

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/279535949>

# Vehicle Control Unit for Drivetrains exclusively from Power Electronics Technology Demonstrators

Conference Paper · June 2015

DOI: 10.1109/ITEC.2015.7165825

---

CITATION

1

READS

3,097

1 author:



**Christian Sültrop**

Fraunhofer Institute for Integrated Systems and Device Technology IISB

7 PUBLICATIONS 7 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



TechReaL [View project](#)



HiBord [View project](#)

# Flexible Vehicle Control Unit for Drivetrains exclusively from Power Electronics Prototypes

C. Sülthrop

Friedrich-Alexander-University Erlangen-Nuremberg

Chair of Electron Devices

Erlangen, Germany

Email: christian.suelthrop@leb.eei.uni-erlangen.de

**Abstract**—Power electronics technology demonstrators for electric vehicle drivetrains are primarily designed to demonstrate the feasibility of novel technical approaches rather than for simple integration into vehicle communication networks. Documentation related to integration issues is often limited. Communication protocols are heterogeneous and tend to change frequently during the development process. Higher level software functions are often not fully implemented in the electronic control unit firmwares. This paper presents the design and implementation of a vehicle control unit software, that is based on a rapid prototyping approach. The software design is flexible and modular enough to cope with the specific characteristics of technology demonstrators. Its features enable to test and recursively integrate these components into a vehicle. The software has been employed to integrate a complete powertrain, exclusively composed of power electronics technology demonstrators, into a converted sports car.

**Index Terms**—power electronics; electronic control unit; vehicle control unit; rapid prototyping; electric vehicle; powertrain; drivetrain

## I. INTRODUCTION

Reducing the number and complexity of electrical, thermal, and mechanical interfaces to a minimum is a current research objective in the field of electric drivetrain power electronics systems. A mechatronic approach to address this goal is to integrate power electronics systems at the physical site of action, i.e. the exact physical location where electric power conversion is needed [1]. Following this idea, the number of electrical interfaces is reduced and AC and DC power lines become as short as possible. Major challenges include small, complexly structured installation spaces and high thermal and mechanical stress. The power electronics systems are often designed with respect to showing the feasibility of novel technical approaches to overcome these issues. Seamless integration into vehicles is not a main objective in this development stage. Even more, the devices are developed in different, unrelated, research projects. Thus, their electronic control units (ECUs) are heterogenous, and often incompatible, in terms of communication protocols and software functions related to vehicle integration.

Three types of experiments are commonly pursued to test and demonstrate drivetrain power electronics systems:

### A. Laboratory and test bed experiments

The unit under test is operated in stationary points of operation to characterize the power electronics in terms of

stationary state system stability, energy efficiency, thermal behavior, etc.

### B. Hardware-in-the-loop experiments

An actual device is coupled with a model based system simulation that is run in real-time on a computer. The mechanical, electrical, thermal, or communication interfaces of the unit under test are stimulated by a computer controlled test bench. The device's reactions are measured and fed back into the simulation. This allows testing the dynamic behavior in a way that is closer to reality than the stationary operation on a standard test bench, but can only be as realistic as the limitations of the underlying system model and the test bench operational range and dynamics permit.

### C. Vehicle integration

Creating prototype vehicles is labourious, but necessary if vehicle components are to be tested in direct interaction in realistic environment conditions. This approach permits to observe effects between units that cannot be modeled in software simulations. Also, only an actual vehicle permits drivers to directly sense the dynamic behavior of drivetrain components in interaction.

Following approach C, an experimental vehicle was created to test the interaction of power electronics technology demonstrators in a realistic environment. Most power electronics systems were initially designed in earlier research projects. This paper describes the design and implementation of a vehicle control unit (VCU) software for the converted ARTEGA GT sports car depicted in Fig 1. The VCU addresses the aforementioned challenges of integrating technology demonstrators into an actual vehicle.

The paper is structured as follows: Section II gives a brief overview of the vehicle topology and the power electronics technology demonstrators that have been integrated into the vehicle. Section III describes the conceptual design of the VCU, which is detailed in Section IV. Section V describes the realized vehicle control functions. Finally, measurement data discussed in Section VI demonstrates the VCU software behavior.



Figure 1. Experimental vehicle, based on a converted ARTEGA GT sports car, during test track experiments.

