Assignment #1

Qualidade e Confiabilidade de Software

Afonso Melo, Francisco Rosendo

2020217360, 2020217697

Department of Informatics Engineering, University of Coimbra

April 6, 2025

1 Introduction

Today's cars rely heavily on software to handle everything from safety systems to comfort features. With millions of lines of code running across many interconnected components, even small faults can lead to unexpected behaviour or even real safety risks. That's why fault tolerance is so important in automotive systems: even if something goes wrong, the system should still work as expected or at least fail gracefully.

In this project, we focused on implementing and improving two specific functional requirements, adding later fault tolerance mechanisms to make each system more robust against likely faults, depending on how critical each one is.

This report explains the faults we considered, the techniques we applied to handle them, and why we made those choices. It also discusses other techniques we looked at but didn't use, along with a reflection on the trade-offs involved. The aim was not just to make the code work but to make it more dependable — which is a key goal in any real-world automotive software.

2 Functional Requirements Specification

2.1 FR3 - Tyre Pressure Warning

The Tyre Pressure Warning system is responsible for detecting tyres that have pressure levels below a safe threshold. It receives recent pressure readings for each tyre and determines whether any tyre should trigger a warning. This functionality is safety-related, as underinflated tyres can lead to increased wear, fuel inefficiency, or even blowouts. Therefore, some level of fault tolerance is necessary.

2.2 FR6 - Climate Control

The Climate Control system is responsible for adjusting the duty cycle of the cooling fans in different zones of the vehicle to maintain a user-specified temperature range. Each zone (e.g., front seats, rear seats) has its own temperature sensor and associated fans.

The system checks if the current temperature in a zone is outside the acceptable error margin from the desired temperature:

- If it's too hot, the fan duty cycles should be decreased.
- If it's too cold, the duty cycles should be increased.
- If the temperature is already within the desired range, no change is made.

Note: Although the spec says fans should be turned down when it's too cold and up when it's too hot, the test cases do the opposite — increasing fan speed when it's too cold. We followed the behavior shown in the test cases.

3 Fault Tolerance Mechanisms

3.1 FR3 - Tyre Pressure Warning

To improve the reliability of this system, several fault tolerance mechanisms were applied in our implementation, as explained below.

3.1.1 Input Validation and Filtering

The first mechanism implemented is basic input validation. Any pressure readings that are zero (typically indicating a sensor malfunction), negative, or unrealistically high (e.g., above 100 PSI) are filtered out. This ensures that sensor glitches or corrupted data do not affect the decision-making logic. If a tyre has no valid readings after filtering, it is skipped entirely, and a warning is logged. This prevents invalid data from leading to false warnings or missed alerts.

3.1.2 Outlier Detection

Outlier detection is used to enhance the quality of the data used to compute each tyre's average pressure. The mechanism compares each reading to the average of the remaining readings. If a value deviates significantly (based on a configurable threshold ratio), it is classified as an outlier and excluded. For example, if a single reading is far lower than the rest due to a temporary sensor glitch, it will not skew the average, ensuring the system remains stable. This adds robustness against transient faults.

3.1.3 Dual Modular Redundancy (DMR)

We simulate DMR by performing the pressure average computation twice and comparing the results. While in hardware this would be done with physically duplicated systems, in our software simulation, we duplicate the computation and compare the results for consistency. If the two computations diverge beyond a small threshold (e.g., 0.001), it indicates a potential transient error, and the system skips the tyre to avoid false results. DMR allows us to detect internal inconsistencies without requiring complex infrastructure.

3.1.4 N-Version Programming (NVP)

To further increase reliability, we implemented N-Version Programming by running two independent implementations of the tyre pressure warning logic: our own fault-tolerant version, and a baseline version developed by another team (group 12). Both versions are fed the same input, and their outputs are compared by a separate script. If there is a discrepancy between the two outputs, a warning is raised, and the code re-runs up to 3 times if the warning is still present. This helps to detect design faults that may be present in one version but not the other. Even if both implementations are imperfect, having them built independently reduces the chance of shared faults. Because we are using another groups code, without fault-tolerance, sometimes the outputs are different and the warning is raised, because of mechanisms like **Outlier Detection** that are present in our fault-tolerant version.

3.1.5 Considered but Not Implemented Mechanisms

- **Triple Modular Redundancy (TMR):** While TMR offers the ability to mask errors using majority voting, it requires three independent computations, which is computationally more expensive and redundant for this task. Since our DMR + NVP strategy already covers both internal and external consistency, TMR was not deemed necessary.
- **Time Redundancy:** Recomputing the same logic over time to catch transient faults was considered, but the low fault rate and the stateless nature of the function made this less effective. Instead, filtering and redundancy at the computation level were more practical.
- **Recovery Blocks and Rollback:** This mechanism was not applicable, as our function is stateless and does not maintain an internal execution state that can be checkpointed or rolled back.

