ISSN 1232-3829

tts 10/2013

Technika Transportu Szynowego koleje ▪ tramwaje ▪ metro

CZEJDO Bogdan D., ZALEWSKI Janusz, TRAWCZYNKI Dawid, BASZUN Mikolaj

DESIGNING SAFETY CRITICAL EMBEDDED SYSTEMS WITH TIME-TRIGGERED ARCHITECTURE

Abstract

The paper presents results of the analysis of safety-critical embedded systems using a time triggered-architecture. First, a distributed safety-critical embedded system is defined in terms of its interfaces with the physical world, and possibilities of failures that can cause safety problems. Then, a model is built that allows mapping the safety functions to the time-triggered architecture. Finally, based on this model, a case study of an anti-lock braking system is developed and analyzed with respect failures that can lead to violations of system safety. The results show that time-triggered architecture can lead to meaningful results in the analysis of safety issues in distributed real-time embedded systems.

INTRODUCTION

Safety of embedded computer systems is of ever-growing importance, because of the continuously increasing functionality and resulting complexity of computer control applications. Safety aspects are important in all embedded systems but are critical in such applications as transportation, where the failure of hardware or software can result in the loss of life, limbs, or large financial losses. The technologies for providing safety assurance for embedded system are therefore of primary concern [1, 2].

The traditional technologies for safety assurance in embedded control systems are typically based on the architecture of feedback control in a single loop (Fig. 1) and they rarely apply results from continuous systems analysis, focusing on discrete models for safety analysis [3], using, for example, finite state machines, queuing theory, Petri nets, including artificial intelligence techniques, such as rule-based reasoning, Bayesian belief networks, even rough sets. The continuous system analysis is much harder to carry out and manage for complex embedded system. There are, however, many efforts to use extended models to define functionality of the embedded system, perform hazard and risk analysis using continuous system approach, systematically transform models into more fault resilient, and verify the fault resilient behavior of transformed models using again continuous systems analysis. Such models can be then used for software development of distributed embedded computer control systems.

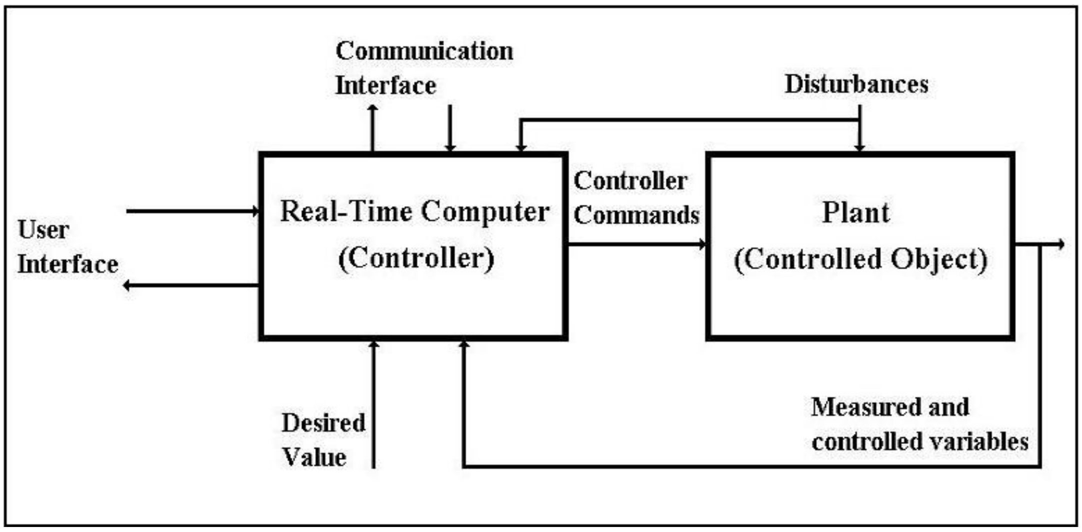

Communication Interface
Disturbances
Real-Time Computer (Controller)
Controller Commands
Plant (Controlled Object)
User Interface
Desired Value
Measured and controlled variables

Fig. 1. General Model of an Embedded Computer Controlled System.

One of the specific technologies for safety assurance of distributed real-time embedded systems, to determine and improve their safety that gained attention in recent years is the time-triggered architecture [4]. A real-time computer system is a computer system in which both the validity of computations and their compliance with specific timing constraints are considered. In some of these systems, computations must be performed within strict deadlines to avoid possibly catastrophic or costly consequences. For such systems, the failure to meet a deadline is considered a failure of the system, and these systems are called hard real-time systems (as opposed to soft real-time systems, where failures are not critical with respect to safety) [4]. A distributed real-time system is a system with computational processes distributed amongst multiple components, each of which is essentially a different computer.

There are presently two major types of technological approaches to design of distributed real-time embedded systems. They are the event-triggered systems and the time-triggered systems. In an event-triggered system, components perform computations in response to various events (environmental and otherwise) and communicate by passing the results of these computations as messages on the communication medium, usually a bus. When multiple events take place, some of them must wait to be handled until preceeding and/or higher priority events have been handled and their results communicated. In periods of high activity, it is possible that some events may never be handled, because of the system overload. This can be problematic in hard real-time systems, in which the delay of a message may render it useless and cause system failure, which could be critical to safety.

