International Journal of Innovative Computing, Information and Control
Volume **11**, Number **4**, August **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 1341–1356

RAPID CONTROL PROTOTYPING FOR AUTOMOTIVE SOFTWARE IN POWER WINDOWS SYSTEMS

Max Mauro Dias Santos[1], João Henrique Neme[1], Felipe Rezende Franco[1]
Sergio Luiz Stevan Jr.[1], Weslley Torres[2], Alexandre Baratella Lugli[3]
Armando A. M. Laganá[4] and João Francisco Justo[4]

[1]Department of Electronics
Federal University of Technology – Paraná (UTFPR) – Campus Ponta Grossa
CEP 84.016-210, Ponta Grossa, PR, Brazil
{ maxsantos; sstevanjr }@utfpr.edu.br; neme@outlook.com; felipefranco@alunos.utfpr.edu.br

[2]Department of Automotive Electronics
FATEC Santo André
CEP 09.020-130, Santo André, SP, Brazil
weslley.torres@fatec.sp.gov.br

[3]Department of Industrial Automation
National Institute of Telecommunications (INATEL)
CEP 37.540-000, Santa Rita do Sapucaí, MG, Brazil
baratella@inatel.br

[4]Escola Politécnica
Universidade de São Paulo
CP 61548, CEP 05424-970, São Paulo, SP, Brazil
lagana@lsi.usp.br; jjusto@lme.usp.br

Received September 2014; revised March 2015

Abstract. *Current and next generations of passenger and commercial vehicles have functions with increasingly higher levels of complexity. The traditional methodologies of software development are not robust enough to meet the requirements for current vehicle systems. Modern features and functions must provide improved properties in terms of drivability, safety, comfort and performance, and also additional features, such as driver assistance and connectivity. In order to address those challenges, new development methodologies and standards for automotive embedded systems are necessary to ensure the correct process for all phases of the automotive software engineering, from development to production. Model-based design (MBD) methodology represents a key methodology which may satisfy those demands of automotive software development. It considers that the first step is to define the requirements based in the physical plant simulated, followed by the development of the controller, and then a set of test cases are performed with the controller, to validate the initial requirements. The first stage of MBD, model-in-the-loop, is presented for the power window function of an automotive system.*

Keywords: Model-based design, Power windows, Automotive software, Embedded system

1. Introduction.

Rapid control prototyping (RCP) is the calibration process of control algorithms on a hardware prototype. This process allows to place a test in operation before there is an electronic control unit (ECU) available for production. The RCP platforms typically offer a method to import mathematical models and execute them in a real time controller operating system connected to an I/O of the real world. RCP allows engineers to test quickly and iterate their control strategies. Consequently, mathematical models can be automatically imported with Matlab/Simulink® on a real-time machine with real

I/O interfaces to connect to real-world systems. RCP solutions can be used with RT-LAB, a user-friendly software interface.

RCP shortens development time, by allowing corrections early in the process. By giving engineering an early view on the product in the design process, errors can be corrected and changes can be incorporated while they are still inexpensive, thereby reducing costs and development time.

Model-based design methodology has been widely used in product development, providing benefits to all parts involved in the process, from the definition of requirements, controller development and test cases to verification and validation of the control strategy of automotive embedded software. This paper presents a workflow of model-based design for automotive exterior lighting system, in the stage of the model-in-the-loop. The approach includes the complete process from requirement definition, modeling the physical plant, design of event-driven controller, simulation and test cases. All those stages were performed using tools from MathWorks® [3].

This paper is organized as follows. Section 2 presents a brief description of rapid control prototyping. Section 3 shows an overview of model-based design methodology. Section 4 describes how to develop a software controller for the power window system using the model-based design, but considering only the model-in-the-loop stage. Section 5 describes the process of rapid control prototyping of the power window function to be tested. Finally, Section 6 presents a summary.

2. **Rapid Control Prototyping.** Design engineers have a major concern: they want to focus completely on their ECU functions, and run quick target trials of their designs, both offline and in a real vehicle or in a test bench. At that stage, they need fast certification that the project is moving forward in the right direction. Since producing a prototype is expensive and time consuming, a far more efficient method is to use a dSPACE prototyping system.

This method ensures flexible development systems, such that engineers can optimize their function designs. Design faults are identified faster and corrections can be carried out immediately. Independent of the ECU application, such as engine, powertrain, vehicle dynamics, airbags, or comfort systems, dSPACE prototyping systems can handle them all.

Diagrams and state diagrams in Simulink®/Stateflow® are the starting points for function prototyping. They represent the controller and its environment graphically and can be simulated offline for a first validation. In order to perform prototyping, the Simulink®/Stateflow® block diagrams are implemented on a dSPACE prototyping system and computed in real time. Therefore, dSPACE prototyping systems act as real ECU prototypes, either onboard the vehicle or in the test bench. To analyze a prototyping experiment, dSPACE prototyping systems record all the data in real time, and any desired control parameters can be optimized online.

The prototyping hardware is far more powerful than the real ECU; therefore, there is no hardware limitations in prototyping. In order to connect sensors and actuators on the prototyping hardware, there is a compact, modular hardware solution consisting of adjustable parameters of the components. These parameters provide the correct signal conditioning and power stages for each sensor and actuator type.

In control rapid prototyping process, the traditional procedures of algorithm development, such as hardware and software design and manual iteration leading to the implementation of a production version, are combined into a single step. Instead of custom made hardware (e.g., a specific embedded processor), off-the-shelf components are used to test control algorithms in real-time hardware implementations. The use of off-the-shelf controller components (and the respective software) allows an automatic code generation, which can then be downloaded to the rapid prototyping controller for testing. Development can proceed much faster in this environment, since the control designer (with the off-the-shelf controller and the automatic code generator) can essentially perform the functions of three traditionally different engineering departments, allowing considerably faster design iterations. Figure 1 compares the traditional and the rapid prototyping procedures.

