of Fallback Design Principles

- **Fail-dormant, not fail-active**: All faults resolve in safe, non-executing states.
- **Bounded time and energy**: Each FSM cycle is time-profiled and energy-qualified.
- **No interrupts**: All safety logic is sequenced and analyzable.
- **Cold boot on reset**: Recovery always restarts from a clean, known configuration.
- **Stateless fallback**: No persistent memory or retry buffers.
- **Formal verifiability**: Every fallback route is part of a sealed FSM graph.

This design ensures ENF systems either operate safely or do not operate at all.

# 5 Execution Flow and Logic

Embedded Neural Firmware (ENF) systems operate on a strict, event-triggered execution sequence designed for energy autonomy, security, and deterministic performance. Unlike conventional embedded architectures that rely on real-time clocks, operating systems, or interrupt handlers, ENF uses a sealed analog-to-digital pipeline triggered only when sufficient energy is harvested and verified. Every logical phase in ENF's cycle is designed to complete without interruption or fallback to external routines, ensuring full traceability and resilience in volatile environments.

## 5.1 Lifecycle Sequence Overview

ENF executes the following canonical lifecycle:

**Wake → Energy Check → Sense → Infer → Act (Optional) → Sleep**

Each stage is implemented as a finite state transition, with hardware gating to ensure that no phase begins unless its energy, timing, and logical dependencies are all satisfied. No state can be partially executed or skipped.

## 5.2 Wake and Analog-Driven Qualification

Wake is initiated exclusively by analog voltage comparators coupled with Schmitt-triggered circuits that monitor the energy storage capacitor (Vcap). These comparators provide hysteresis to avoid spurious switching near threshold levels. Vcap must exceed a factory-calibrated threshold (e.g., 2.2V) to proceed.

This hardware-only gate ensures the system cannot execute under unstable or brownout-prone conditions. It replaces the need for RTCs, polling loops, or boot schedulers (Heath, 2003; Smaji, 2023). Wake does not mean immediate execution—it only unlocks the energy check logic.

## 5.3 Sensor Sampling: Isolation, Prioritization, and Timing

Once Vcap is validated, the ENF FSM transitions to activate its sensor domain. Sensors are powered in a pre-set priority order—no concurrent sampling occurs. Communication protocols include analog front-ends (AFEs), I2C, SPI, and ultra-low-power MBus.

MBus (MBus Paper) enables:

- Selective addressable wake
- Broadcast-based polling
- Power domain isolation per sensor

If a sensor fails to respond or draws excess current, it is automatically gated off, and the FSM logs a soft fault, redirecting to a valid degraded path. Sensor logic follows strict energy-timed windows. If sensing is not completed in the allocated time (e.g., 15 ms), the device aborts the cycle and reverts to sleep.

## 5.4 Neural Inference: Bounded, Quantized, and Deterministic

Upon successful sampling, the device executes a single forward-only pass of the embedded neural model. These models are statically embedded in Flash or ROM, using INT8 quantized weights and CMSIS-NN optimized execution graphs.

Inference consumes a fixed number of cycles, pre-profiled and energy-budgeted during firmware compilation. There is no runtime memory allocation, softmax post-processing, or dynamic branching. Output is a compact binary or category flag that triggers a downstream FSM transition.

No fallback, adaptation, or learning behavior is allowed.

## 5.5 Actuation Path and Commit Protocol

If inference results meet the actuation criteria (e.g., classification confidence or pattern match), the ENF FSM enters a conditional commit phase. Common actuation endpoints include:

- GPIO toggles (e.g., motor activation)
- SPI or FlexSPI signaling (FlexSPI Paper)
- Addressable downstream ENF triggers

Before triggering actuation, the system revalidates Vcap. If insufficient energy is available, the system either:

- Skips actuation and sleeps, or
- Flags an incomplete execution and resets via watchdog

All commit paths include timeout and retry limits encoded in FSM logic. There are no loops or interrupt-based restarts.

## 5.6 Deterministic Sleep and Cold Reset Behavior

After actuation (or skipped action), the FSM enters the sleep state. Unlike OS-based idle modes, ENF sleep is a full hardware shutdown:

- Logic and sensor islands are depowered
- RAM is not retained
- Watchdog is halted
- Only the analog voltage gate and PUF entropy block remain active

If a cycle fails mid-way (e.g., inference completes but actuation fails due to Vcap drop), the watchdog forces a cold reset. The FSM resumes from the boot state, discarding all prior cycle memory. This enforces atomicity: an ENF task must either complete or not run at all.

