simms

Understanding the Functional Layers of Edge AI

Whitepaper

Abstract

By breaking an Edge AI system into functional layers—spanning hardware, firmware, operating system (if applicable), AI runtime, perception, decision logic, security and cloud/edge management, it becomes easy to appreciate the responsibilities, interfaces and dependencies within the system.

This layered view enables teams to map use-case requirements onto concrete architectural components, identify capability gaps and plan implementation pathways without being overwhelmed by technology choices.

This white paper provides a practical, structured framework to help engineering-sector organisations progress from Edge AI ideation to real-world deployment. We also explain how we, Simms, can support you on your Edge AI journey.

Index

Introduction

All companies operating in the engineering sector are aware of Edge AI. Some are already dabbling with AI deployment and are in pilot mode. However, most are in AI ideation mode, in that they have use cases but no clear route to implementation.

We regard an Edge AI system as comprising eight functional layers, each representing a coherent subsystem that provides services upward and depends on well-defined interfaces downward. Human-system interaction is external and can be thought of as a 'touch point' coming in between layers 7 and 8.

Layer		Main Roles
8	Cloud/Edge Coordination & Management	Handles model and configuration deployment, telemetry ingestion, fleet management, and operator interfaces.
7	Security, Trust & Governance	Enforces cryptographic identity, authentication/authorisation, integrity, audit, policy enforcement, and confidentiality.
6	Decision, Control & Application Logic	Implements mission logic, rules, control loops, UI behaviours and high-level workflows based on perception outputs.
5	Data Pipeline & Perception	Acquires raw sensor data, performs synchronisation, preprocessing, fusion, inference invocation, and post-processing.
4	AI Runtime & Model Infrastructure	Provides accelerated inference, model loading, model integrity checks and hardware-optimised computation.
3*	Operating System & Execution Environment	Manages processes, inter-process communication, memory, networking, and containerisation. Provides secure, deterministic runtime conditions.
2	Firmware & Hardware Abstraction	Abstracts hardware complexity, initialises components, validates firmware integrity, and exposes deterministic APIs upward.
1	Physical (Hardware)	Provides compute, memory, storage, sensing, power, and connectivity. Defines physical constraints and performance ceilings.

* For the purpose of this white paper we shall assume there is an operating system (OS) present. We do though include sections on the ramifications of there being no OS.

Return to Index ^^

Layer 1 - Physical (Hardware)

This is the foundation/ substrate of the Edge AI system. The layer includes a primary compute element, such as a GPU, NPU, CPU or DSP. There might also be a dedicated accelerator, such as an FPGA. Also, sometimes compute elements are paired, e.g. an NPU plus CPU, to make take advantage of their respective strengths.

There is also volatile memory such as DRAM designed for high-throughput inference workloads, and non-volatile memory (such as Flash), present for model storage, data recording and log keeping.

The physical layer of an AI at the edge system also includes sensor interfaces, such as cameras, microphones, inertial sensors, environmental probes and industrial transducers. These capture the real-world environment in which the edge system operates and provide the raw data needed by the higher-level perception functions.

Network interfaces such as Ethernet, Wi-Fi and Bluetooth are also present. They create the physical communication pathways that allow an AI-enabled device (such as a vision-based IIoT device) to be part of a bigger system and for firmware updates to be possible.

Layer 1 also includes power elements such as battery modules, converters, PMICs, and thermal management components, ensuring that the device remains within operational limits despite the computational intensity of AI inference.

Note:
At Simms we believe enclosures, ruggedisation features and environmental protections (IP ratings, shock isolation, EMI shielding) should also be regarded as aspects of layer 1. This is because they perform a function - to protect the system's electronics in harsh environment deployments, for example – so belong in our functional layer model.

Importantly, layer 1 must guarantee predictable electrical, mechanical and thermal behaviour, providing stable, deterministic performance to the layers above. Its constraints—compute ceilings, memory limits and power envelopes, for example—shape the architectural feasibility of all higher-level functionality.

Return to Index ^^

Layer 2 - Firmware & Hardware Abstraction

This layer creates the bridge between raw hardware and the general-purpose software environment. It initialises and configures hardware blocks, buses, clocks and power domains, while enforcing secure boot and firmware validation to establish a trusted execution baseline.