3.2 FR6 - Climate Control

In order to ensure that the climate control system delivers a correct service even in the presence of faults, several fault tolerance techniques were applied to the FR6 implementation. Although climate control is primarily a comfort feature rather than a safety-critical system, improving its reliability enhances overall user experience and system robustness. The key fault tolerance mechanisms implemented in our solution are described below.

3.2.1 Consistency Checks

The first line of defense in our implementation is the use of consistency checks. These are implemented using assertions that validate all sensor inputs and fan duty cycle values. For example, temperature readings are checked to ensure they are numeric and fall within a plausible range (–40°C to 60°C), while fan duty cycles are verified to be within the range of 0% to 100%. This mechanism provides early error detection and prevents the propagation of erroneous data through the control algorithm. The low overhead and simplicity of consistency checks make them an ideal choice for a climate control system, ensuring that the system always operates on valid data.

3.2.2 Triple Modular Redundancy, TMR

To further improve fault tolerance, we replicated the core computation of the FR6 function using Triple Modular Redundancy (TMR). In this approach, the primary function—responsible for adjusting the fan duty cycles based on the current temperature and target range—is executed three independent times. The outputs from these three replicas are then compared using a majority voting mechanism. For each zone and for each fan index, if at least two replicas agree on a value, that value is adopted as the final output. This method effectively masks transient errors; even if one replica produces an incorrect result due to a temporary fault, the majority decision will yield the correct value. In the context of climate control, where transient sensor glitches or computation anomalies might occur sporadically, TMR is particularly beneficial despite the additional processing overhead. It enhances the reliability of the system without requiring complex hardware redundancy.

3.2.3 Outlier Detection

Given that climate control relies heavily on sensor data, ensuring the quality of these inputs is crucial. We integrated an outlier detection mechanism that processes temperature readings from each zone. In scenarios where multiple sensor readings are available for a zone, the mechanism calculates the median value and then filters out readings that deviate significantly (beyond a predetermined threshold) from the median. This process effectively eliminates spurious data that could otherwise trigger inappropriate fan adjustments. Outlier detection is implemented with minimal computational cost and significantly increases the robustness of the sensor data, ensuring that only consistent and reliable values are used in the control logic.

3.2.4 Considered but Not Implemented Mechanisms

In addition to the techniques implemented above, other fault tolerance methods were evaluated:

- **Dual Modular Redundancy (DMR):** DMR duplicates the computation and compares the outputs; however, it lacks a majority voting mechanism to resolve disagreements. Since DMR does not provide a clear method to choose the correct output when discrepancies occur, it was deemed less effective for our comfort-oriented climate control system.
- **Time Redundancy and Recovery Blocks with Checkpointing:** Although time redundancy (re-executing computations sequentially) and recovery blocks (which roll back to a known-good state in case of failure) can be beneficial in safety-critical systems, their added processing overhead and complexity were not justified in the context of a climate control system. The transient fault rate in such comfort systems is generally low, and simpler mechanisms like consistency checks and TMR provide sufficient fault masking.

4 Conclusion

In this assignment, we designed, implemented, and improved two automotive subsystems — Tyre Pressure Warning and Climate Control — by integrating a variety of fault tolerance mechanisms tailored to the criticality of each system. Our main goal was to ensure that even in the presence of faults, these systems would behave reliably, either by correcting or safely handling the fault.

For FR3, which is more safety-critical, we employed techniques such as input validation, outlier detection, Dual Modular Redundancy (DMR), and N-Version Programming (NVP). These mechanisms helped ensure that sensor malfunctions, transient faults, and design errors were either caught or mitigated. The combination of internal checks (DMR) and external comparison (NVP) gave us confidence in the robustness of our implementation.

For FR6, we focused on improving the resilience of a comfort-focused feature by applying lighter-weight techniques such as consistency checks and outlier filtering, while also experimenting with Triple Modular Redundancy (TMR) to simulate stronger guarantees in the presence of sensor anomalies.

Throughout this process, we carefully evaluated multiple fault tolerance strategies, implementing those that provided the best trade-off between robustness, complexity, and performance. We also documented mechanisms that we opted not to implement, providing justification based on the context and goals of the system.

Overall, this assignment strengthened our understanding of fault tolerance principles in software design and gave us practical experience in applying these concepts to real-world scenarios, especially in domains where correctness and reliability are critical.