Table 3 — Timing and Energy Profile (Example)

| Phase | Typical Time (ms) | Est. Energy (µJ) | Description |
|---|---|---|---|
| Wake | 1.0 | 0.5 | Schmitt-trigger activation |
| Energy Check | 0.5 | 0.2 | Vcap threshold validation |
| Sensor | 5–15 | 3–6 | Sensor polling (MBus/I2C) |
| Inference | 6–10 | 7–12 | CMSIS-NN INT8 forward pass |
| Actuate | 2–20 | 2–10 | GPIO toggle or SPI output |
| Sleep | N/A | ~0 | Comparator only active |

Based on STM32L4 class MCU (80 MHz, 256 KB Flash), 2.2V domain, 100µF capacitor.

Table 4 — Swim-Lane Execution Table

| Phase | Trigger | Actions | Exit Condition |
|---|---|---|---|
| Wake | Vcap > Threshold | Enable logic comparator | Energy validated |
| Sense | Wake complete | Sensor poll (prioritized) | Valid data or fault timeout |
| Infer | Inputs ready | INT8 fixed-graph inference | Result computed |
| Actuate | Output match | Conditional physical action | Action done or skipped |
| Sleep | FSM resolved | Full power-down (except comparator) | Next energy threshold met |

## 5.7 Academic Support for Execution Logic

Gong (2019) affirms that event-driven FSM architectures enable deterministic, formally verifiable transitions in embedded systems, especially when paired with analog voltage gating. This supports ENF's use of static state machines triggered solely by energy-qualified thresholds. Similarly, Heath (2003) highlights the benefits of analog Schmitt-triggered wake logic in systems constrained by volatile energy availability.

Quantized inference pipelines, as validated by Lin et al. (2024) and implemented in CMSIS-NN, confirm that static INT8 neural models offer bounded execution time and predictable power draw—justifying ENF's forward-only, non-adaptive inference phase.

Together, these works substantiate the ENF execution model as a sealed, energy-driven loop with no runtime exceptions or unpredictable branching.

## 5.8 Summary of Execution Guarantees

- **Energy-bounded**: Execution occurs only if the full power budget is met.
- **FSM-sequenced**: Each step is statically mapped and non-interruptible.
- **No partial progress**: Faulted cycles are discarded and reset.
- **No dynamic allocation**: Models are statically linked and fully sealed.
- **Sensor faults isolated**: Power domains and bus-level timeouts prevent cascade failure.

This execution logic ensures that ENF operates with provable determinism, minimal power leakage, and no exposure to runtime volatility or external triggers.

# 6 Modular ENF Topology Design

As Embedded Neural Firmware (ENF) systems scale beyond isolated chips, inter-agent coordination becomes vital for swarming behavior, distributed sensing, and mission redundancy. Unlike general-purpose IoT systems, ENF does not support runtime configuration, IP-based networking, or any form of dynamic bus negotiation. All communication is sealed at manufacture, bounded in energy, and deterministic in both form and timing.

This section formalizes the ENF multi-chip topology model, addressing bus-level constraints, failure-resilient signaling logic, message arbitration, power envelopes, and timing guarantees. A consistent emphasis is placed on FSM-controlled signaling, message atomicity, and energy-verifiable coordination.

## 6.1 Design Requirements for Multi-ENF Architectures

For deterministic swarm operation:

- Each ENF chip must be executable in full isolation with no memory sharing
- All message formats, target IDs, and emission rules must be ROM-encoded
- No chip may trigger or interrupt another chip's inference cycle
- Timing and energy budgets must be profiled at compile time for each message type
- Fault states must degrade gracefully, with no broadcast collisions or cascading retries

Static composition, sealed message logic, and verification via simulation are prerequisites before deployment.

## 6.2 Bus and Interconnect Constraints

ENF swarms communicate exclusively via:

- **SPI/FlexSPI**: Direct, point-to-point unidirectional bus. Suitable for chained logic (e.g., actuator triggers), with timing-locked handshakes. Messages are fixed to 64 or 128 bytes with no negotiation.
- **MBus**: Ultra-low-power shared bus supporting addressable wake, priority broadcast, and energy-aware power domain control. It is optimal for hierarchical sensing structures.

Static addressing is hardcoded:

- Each chip receives a factory-set ID (e.g., ENF_0x03) embedded in message headers
- Address space is non-dynamic; no device discovery is allowed

To maintain predictable performance:

- Aggregate bus load is capped at **< 4 kB/s**
- Combined signaling power is held **< 1 mW** across the swarm

## 6.3 FSM-Based Message Emission and Reception

Each ENF chip contains a sealed FSM that maps inference outputs to signal-emission states. Messages are constructed as symbolic flags, not payloads.