This communication issue is avoided by the use of a time-triggered architecture. In a time-triggered architecture, the only events that trigger component behavior are based on time. A global communications schedule is created prior to run-time and completely determines the order of each component's communication, as components may only send messages during the time assigned to them by the schedule. As a consequence a time-triggered system is deterministic, because the timing of events is known in advance. That is, given the current state of a time-triggered system and all future inputs, it is possible to predict any future state of the system [4].

In this project we concentrated on studying distributed real-time embedded systems to address and determine their safety by introducing fault tolerance. Our emphasis was on distributed real-time embedded systems with redundant nodes to investigate how redundancy and fault tolerance can improve safety. The time-triggered architecture was chosen for a couple of reasons. First, such architecture allows avoiding communication issues described above and, secondly, allows for a much more thorough analysis of the behavior of distributed real-time systems by using continuous models.

The rest of the paper is structured as follows. Section 1 outlines some previous work in automotive system safety and sets the stage for the current project. Section 2 defines the objectives of this project, the basic model selected for the analysis of safety, and technologies to address the related issues. Section 3 discusses the case study and the experiments conducted. The paper ends with a conclusion.

1. IMPACT OF EMBEDDED SYSTEMS ON AUTOMOTIVE SAFETY

According to a recent study of the U.S. Transportation Research Board [5], spurred by a series of accidents related to unintended acceleration:

> *Proliferating and increasingly interconnected electronics systems are creating opportunities to improve vehicle safety and reliability as well as demands for addressing new system safety and cybersecurity risks.*

The report has acknowledged that, while electronics are central to the basic functionality of modern automobiles, at the same time they lead to new demands for ensuring the safe performance of these systems. In particular, even though it is clear that automotive electronics provide numerous benefits to reliability and safety of vehicles, they also present safety challenges. Specifically, “development of vehicle control strategies that are fail-safe (or fail-soft) in the event of some unforeseen and potentially unsafe vehicle operating condition is a critical goal for automotive manufacturers” [5].

What is stated in this report, has been pursued by researchers and manufacturers over some time. Only in the last decade, embedded systems safety and software safety in automotive systems have been investigated in a wide range of publications. For example, Papadopoulos et al. proposed conducting the modeling and analysis of automotive applications safety by using fault trees and failure propagation [6]. They were able to generate safety analysis from design models using in-house tool cooperating with Matlab/Simulink simulations. Czerny et al. [7] and Leaphart et al. [8] reviewed software fail-safe techniques for safety in automotive applications, to detect failures in Electronic Control Unites (ECU) of vehicle. Among those analyzed were: complement storage read/write, checksum compares, redundant coding, redundantly orthogonal, program flow monitoring, initialization tests, and others. A case study of Delphi’s Electric brake system was used as an application, with software safety lifecycle as a process.

Panaroni et al. [9] focused on discussing how to address automotive safety through fault tolerance, including traditional techniques of fault prevention/avoidance, fault removal, fault detection, fault isolation/containment, and fault recovery. They advocate using safety lifecycle standards, such as IEC 61508 and ISO 26262, to mitigate potential failures in early phases of system and software development.

More recently, Stringefellow et al. [10] outlined a safety-driven design technique that addresses vehicle hazard analysis. Their approach is consistent with a general model of a control system presented in Figure 1. They describe a new hazard analysis approach based on an expanded model of causality derived from system theory. A case study from the aerospace domain is discussed, but the technique applies to automotive systems, as well.

Heckemann et al. [11] discussed safety challenges of autonomously driven vehicles vs. requirements of the ISO 26262 standard, and proposed a safety cage architecture to allow a formal verification of the system behavior. They claim that combining this concept with multisensory data fusion will improve dependability, thus, enabling automotive safety.

All these papers provide a cross-section of current methods for addressing safety issues in embedded systems for automotive applications, which we followed in the current project, as described in the next section.

2. ADDRESSING SAFETY ISSUES IN AUTOMOTIVE SYSTEMS

Following the most recent work discussed in [10] and [11], we adopt a general model of a control system, as outlined in Figure 1, consistent with [10], and add to it a concept of a safety shell composed of guards, which follows the idea of a safety cage described in [11] (Figure 2). With this in mind, with respect to safety, such a system is subjected to various hazards related to a possibility of numerous failures of the controller itself and its interfaces to the operator, the network and the physical plant (its physical objects or subsystems).