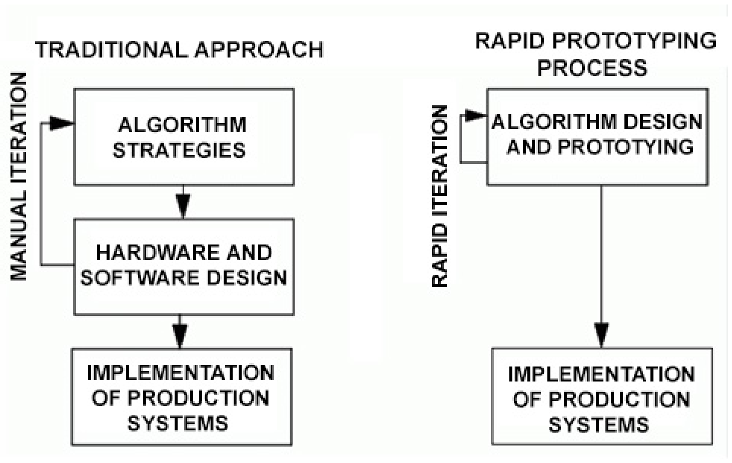
TRADITIONAL APPROACH
MANUAL ITERATION
ALGORITHM STRATEGIES
HARDWARE AND SOFTWARE DESIGN
IMPLEMENTATION OF PRODUCTION SYSTEMS
RAPID PROTOTYPING PROCESS
RAPID ITERATION
ALGORITHM DESIGN AND PROTOTYING
IMPLEMENTATION OF PRODUCTION SYSTEMS

FIGURE 1. Traditional versus rapid prototyping

2.1. **Control strategy development and verification.** In this case, the controller plays a role as the feature function supervisory control algorithm, while the plant consists of the Radar Sensor and several other parts of the vehicle that must be controlled. Development involves highly complex safety critical supervisory control strategies, which should perform consistently under all kinds of input/traffic scenarios, with no tolerance for failure. This requires the control algorithm to be validated and verified thoroughly before the design is finalized for software implementation. In order to benefit from MBD, the control strategies for these feature functions are developed as executable specification using Matlab®/Simulink®/Stateflow which is verified on the host computer as well as on the actual vehicle using a Rapid Prototyping environment.

Typical control software for automotive applications undergoes four stages of verification (shown in Figure 2): Offline Simulation, Rapid Control Prototyping, Hardware In Loop Simulation, and ECU Validation. The first two stages verify the functional performance of the control algorithm. It is performed on the control strategy model (specification/algorithm) before the production software is generated and is described in detail here. The last two stages are performed on production software embedded in the final ECU.

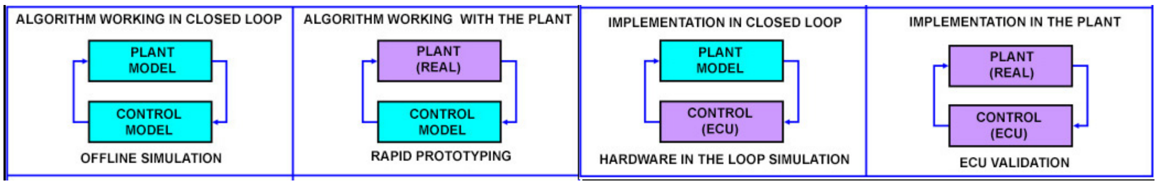
ALGORITHM WORKING IN CLOSED LOOP
PLANT MODEL
CONTROL MODEL
OFFLINE SIMULATION
ALGORITHM WORKING WITH THE PLANT
PLANT (REAL)
CONTROL MODEL
RAPID PROTOTYPING
IMPLEMENTATION IN CLOSED LOOP
PLANT MODEL
CONTROL (ECU)
HARDWARE IN THE LOOP SIMULATION
IMPLEMENTATION IN THE PLANT
PLANT (REAL)
CONTROL (ECU)
ECU VALIDATION

FIGURE 2. Control strategy and verification stages

2.2. **Model-based design for automotive software.** RCP provides automatic execution of system identification and system modelling. During identification, RCP identifies parameters of both electrical motor and mechanical parts of the system. The model determination is based on data acquisition, which is used for selection of a suitable plant model with possible nonlinearities. System identification and model determination are the basis for the next automated parameter calculation of the torque control, speed control and position control loops.

RCP developing software is compatible with Matlab®/Simulink® and can provide interaction with other control systems. Unlike existing RCP systems, the proposed RCP decouples the platform from PC-supported simulation and developing methods. System identification, model determination and draught regulation are completely processed in the RCP hardware, which reduces development expenses and provides expertise.

The hardware includes processor unit, power electronics, instruments to measure motor current and supply voltage, digital I/Os, A/D and D/A converters. The hardware provides communication via CAN-Bus, Serial Peripheral Interface Bus (SPI), RS232/USB interfaces, and Ethernet. The core of the platform is the microcontroller 16-bits μC Infinion XC167, which has been designed for automotive applications and is often used as a target processor for engine management and other vehicular systems.

The power electronics circuit consists of four high current PWM (Pulse Width Modulation) controlled half bridges, in order to commutate one three-phase AC motor or two brush DC motors or electromagnetic actuators. The power electronics circuit can operate at a maximum current of 30A and a maximum voltage of 30V with PWM frequency up to 25kHz. Each half bridge contains a current sensor which is used for torque/force control. The value of PWM is controlled by the supply voltage level. The platform provides 8 digital inputs and 8 digital outputs. There are also 8 A/D and 8 D/A channels. The system supports two CAN channels, two RS232 (or 1xRS232 + 1xUSB) ports and Ethernet interface.

Matlab® toolbox, which allows automatic code generation for a variety of off-the-shelf embedded controller prototypes, is called the Real-Time Workshop. Simulink® can be used to create a hi-fidelity model of the system to be controlled, and to develop prototype control systems for this model. Real-Time Workshop can then be used to generate controller code from the control portion of the Simulink® model, which can then be downloaded to the prototype controller for real-time, hardware-in-the-loop testing. Figure 3 illustrates this concept of exporting a controller developed in Simulink® to control a real system.

2.3. **MicroAutoBox hardware.** MicroAutoBox is a real-time system to perform fast function prototyping in fullpass and bypass scenarios. It operates without user intervention, like an ECU. It can be used in many different RCP applications, such as chassis control, powertrain, body control, electric drives control, X-by-wire, advanced driver assistance systems (ADAS), and aerospace applications.

The special strength of MicroAutoBox hardware is its unique combination of high performance, comprehensive automotive I/O, and an extremely compact and robust design (extreme shock and vibration tests according to ISO 16750-3:2007). In addition to the standard I/O, MicroAutoBox offers variants with FPGA (Field-Programmable Gate Array) functionality for application-specific I/O extensions and for user-programmable FPGA applications. Moreover, there are MicroAutoBox variants with interfaces for all major automotive bus systems: CAN, LIN, K/L line, FlexRay, and Ethernet. As an option, an additional embedded PC can be integrated into the MicroAutoBox II.