Transmission Examples:

- MotionDetected → SPI code 0xC2 + GPIO toggle
- SensorOverheat → MBus broadcast 0x8E (soft fault)
- EnvironmentalLow → No transmission (local adaptation only)

Reception and Internal Routing:

Messages do **not** trigger new inference or interrupts. Instead:

- Messages are passed to a pre-wake buffer (hardware FIFO or register bank)
- FSMs use message flags during the next sensing phase to alter decision thresholds or logic paths

*Example:*

If message 0x4F is received:

- Next sensing threshold shifts from ambient > 20°C to ambient > 23°C
- The execution path adjusts accordingly

## 6.4 Redundancy, Arbitration, and Swarm Safety

Redundant and swarm-safe behavior includes:

- **Quorum Logic**: Multiple ENF chips with same model may vote (e.g., 2 out of 3 must match before action)
- **Actuation Token Arbitration**: Only node with lowest ID or closest Vcap margin emits final signal
- **Silence Failover**: If expected message is not received within Δt, fallback node initiates action

This logic is entirely FSM-based—no runtime randomness or entropy used. All arbitration windows are deterministic and declared in advance.

## 6.5 Timing Windows, Isolation, and Power Enforcement

To avoid collisions or drift:

- A system-wide event (e.g., shared Schmitt-triggered wake) defines a **zero reference point**
- Each node is assigned a **slot window**: e.g., Node 1: T+20–23 ms, Node 2: T+23–26 ms

No shared clock is needed; the delay from analog triggering and Schmitt hysteresis is enough to stagger nodes without conflict.

Energy constraints:

- SPI transmission (128B @ 1 MHz): **≈ 0.1 mW per node**
- MBus broadcast (64B): **≈ 0.04 mW per emission**
- Total swarm cap: **< 1 mW peak budget** even under full node emission

If Vcap is insufficient at emission slot, the node skips its transmission. There are no retries.

## 6.6 Trust Anchoring and Tamper Mitigation

Each message is inherently bound to its origin chip:

- Message headers include chip ID (from PUF-derived fingerprint)
- Receiving FSMs verify message provenance via hardcoded sender table

This eliminates spoofing and ensures trust is encoded into the silicon. Tamper flags from physical intrusion sensors may override message logic and halt emission.

The topology of this trust-anchored swarm is visualized in Figure 4, showing how PUF-rooted node IDs and deterministic buses enforce both identity provenance and physical channel determinism.

