

# SYSTEM AND SOFTWARE DESIGN DESCRIPTION (SDD - IEEE 1016 STANDARD)

**Project:** EdgeStream-GW

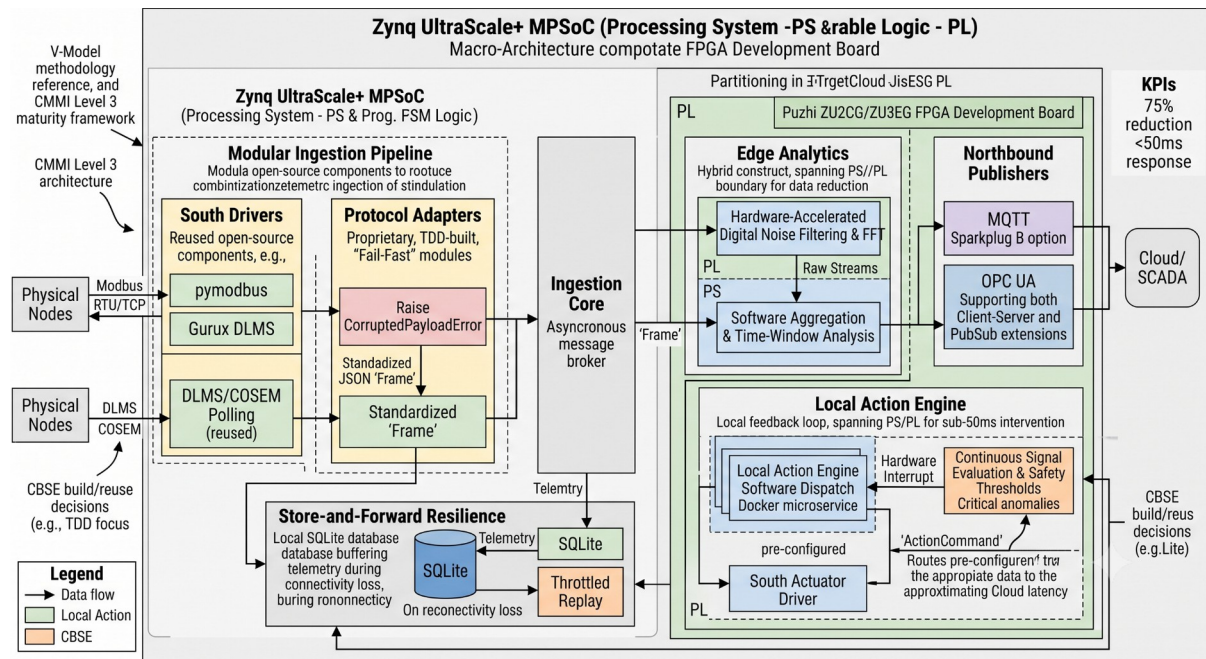
**Developed by:** Helveticore OÜ Estonia

**Target Hardware:** Puzhi ZU2CG / ZU3EG FPGA Development Board (Zynq UltraScale+)

**Maturity Framework:** CMMI Level 3 (Defined)

## 1. System Overview and Strategic Alignment

**EdgeStream-GW** is an industrial IoT Edge Gateway operating as an intelligent, resilient bridge between physical perception nodes (sensors, meters, PLCs) and the Cloud. Positioned in the "hero layer" of a Three-Layer Edge Ecosystem, it intercepts raw time-series data streams, processes them locally, and prevents data bottlenecks in harsh industrial environments.



## Methodological Framework:

To align with a **CMMI Level 3 (Defined)** maturity framework, the software development lifecycle (SDLC) employs a specialized hybrid approach:

- **Macro-Architecture (V-Model):** System validation and verification follow top-down design and bottom-up testing.
- **Micro-Architecture (TDD):** Internal proprietary software components are built using Test-Driven Development (TDD).
- **Component-Based Software Engineering (CBSE):** Integration of ready-to-use open-source components to accelerate development while guaranteeing portability and extensibility.

**Performance KPIs:** Achieve a **75% reduction in data volume** transmitted to the cloud and a sub-50ms local actuation response time.

---

## 2. IEEE 1016 Global Definitions

To ensure strict adherence to design specifications, the following definitions and concerns apply:

### 2.1. System Glossary:

- *Frame*: The standardized internal JSON object model representing parsed telemetry data.
- *ActionCommand*: The strict data structure defining an immediate hardware write command (e.g., Modbus FC 05).
- *Fail-Fast*: A system design property where a component immediately reports a failure and aborts operation upon encountering corrupted data, rather than attempting to silently proceed.
- *CorruptedPayloadError*: A proprietary exception raised strictly by Protocol Adapters during TDD to reject malformed or incomplete hardware payloads.
- *Edge Analytics*: A logical processing construct spanning the PS/PL boundary, responsible for data reduction via hardware-accelerated noise filtering and software-based mathematical aggregation.
- *Northbound Publishers*: The software modules within the Ingestion Core dedicated to packaging and transmitting normalized telemetry to the Cloud or SCADA systems.
- *AXI Bus (Advanced eXtensible Interface)*: The high-bandwidth, low-latency internal hardware bus used for memory-mapped communication between the PS (software) and PL (FPGA fabric).
- *Sparkplug B*: An industrial MQTT specification that standardizes namespace, state management, and payload architecture (using Protobuf) for seamless OT/IT interoperability.

