Lecture 2 — Physical Architecture: Nodes & Gateways

This lecture introduces the fundamental components of Internet of Things (IoT) physical architecture, focusing specifically on the nodes and gateways that form the edge layer of any connected system. We will explore their classifications, inherent design constraints, critical functions, and the essential requirements for secure and reliable deployment.

Learning Objectives

Classify Nodes & Gateways

Identify and classify various IoT nodes and gateways and understand their specific roles within physical architectures.

Hardware & Power Constraints

Explain the limitations imposed by hardware and power, and analyse their consequences for design choices.

Gateway Functions & Security

Describe core gateway functions, architectural patterns, and their crucial security responsibilities at the network edge.

Device Lifecycle & Reliability

Define device lifecycle requirements (e.g., provisioning, OTA) and techniques for ensuring system reliability and physical hardening.

Node Taxonomy: Types & Roles

IoT nodes form the foundation of connected systems, each serving distinct purposes based on their capabilities and constraints. Understanding their taxonomy is crucial for effective system design. These devices are generally resource-constrained, operating at the very edge of the network.

Dev Boards

Microcontroller boards, development kits; characterised by high compute resources relative to deployed nodes, and flexible Input/Output (I/O) for prototyping and advanced applications. Often mains-powered.

Smart Objects

End-user devices such as smart appliances and wearables. They offer integrated services and are typically constrained by battery life and physical form factor.

Actuators

Devices like motors, valves, and relays responsible for physical action. They require real-time control, often consuming moderate power, and demand high reliability.

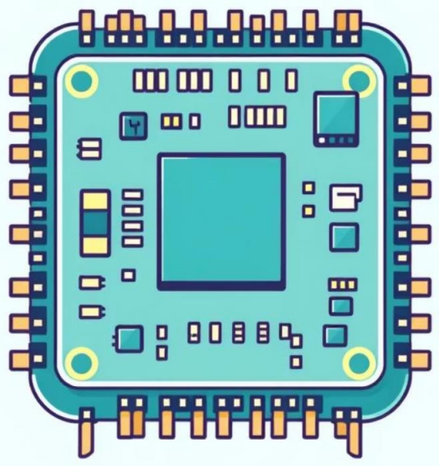
Sensors

Devices gathering environmental data (temperature, motion, light). They are primarily defined by low power consumption, minimal compute capabilities, and event-driven data transmission.

The key differentiator between these nodes is the balance they strike between processing power, connectivity needs, and energy constraints. Choosing the right node type dictates the entire system architecture.

Hardware Constraints: Compute, Memory, I/O

The core challenge in IoT node design stems from inherent resource limitations. Devices must operate efficiently within tight constraints on processing power, memory footprint, and I/O availability.



Resource Limitations

- **Compute:** Typically 8-bit to low-end 32-bit MCUs (e.g., ARM Cortex-M class). They often lack a Memory Management Unit (MMU), simplifying the hardware but requiring more careful software design.
- **Memory:** RAM is measured in kilobytes (KB) to a few megabytes (MB). Flash/ROM storage is similarly limited, enforcing the necessity for exceptionally lightweight software stacks and bootloaders.
- **I/O:** Scarce serial ports, limited Direct Memory Access (DMA) channels, and constrained peripheral sets demand highly efficient hardware abstraction layers (HALs) and device drivers.

Design Implications

Software Optimisation

Prioritise compact Real-Time Operating Systems (RTOS) or bare-metal implementations. Use static memory allocation exclusively and rely on small C libraries. **Avoid** memory-heavy runtimes like JVM or Python interpreters.

Protocol Selection

Select lightweight communication protocols designed for constrained environments. Examples include CoAP (Constrained Application Protocol), MQTT-SN (MQTT for Sensor Networks), and compressed data formats like CBOR/CBOR-Lite to minimise payload size and transmission time.

Robust Profiling

Always profile worst-case memory and stack usage before deploying any code change. Resource exhaustion is a primary cause of unexpected node failure in production environments.

Note: The absence of an MMU means that memory protection is not guaranteed in hardware. A single faulty pointer can crash the entire system, highlighting the need for rigorous code quality.

Power Sources & Power-Aware Design

Power management is perhaps the most critical constraint in IoT, determining a node's operational lifespan and maintenance cycle. Energy efficiency must be factored into every architectural decision.