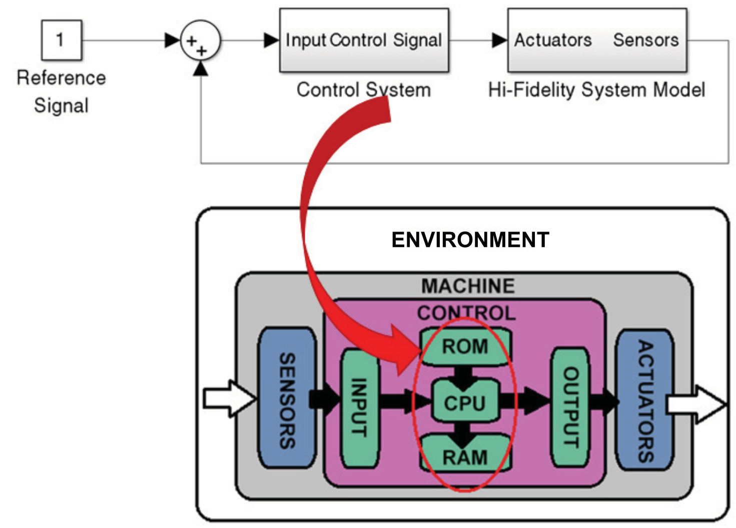

FIGURE 3. Exporting a Simulink® controller for the RCP platform

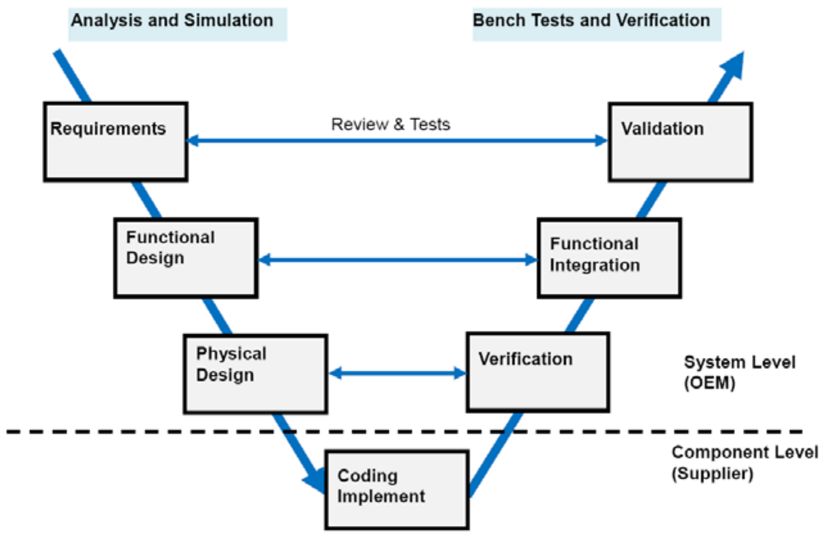

FIGURE 4. V-Model for development of automotive software architecture in E/E

3. **Model-Based Design.** MBD is a mathematical and visual design methodology for complex systems in a number of domains. The system model is in the center of the development process, from requirement development, through design, implementation and test [2]. MBD provides an efficient approach for establishing a common framework for communication throughout the design process, while supporting the development cycle ("V" diagram). Figure 4 shows those processes organized for the automotive embedded systems in a V-Model for hardware and software.

In MBD of complex control systems, the development is divided into four main steps: 1) modeling a physical plant, 2) analyzing and synthesizing a controller for the physical plant, 3) simulating the plant and controller, and 4) integrating all those phases by deploying the controller. The MBD paradigm is significantly different from traditional design methodologies, in which, rather than using complex structures and extensive codes, designers can use the MBD approach to define models with advanced functional characteristics using continuous-time and discrete-time building blocks. Those models, used with simulation tools, can lead to rapid prototyping, software test and verification. Not only the test and verification processes are improved, but also the hardware-in-the-loop simulation can be used to perform tests of dynamic effects on the system faster and more efficiently than with traditional design methodologies. It is important to consider the players involved in this process: OEM (Original Equipment Manufacturer, which makes a final product for the final consumer), Tier-1 (direct suppliers to OEMs), and Tier-2 (key suppliers to Tier-1, without supplying directly to OEM).

The main steps in MBD approach are:

a) Plant modelling. It can be data-driven or first principles based and the data-driven plant modeling uses system identification techniques. With system identification, the plant model is identified by acquiring and processing raw data from a real-world system and choosing a mathematical algorithm (mathematical model). First principles based modeling is based on creating a block diagram model that implements known differential-algebraic equations governing plant dynamics. A type of first principles based modeling is physical modeling, where a model is created by connecting blocks that represent the physical elements to the real plant.

b) Controller analysis and synthesis. The mathematical model conceived in step a) is used to identify dynamic characteristics of the plant model and then the controller can be synthesized based on these characteristics.

c) Simulation. In order to solve the problem, usually multiple models of computation should be used during the construction of the dynamic system, and during the simulation, errors due to steps a) and b) can be identified immediately, rather than later in the design phase. During this step, it is possible to capture all the system interation, and test against theoretical model provides a robust design. Normally, the code synthetizers are incorporated into simulation environment, assuming that the simulation is working properly, and the code is then corrected by construction.

d) Deployment. At this point, the automatic generated code, from the controller developed in step b), will be running in the real system, similar to simulation, allowing debugging the real application.

MBD offers several advantages when compared to traditional approaches. It provides a common design environment, which facilitates general communication, data analysis, and system verification between development groups. Engineers can locate and fix errors early, when time and financial impact are minimized. Design reuse, for upgrades and derivative systems with expanded capabilities, is eased.

4. **Model-Based Design for Power Windows.**

The power window system is a vehicle subsystem that comprehends a set of mechanical components to open/close the windows, lifting glass machine, control switches, and control unit.

We considered the power window system in the passenger side. The driver and passenger can control the system by pressing a switch, where the driver side has priority over the passenger one. The switch activation allows the system to feed an electrical motor moving up or down the glass respectively to the top or bottom and can be stopped by the user or if there is an obstacle for motion. Figure 5 illustrates the traditional power window system in a vehicle. This work examined the controlling software power window operation, and later implemented the code using MBD, then testing and comparing with requirements of ISO 26262 standard.

4.1. **Requirements.**

The first step of the process defines requirements of a specific subsystem to be controlled. The product development workflow used Mathworks® tools. The requirements are:

- **REQ-01:** The glass must be fully opened or closed in 4s;

- **REQ-02:** The glass starts to move 200ms after the command is selected;
- **REQ-03:** After 4s of uninterrupted motion in one direction, the motor should turn off;
- **REQ-04:** The control system voltage operation is between 12.5V and 14.5V;
- **REQ-05:** If the command to open or close the glass is kept pressed between 200ms and 1s, the glass opens or closes completely;
- **REQ-06:** If an obstructing object is detected, the glass must come down approximately 10cm;
- **REQ-07:** The obstacle detection has priority over the driver and passenger switch commands;
- **REQ-08:** The driver switch commands have priority over the passenger ones;
- **REQ-09:** When triggered, the security system must close completely unless there is an obstacle;
- **REQ-10:** The control system operates only with ignition position in "On", Accessory and Run.

