PROJECT ABSTRACT

IoT Device Provisioning and Fleet Manager

A fleet-operations console for IoT deployments: zero-touch provisioning wizard (device ID, X.509 fingerprint, single-use claim token), live telemetry dashboard for every device, staged OTA campaigns (canary → 25% → 100%) with failure handling, and a threshold alert feed. Built to order with source, design docs, fleet-config schema and complete viva kit.

01. Title

IoT Device Provisioning and Fleet Manager

02. Introduction / Background

A single IoT prototype is easy; twenty of them in a greenhouse is a different problem. Each device needs unique credentials flashed securely, a way to know it is alive and healthy, firmware updates without a site visit, and alerts when something fails — the unglamorous operations work that decides whether a deployment survives. This project builds the operations console for that job: a provisioning wizard generating per-device identity, a fleet dashboard with live telemetry and sparklines refreshing every two seconds, staged OTA campaigns with per-device progress and automatic failure handling, and an alerts feed collecting offline, low-battery, threshold-breach and stale-firmware events.

03. Problem Statement

IoT coursework usually ends at one working node, skipping the operations layer — secure provisioning, fleet health visibility, staged firmware rollouts and alerting — that real deployments depend on. This project fills that gap with a complete fleet-operations console, demonstrating IoT security, OTA design and full-stack dashboard engineering through a demonstrable 9-device greenhouse pilot scenario.

04. Objectives

- Provision devices in three wizard steps: device info, credential generation, done — producing a unique device ID, 256-bit X.509 fingerprint and single-use claim token (15-minute window) per device.
- Render a live fleet dashboard: per-device temperature, humidity, soil moisture, battery, signal bars, firmware version and telemetry sparklines, updating every 2 seconds.
- Run staged OTA campaigns: canary (2) → 25% → 100%, with signed 128 KiB resumable chunks and per-device states.
- Handle failures automatically: offline devices marked failed instead of blocking; a failure threshold halts the rollout.
- Raise an alerts feed for offline, low-battery, threshold-breach and stale-firmware events with severity and timestamps.
- Export fleet-config.json for flashing each provisioned device.
- Deliver the provisioning/OTA design docs, schema, flashing guide and full viva kit (report, PPT, Q&A;).

05. Proposed Methodology

Provisioning: the operator enters the device name and picks a hardware profile (ESP32-S3, ESP32-C3, Raspberry Pi gateway); the wizard generates the device ID, X.509 fingerprint, MQTT endpoint and single-use token, and exports fleet-config.json for flashing — on first boot the device claims its identity and starts publishing telemetry. Dashboard: the console subscribes to each device's telemetry topic (simulated live in the demo) and renders cards with sensor values, battery, RSSI and sparklines. OTA: campaigns move devices through canary, quarter and full phases with signed, resumable chunks; each device reports download/install/reboot states and failures count against the halt threshold. Threshold rules (battery < 35%, soil < 25%, offline > 30 min, firmware behind) raise alerts automatically.

06. System Architecture / Working

The header shows MQTT/TLS broker connection state, documenting the secure transport design. Fleet health cards summarize devices online, 24-hour uptime, average battery and active alerts. Each device card carries live sensor values, battery, signal strength, firmware version and a canvas sparkline of its history. The OTA panel tracks per-device campaign states (pending, downloading, installing, rebooting, done) with progress bars. The alerts feed timestamps every event with severity. In the demo the fleet and broker are simulated in-browser — zero dependencies, fully demonstrable anywhere; the design docs specify exactly how real ESP32 firmware and an MQTT broker plug in.

07. Hardware / Software Requirements

- Any modern web browser with JavaScript and HTML5 canvas (demo console)
- No server, no libraries — single HTML file, zero dependencies
- For the real-hardware path: ESP32-class devices, an MQTT broker with TLS (port 8883), and a server-side credential vault/HSM (documented as the production requirement)
- Git

08. Expected Outcomes

- A working provisioning flow producing unique, secure per-device identity.
- A live fleet dashboard with telemetry, health summaries and sparklines.
- A staged OTA state machine with per-device progress and automatic halt on failure threshold.
- A timestamped, severity-tagged alerts feed driven by threshold rules.
- Honestly documented demo limits: simulated fleet/broker, client-side credential generation in the demo, design targets at scale.
- A full viva kit: report on fleet concepts and security design, PPT and Q&A; on provisioning, MQTT, TLS, OTA strategies and fleet ops.

09. Applications

- Operating small IoT deployments (greenhouse, campus, lab sensor networks).
- Teaching IoT security, device identity and fleet operations.
- Demonstrating OTA rollout design and failure handling.

10. Future Scope

- Server-side provisioning service with HSM-backed credential issuance.

- Real MQTT broker integration with live device firmware.
- Delta OTA updates and A/B firmware slots with rollback.
- Role-based operator access and audit logging.