Riassunto della Tesi di Laurea Magistrale in Ingegneria Informatica (Sessione Ottobre 2024)

“Library definition for an automotive ECU API layer using Model Based approach - Memory Management & OBD”

Candidato: Mattia Oliva

Relatore: Massimo Violante Tutor Aziendale: Emilio Bertrand

Summary:

Introduction

This thesis addresses the redesign of the On-Board Diagnostic (OBD) system and the management of memory within the API layer of an automotive ECU using a model-based approach. The project is a result of the collaboration between Polytechnic of Turin and the Metatron Research Centre, a company specialising in the development of electronic and mechanical components for alternative fuel vehicles.

In recent years, the automotive industry has experienced a rapid evolution in standards and regulations. To remain compliant, companies such as Metatron have had to develop extensive and intricate codebases. The proliferation of these standards, coupled with differing interpretations and diverse customer requirements, has resulted in a convoluted and fragmented software architecture, making it difficult to manage and adapt.

This thesis aims to redesign the OBD strategy while adhering to the latest standards, reduce complexity, and improve maintainability, in addition to optimising the non-volatile memory (NVRAM) management for better performance and robustness, to lay the foundation for future projects.

It tackles these challenges starting with a thorough study of the existing ECU API layer and Model-based Software Level (MSBL) libraries, adopted strategies, and international guidelines requirements. The work then moves to the design and implementation of model-based (SIMULINK) libraries and support functions according to an automatic/user-friendly process to support diagnostic protocols management, first with the definition of a robust NVRAM handling strategy and then of a standard-compliant OBD strategy. Both memory and diagnostic solutions are then tested with appositely made demo applications.

Development

The initial phase of this project focused on setting the groundwork for the system by developing and implementing a new NVRAM management strategy. It started with examining and redesigning the current solution to strengthen the system's robustness, creating methods for verifying the integrity of stored data, performing and recovering backups, and, of course, performing regular R/W operations. Every choice was made trying to keep the approach as versatile as possible, given that the structure of the OBD memory was still unknown at the time. This section of the thesis also involved the development of several testing programs in LabVIEW to automate and carry out stress-intensive testing on the implemented solution.

The second, and main, part of this thesis was the redesign of the On-Board Diagnostic (OBD) system, which involved several crucial steps. Initially, a comprehensive analysis of the existing Engine Control Unit (ECU) API layers and the requirements of the state-of-the-art standards for heavy-duty and passenger systems (WWH-OBD, J1939, EURO VI, CHINA VI, etc.) was conducted. From the information gathered, a precise diagnosis flow (*Figure 1*) was designed, consisting of several parts: fault detection, fault validation, reaction to the fault, storage of the fault (error memory), and management of the malfunction indicator lamp (MIL). The fault detection part is outside the scope of this thesis work, as it strongly depends on the board’s

hardware and base-level software (sensors) or is up to the final manufacturer that uses Metatron's system (fault check test routine).

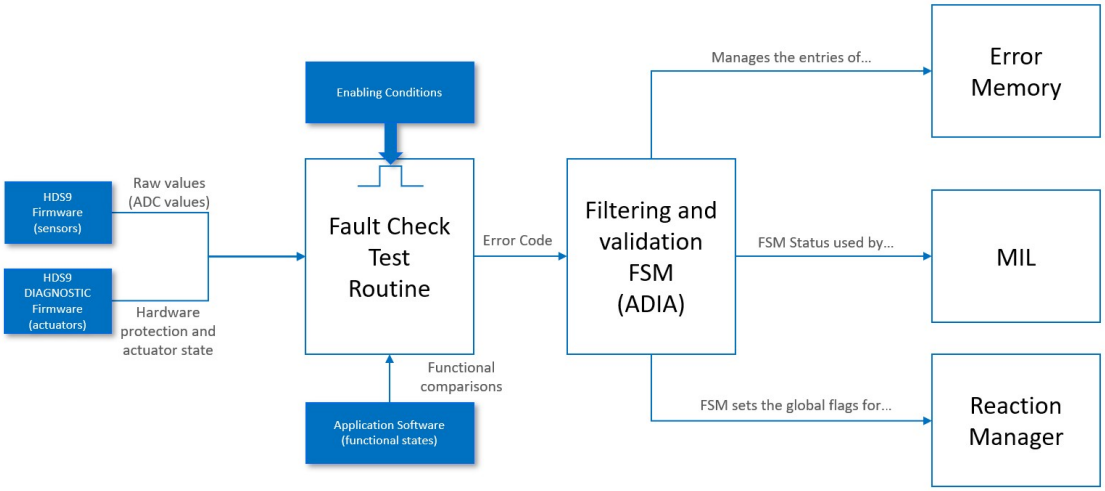

Figure 1 - Diagnosis flow

The main component of the diagnostic flow is the filtering and validation finite state machine (FSM), called ADIA (from Automatic DIAgnostic). For each fault defined in the system, an automaton cycles through its states to validate (or de-validate) the error conditions identified by the dedicated, user-defined fault check test routine. The ADIA is the core of the diagnostic system, as it's in charge of triggering the faults' memory management strategies and preparing the field for the recovery strategies. The original code defined multiple types of ADIAs that have been now condensed into a simpler, standard-compliant single type of FSM (*Figure 2*). All the necessary functions to make the strategy work have been implemented in C language code, making available at model-based level (Simulink) the minimum number of APIs for the customers to use. The original process to define the faults associated with the ADIAs and the parameters of the FSMs has been simplified thanks to the introduction of Simulink mask blocks to facilitate the setup of the faults.