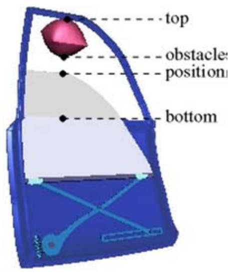

FIGURE 5. Electrical power window system [5]

4.2. **Model of physical plant.** The power window can be modelled virtually using a computer tool that presents its dynamic behavior. The model physical plant is the main component of MBD, since the controller is designed around the virtual system. Using the requirements, the controller is designed using Simulink® dynamic system libraries to describe the behavior model (internal block diagrams with I/O interface, which can connect the controller with a specific hardware) and all tests can be performed using the virtual physical plant. Figure 6 shows a block diagram representation of the power window system.

4.3. **Event driven controller.** In order to meet the requirements, the speed and glass position sensors are incorporated in the power window embedded control system. For the simulation, some input variables of glass state were added. The model only considered one electric glass, since the other glasses were just a replication of the developed code. The input variables were:

- **Passenger Up** and **Passenger Down** – representing the signal trigger that the passenger window goes up and down;
- **Driver Up** and **Driver Down** – representing the signal trigger that the driver controls the passenger window up and down and have priority over passenger controls;
- **End Stop** – limit switch handling the glass, rising or lowering the glass;
- **Obstacle** – obstacle detection in the course of glass;
- **Alarm** – safety system status.

Subsequently the signals that represent the commands to the physical plant of the system were analyzed. Those signals are "moveUp" (signal to raise the glass) and "moveDown" (signal to lower the glass). After the implementation following the requirements,

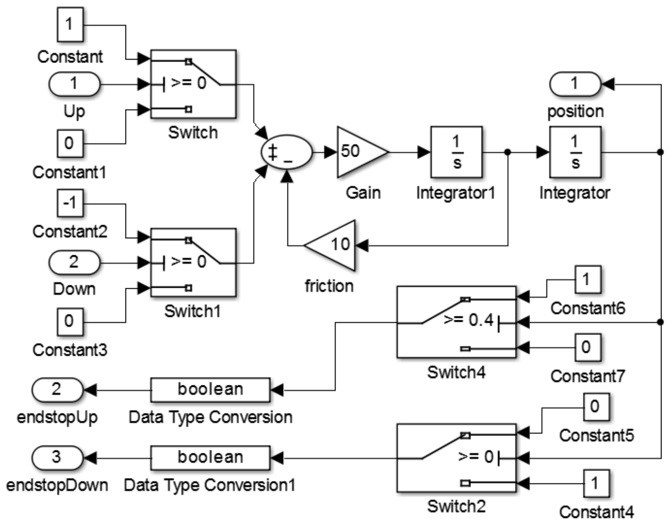

FIGURE 6. Model of the power window system

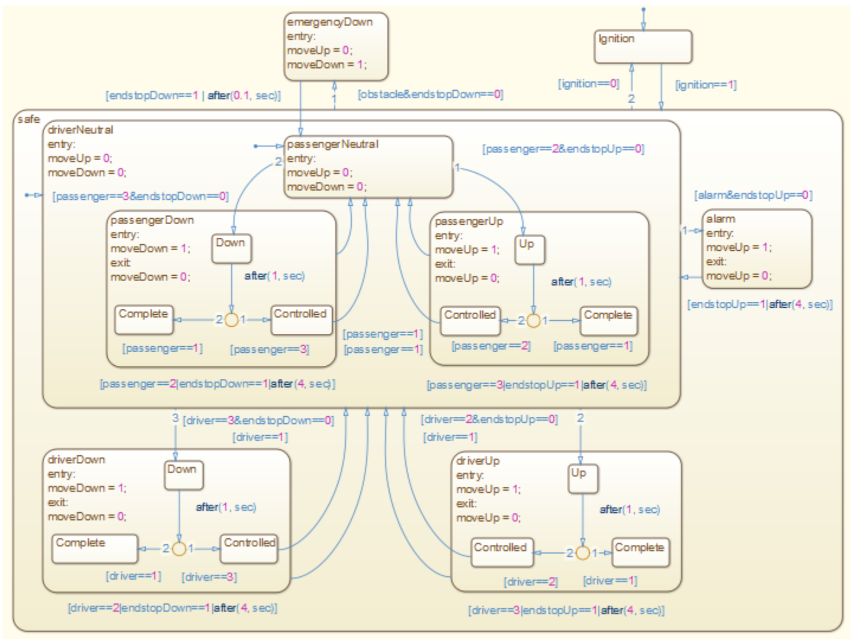

FIGURE 7. State diagram of power windows control

the state transitions diagram of the power window system, which represents the event driven controller with adjustment of the signal alarm, is presented in Figure 7.

In order to improve the signal trigger analysis, both the passenger and driver signals were processed using *truth tables*, and then they were sent to the control system. The truth table that analyses the passenger signals is shown in Figure 8.

This strategy of analysis was used for the ignition signals too (Figure 9).

The final diagram also takes account of the vehicle ignition signal (Off, Accessory, Run, and Crank) and battery voltage. The ignition system consisted of the following requirements.

- Case voltage reading is not between 12.5V and 14.5V and the system does not work;
- The ignition key position is not required to trigger control.

For the power window system to enter in operation, the battery voltage and ignition position must be ready. Table 1 shows the *truth table* of the system.