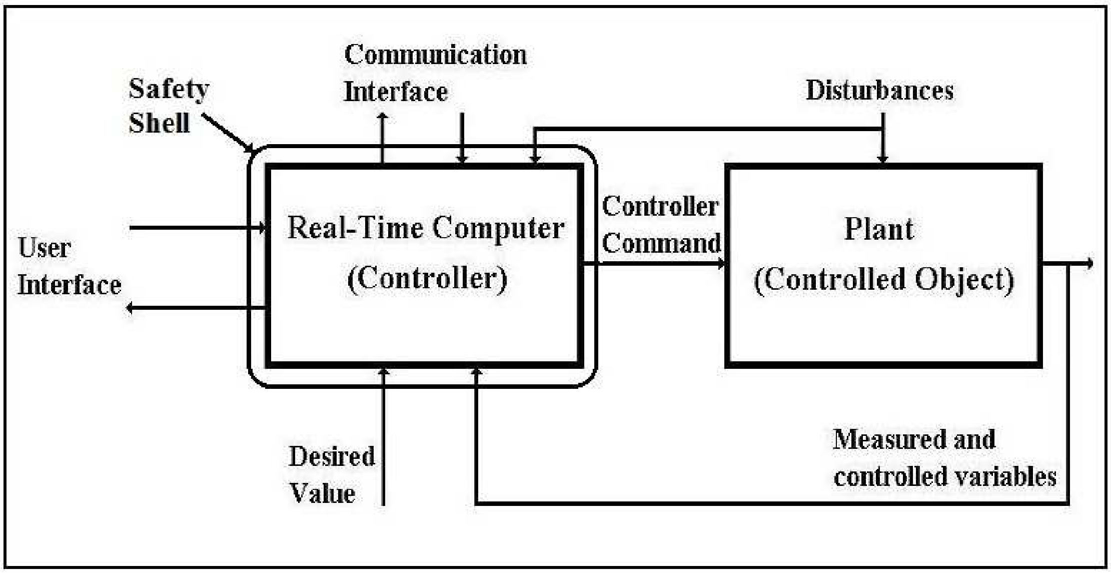


Fig. 2. Illustration of a Safety Shell Concept.

As described in previous work [12], the failures of the controller can involve: (a) “omission failure” caused by the controller to react correctly, (b) “commission failure” caused by controller providing incorrect commands or inputs, or (c) “timing failure” caused by controller responding outside of timing constraints. The failures of physical subsystems in the plant can not only involve providing incorrect response or no response at all but also can cause important “physical” failures whose consequences need to be clearly identified. The operator failures are also of crucial importance but, actually, they can be modeled as another physical subsystem allowing for more uniform hazard analysis. All these failures need to be identified and systematically classified as part of proper hazard analysis. This leads to a concept of splitting the safety shell into guards, which are individually assigned to respond to safety related issues at a specific interface.

With this model in mind, involving a shell and its guards, we decided to focus only on once specific boundary, which is a communication interface, and adopt principles of fault tolerance in achieving safety, as per [9]. The tools selected for the study follow the industry standard Matlab/Simulink, which have been used for years in similar projects in the automotive industry [6]. A more educated decision was required to select a case study.

Even though modern automotive vehicles are equipped in around a dozen of embedded computers or microcontrollers, among them: control of engine, transmission and throttle, brake power assistance and lockup control, traction and stability control, suspension control, power steering assist, adaptive cruise control, occupant protection (air bags, seat belts), etc. [5], a decision to choose the right example to demonstrate benefits of safety analysis is not an easy one, because the results of the analysis should be applicable to a range of subsystems. The initial selection of a brake system in [8] was supported, with further backup from the TRB report [5].

As described in the TRB report [5], it is an Antilock Brake System (ABS), which most of the new automotive vehicles are equipped with. Since it has been introduced in the early 1980’s, it provided multiple benefits to safety improvement:

> *A typical system uses an electronic control unit and speed sensors in the wheels. The control unit constantly monitors the speed of each wheel. If it detects a wheel rotating more slowly than the others, which indicates an impending wheel lock, the unit will reduce the brake pressure at the affected wheel. In the event of an ABS failure, the system reverts to conventional braking, in which the pressure applied to the brake pedal by the driver is not modulated by the computer and skidding can occur on slippery surfaces.*

ABS safety, however, has been analyzed many times since its invention, for example, in [13], What was more interesting to us in this research, is not the ABS itself, but its further development, a brake-by-wire (BBW) system, that is, a system which would eliminate the mechanical connection between pedal and the actual brakes, by electronically activating motors of each wheel of a vehicle. As stated in [5], since this is an advanced concept, not yet fully utilized, a convincing case must be made with regard to its operational reliability. One approach to do this would be to apply a redundant system, as in the apparently similar case of aircraft fly-by-wire.

This leads us to the objective of this project: *investigate whether redundancy is a technology which improves safety of a brake-by-wire system*. To study the answer to this question, we make the following technical assumptions:

– use the model of a safety shell and its guard for communication interface only
– use a time-triggered system from TTTech [14] to implement the ABS/BBW systems
– use Matlab/Simulink with RealTime Workshop [15] for simulation support.

With this in mind, the rest of the paper is structured as follows. First, we define the basic model’s system functionality by simulation diagrams in Simulink. Second, we develop and classify all identifiable failures. Third, we test the simulation system assuming different failures to understand the fault tolerance of the system. Fourth, we introduce the node redundancy in the implementation and observe the synchronization problems. Fifth, we insert a “guard” in the simulation models to improve the system safety. Sixth, we verify correctness of the improved simulation models.

3. CASE STUDY: BREAK-BY-WIRE SYSTEM

Following the recent work discussed in [10] and [11], we adopted a general model of a control system, as outlined in Figure 1, which consistent with the one presented in [10], and added to it a concept of a safety shell composed of guards, which is consistent with a safety cage, as described in [11] (Figure 2).