This layer creates the bridge between raw hardware and the general-purpose software environment. It initialises and configures hardware blocks, buses, clocks and power domains, while enforcing secure boot and firmware validation to establish a trusted execution baseline.

At startup, layer 2 authenticates firmware images, configures memory mappings and exposes standardised interfaces to the operating system (OS) - if the system has one, which we shall discuss shortly - thus hiding complex hardware behind stable and trustworthy abstractions.

Layer 2 typically includes boot ROM logic, low-level drivers, and direct memory access (DMA) configurations. *It will also include firmware from level 1: remember we are talking in terms of functionality, not physical hardware.*

Sensor drivers will also be present to ensure deterministic timing constraints are met, i.e. to provide consistent sampling cycles and inference throughput.

This layer also provides foundational diagnostics such as temperature readings, voltage telemetry and hardware health monitoring.

In essence, layer 2's primary role is to guarantee that all hardware components behave predictably and securely, regardless of software complexity. It enables upward portability, while its secure boot and firmware integrity mechanisms ensure downward trust, thus forming a 'root of security' for the entire system stack. A well-designed hardware abstraction layer (HAL) therefore reduces porting effort, and AI runtimes should be able to operate across hardware revisions with minimal modification.

Return to Index ^^

Layer 3 - Operating System & Execution Environment

This layer provides the general-purpose computational substrate on which higher-level services can execute reliably.

Commercially available operating systems (OSs) that might be used include:

- VxWorks - a real-time operating system (RTOS) used in aerospace, automotive and industrial automation.
- QNX Neutrino RTOS - a microkernel real-time operating system popular in automotive applications, medical devices, industrial controllers and energy systems.
- Android - a non-real-time OS used in smart cameras, consumer robotics, handheld industrial devices, and wearable edge-AI systems.

There are also several commercial versions of Linux available (such as Yocto, Ubuntu Core, Wind River), where Linux is by far the most popular choice for Edge AI applications that do not require strict real-time performance. That said, PREEMPT_RT and similar patches can give Linux 'soft' real-time characteristics.

Core functions of layer 3 include process scheduling, task prioritisation, memory management, networking protocols, filesystem operations and device lifecycle control.

This layer arbitrates access to hardware resources exposed through layer 2, offering abstraction via kernel drivers, system calls and inter-process communication mechanisms.

The OS must enforce isolation, manage contention between concurrent AI workloads, and maintain predictable latency for time-sensitive inference paths. It also provides secure communication channels, cryptographic libraries and kernel-level security controls.

From an architectural standpoint, this layer's robustness ensures the system's resilience to faults, resource exhaustion and environmental variability across diverse deployment conditions. It also ensures determinism under dynamic load, supports maintainable software packaging and exposes stable APIs to the AI runtime and data pipeline layers.

No Operating System?

If there is no OS, i.e. the system is 'bare metal', layer 3 does not exist and our architecture will have seven and not eight layers. However, the system still requires an execution environment for interrupt handling, timing primitives, memory initialisation, bare-metal startup code and possibly a cooperative scheduler.

The execution environment responsibilities of what had been layer 3 must be picked up by layer 2 and what will become new layers 3, 4 and 5 (formerly 4, 5 and 6). Layers 7 and 8 become 6 and 7 respectively but have no additional duties.

- Without an OS within the Edge AI system, our new layer 2 must now provide OS-like duties, implemented in firmware. These include:
- Hardware abstraction for peripheral drivers, sensor interfaces/drivers, accelerator control registers and DMA configuration.
- Resource setup, such as memory layout configuration, interrupt vector setup and stack initialisation and boot sequence.
- Base services (if applicable) that might include simple scheduler loops, interrupt handlers, timing utilities and power-state management.

Note:
For the convenience of this white paper, we are differentiating between systems with no OS (bare metal) and those with a full OS. In reality these are the extremes of a spectrum of scenarios. Some systems might have a minimal RTOS, for example.

