Project Overview & Motivation

1. Introduction

Modern embedded systems rarely exist in isolation. In domains such as automotive, industrial automation, and smart infrastructure, devices must reliably interact with complex communication networks, process large volumes of data, and support higher-level analytics such as diagnostics or predictive maintenance.

This project demonstrates, step by step, how to design and build a **CAN-based embedded monitoring system** using a modern dual-processor development board. The focus is not only on making the system work, but on **documenting the engineering thought process** behind it, from requirements definition through to validation.

The project is intentionally structured as a learning journey, suitable for beginners, while still reflecting **real-world engineering practices** used in professional environments.

---

2. Problem Statement

CAN (Controller Area Network) remains one of the most widely used communication protocols in embedded and automotive systems. While many tutorials show how to read CAN frames on a microcontroller, fewer examples demonstrate how to:

- Integrate CAN data into a larger system architecture
- Separate real-time and non-real-time responsibilities correctly
- Decode CAN signals using standardized descriptions (DBC)
- Store data in industry-standard measurement formats
- Perform basic condition monitoring or predictive maintenance

This project addresses these gaps by building a complete, end-to-end system rather than a single isolated sketch or script.

---

3. Project Goals

The primary goals of this project are:

1. To design a system capable of receiving CAN frames from an external CAN transmit software tool.
2. To decode CAN signals using a user-defined DBC file.
3. To process CAN-derived data for simple anomaly detection and predictive maintenance use cases.
4. To store measured data in an industry-standard file format for later analysis.
5. To notify a user when defined events or abnormal conditions occur.
6. To document each design decision and development step clearly and progressively.

Equally important is the **learning objective**: to show *how* an engineer thinks when approaching such a system, not just *what code to write*.

---

4. Target Audience

This project is intended for:

- Engineers and students new to CAN-based systems
- Embedded developers transitioning toward system-level design
- Test and validation engineers interested in data-driven monitoring
- Anyone wishing to understand how embedded devices integrate with Linux-based applications

No prior experience with predictive maintenance or edge AI is assumed. Concepts are introduced gradually and explained in plain language.

---

5. High-Level System Concept

At a high level, the system consists of:

- An **external CAN transmit software tool** running on a PC
- A **physical CAN bus**
- An **embedded system** that:
- Receives raw CAN frames
- Transfers data between a real-time microcontroller and a Linux-based processor
- Decodes and stores CAN signals
- Performs basic monitoring and alerting

The embedded system intentionally uses a dual-processor architecture to demonstrate how real-time constraints and high-level processing are typically separated in professional designs.

---

6. Scope and Non-Scope

In Scope

- Classical CAN communication
- CAN frame reception and validation
- DBC-based signal decoding
- Simple predictive maintenance techniques
- Linux-based data storage and notification
- System architecture documentation

Out of Scope

- Vehicle-level functional safety (ISO 26262)
- Complex machine learning models
- High-speed CAN FD optimization
- Cloud-based analytics platforms

Keeping the scope controlled ensures the project remains understandable and achievable for beginners.

---

7. Why This Project Matters

From a professional perspective, this project demonstrates:

- Systems engineering thinking
- Clear separation of concerns
- Awareness of industry standards
- Validation and traceability mindset
- The ability to communicate complex ideas clearly

Rather than focusing on a single technology, the project highlights how **multiple disciplines come together** in real embedded systems.

---

8. What Comes Next

The next step in this series is to formally define the **system requirements**, as a systems engineer would, before any hardware is wired or any code is written.

This ensures that all later design decisions can be traced back to clearly stated goals and constraints.