3.1. Basic Assumptions

To analyze the behavior of time-triggered systems under various faults, an anti-lock braking system (ABS) in a Break-by-Wire (BBW) version was chosen as a case study. For simulations a 4-node TTTech development cluster [14] was used in conjunction with Matlab's Simulink and Real-Time Workshop [15]. The basic models of anti-lock braking systems were transformed to improve performance under various faults. Models of an ABS system on a single wheel and of an entire car were also used in the simulations but to simplify the presentation only single wheel models are described in this paper.

An ABS system serves to prevent the wheel lock. The ability of a vehicle to lock the brakes is related to the fact that the static coefficient of friction between tires and the road surface is much greater than the dynamic coefficient of friction. That is, when the part of the tire in contact with the road is moving relative to the road, it takes less force to continue moving. This results in the wheel lock and increased stopping time and distance traveled by the vehicle between the time of brake application and the end of the vehicle's movement. Preventing the wheel lock is accomplished by appropriately regulating the application of the vehicle's brakes.

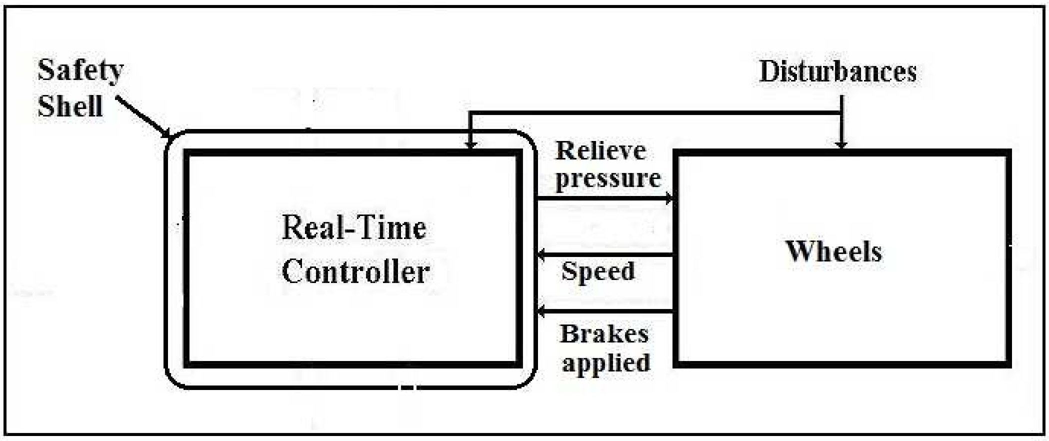


Fig. 3. Basic Model of an ABS/BBW Controller.

In a typical ABS system (Figure 3), there are several major components. These are speed sensors, brake line valves to relieve pressure, a pump, and the rea-time controller. When the operator of a vehicle attempts to apply the brakes, the wheel-speed sensors detect the deceleration of the wheels. If the deceleration on a wheel is too large, the ABS controller forces the appropriate valves to relieve pressure on that wheel's brake. This decreases the brake-force on that wheel, and decreases the deceleration of the wheel. This is crucial as great deceleration typically leads to wheel-lock. Two preexisting Simulink ABS models are considered in this paper.

3.2. Representing Simulation Models

As a first basic ABS model the quarter-vehicle model (QVM) was used. This model was created by TTTech and included with the TTP-Matlink setup software. The QVM consists of subsystems composed of Simulink block diagrams, with each subsystem simulating a necessary part of an ABS for a single wheel. In all, this model has four unique messages being sent from its subsystems in every TDMA (Time-Division Multiple Access) round.

There are four distinct subsystems: calculation of brake force as a main function of ABS controller, pedal position sensing, wheel speed modeling, and monitoring of wheel speed values. The subsystem responsible for calculating brake force (Figure 4) is composed of two blocks implementing logic to calculate brake force and clamp the brake force to 0 if required. The brake force computation block will only generate a non-zero brake force if the pedal sensor subsystem sends `pedal_pos` signal.