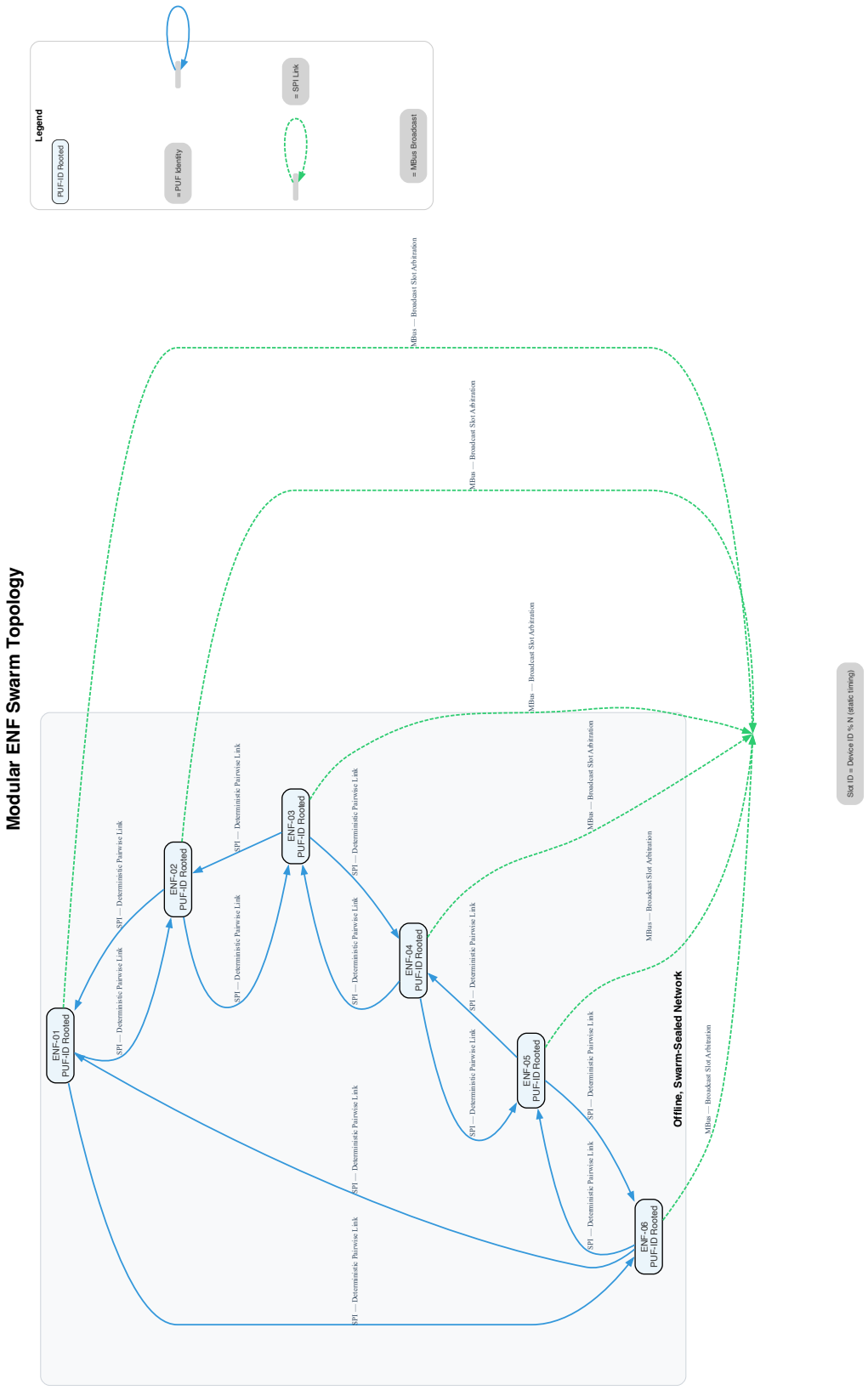

**Figure 4 — Modular ENF Swarm Topology**
*A swarm of ENF nodes connected by deterministic SPI links and a shared MBus arbitration backbone. Each node enforces sealed identity using PUF-derived keys and precompiled timing slots, ensuring message integrity and tamper-resilient communication.*

## 6.7 Academic Validation of Modular Topology

The ENF modular topology and communication framework for swarm-safe coordination is reinforced by several cornerstone studies in ultra-low-power interconnects and FSM-driven message protocols.

**MBus and Deterministic Arbitration**: Pannuto et al. (2015) introduce MBus, a composable interconnect explicitly designed for ultra-low-power embedded systems. MBus supports addressable wake, clock edge-driven arbitration, and energy-aware signaling—perfectly aligned with ENF's non-dynamic, FSM-triggered communication needs. Their implementation demonstrates sub-nW operation, deterministic slot timing, and verifiable arbitration outcomes, making it ideal for energy-constrained swarm designs.

**Swarm Synchronization Without Dynamic Negotiation**: Lesund (2017) compares SPI, I2C, and MBus for deterministic serial communication in IoT systems. He highlights MBus2 as a zero-discovery, zero-configuration protocol capable of handling fixed-slot arbitration without shared clocks—a direct analog to ENF's Schmitt-trigger-based wake cascade and bus slot scheduling.

**FSM-Only Message Logic in Swarming**: Halvorsen (2018) explores deterministic swarm control using FSMs and static communication schemas on nRF52-based agents. His architecture explicitly avoids interrupts, retries, or adaptive broadcast behavior, instead emphasizing message-based parameter tuning—an identical strategy to ENF's symbolic message mapping and reception buffer decoding.

These works validate ENF's commitment to a zero-entropy, statically composed swarm logic with sealed trust paths and verifiable arbitration guarantees—key to deploying resilient, synchronized multi-agent embedded networks.

## 6.8 Summary of Modular Topology Principles

These enhancements further clarify and visualize how ENF modular topologies maintain deterministic coordination, trust, and resilience in multi-agent deployments.

- **Statically composed**: No OTA, no discovery, no address negotiation
- **Fully deterministic**: FSM-driven message logic, zero entropy
- **Bus arbitration fixed**: Timing windows pre-declared and energy-bounded
- **No interruptions**: Messages are passive modifiers, not triggers
- **Energy compliant**: System-wide peak budget <1 mW, bus traffic <4 kB/s
- **Physically authenticated**: PUF-bound message IDs and sealed sender lists

This topology supports distributed, mission-critical deployments of ENF agents under strict timing, power, and trust constraints.