Return to Index ^^

Layer 4 - AI Runtime & Model Infrastructure

This layer is responsible for executing optimised machine-learning models efficiently and deterministically on often resource-limited edge hardware. It includes inference engines, compilers, runtime schedulers, accelerator interfaces and memory orchestration frameworks capable of exploiting hardware parallelism.

It also manages model loading, caching and version control, ensuring that inference code paths remain consistent despite updates or reconfigurations.

The runtime often performs dynamic optimisations such as tensor pre-allocation, operator fusion, quantised computation, mixed-precision execution and hardware-specific graph compilation. It must coordinate with lower layers to access accelerators, regulate power consumption and ensure predictable latency under variable input rates.

Model infrastructure components validate model signatures, maintain model metadata and expose introspection interfaces for performance monitoring.

This layer ultimately provides a stable inference API to the Data Pipeline Layer. For example, given input tensors, the layer guarantees deterministic outputs within bounded time and memory constraints.

Architecturally, layer 4 forms the system's inference engine, converting trained models into operational capabilities. Its efficiency directly affects battery life, thermal stability, and real-time responsiveness - all of which are fundamental constraints in edge-AI deployments – and its correctness and determinism underpin the integrity of all downstream perception, decision, and application workflows.

No Operating System?

If there is no OS, layer 4 becomes our new layer 3. In addition to everything described above (in this section), the layer must also perform memory allocation, basic task scheduling, timing control, interrupt handling, buffer management and direct hardware invocation.

Return to Index ^^

Layer 5 - Data Pipeline & Perception

We consider 5 the hardest working in our system-level view, and here's why. This layer is most responsible for transforming raw sensor data into actionable semantic representations. It performs data acquisition, synchronisation, temporal buffering and pre-processing tailored to the specific sensing modalities, such as video decoding, resampling, denoising, spectral transforms or feature normalisation.

Layer 5 coordinates signals from heterogeneous sensors and performing sensor fusion (when required) to create coherent intermediate representations. It is also tightly integrated with the AI runtime (layer 4) and manages batching, rate controls and fallback logic when inputs exceed compute capacity.

Post-processing logic refines inference outputs into usable perception artifacts: object tracks, classification labels, segmentation maps, anomaly scores, or event triggers. The layer also encapsulates confidence scoring, temporal filtering and scenario interpretation, producing structured outputs for the decision logic layer.

Because edge systems must operate under bandwidth and compute constraints, this layer performs aggressive data reduction, extracting only the information necessary for local decision-making or occasional upstream transmission.

Architecturally, layer 5's role is to impose order and meaning on high-volume sensor streams. Its correctness determines system sensitivity, false-positive rates and overall situational awareness.

No Operating System?

If there is no OS, layer 5 becomes our new layer 4. In addition to everything described above (in this section), the layer must also directly manage sensor I/O, implement deterministic acquisition loops, handle raw data buffering, perform timing/trigger coordination and orchestrate preprocessing pipelines.

Return to Index ^^

Layer 6 - Decision, Control & Application Logic

This layer governs system behaviour based on perception results and operational policies. It integrates outputs from the perception layer with one, more or all of the following: domain logic, state machines, rule engines and control algorithms. This is done to produce meaningful decisions or actions: which might include threshold-based event triggers, complex decision fusion, safety interlocks or predictive logic derived from system context.

For systems with actuators as well as sensors, this layer will probably be most responsible for the system's closed-loop behaviour.

The layer manages application workflows, integrates with local storage and logs, and interacts with the user interface where appropriate. All decisions must operate under constraints imposed by the security layer.

This layer determines how perception translates to outcomes such as alerts, actuation, UI feedback, local decisions or data transmission to cloud services.

Architecturally, layer 6 is the 'mission logic' of the system, the part that expresses its intended operational purpose. Its correctness affects safety, reliability and user trust. Its implementation must be deterministic, auditable and resilient to noisy inputs. Because this layer directly governs real-world consequences, it often incorporates redundancy, sanity checks and fail-safe behaviours to mitigate perception errors.

No Operating System?

If there is no OS, layer 6 becomes our new layer 5. In addition to everything described above (in this section), the layer must also coordinate control loops, enforce execution order, manage simple state machines, handle error recovery, implement timing-critical behaviours and directly interface with layer 2 (hardware abstraction) for actuation.