## II. HARDWARE INTEGRATION

The topology of the experimental vehicle is depicted in Fig. 2. The main drivetrain components are an integrated twin-drive unit and a battery system with an integrated ACDC charger. The drive unit shown in Fig. 3 is highly integrated and combines two permanent magnetic synchronous motors (PMSM), two sets of gears, the drive inverters, and the ECU. The drive inverters are based on inverter building blocks [2]. The mechanic and electronic components are connected to a common liquid coolant circuit. A summary of technical data is given in Table I.

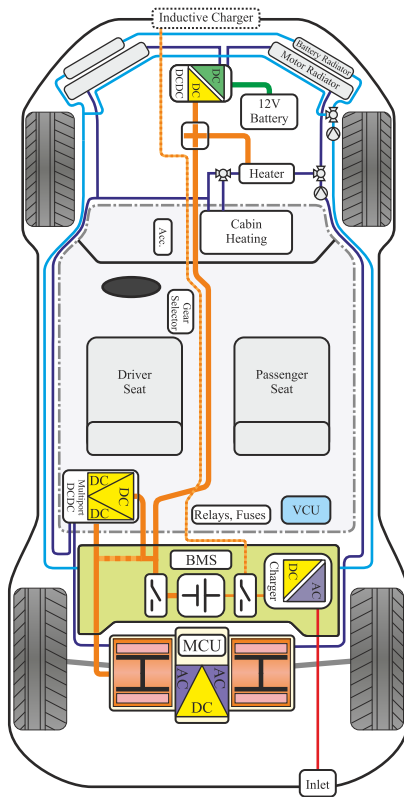


Figure 2. Packaging, HV network, and coolant circuits of the experimental vehicle. The battery system (green) and the integrated twin-drive unit (orange) are located in the vehicle's rear.

The battery system consists of eight battery modules. Each module is equipped with 12 nickel manganese cobalt oxide (NMC) lithium ion pouch cells and a monitoring board. The

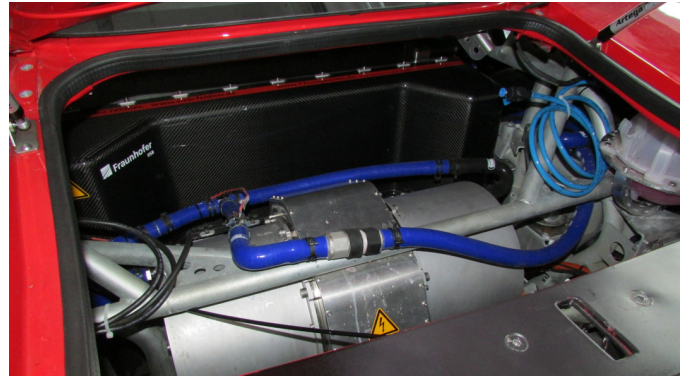


Figure 3. Integrated drive unit with two independent PMSM motors, gearing and modular inverters in the vehicle's rear. The battery system in its black carbon fibre composite case is located behind the drive unit.

Table I  
INTEGRATED DRIVE UNIT SPECIFICATION

Parameter	Value
construction	twin drive with integrated inverters and transmissions
motor type	permanent magnet synchronous motors (PMSM)
maximum power	$2 \times 80$ kW
maximum motor torque (60 s)	$2 \times 230$ Nm
maximum motor speed	10,000 RPM
transmission factor	7:1
controller	field oriented control

monitoring boards measure cell voltages and temperatures and are capable of passive cell balancing. The battery is controlled by a battery management system (BMS). The monitoring boards are connected to the BMS in a galvanically isolated daisy chain. The modules are removable and thermally connected to a liquid cooled framework by a combination of heat spreader sheets and a thermally conductive case. A battery charger and a junction box with distinct power paths for propulsion and charging mode are also integrated into the battery system. The battery system offers multiple connectors for connecting power electronics systems to the high-voltage (HV) traction and charging networks. Thus, future developments can be integrated into the experimental vehicle, such as the inductive charging unit drawn dashed in Fig. 2. Technical data of the battery system is stated in Table II.

A bi-directional multi port HV DCDC converter can be used to regulate the drive unit DC-link voltage, to add additional energy storage capacity, or to connect the vehicle to a DC grid. The converter can be bypassed by the orange HV connection drawn dashed in Fig. 2.