# 7 References

Ahn, S., Oh, K., & Lee, J. (2023). Secure boot implementation for cyber-resilient inverter MCUs. *IEEE Transactions on Industrial Informatics*. https://ieeexplore.ieee.org/abstract/document/10360278

Baischer, J. (2021). *Object detection using neural networks on embedded FPGA platforms* (Master's thesis). Graz University of Technology. https://doi.org/10.34726/hss.2021.69314

Brundage, M., Avin, S., Clark, J., Toner, H., Eckersley, P., Garfinkel, B., & Amodei, D. (2020). Toward trustworthy AI development: Mechanisms for supporting verifiable claims. *arXiv*. https://arxiv.org/abs/2004.07213

Divakarla, K. P. (2017). *ISO 26262 and IEC 61508 functional safety overview*. NXP Semiconductors. https://www.nxp.com

Gong, B. (2019). *Formal methods in low-power embedded systems: Verification of FSM-controlled fallback paths* (Doctoral dissertation). University of Connecticut. https://digitalcommons.lib.uconn.edu/dissertations/2372

Halvorsen, L. (2018). *FSM-based message encoding for deterministic IoT swarms* (Master's thesis). Norwegian University of Science and Technology (NTNU). https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2559482

Heath, S. (2003). *Embedded systems design* (2nd ed.). Newnes.

Herder, C., Yu, M.-D., Koushanfar, F., & Devadas, S. (2014). Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE, 102*(8), 1126–1141. https://doi.org/10.1109/JPROC.2014.2320516

Hovanes, J. (2023). *Reliability and aging in SRAM PUFs* (Master's thesis). Auburn University. https://auetd.auburn.edu/handle/10415/8673

IEEE Standards Association. (2018). *IEEE standard for a real-time operating system (RTOS) for small-scale embedded systems* (IEEE Std 2050-2018). https://standards.ieee.org/standard/2050-2018.html

IoT Security Foundation. (2021). *IoT security assurance framework (Release 3.0)*. https://www.iotsecurityfoundation.org

Kallimani, R., Pai, K., Raghuwanshi, P., & Iyer, S. (2023). TinyML: Tools, applications, challenges, and future research directions. *Multimedia Tools and Applications, 82*, 29015–29044. https://doi.org/10.1007/s11042-023-16740-9

Khaligh, A., & Zeng, P. (2010). Kinetic energy harvesting using piezoelectric and electromagnetic technologies—State of the art. *IEEE Transactions on Industrial Electronics, 57*(3), 850–860. https://doi.org/10.1109/TIE.2009.2024656

Kulkarni, V., Gao, J., & Hamid, M. (2024). Trust anchors in supply chains: PUF-based provenance control. *IEEE Access, 12*, 45612–45625. https://ieeexplore.ieee.org/document/10570172

Lai, L., Suda, N., & Chandra, V. (2018). CMSIS-NN: Efficient neural network kernels for Arm Cortex-M CPUs. *arXiv*. https://arxiv.org/abs/1801.06601

Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., & Han, S. (2024). Tiny machine learning: Progress and futures. *arXiv*. https://arxiv.org/abs/2403.19076

Matiko, J. W., Grabham, N. J., Beeby, S. P., & Tudor, M. J. (2014). Review of energy harvesting techniques for wireless sensor networks. *Renewable and Sustainable Energy Reviews, 34*, 225–235.

Pannuto, P., Gummeson, J., Kempke, B., & Dutta, P. (2015). MBus: An ultra-low-power interconnect for next-generation nanodevices. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys 2015)*. https://doi.org/10.1145/2872887.2750376

Rutishauser, P. (2024). *Deterministic graph-based quantization for embedded neural inference*. ETH Zurich. https://doi.org/10.3929/ethz-b-000675547

Sami, Y. (2024). *Zero-trust principles in embedded cyber-physical devices* (Doctoral dissertation). Ontario Tech University.

Smaji, M. (2023). *Toward safe, trustable, and autonomous embedded AI systems* (PhD thesis). Technical University of Denmark.

Verizon. (2024). *2024 data breach investigations report*. https://www.verizon.com/business/resources/reports/dbir/

Zhang, L., Zhuang, L., & Zhao, Y. (2014). Hardware-based secure boot and PUF integration for embedded systems. *IEEE Transactions on Emerging Topics in Computing, 2*(1), 67–75. https://doi.org/10.1109/TETC.2014.2305994