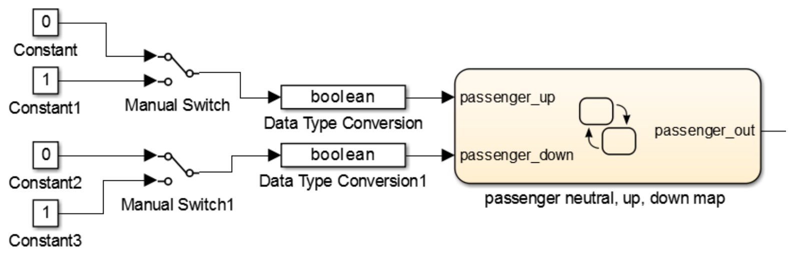

FIGURE 8. Electrical power window system

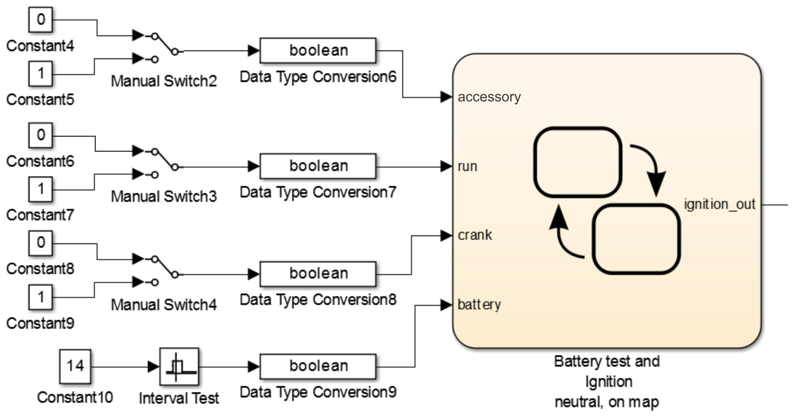

FIGURE 9. Battery test and ignition system

TABLE 1. The *truth table* of the power window controller

Inputs				Outputs
Accessory	Run	Crank	Bat_Volt	NEU [1 0] ON [0 1]
0	0	0	0	1 0
0	0	0	1	0 1
0	0	1	0	1 0
0	0	1	1	1 0
0	1	0	0	1 0
0	1	0	1	0 1
0	1	1	0	1 0
0	1	1	1	1 0
1	0	0	0	1 0
1	0	0	1	0 1
1	0	1	0	1 0
1	0	1	1	1 0
1	1	0	0	1 0
1	1	0	1	1 0
1	1	1	0	1 0
1	1	1	1	1 0

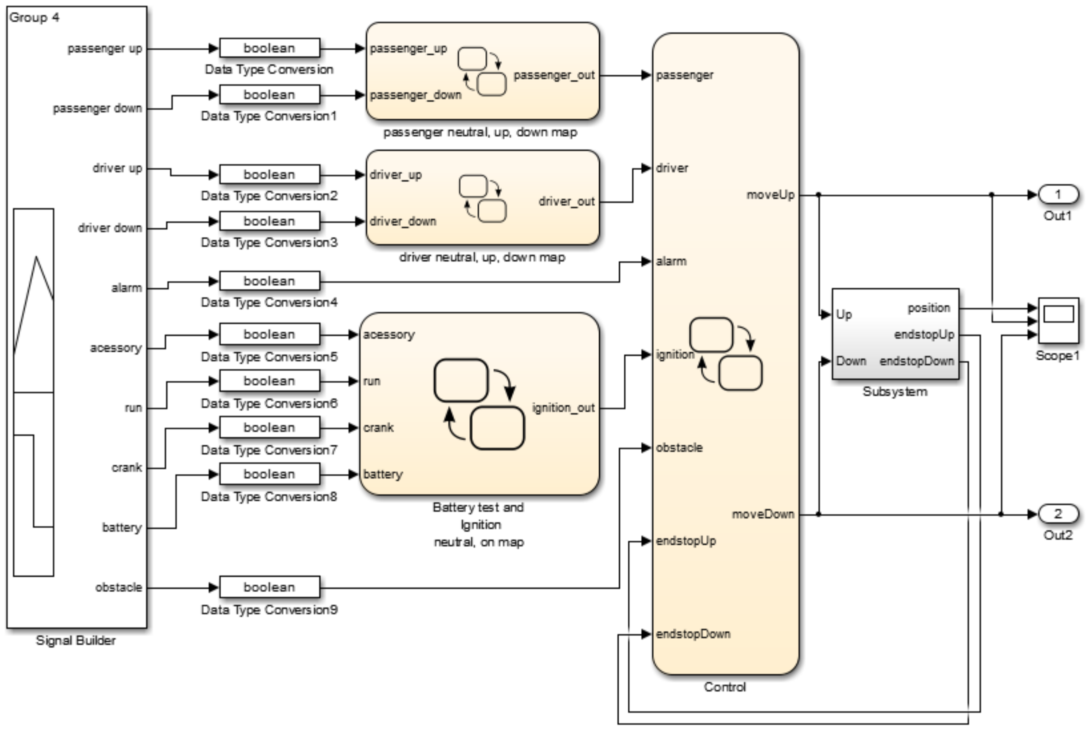

FIGURE 10. Electrical power window system

5. **Rapid Control Prototyping of Power Window System.** Figure 10 shows the complete power window system implemented in Simulink®.

5.1. **Inputs for test.** In order to demonstrate that the event driven controller was designed according to requirements defined in preliminary phase, a set of tests with signal input was designed for the controller. The controller has nine external inputs that affect the behavior of power window system: Driver-side window UP, Driver-side window DOWN, Passenger-side window UP, Passenger-side window DOWN, EndStop, Accessory, Run, Crank, and BatteryVoltage (Bat_Volt). The first four inputs correspond to the buttons power windows available to the driver or passenger and the others are associated with safety and performance. The last external input represents an obstacle in the glass path. Four test cases with different input combinations were constructed using the *Signal Builder*® block. Those test cases allowed verification if the model meets specified performance criteria.

5.2. **Test Case 1.** *The test cases* were performed with a signal generator block (*Signal Builder*®) to validate the requirements of the controller of the diagram. Therefore, it can check if system requirements were met and if changes were necessary in the diagram.

The requirements 1, 2, 3, 4 and 5 are shown in Figure 11, with a set of inputs of Test Case 1. The battery voltage is in normal operation ($12.5\text{V} < \text{BatteryVolt} < 14.5\text{V}$). When battery voltage is in a threshold value, the battery signal is 1; otherwise it is 0.

Figure 12 shows the relevant output signal function of time and how the requirements are met. The window starts moving up approximately at $\text{T} = 0.25\text{s}$ and stops at $\text{T} = 4.25\text{s}$ and starts moving down at $\text{T} = 5.0\text{s}$ (The glass begins moving down at 200ms) and stops at $\text{T} = 9.0\text{s}$. Therefore, it is concluded that the following requirements have been met.