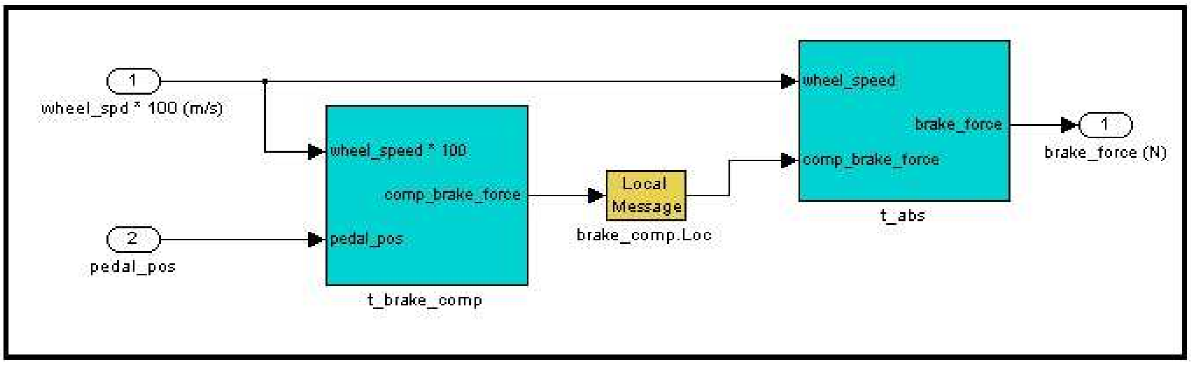


Fig. 4. The Brake Force Calculation Subsystem

The subsystem responsible for representing the brake pedal just sends a constant value every round – the presence of this signal is considered by other subsystems to indicate the intent to apply the brakes by the driver of the vehicle. The Brake Model subsystem takes the last-calculated `brake_force` and `prev_wheel_speed` values and uses them to calculate a new `wheel_speed` value. The `brake_force` message is adjusted for the vehicle's mass, and used for the integration step. The Previous Wheel Speed Monitor subsystem reads the value of wheel_speed and broadcasts it again during its next time slot.

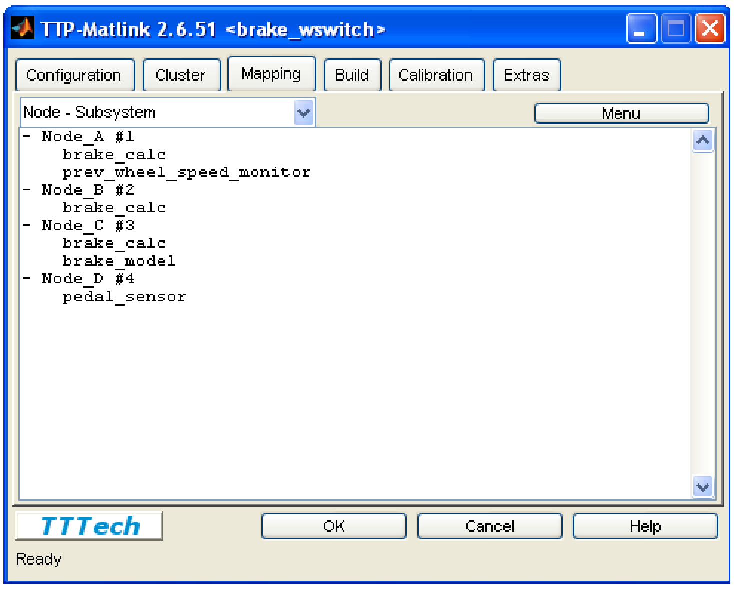
Resetting the wheel speed to full speed instantaneously is an important part of the simulation. This allows the simulation to repeat itself quickly without simulation of vehicle acceleration, which is not interesting from the point of view of brake behavior analysis.

The four-wheel vehicle model (FVM) that simulates the brakes of four wheels and their relationships is obviously much more complicated but the structure of the whole systems remains the same. For example, there are differences in the wheel speed model, that needs additional subsystem, the wheel-speed averager. The wheel-speed averager subsystem takes as inputs the speed of each node's wheel and produces the speed of the vehicle.

3.3. Analysis of the Vulnerability of the Simulation Models

The main emphasis of this project was to weigh the impact of the assignment of redundant computational processes to components on the simulated system with the presence of various faults. This objective was met by comparing the results of multiple fault-injection scenarios over multiple versions of simulation models. An important part of the project was to develop techniques to transform simulated models to reduce the impact of these faults, and evaluating the usefulness of the different solution. As a result, we determined how the distribution of the system's processes over its components changes both the systems susceptibility to faults and effects fault-correcting techniques.

For illustration of the encountered problems and solution,s let us consider the following distribution of processes as shown in Figure 5. The pedal sensor runs on separate node (Node D) while the remaining three nodes redundantly calculate the brake force calculation. The wheel speed monitor runs on one of the brake force calculating nodes. Similarly, the brake model runs on another node containing of the brake force calculation.



```
- Node_A #1
    brake_calc
    prev_wheel_speed_monitor
- Node_B #2
    brake_calc
- Node_C #3
    brake_calc
    brake_model
- Node_D #4
    pedal_sensor
```

Fig. 5. First QVM Mapping

Different fault models were selected for this and other differently distributed QVM/FVM simulations. These fault models define the disruption of the communication of individual nodes via the TTTech disturbance node, to simulate component disruption in a manner that may be observed in a real ABS.

It is expected that this mapping should perform without failure when subjected to fault injections that disturb and disable the node performing only brake force calculations, as this is redundantly-performed, but fail when the non-redundant braking model and pedal sensor nodes are disrupted.

3.4. Basic Model Simulation Results

Simulation models were tested under individual node disturbance scenarios to simulate various component failures. The nodes of the development cluster communicated via a dual-channel bus, to which each node is connected. The fault injection scenarios specified a particular schedule for the disturbance node to “communicate” on the bus. Essentially, the disturbance node was set to send a high signal on the bus at specific time periods for specific lengths of time so as to prevent a specific node from being able to send its messages. In all experiments shown in disturbance scenario execution began between rounds 1100 and 1400 and lasted 3000 rounds.