Auxiliary devices include an on-board electrical system DCDC power converter that transports power from the HV traction battery to the low voltage (LV) electrical system, a climatization system with a HV PTC heater, valves to control the coolant flow in the motor and battery circuits [3], and

Table II  
BATTERY SYSTEM SPECIFICATION

Parameter	Value
cell chemistry	nickel manganese cobalt oxide lithium-ion (NMC)
cells per module	12
modules	8
capacity	14 kWh
min/nom/max voltage	260/355/403 V
min/max current	-120/480 A
charger power	3.6 kW

a vacuum pump for braking support. In total, the VCU is connected to 19 power electronics systems, auxiliary units, and original vehicle ECUs.

### III. VEHICLE CONTROL UNIT CONCEPT

The creation of a functional vehicle from the prototypical technology demonstrators described earlier was the primary goal of the powertrain integration project. Incomplete documentation and reduced system maturity are inherent properties of technology demonstrators. The systems were not designed for operating together. Software functions and communication protocols tend to change during the ongoing development process. These properties create the need for a flexible vehicle control unit (VCU), that can be adapted to changes in functions and interfaces quickly. Four fundamental design decisions provide the flexibility needed for the presented VCU concept: First, all higher level vehicle control functions are centralized in a single unit. This decision is similar to the domain controller concept [4], [5], [6]. The concept moves functionality from the ECUs to a central controller, which simplifies the development process as fewer hardware devices and physical data interfaces are involved. Second, presentation of abstractions of the actual units to the higher level functions. Third, no direct communication between ECUs; all communication between units is routed virtually within the VCU. Fourth, the unit abstractions are as self contained as possible, in order to minimize interdependencies between higher level software and unit abstractions.

The software structure resulting from the four design decisions is shown in Fig. 4. Each physical unit is abstracted and represented by a driver module. These modules are on the bottom side connected to the communication interfaces of the VCU hardware and communicate with the physical units. On the top side, they are connected to a virtual data bus that interconnects the modules with each other and with the higher level VCU software. The inner structure of the driver modules is described in Section IV.

The VCU implementation is based on a model-based-design approach. C-code is generated from SIMULINK and STATEFLOW models and compiled in an automated process, which effectively eliminates time consuming errors during code creation. Fig. 5 shows the top level of the Simulink

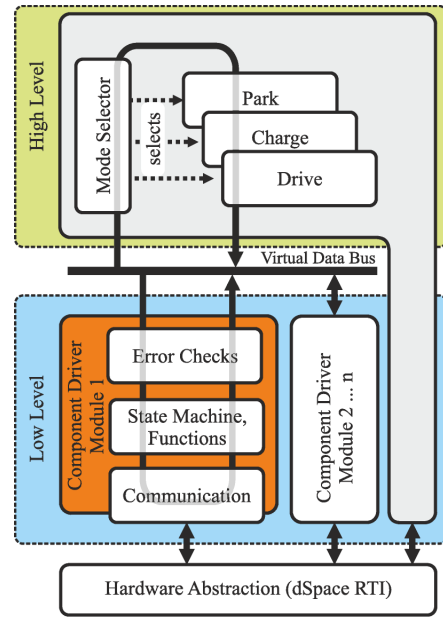


Figure 4. VCU software architecture. Component driver modules abstract the communication between ECUs and higher level VCU functions. Data is shared between high level functions and driver modules via a virtual common data bus.

implementation of the software structure from Fig. 4. The firmware created is run by a DSPACE MICROAUTOBOX that supplies digital and analog interfaces and CAN bus interfaces to connect to the powertrain ECUs and auxiliary units.

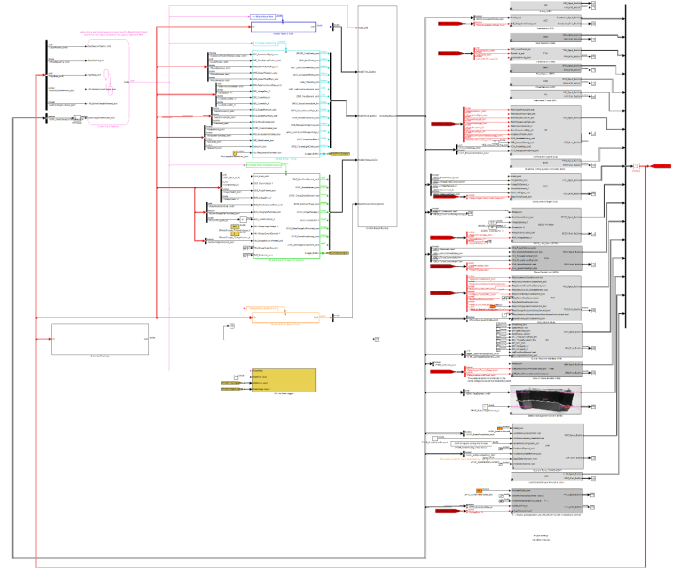


Figure 5. Top level of the SIMULINK VCU software implementation. From left to right: Mode selector (pink), higher level functions for park (blue), drive (cyan), and charge (green), component driver modules (gray).