- **REQ-01:** The glass must be fully open or closed in 4s;
- **REQ-02:** The glass begins to move 200ms after the command is selected;

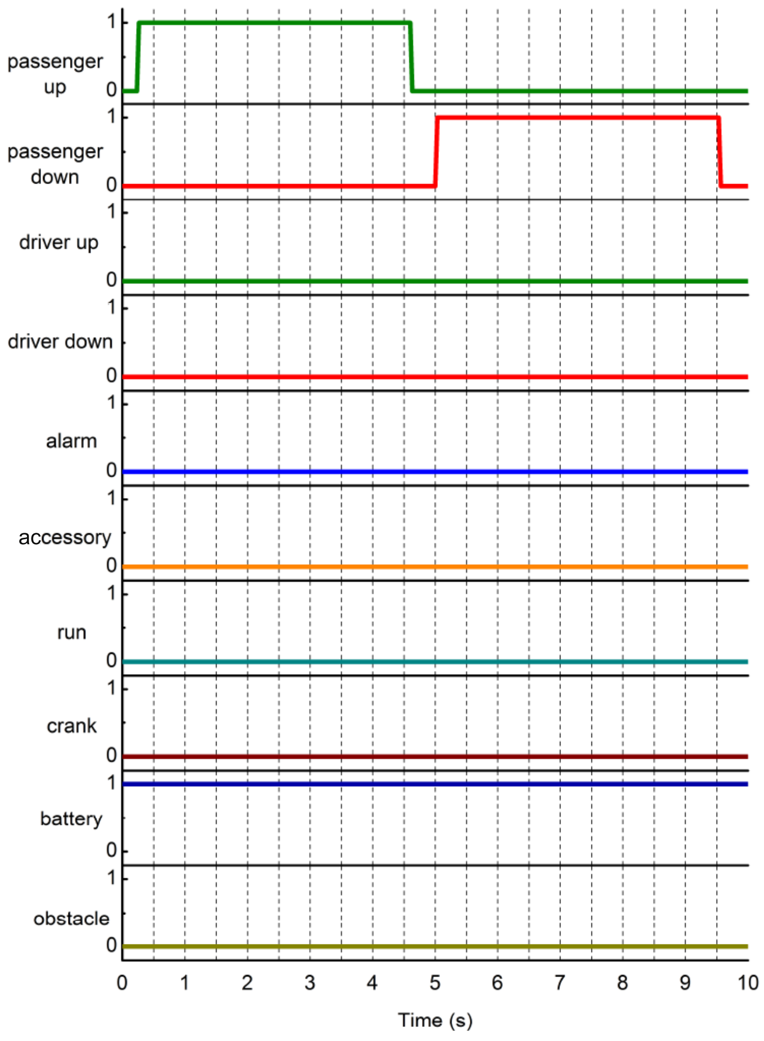

FIGURE 11. Signals input for *Test Case 1*

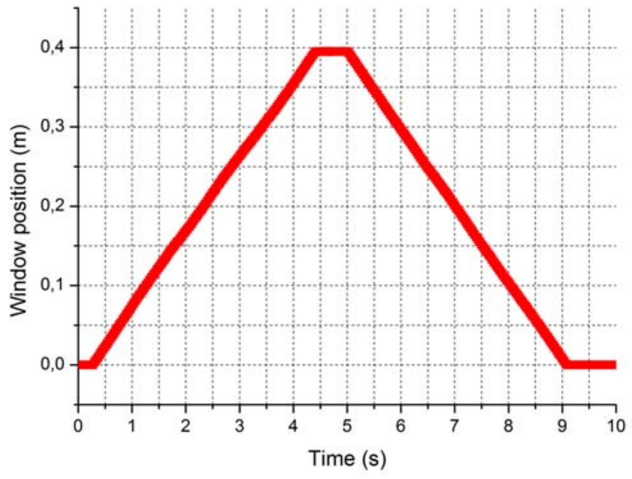

FIGURE 12. Power window position vs. time for *Test Case 1*

- **REQ-03:** After 4s of uninterrupted motion in one direction, the motor turns off;
- **REQ-04:** The control system voltage operation is between 12.5V and 14.5V.

5.3. **Test Case 2.** *Test Case 2*, with the input set shown in Figure 13, tests the REQ-05 and is similar to *Test Case 1*. Figure 14 shows the relevant output signals as a function of time and explains how the requirements are met.

According to Figure 14, the window starts moving up at approximately T = 1.0s and stops at T = 5.0s and starts moving down at T = 5.4s (The glass begins moving down 200ms) and stops at T = 9.4s. Therefore, the following requirement has been met:

- **REQ-05:** If the command to open or close the glass is kept pressed in between 200ms and 1s, the glass will be opened or closed completely.

5.4. **Test Case 3.** *Test Case 3*, with input set shown in Figure 15, tests the requirements 6, 7 and 8. In this case the system was tested in a situation with the occurrence of an obstacle. Figure 16 shows that the window moves down after detecting the obstacle, even though the passenger is requesting the window to move up and down. This satisfies REQ-06 and REQ-07.

At T = 4.25s, the passenger triggers the glass down and once the driver activates the glass to rise. Figure 16 shows that in this range the glass increases from T = 4.25s to T