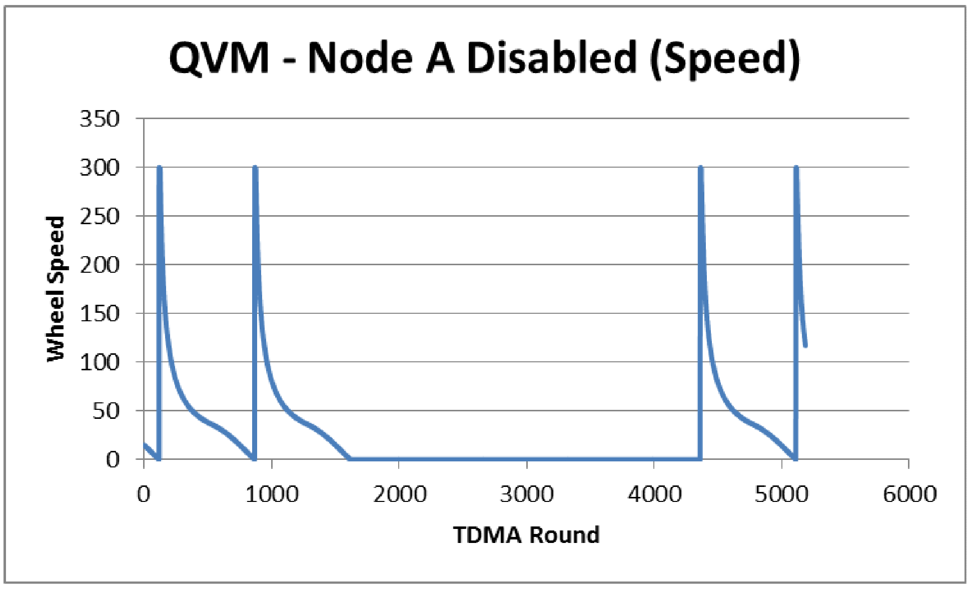


Fig. 6. Results of disabling Node A of QVM

In Figure 6, the results of preventing Node A from communicating are presented. As Node A runs the `prev_wheel_speed` subsystem, responsible for restarting the simulation when wheel speed reaches zero, and the `brake_calc` subsystem, responsible for calculating the speed of the wheel, disturbance of Node A, was expected to prevent the simulation from continuing to repeat after it finished. Since the `brake_calc` subsystem is redundant, however, the simulation in progress at the time of the initial disturbance was expected to run to completion without issue. Figure 7 shows that that is exactly what happened. The simulation restarted and ran normally as if no disturbance had happened.

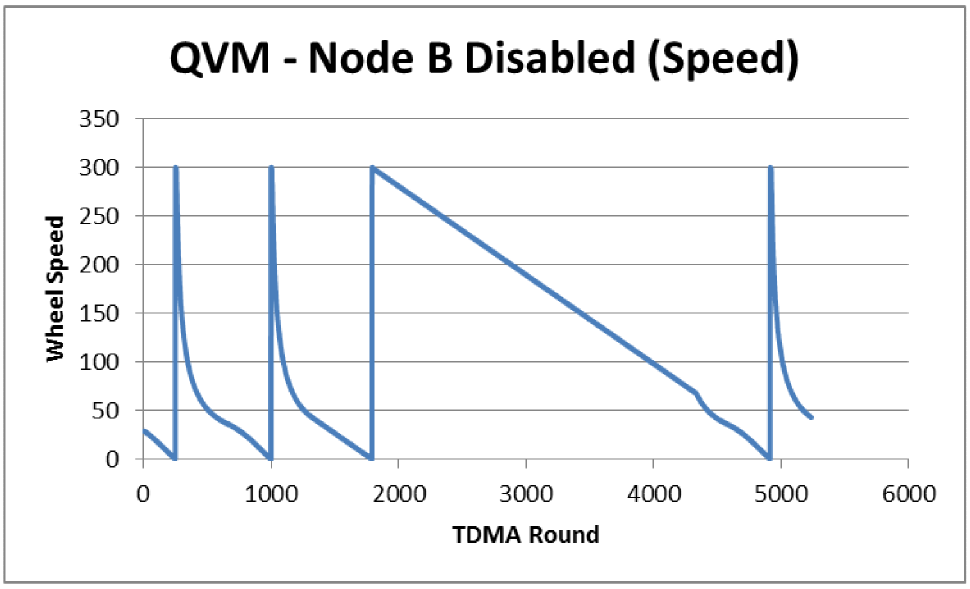


Fig. 7. Results of Disabling Node B of QVM

Further, disabling Node B, was expected to have no effect on the system because of the redundancy. This was not the case, since the disturbance scenario was executed prior to round 1100, and the current simulation at that time ran to completion. However, the decline of the wheel speed became linear. Disabling Node C was expected to lead to simulation failure, and this is what happened. This is because Node C is the only node in the QVM to calculate the wheel speed. Node D is the only node of the QVM that runs the pedal sensor subsystem, and it runs nothing else. Thus, when Node D is disturbed, the simulation is expected to halt application of the brake and allow the wheel speed to decelerate linearly until it stops. That was observed in the experimental results.