### 2.2. Stakeholders & Concerns:

- **Helveticore OÜ Estonia**: The primary engineering organization responsible for developing EdgeStream-GW as a proprietary, industrialized technological brick.
  - **Distribution System Operators (DSOs) & Smart Grid Managers**: Concerned with grid stability, scalability, and the secure, low-latency integration of distributed renewable assets.
  - **Estonian / Swiss GreenTech Partners**: Concerned with technology maturity, portfolio strategic alignment, and pilot deployment feasibility.
  - **Network Operators**: Concerned with <50ms actuation latency for local safety and equipment protection.
  - **SCADA Integrators**: Concerned with strict OPC UA and MQTT Sparkplug B compliance for seamless IT/OT convergence.
- 

## 3. System-Level Architectural Viewpoints

### 3.1. Containerization and Deployment Viewpoint

The software stack is containerized using **Docker** for "zero-configuration" deployment. By encapsulating microservices (**South Drivers**, **Protocol Adapters**, **Ingestion Core**, the software-aggregation layer of **Edge Analytics**, and the software-dispatch layer of the **Local Action Engine**), vulnerabilities are isolated, and the gateway supports secure over-the-air (OTA) updates.

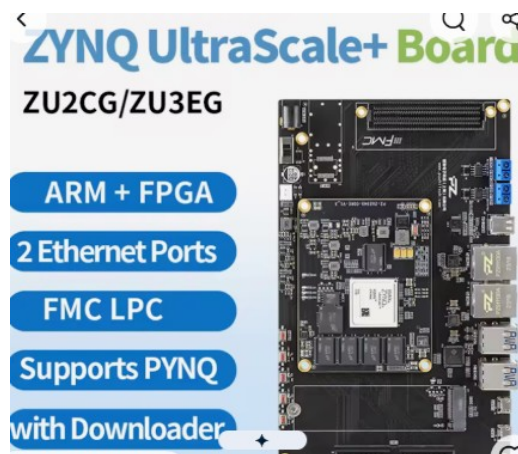
### 3.2. Network Boundaries and Data Routing Viewpoint

- **Southbound (Collection Boundary):** Synchronous, polling-based communication via industrial legacy protocols like **Modbus (RTU & TCP)** and **DLMS/COSEM**.
- **Northbound (Telemetry Boundary):** Telemetry routing relies on asynchronous pub/sub protocols, specifically **MQTT (with optional Sparkplug B)**. It also integrates **OPC UA**, supporting both classic synchronous client-server (request/response) architecture and newer PubSub extensions to satisfy diverse SCADA integrator requirements.

### 3.3. Component Integration and Management Viewpoint (CBSE)

- **Component Qualification (Build vs. Reuse):** The architecture defines strict criteria for adopting open-source libraries (e.g., pymodbus, Eclipse Mosquitto). If an open-source component meets the footprint and latency criteria, it is reused. If specific "Fail-Fast" parsing is required, the component is built from scratch (Proprietary Adapters).
- **Component Adaptation & Composition:** Connection models (Interfaces) ensure external open-source components integrate safely with proprietary TDD code without compromising system stability.

### 3.4. Hardware/Software Partitioning Viewpoint (Puzhi ZU2CG / ZU3EG Target)



- **Processing System (PS - Software):** Manages high-level asynchronous operations (Docker, Python asyncio ingestion, MQTT/OPC UA telemetry).
- **Programmable Logic (PL - Hardware/FPGA):** Reserved for deterministic, hardware-accelerated tasks (digital filtering, fast-Fourier transforms) and ultra-low latency hardware triggers.
- **PS/PL Interfaces:** Communication is brokered via high-speed **AXI buses** and **hardware interrupts**, defining strict contracts for software engineers to mock during TDD.

---

## 4. Modular Ingestion & Analytics Pipeline

This pipeline translates and processes legacy industrial languages into a unified data schema, decoupled into distinct functional layers:

- **South Drivers (Collection):** Manage physical connections and protocol-specific polling (Modbus, DLMS/COSEM). This layer explicitly *reuses* qualified open-source components.
  - **Protocol Adapters (Normalization):** Translates raw binary payloads from the third-party South Drivers into the standardized **Frame** object. Built entirely from scratch using isolated TDD via mocked inputs to enforce strict **"Fail-Fast" behavior**. If a reused driver delivers a corrupted payload, the Adapter instantly raises a **CorruptedPayloadError** and discards the frame.
  - **Edge Analytics (PS/PL Hybrid Construct):** The mathematical processing core fulfilling the 75% volume reduction KPI.
    - *PL (Hardware-Accelerated Filtering):* The FPGA fabric executes raw, high-frequency digital noise filtering and fast-Fourier transforms (FFT) directly on the incoming streams.
    - *PS (Software Aggregation):* The Dockerized Python microservice receives the pre-filtered data and applies time-window aggregations (e.g., RMS, min/max trends, deduplication) on the **Frame** objects.
  - **Ingestion Core & Northbound Publishers:** The central asynchronous message broker that routes the scrubbed **Frame** data to the **Northbound Publishers** (the MQTT clients and OPC UA servers) for transmission to the Cloud.
- 