### IV. DRIVER MODULES

The development process is orientated along the V-model of software development, which partitions the development

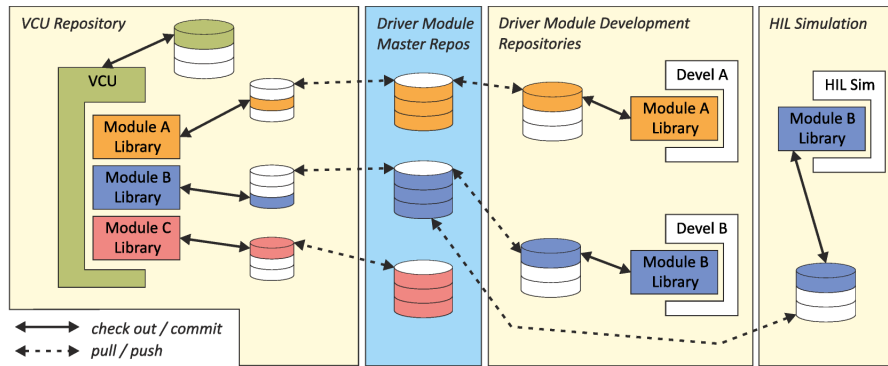


Figure 6. A GIT repository (oval container) exists for each driver module SIMULINK library. It contains the history of all files belonging that belong to a driver module. The modules are pushed/pulled from/to master repositories (short dashed lines). The version checked out (colored oval container) of the same module can be different in different projects. Repository clones can also be used in other projects, e.g. a HIL simulation. Local repositories are synchronized via pushing/pulling to/from master repositories.

process into different phases that build up on another. The problem to solve is gradually broken down and partitioned, while the level of detail is increasing. The software design phase is followed by test and integration phases, which eventually end with the release of the finished software. The effort put in the inevitable recursions that occur in practice should be as small as possible. Therefore, the interfaces to the various ECUs are implemented as modules that can be developed and tested individually. These modules are implemented as SIMULINK libraries. The component driver module interfaces are designed as lean as possible to reduce the complexity of the next higher VCU software level.

Each module is managed in a distinct repository by the GIT version control system (VCS). The VCS provides version history documentation and tracking. More important, it permits to deploy the modules not only in the VCU context, but also for unit based function verification or for test bed experiments, without losing version history. If a driver module is altered due to experiment results, it can be loaded back into the VCU project. Most driver modules were primarily developed in their own minimized development environments, before they were integrated into the main VCU project. This procedure accelerates the development process, as software complexity and compile time are reduced. These minimal environments are especially handy during basic integration tests and bug tracking on ECU level. They can also be used during bench tests. A relay board, which can supply each ECU with power individually, supports this procedure. Fig. 6 illustrates how driver modules stored in master repositories are deployed to the VCU repository, to module specific development repositories, and to HIL simulation repositories for different use cases.

All component driver modules are based on a common structure. The module functions are organized in three layers: Error-checking blocks control incoming and outgoing communication for plausibility and generate error signals if SOAs boundaries are hurt. The calculation blocks implement all primary functions. Finally, the communication block acts as an interface to the VCU's communication hardware. The driver

modules are implemented as SIMULINK libraries, as depicted in Fig. 7.

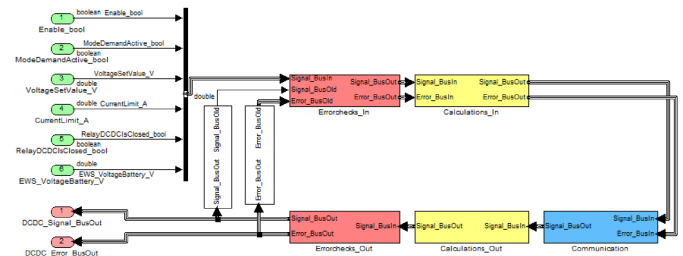


Figure 7. Top level of the HV-to-LV DCDC converter driver module SIMULINK block. All driver modules are divided into error checking (red), calculation (yellow), and communication blocks (blue).