3.5. The Modified Model and Its Verification

The modified models had to correct all problems identified in the previous simulations. Let us concentrate on two first problems discussed above. The first problem was related to the simulation itself running properly, more specifically properly restarting the simulation. The model improvement was accomplished by designing a separate meta-system Simulation Control. The second problem related to the lack of differentiation between failure of a subsystem and failure of communication with the subsystem required much more complex model transformation. The “live” signal was introduced and the necessary estimation logic to calculate estimated `wheel_speed` and estimated `brake_force` was introduced. Since the `brake_model` calculates `wheel_speed` as a function of `brake_force`, the removal of a non-redundant `brake_calc` subsystem previously resulted in a complete breakdown of the simulated embedded system. By giving the brake_model the ability to estimate the correct amount of `brake_force` to apply, it is possible to partially mimic the ordinary, faultless behavior of the system.

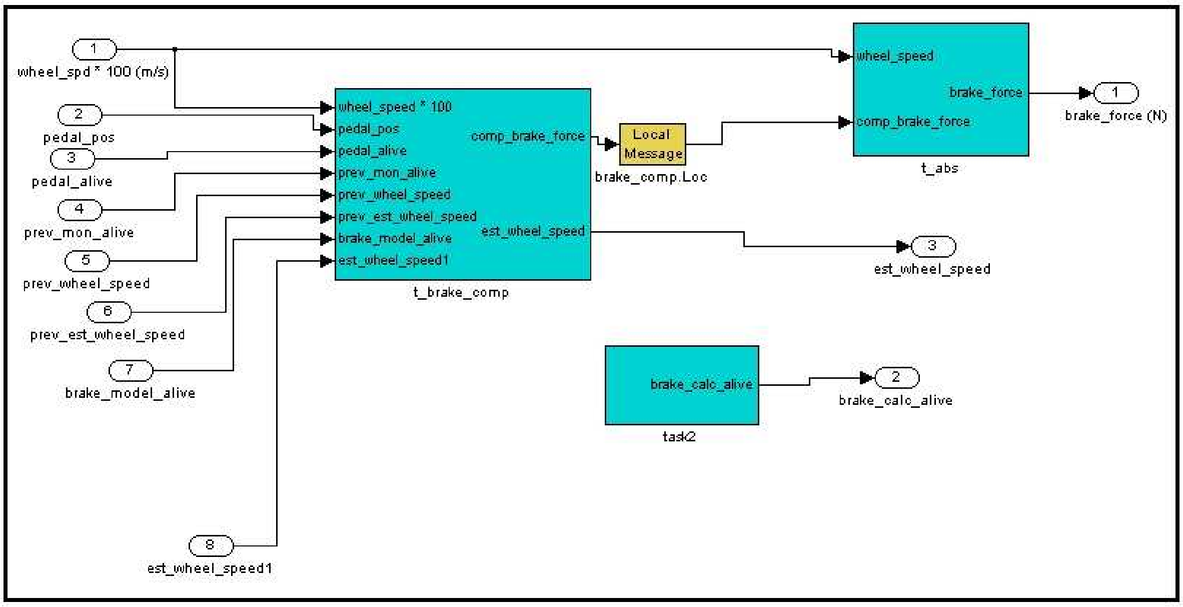


Fig. 8. High-level View of Brake Force Calculator Subsystem

The brake-force-calculating subsystem performs the same role as before, calculating and broadcasting the brake_force message as a function of wheel speed and its clamping block is unchanged from the unmodified model. However, an "alive" message block (Figure 8) has been added, and the block responsible for computing the brake force as function of wheel speed, shown in Figure 8, has been greatly expanded. The brake force computation block now checks whether the `brake_model` subsystem is "alive" prior to computing the necessary brake force. The simulation results showed that the modified QVM model now behaves normally in response to disturbances that disable any individual node of Node A, B, or C as shown in Figure 9.