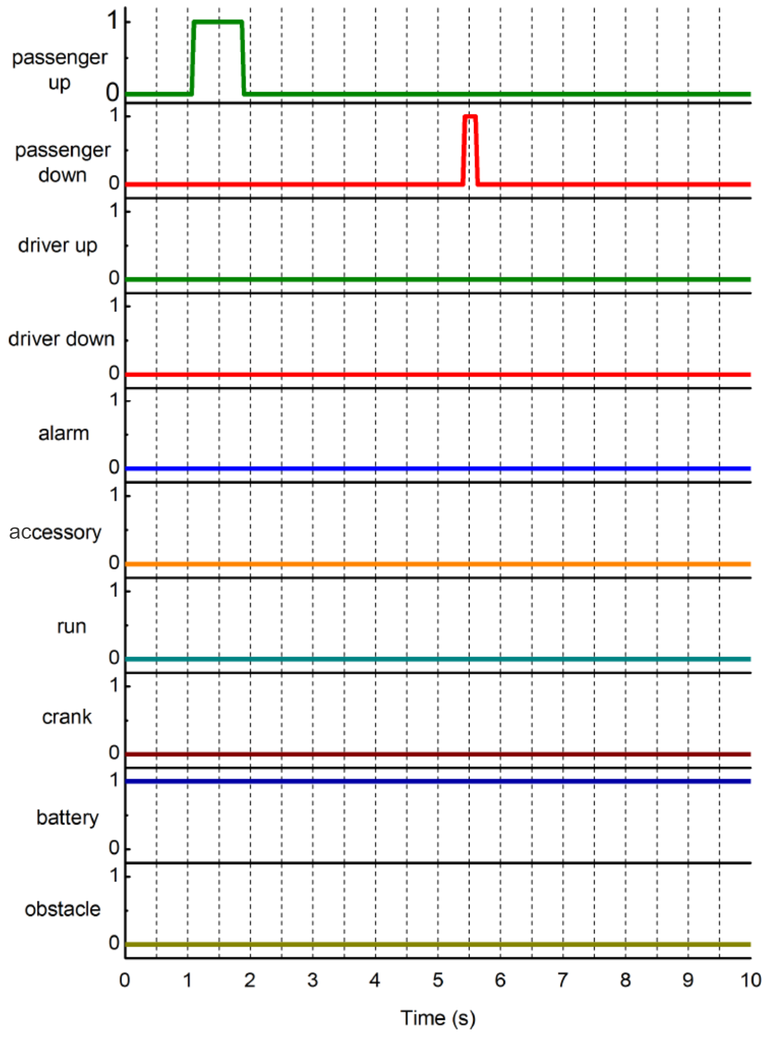

FIGURE 13. Signals input for *Test Case 2*

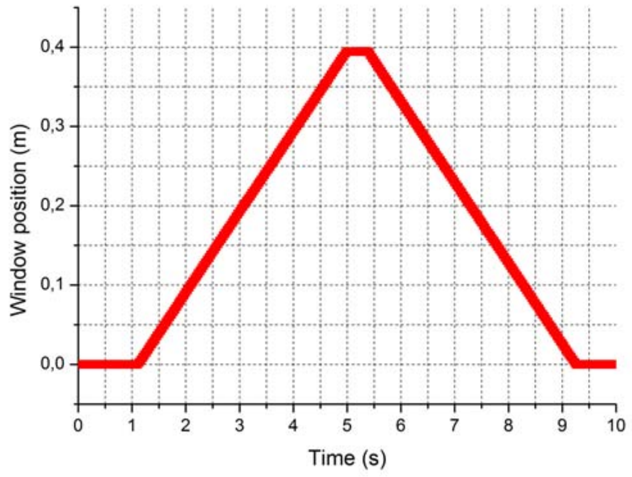

Figure 14. Power window position vs. time for *Test Case 2*

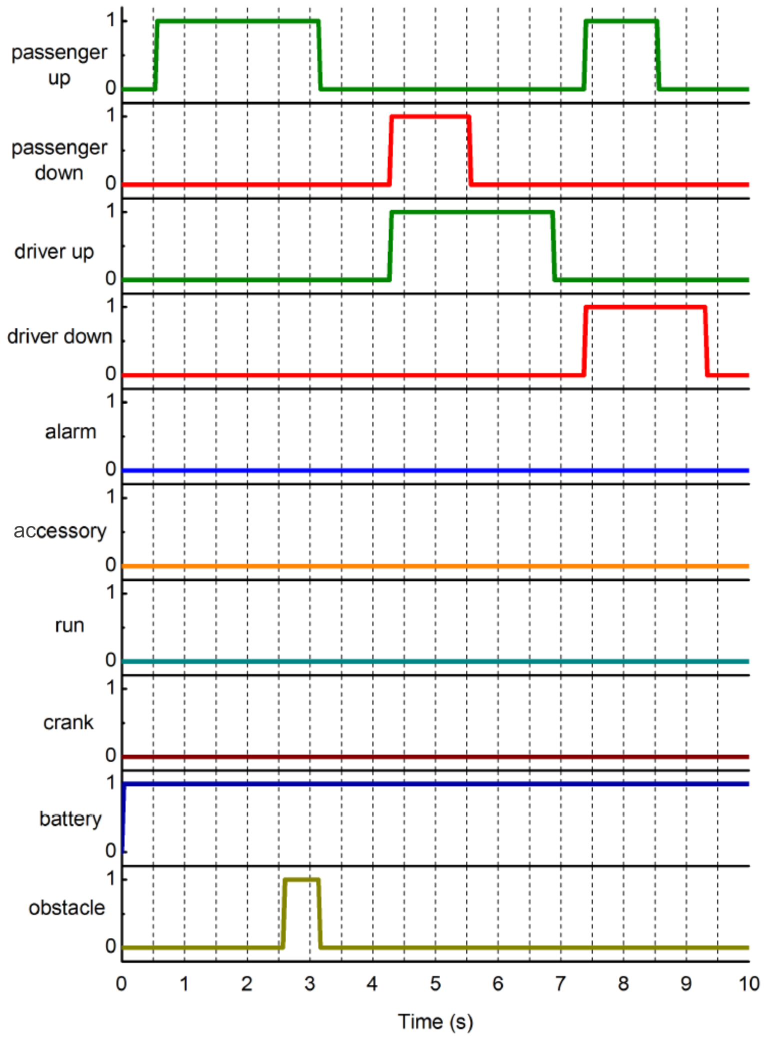

Figure 15. Signals input for *Test Case 3*

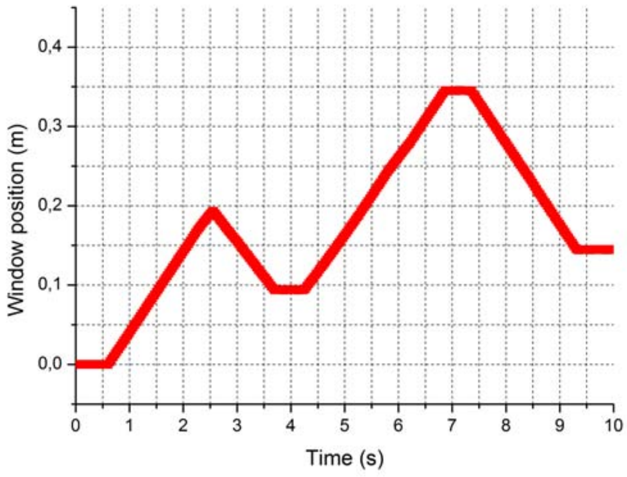

FIGURE 16. Power window position vs. time for *Test Case 3*

= 6.75s. At T = 7.5s, the passenger triggers the glass to rise and at the same time the driver activates the glass down. It is shown that in this range the glass down from T = 7.5s to T = 9.25s. Therefore, REQ-08 is satisfied. The following requirements have been met:

- **REQ-06:** If an obstructing object is detected, the glass must go down approximately 10cm;
- **REQ-07:** The obstacle detection has priority over driver and passenger switch commands;
- **REQ-08:** The driver switch commands have priority over the passenger one.

5.5. **Test Case 4.** *Test Case 4*, with the input set shown in Figure 17, tests requirements 9 and 10. At T = 0.25s, there is a signal of the security system, showing that the glass began to rise, and finished its motion after 4.25s, closing the glass, satisfying the REQ-09. At T = 5.0s, the command to move down is executed; however, the glass only moves when ignition is Accessory or Run, as shown in Figure 18. This satisfies REQ-10. At T = 8.5s, the battery signal is turned off, making the power window system inoperative, which satisfies REQ-04.