Power Options for Constrained Nodes

- **Batteries:** Primary (non-rechargeable, e.g., Lithium Thionyl Chloride) for long, infrequent use, or rechargeable (e.g., Li-ion/LiPo) for devices with access to charging or higher duty cycles.
- **Energy Harvesting:** Sustainable, environment-dependent options such as solar photovoltaics, thermal gradient generators (TEGs), vibration kinetic energy, or RF energy harvesting. Often require supercapacitors for short bursts of high power.

Energy Management Techniques

The goal is to minimise active time and maximise sleep time. Key techniques include:

- **Deep Sleep & Power Gating:** Shutting down non-essential components entirely.
- **Duty Cycling:** Periodically waking up, performing a task quickly, and returning to sleep. This is fundamental to extending battery life.
- **Event-Driven Wake-ups:** Utilising interrupts (e.g., motion detection, RTC alarm) rather than polling.
- **Adaptive Sampling:** Adjusting data collection frequency based on system state or measured variance.

> **Crucial Insight:** Wireless transmission (TX) typically consumes orders of magnitude more energy than simple sensing or MCU computation. Therefore, the single most effective power-saving measure is reducing the frequency and duration of radio communication.

Design by Energy Budget

For long-life devices, design must be based on a fixed energy budget (measured in mAh/day or Joules/day). Architectural choices should favour technologies that enable an optimal energy profile, such as offloading heavy processing to the gateway or cloud, and selecting Low-Power Wide-Area Network (LPWAN) technologies like LoRaWAN or NB-IoT for low duty-cycle applications.

Radio Interfaces & Connectivity Tradeoffs (Node Perspective)

Connectivity defines an IoT node's capability. The choice of radio interface is a multi-dimensional optimisation problem, balancing range, data rate, power consumption, and latency.

Short-Range	BLE, Zigbee, Thread. Low power, mesh/star topology, limited range (10-100m).	Personal area networks, smart home devices, local health monitoring.
Local Wi-Fi	ESP32, high bandwidth, high power consumption, IP-based protocol stack, requires dedicated gateway/router.	Mains-powered smart devices, high-throughput data streams, video streaming, factory floor monitoring.
LPWAN / Wide Area	LoRaWAN, NB-IoT, LTE-M. Very long range (1-15km), very low throughput, excellent battery life (years).	Remote sensing, agricultural monitoring, asset tracking, utility metering.

Beyond the protocols, physical layer design is paramount. Hardware considerations include meticulous antenna tuning, proper RF Printed Circuit Board (PCB) layout design to minimise signal loss and interference, and achieving regulatory certifications (e.g., CE, FCC) which can be complex and costly.

Range vs. Power

Longer range usually implies higher transmit power, reducing battery life. LPWAN solves this by using lower data rates and narrower bandwidths to increase link budget.

Throughput vs. Latency

High throughput (Wi-Fi) supports near real-time interaction but uses significant power. Low throughput (LoRa) is suitable for infrequent status updates but results in higher latency.

Cost & Complexity

Proprietary standards may offer performance advantages but increase hardware and licensing costs compared to open standards like BLE or Wi-Fi.

Gateways: Definition & Primary Functions

An IoT gateway acts as a crucial intermediary, bridging constrained IoT devices (using short-range or LPWAN protocols) with broader IP networks and cloud services. It is an intelligent edge component, often running a fully-featured OS and possessing significant compute resources.

> The gateway concentrates device traffic, manages local security, and performs essential edge processing, acting as the operational linchpin between the physical and digital worlds.

Core Functions of an IoT Gateway

Protocol Translation

Converts non-IP field protocols (e.g., Modbus, Zigbee, BLE) into standard internet protocols (e.g., MQTT, HTTPS, AMQP) required for cloud integration.

Data Aggregation & Pre-processing

Collects, batches, filters, and compresses data from multiple nodes before transmission to the cloud. This reduces network overhead and cloud ingestion costs.

Local Edge Processing

Executes business logic, performs time-sensitive analysis, and runs machine learning inference locally, enabling low-latency responses and autonomous operations even during network outages.

Security Enforcement & Proxy

Manages authentication for constrained devices, handles Transport Layer Security (TLS) termination, and enforces network segmentation between the local device network and the public internet.

Resilience & Reliability

Provides local storage and message queuing (store-and-forward) to maintain continuous operation and prevent data loss during interruptions to the backhaul connection.

