System Context & Boundaries

1. Purpose of This Document

This document defines the **system context and boundaries** for the CAN-based embedded monitoring system. Its purpose is to clearly identify what is considered *inside* the system, what is *outside* the system, and how the system interacts with external entities.

Establishing system boundaries at this stage helps:

- Prevent scope creep
- Clarify responsibilities of each subsystem
- Support correct architectural decisions
- Maintain traceability back to system requirements

This document intentionally remains **high-level** and avoids implementation detail.

---

2. System of Interest (SoI)

The **System of Interest (SoI)** is defined as:

> *An embedded monitoring system that passively receives CAN traffic, processes selected data, stores measurements, and notifies a user when defined conditions are detected.*

The SoI includes both embedded hardware and software elements required to achieve this functionality.

---

3. External Entities

The following entities interact with the system but are **outside the system boundary**:

3.1 External CAN Transmit Software

- Generates CAN frames for transmission onto the CAN bus
- Is responsible for defining CAN identifiers and data content
- Is not controlled or modified by the system

3.2 Physical CAN Bus

- Provides the electrical communication medium
- Includes wiring and termination
- Must comply with standard CAN electrical characteristics

3.3 User

- Configures the system
- Reviews logged data
- Receives notifications or alerts

3.4 External Analysis Tools

- Used to post-process stored measurement data
- Are not part of the real-time system

---

4. System Boundary Definition

The system boundary encloses all components that are designed, implemented, and validated as part of this project.

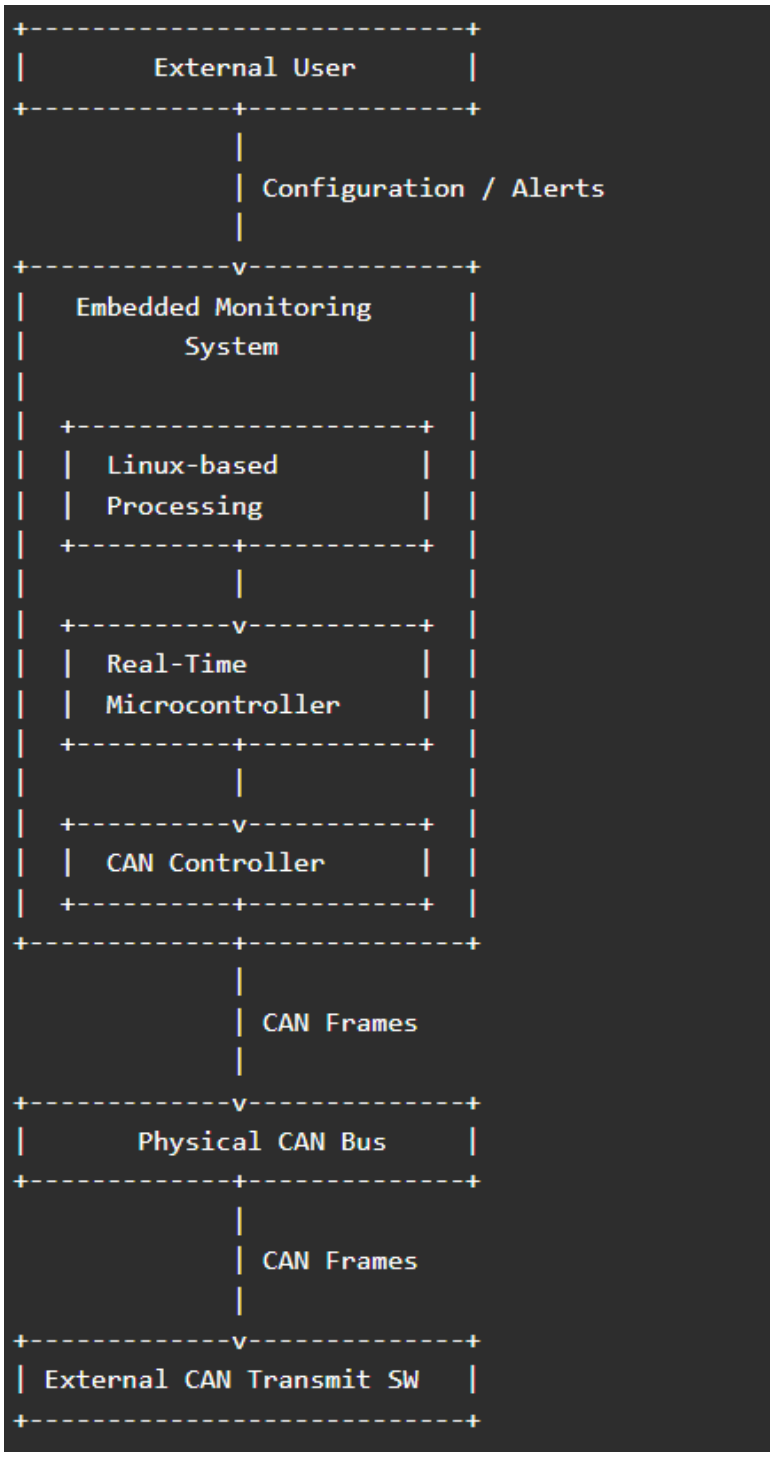
Inside the system boundary:

- Embedded hardware platform
- CAN controller and interface circuitry
- Microcontroller firmware
- Linux-based application software
- Local data storage
- Notification logic

Outside the system boundary:

- CAN frame generation tools
- External CAN nodes
- CAN cabling and termination
- External data analysis environments

5. High-Level Context Diagram (Textual Representation)



This diagram is conceptual and will be refined into graphical form in later documents.

6. Interfaces

At this level, interfaces are defined only in functional terms:

- **CAN Interface**: Receives raw CAN frames from the physical bus
- **Data Transfer Interface**: Transfers CAN data between real-time and non-real-time processing domains
- **User Interface**: Allows configuration and notification
- **Data Storage Interface**: Persists measurement data for offline analysis

Detailed interface specifications are intentionally deferred.

---

7. Relationship to System Requirements

This context definition supports the system requirements by:

- Identifying where CAN reception (FR-01 to FR-03) occurs
- Clarifying the separation between real-time and Linux-based processing (FR-04, NFR-03, NFR-04)
- Establishing boundaries for data storage and notification (FR-08, FR-09)

Formal allocation of requirements to subsystems will be introduced in the next architectural document.

---

8. What This Document Does Not Define

This document deliberately does **not** define:

- Hardware pin assignments
- Communication protocols between processors
- CAN message or signal definitions
- Monitoring algorithms or thresholds

These topics belong to later design stages.

9. Next Steps

With system boundaries clearly defined, the next step is to describe the **high-level system architecture**, including major functional blocks and their responsibilities.