System Requirements Definition

1. Purpose of This Document

This document defines the **system-level requirements** for the CAN-based embedded monitoring system described in PDF 0.1. The goal is to clearly state *what the system must do* and *under what constraints*, before any detailed hardware wiring or software implementation is performed.

Defining requirements early helps: - Guide architectural decisions - Avoid premature design choices - Enable traceability and validation later in the project

At this stage, requirements are intentionally **technology-agnostic** and **signal-agnostic**.

---

2. Requirements Scope

The requirements in this document apply to the complete system, including: - External CAN communication - Embedded hardware - Microcontroller firmware - Linux-based application software - Data storage and notification mechanisms

Detailed signal definitions, algorithms, and tuning parameters are explicitly **out of scope** for this document and will be introduced later during system design.

---

3. Stakeholders and Use Perspective

Primary stakeholders include: - A developer or engineer building the system - A user interested in monitoring system health - A learner seeking to understand system-level embedded design

From a usage perspective, the system acts as a **passive CAN listener** that observes traffic, processes selected data, and reports meaningful events.

---

4. Functional Requirements (FR)

CAN Communication

- **FR-01**: The system shall receive CAN frames transmitted by an external CAN transmit software tool.
- **FR-02**: The system shall support standard CAN identifiers and data payloads.
- **FR-03**: The system shall timestamp received CAN frames.

Data Transfer and Processing

- **FR-04**: The system shall transfer CAN frame data from a real-time microcontroller to a Linux-based processing environment.
- **FR-05**: The system shall decode CAN frames into signals using a user-defined DBC file.

Monitoring and Analysis

- **FR-06**: The system shall support monitoring of one or more CAN-derived signals representing system health.
- **FR-07**: The system shall support algorithmic processing of decoded signals to detect abnormal conditions.

Data Storage and Reporting

- **FR-08**: The system shall store measured data in an industry-standard file format suitable for offline analysis.
- **FR-09**: The system shall notify a user when a defined event or abnormal condition is detected.

---

5. Non-Functional Requirements (NFR)

Performance and Reliability

- **NFR-01**: The system shall not intentionally modify CAN traffic on the bus.
- **NFR-02**: The system shall minimize loss of received CAN frames under normal operating conditions.
- **NFR-03**: Time-critical CAN reception shall be handled by a real-time execution context.

Maintainability and Modularity

- **NFR-04**: The system architecture shall separate real-time tasks from non-real-time tasks.
- **NFR-05**: Monitoring algorithms shall be updatable without requiring hardware changes.

Usability and Learning

- **NFR-06**: The system shall be understandable by beginners with basic embedded knowledge.
- **NFR-07**: The system shall be developed and documented incrementally.

---

6. Constraints

- **C-01**: The system shall use the Arduino UNO Q development board.
- **C-02**: CAN communication shall be implemented using an external CAN controller connected via SPI.
- **C-03**: The system shall use a dual-processor architecture consisting of a microcontroller and a Linux-capable processor.
- **C-04**: All public documentation shall avoid vendor-specific CAN tool references.

---

7. Assumptions

- The CAN bus is correctly terminated and operates within standard electrical specifications.
- The external CAN transmit software generates valid CAN frames.
- The system operates as a monitoring node and does not require real-time control of actuators.

---

8. Requirements Traceability (Initial)

At this stage, requirements are defined but not yet allocated to specific hardware or software components. Allocation and traceability will be introduced once the system architecture is defined in later documents.

---

9. Next Steps

With system requirements defined, the next step is to describe the **system context and high-level architecture**, identifying system boundaries and major functional blocks.