## 5. Local Action Engine and System Resilience

### 5.1. Local Feedback Loop (Edge Actuation / Fast-Path)

The **Local Action Engine** is a logical construct spanning the PS/PL boundary to guarantee sub-50ms intervention:

- **PL (Detection & Trigger):** The FPGA fabric continuously evaluates raw signal data against strict safety thresholds in real-time. If a critical anomaly is detected, it triggers an immediate hardware interrupt.
- **PS (Dispatch & Logging):** The Dockerized microservice receives the interrupt and instantly routes the pre-configured *ActionCommand* to the appropriate South Actuator Driver, triggering an immediate write command to field actuators in **under 50 ms**, bypassing cloud latency.

### 5.2. Store-and-Forward Resilience

During internet connectivity loss, the gateway buffers outbound telemetry data into local, lightweight solid-state databases (e.g., SQLite). Upon reconnection, the gateway intelligently throttles and replays the cached historical data sequentially.

---

## 6. Design Rationale & Decision Points (IEEE 1016 Decision Log)

### Decision Point 1: Build vs. Reuse for Protocol Collection (South Drivers)

- **Criteria:** Development speed, protocol compliance, hardware resource footprint.
- **Options Considered:** Build proprietary C/Python drivers vs. Reuse open-source libraries (pymodbus, Gurux DLMS).
- **Decision & Rationale: REUSE.** The architecture will utilize qualified open-source libraries for basic I/O and polling. Rebuilding compliant Modbus/DLMS protocol stacks from scratch offers no strategic business value and risks protocol non-compliance.

### Decision Point 2: Build vs. Reuse for Data Normalization (Protocol Adapters)

- **Criteria:** "Fail-Fast" error handling, TDD compliance, strict JSON schema alignment.
- **Options Considered:** Utilize open-source parsers vs. Build proprietary Adapters.
- **Decision & Rationale: BUILD FROM SCRATCH.** Open-source parsers often attempt to silently correct bad data or pass incomplete arrays. Custom Adapters must be built to enforce strict exceptions (e.g., CorruptedPayloadError) to guarantee CMMI Level 3 quality.

#### Decision Point 3: Store-and-Forward Database Selection

- **Criteria:** Lightweight footprint on edge SSDs, power-loss corruption resistance, open-source availability.
- **Options Considered:** Proprietary Binary Files vs. MongoDB vs. SQLite.
- **Decision & Rationale: REUSE SQLite.** MongoDB is too heavy for the PS RAM limits. Proprietary binary files risk data corruption during unexpected industrial power losses. SQLite is an open-source, serverless, ACID-compliant database perfectly suited for embedded edge memory constraints.

#### Decision Point 4: HW/SW Partitioning Boundary (Zynq UltraScale+)

- **Criteria:** <50ms response latency, CPU load optimization.
- **Options Considered:** 100% Software Evaluation vs. PS/PL Split.
- **Decision & Rationale: PS/PL SPLIT.** Routing high-frequency continuous signals through the Linux kernel (PS) introduces non-deterministic OS latency. Delegating noise filtering and initial threshold breaches to the FPGA fabric (PL) mathematically guarantees sub-millisecond detection, leaving the PS solely responsible for routing the asynchronous *ActionCommand* to the South driver.

### 7. Document Control and Traceability

#### 7.1. Requirements Traceability Matrix (RTM)

- *REQ-001a (Hardware Data Reduction - Noise/FFT):* Traced to Section 4 (Edge Analytics - PL).
- *REQ-001b (Software Data Reduction - Time-window Aggregation):* Traced to Section 4 (Edge Analytics - PS). (*Note: Combined, REQ-001a and REQ-001b satisfy the overall  $\geq 75\%$  volume reduction KPI.*)
- *REQ-002 (Offline Resilience):* Traced to Section 5.2 (Store-and-Forward / SQLite).
- *REQ-003 (<50ms Actuation):* Traced to Section 3.4 & 5.1 (Hardware/Software Partitioning & Local Action Engine).

#### 7.2. Revision History

- **v1.0.0:** Initial Architecture Draft (High-Level Framework).
- **v1.1.0:** Integration of Component-Based Software Engineering (CBSE) and CMMI Level 3 alignments.
- **v1.2.0:** Resolution of PS/PL construct mapping for Local Action Engine & Edge Analytics, inclusion of GreenTech stakeholders, restoration of evaluation options, and correction of terminology gaps per formal IEEE 1016 review.
- **v1.2.1:** Microservice enumeration refined in Section 3.1 and RTM split applied for PL/PS Edge Analytics traceability per audit requirements.

#### 7.3. Approvals

- **Prepared By:** Senior Systems Architect, Helveticore OÜ.
- **Approved By:** [Pending Management Review]