Return to Index ^^

Layer 7 - Security, Trust & Governance

This layer enforces the system's integrity and operational trustworthiness, and provides device identity, secure boot*, cryptographic key management, firmware validation and secure update flows. It also enforces access control policies that govern which entities-human or machine-may read data, change configurations and deploy new models.

Note:
secure boot was also mentioned layer 2. For clarification, layer 2 implements the mechanism (root of trust, boot chain), while layer 7 defines and manages the policies, keys and lifecycle that those mechanisms enforce.

Communication between layers relies on security primitives from this layer: *and you'll see in our Layer-to-Layer Dependency diagram how (uniquely) all other layers depend on layer 7*. Security primitives include encryption, certificate-based authentication and policy-driven authorisation. Governance functions include audit logging and operational mode enforcement, as well as relevant regulatory and/or safety standards compliance mechanisms.

This layer's correctness is essential, as a compromised device or model directly undermines the credibility of the entire Edge AI deployment. It must balance strong security guarantees with minimal performance overhead: quite a balancing act given the often-tight compute and memory constraints. It also mediates human interaction, validating operator credentials and policing permissions.

Architecturally, layer 7 forms the system's trust boundary, ensuring that every action executed by the system or its operators is authorised, validated and securely logged.

No Operating System?

If there is no OS, layer 7 becomes our new layer 6, but it does not need to perform any duties above and beyond those described above (in this section).

Return to Index ^^

Layer 8 - Cloud/Edge Coordination & Management

This layer governs the system's lifecycle, configuration and fleet-scale orchestration. It handles model deployment pipelines, distributing updated or optimised models to devices, verifying signatures and coordinating staged rollouts or rollback strategies.

Configuration management is a key responsibility for layer 8, as administrators can, for example, update operating modes via secure management APIs. Also, within this layer, telemetry ingestion pipelines collect logs, performance metrics, inference statistics and health data from edge devices, thus enabling diagnostics and analytics. Indeed, this layer supports remote debugging and automated anomaly detection for operational resilience.

Fleet management functions within layer 8 support discovery, onboarding and the retirement of devices at scale: maintaining consistency even under intermittent connectivity.

Architecturally, layer 8 forms the primary human–system interaction (HMI) surface for operators, abstracting the complexity of distributed edge deployments behind dashboards. The layer's efficiency (governed by design) influences maintainability and upgradeability, making it essential for long-term Edge AI deployments.

No Operating System?

If there is no OS, layer 8 becomes our new layer 7, but it does not need to perform any duties above and beyond those described above (in this section).

Return to Index ^^

Layer-to-Layer Dependency

	L1	L2	L3	L4	L5	L6	L7	L8
L2	P		X	X	X	X	S	X
L2	P	P		S	S	S	S	S
L2	P	P	P		S	S	S	S
L2	P	P	P	P		S	S	S
L2	S	S	P	P	P		P	S
L2	S	P	P	P	P	P		P
L2	X	S	P	P	S	P	P	

Key: L1 = Physical (Hardware), L2 = Firmware & Hardware Abstraction, L3 = Operating System & Execution Environment, L4 = AI Runtime & Model Infrastructure, L5 = Data Pipeline & Perception, L6 = Decision, Control & Application Logic, L7 = Security, Trust & Governance, L8 = Cloud/Edge Coordination & Management.

*The layer named in the left-hand vertical column depends on the layer identified in the horizontal header, with **P** = primary dependency and **S** = secondary dependency. **X** denotes no dependency. Note how security (L7) is the only layer that interacts with all other layers.*

Return to Index ^^

The Human-System Interface (HSI)

This is not a functional layer, but rather it is a cross-layer interface that primarily spans layers 7 and 8. Note, some frameworks you may have seen include HSI within layer 8, but we firmly regard it as a touchpoint, and here's why:

Layer 7: Security, Trust & Governance

The HSI defines what the human is allowed to do. For example:

- Authentication
- Authorisation
- Policy-driven access boundaries
- Audit and compliance

Layer 8: Cloud/Edge Coordination & Management

The HSI defines what the human intends to do. For example:

- Deploy models
- Push updates
- Modify configuration
- Observe telemetry
- Manage the fleet
- Conduct root-cause analysis
- Trigger retraining workflows

Essentially, human operators – such as developers, maintainers, analysts, field engineers and end users - typically interact with cloud dashboards, mobile apps or device UIs that invoke functionality exposed by layer 8, but which are constrained by layer 7.