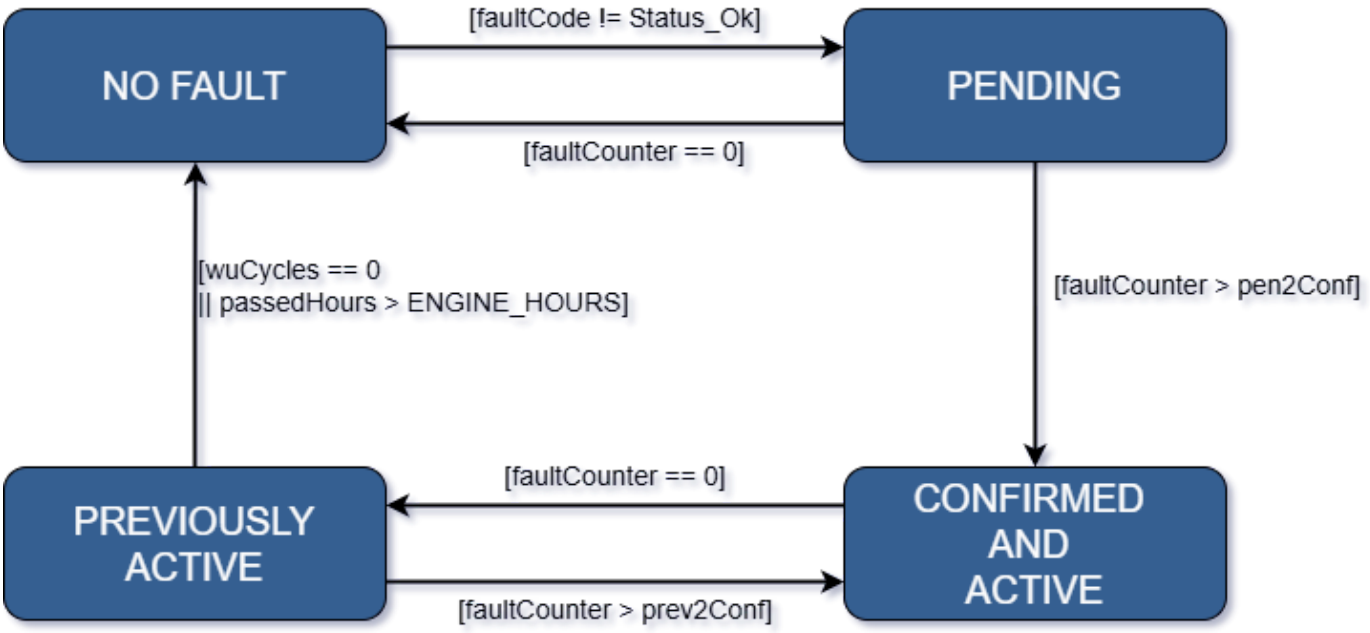

Figure 2 - New FSM scheme, here extremely simplified to protect the actual know-how

Once the ADIA confirms the fault, the associated recovery lines must be activated. The actual implementation of the recovery function is handled by the users, as it depends solely on how the faults have been defined. The strategy for the correct activation, deactivation, and management of the recovery lines is instead part of this thesis.

The ADIA is also partially responsible for the management of the MIL, updating certain required information, like the number of active faults of certain classes. The behaviour of the lamp is strictly defined by the studied regulations, and it has been implemented with the use of different FSMs that cover the totality of possible factors and their combinations to determine the MIL activation patterns.

Another fundamental component of the flow is the management of the error memory, used to store information related to the active or previously active faults and additional data that could be used by an external operator to better discern and repair the causes of the issues.

Strictly associated with the error memory is the management of the *Freeze Frames*, snapshots of the engine status taken when a fault is inserted in the error memory. By integrating freeze frame data with the diagnostic error memory, the system is better equipped to improve the accuracy of fault identification and resolution, resulting in a more efficient and reliable diagnostic process. Once the set of required and optional fields defined by the guidelines had been merged into a final batch, an additional Simulink mask block was designed to facilitate the setup of the system.

The error memory and the freeze frame are the components that most benefit from the NVRAM work, although the entirety of the system rests on the memory strategy.

Each of the diagnosis flow parts was implemented with low-level code, and APIs (available as Simulink blocks) have been designed and integrated with Metatron's library for correctly interfacing with the system following a model-based approach and providing the OBD information as per guidelines requirements.

Every implementation step was accompanied by a testing phase. To ensure the redesigned OBD system performs reliably, Hardware-in-the-Loop (HIL) testing was used to simulate real-world conditions by combining actual hardware with software models, creating a controlled environment to test the system's resilience and memory management strategies under stress. Additionally, a demo application has been designed to test the implemented solution as a normal user would have, and part of the system has also been integrated on a pre-existing Metatron project.

Conclusion

This project has successfully simplified the On-Board Diagnostic (OBD) system architecture, enhancing the efficiency of fault management. Key achievements include increased modularity, reduced error points, and minimised setup times, all while ensuring full compliance with the latest diagnostic standards. The next steps involve integrating the redesigned protocols with the Basic Software Layer (BSWL) and selecting a pilot project to fully evaluate the system's functionalities in a real-world environment. This phase will provide valuable insights and allow for further refinements, ultimately contributing to the development of an even more solid base for future projects.