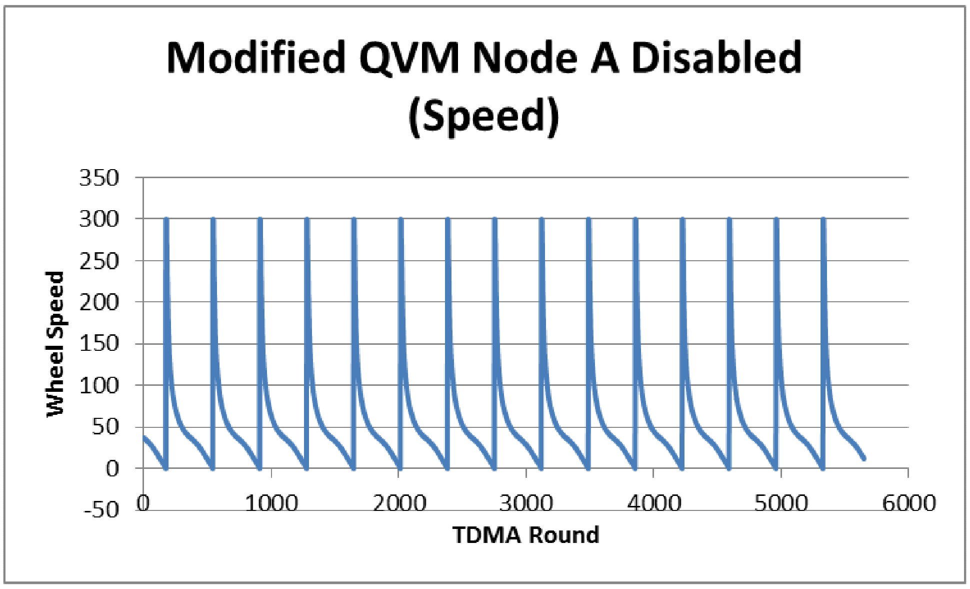


Fig. 9. Simulation with Node A Disabled

CONCLUSION

The goal of this project was to gain an understanding of how to improve vehicle safety by using fault tolerance in a time-triggered embedded system with redundant nodes by transformations of simulation models. The introduction of redundant nodes is a well-established technique to improve safety of embedded systems [1-4]. Unfortunately, only advantages of redundant nodes are usually considered, while the consequences of introduction of redundant nodes are not properly addressed or not addressed at all. Simply increasing redundancy of subsystems might not be sufficient to increase fault tolerance.

In this project we have shown that introduction of the redundant nodes can cause possibly catastrophic or costly problems in functioning of the embedded systems. The redundant nodes might introduce new synchronization problems and can cause an overhead resulting in computations not being performed within strict deadlines, and thus have unforeseen consequences for safety.

The main contribution of this paper is not only identifying the problems related to introduction of redundant nodes in embedded systems but also proposing some solutions. We solved the “redundant nodes problem” by expanding the safety shell technique originally introduced in [12]. The “shell” concept is based on development of a “guard” for each physical subsystem. Our shell had not only a low level guard but also generated proper fault signal for the controller, e.g., “not alive” or “alive”, informing whether the correct communication between subsystems is still taking place.

It was shown that there exist methods to construct the correct modified model with redundant nodes in such a way as to preserve normal behavior during the specified fault scenarios. The next step in this work is to formalize this modification by transformation rules. More specifically, we are working on defining transformation rules so that the models can be transformed using well defined templates, such as, “alive”/”not_alive” template.

REFERENCES

1. Storey N. *Safety-Critical Computer Systems*. Prentice Hall, Englewood Cliffs, NJ, 1996.
2. Zalewski J. et al., *Safety of Computer Control Systems: Challenges and Results in Software Development*. Annual Reviews in Control, Vol. 27, No. 1, pp. 23-37, 2003.
3. *Joint Software Systems Safety Engineering Handbook*, Naval Ordnance Safety and Security Activity, Indian Head, MD, August 27, 2010.
4. Obermaisser R., *Event-Triggered and Time-Triggered Control Paradigms*, Springer-Verlag, New York, 2005.
5. Transportation Research Board, *The Safety Promise and Challenge of Automotive Electronics*. Special Report 308, National Research Council, Washington, DC, 2012.
6. Papadopoulos Y. et al., *Model-based Semiautomatic Safety Analysis of Programmable Systems in Automotive Applications*. Proc. ADAS2001, Intern. Conference on Advanced Driver Assistance Systems, Birmingham, UK, September 18-21, 2001, pp. 53-57.
7. Czerny B., D'Ambrosio J., Murray B., Sundaram P., *Effective Application of Software Safety Techniques for Automotive Embedded Control Systems*, SAE Technical Paper 2005-01-0785, 2005.
8. Leaphart E.G., et al., *Survey of Software Failsafe Techniques for Safety Critical Automotive Applications.* SAE Technical Paper 2005-01-0779, 2005.
9. Panaroni P. et al., *Safety in Automotive Software: An Overview and Current Practices*. Proc. COMPSAC 2008, 32nd Annual IEEE Intern. Computer Software and Applications Conference, Turku, Finland, July 28 – August 1, 2008, pp. 1053-1058.
10. Stringefellow M.V., Leveson N.G., Owens B.D., *Safety-Driven Design for Software-Intensive Aerospace and Automotive Systems*, Proceedings of the IEEE, Vol. 98, No. 4,

pp. 515-525, April 2010.
11. Heckemann K. et al., *Safe Automotive Softwar*e, Proc. KES2011, 15th Intern. Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Kaiserslautern, Germany, September 12-14, 2011, Part 4, pp. 167-176.
12. Anderson E., van Katwijk J., Zalewski J., *New Method of Improving Software Safety in Mission-Critical Real-Time Systems*, Proc. 17th International System Safety Conference, System Safety Society, Unionville, Virginia, 1999, pp. 597-596.
13. Burton D. et al, *Effectiveness of ABS and Vehicle Stability Control Systems*. Research Report 04/01, Royal Automobile Club of Victoria, Noble Park North, Victoria, Australia, April 2004.
14. TTTech Computertechnik AG, *TTP-Powernode –Development Board. Product Description*. URL: http://www.tttech.com/products/ttp-product-line/ttp-powernode/
15. MathWorks, *Simulink. Product Description*. http://www.mathworks.com/products/simulink/

Authors:
CZEJDO Bogdan D., Fayetteville State University, Fayetteville, North Carolina, USA
ZALEWSKI Janusz, Florida Gulf Coast University, Ft. Myers, Florida, USA
TRAWCZYNKI Dawid, Advanced Micro Devices, Orlando, Florida, USA
BASZUN Mikolaj, Warsaw University of Technology, Warsaw, Poland