Return to Index ^^

How Simms Can Help You on Your Ideation-to-Implementation Journey

At Simms, we recognise that the challenge organisations face is not a lack of ideas for Edge AI, but rather a lack of clarity on how to translate those ideas into robust, deployable systems.

Our role is to help you move confidently from concept to implementation by applying a structured, engineering-led approach aligned to the functional layers described in this paper. We will help with:

Clarifying your use case and system requirements

We begin by working with you and your team to define the operational problem, performance constraints and success criteria. This ensures that your Edge AI is applied where it delivers measurable value, rather than as a technology experiment.

Mapping requirements to architecture

Working with our trusted vendors we will assist with translating your use case into a coherent system design. This includes identifying which layers are most critical to your application, where complexity will arise and how responsibilities should be distributed across hardware, software and cloud components.

Selecting and integrating the right technologies

We support the evaluation and selection of compute platforms, sensors, connectivity options, AI runtimes and operating environments. Our focus is on achieving the right balance between performance, power, cost and scalability, rather than defaulting to any single vendor or approach.

Building and validating prototypes

Our vendors and partners can develop proof-of-concept and pilot systems via bundles or systems that validate both technical feasibility and operational fit. This includes integration across layers, from hardware bring-up and firmware through to perception pipelines and application logic, ensuring that your Edge AI system performs reliably under real-world conditions.

Enabling deployment and lifecycle management

Also through the close relationship we enjoy with our trusted vendors, we can help you establish the cloud/edge coordination capabilities required to operate at scale. This includes model deployment pipelines, telemetry, remote management and fleet orchestration, ensuring your solution remains maintainable and adaptable over time.

Enabling scale through distribution and supply chain expertise

Beyond initial deployment, we can support your transition from pilot to production at scale. We leverage our established distribution capabilities, supplier network and logistics expertise to ensure that your Edge AI solution can be manufactured, delivered and supported reliably in volume.

Return to Index ^^

Summary

In this white paper we presented an eight-layer architectural model for Edge AI systems (with operating systems), offering a technology-agnostic framework to help you move from Edge AI ideation to practical implementation.

We described eight functional layers, each representing a coherent subsystem that provides services upward and depends on well-defined interfaces downward. These layers span from physical hardware (compute, memory, sensors, power and connectivity) through firmware, operating systems (if employed), AI runtimes, data pipelines, decision logic, security, and finally cloud/edge coordination.

We also explored the impact of there being no operating system - explaining how layer 2 and new layers 3, 4 and 5 (within a seven-layer system) need to take on additional responsibilities – and we described how the human-system interface is touchpoint between the final and penultimate layers.

In terms of how Simms can accelerate your journey from concept to deployment we do this by grounding every step in the structured, eight-layer framework outlined in this paper. We help you clarify your use case, map requirements onto a viable architecture, and select the right compute, storage, sensing and software components through our close relationships with trusted vendors.

We also support prototype development, validation and optimisation across the full stack, from hardware bring-up and firmware through to perception pipelines, decision logic and cloud/edge coordination.

Finally, we enable real-world deployment at scale through robust supply-chain management, lifecycle support and fleet-level operational tooling, ensuring your Edge AI solution is not only technically sound but commercially sustainable.

Further reading

Whilst convenient to think of an AI Edge system as comprising functional layers the reality is that the layers are interwoven and non-linear - and sometimes collapse or bypass one another. For information on this we recommend you check out our ‘Edge AI Implementation & Integration Challenges’ white paper.

Click Here to Download

Return to Index ^^

About the Authors

This white paper was written by Simms technical specialists.

Return to Index ^^

simms

Get in touch

01622 852800
www.simms.co.uk/intelligent-solutions
sales@simms.co.uk

Simms International
Northdown Close
Northdown Business Park
Ashford Road
Lenham
Kent
ME17 2DL