Gateway Architectures & Software Stacks

Gateway deployment involves selecting the right hardware form factor and designing a robust, layered software stack that supports edge functionality, management, and security.

Hardware and Software Choices

- **Hardware Options:** Range from Single Board Computers (SBCs) like Raspberry Pi for prototyping and non-critical applications, to industrial-grade rugged ARM/x86 gateways designed for harsh environments, or virtualised edge appliances running on existing infrastructure.
- **Software Building Blocks:** Typically based on a secure Linux OS (e.g., Yocto, Debian), leveraging container runtimes (Docker/Containerd) for application isolation. Essential services include an MQTT broker, edge application frameworks (e.g., EdgeX Foundry, AWS Greengrass), and lightweight local databases (SQLite, InfluxDB).

Operational and Design Patterns

- **Resilience Features:** Crucial for maintaining data integrity. Implement local buffering, intelligent retry mechanisms, A/B updates for fault-tolerant software rollouts, constant health monitoring, and dual-WAN or cellular backup for connectivity.
- **Integration Patterns:** Architectural decision between **Thin Gateways** (minimal processing, acting mainly as protocol translators) and **Thick Gateways** (full edge computing capabilities). The choice is driven by latency requirements, payload volume, and cost constraints.

Security Note: Gateways must implement secure boot mechanisms, ensure all firmware is cryptographically signed, and strictly enforce network segmentation to isolate the sensitive IoT network (OT) from the enterprise network (IT).

Device Lifecycle Management (DLM)

Managing IoT devices from manufacturing through to decommissioning is complex due to scale, location, and resource limitations. Robust DLM is essential for security and long-term viability.

1 Provisioning & Onboarding

Secure bootstrapping with pre-provisioned credentials or certificates. Zero-touch enrolment systems are preferred to automate deployment and minimise human interaction.

2 Identity & Authentication

Utilise X.509 certificates and Pre-Shared Keys (PSKs) for device identity. Hardware Secure Elements (e.g., ATECC or TPM modules) are mandated for robust key storage and cryptographic operations.

3 OTA & Firmware Management

Implement secure Over-The-Air (OTA) updates. This includes signing images, using TLS transport for delivery, incorporating A/B partitions for safe rollback, and conducting integrity checks post-installation.

4 Monitoring & Inventory

Maintain a continuous, accurate asset inventory of all deployed devices. Track their firmware versions, battery health, and operational status to ensure ongoing policy compliance and proactive maintenance.

5 Decommissioning

Establish clear and secure procedures for retirement. This involves revoking all device credentials and certificates, securely wiping sensitive data/keys, and managing the ethical disposal or transfer of the device hardware.

Compliance Focus: Every action within the device lifecycle, particularly updates and provisioning, must be logged and cryptographically auditable to meet industry and regulatory requirements.

Reliability, Fault Tolerance & Physical Hardening

IoT systems often operate in inaccessible or harsh environments where failure can lead to significant cost or safety risk. Therefore, design must inherently incorporate resilience against physical and software faults.

Ensuring Uptime and Data Integrity

Redundancy Strategies

Employ physical redundancy (duplicate critical sensors), logical redundancy (sensor fusion algorithms), and network redundancy (gateway clustering, multi-path connectivity) for high availability and consistent data streams.

Hardware Robustness

Select hardware with appropriate Ingress Protection (IP) ratings (e.g., IP67 for dust and water), extended temperature ranges, conformal coatings against moisture, and built-in surge protection against electrical transients.

Runtime Safeguards

Implement embedded mechanisms such as watchdog timers (to automatically reset stalled processes), brown-out detection (to handle low voltage gracefully), and transactional file system writes to prevent data corruption during unexpected power loss.

Communication and Testing

- **Communication Reliability:** Ensure data integrity through the use of checksums, Cyclic Redundancy Checks (CRCs), sequence numbers to detect dropped packets, and acknowledgement/retransmission protocols at the application layer.
- **Testing & Validation:** Conduct Accelerated Lifetime Testing (ALT) to predict wear-out failures. Implement fault injection testing to verify recovery procedures. Collect detailed field failure telemetry to inform future design revisions.
- **Operational Protocols:** Systems must allow for remote diagnostics and logging. Critical actuators require safety interlocks and well-defined, safe failure modes, often including manual override capabilities for human intervention.