Therefore, the following requirements have been met:

- **REQ-09:** When triggered, the security system glasses should close completely unless there is an obstacle;
- **REQ-10:** The control system operates only with ignition position "On", Accessory and Run.

It should be emphasized that when the ignition is in Accessory and Run position, and the signal from the battery sensor is available, the control system is enabled. Also, if the ignition is in the "Off" position (no signal) and the signal from the battery sensor is available, the control system is enabled. However, when the ignition is in the Crank position, the control system is not activated, regardless of the battery sensor signal.

6. **Summary.** In summary, we have shown how to design software to control functions in a vehicle using MBD. Additionally, it has been shown how to handle the whole process using MBD for functional safety applied to power window systems. This system satisfied the requirements as tested and additional test cases could be used for new tests with wider

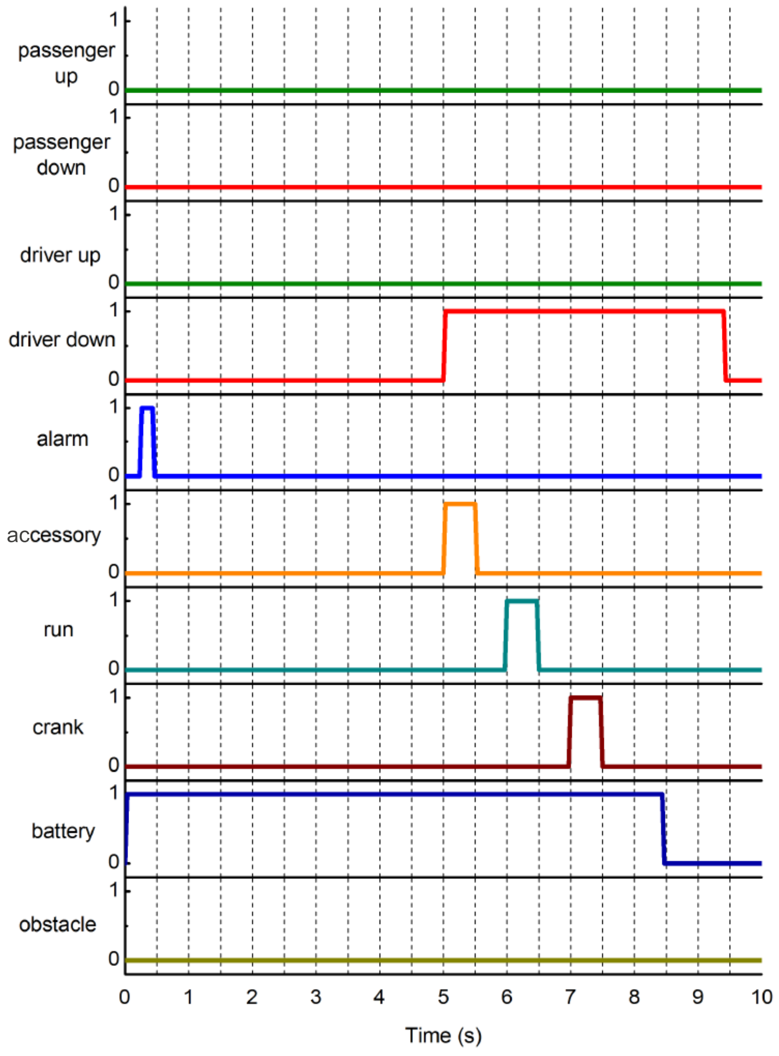

FIGURE 17. Signals input for *Test Case 4*

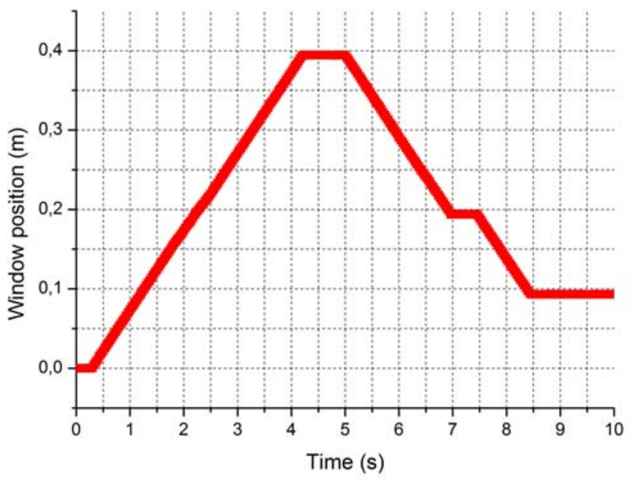

FIGURE 18. Power window position vs. time for *Test Case 4*

coverage. Using MBD, it is possible to design and test controllers with a plant model in a closed loop. The level of fidelity of the plant model depends on system requirements, but for this first work we use a simplified plant model only to illustrate the methodology. Once the controller has been verified and meets the system requirements, the system can be tested in real time in different configurations.

REFERENCES

[1] A. Sangiovanni and M. D. Natale, Embedded system design for automotive applications, *IEEE Computer*, vol.40, no.10, pp.42-51, 2007.

[2] M. Brou, S. Kirstan, H. Krcmar and B. Schatz, What is the benefit of a model-based design of embedded system in the car industry? *Emerging Technologies for the Evolution and Maintenance of Software Models*, pp.343-369, 2012.

[3] E. Eyisi, Z. Zhang, X. Koutsoukos, J. Porter, G. Karsai and J. Sztipanovits, Model-based control design and integration of cyber-physical systems: An adaptive cruise control case study, *Journal of Control Science and Engineering*, 2013.

[4] I. Fey, J. Muller and M. Conrad, Model-based design for safety-related applications, *SAE World Congress 2008*, Detroit, USA, 2008.

[5] S. M. Prabhu and P. J. Mosterman, Model-based design of a power window system: Modeling, simulation and validation, *Proc. of IMAC-XXII: A Conference on Structural Dynamics*, Dearborn, MI, 2004.

[6] M. Conrad, P. Munier and F. Rauch, *Qualifying Software Tools According to ISO 26262*, http://www.mathworks.com/tagteam/61793_CMR10-16.pdf, 2014.

[7] M. Conrad, *Verification and Validation According to ISO 26262: A workflow to Facilitate the Development of High-integrity Software*, http://www.mathworks.com/tagteam/71300_1D-4.pdf, 2014.

[8] M. K. Lee, S.-H. Hong, D.-C. Kim and H. M. Kwon, Incorporating ISO 26262 development process in DFSS, *Proc. of the Asia Pacific Industrial Engineering & Management Systems Conference*, 2012.

[9] *Official Website of the Mathworks: www.mathworks.com.*