### A. Communication level

The communications block, located on the lowest level of each component driver module, implements the actual interface to the ECU. The CAN interface description defines messages and signals in form of data base container (DBC) files.

Devices that do not have a microcontroller of their own, such as the accelerator, the relay boards, or the vacuum pump, communicate with the VCU via analog or digital I/O ports.

Most driver modules send a condensed set of state variables to the CAN bus, which can be easily recorded with a CAN data logger for analysis.

### B. Function level

A state machine implements the communication protocol of the ECU, defines startup and shutdown procedures, and implements error reactions. The state machine ensures that the component module and the ECU are always in a defined state. It simplifies the communication with the ECU, as it organizes the — often only implicitly defined — communication protocols in a state based form. An example for the BMS is given in Fig. 8. This is especially important if the ECU itself does not implement such features. On this level, also necessary



calculations are located, such as conversion from electrical to mechanical motor speed, calculation of accelerator position from measured voltages, conversion from coolant flow request to pump speed, etc.

Often, technology demonstrator firmware is not functionally complete. Functions that are missing in the ECU firmware are implemented on the function level of the driver modules.

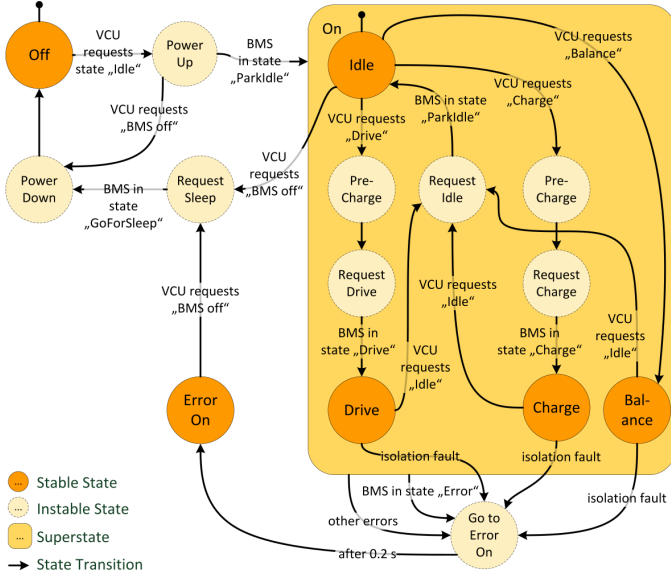


Figure 8. Simplified example of a state machine, taken from the BMS device driver module. Only a few stable states are presented to the higher level functions. In case of unexpected behavior of the BMS ECU, a safe error state is entered.

### C. Safety level

Each power electronics device can only be operated in a limited operating range. This range is defined by electrical, thermal, and mechanical boundaries. Obedience of these safe operating areas (SOAs) is ensured by restricting, statically or dynamically, the set values that are sent to the ECUs. In general, the driver modules ensure safe operation on component level, autonomous of the higher VCU levels, in sense of self-protecting behavior. For instance, the permitted current in a motor is calculated taking into account the temperatures of the motor windings, power transistors, and DC-link capacitors. If one of these temperatures reaches the edge region of its SOA, torque request derating is enabled; derating limits the motor current and thus makes sure that the temperature cannot increase further.

## V. HIGHER LEVEL FUNCTIONS

The higher level functions shown in Fig. 4 implement the vehicle functions that coordinate and control the underlying individual units. Three different global modes of operation were identified: Parking, charging, and driving. In future, more modes can be added, such as different charging modes for DC and inductive charging. The global modes are implemented

independently from each other, which reduces the software complexity. A mode selector interprets the driver commands received from the human machine interfaces and selects the appropriate mode. Locking mechanisms ensure that transitions between modes are only possible in predefined system states, such that global modes are decoupled and mutual interference is impossible.

## VI. MEASUREMENT AND VALIDATION

Extensive data logging has been performed during vehicle tests on both roller dynamometer and test track. Data recorded during a dynamometer test is presented in the following to exemplarily demonstrate the functionality of the VCU control mechanisms. For orientation, vehicle speed and travelled distance are shown in Fig. 9.

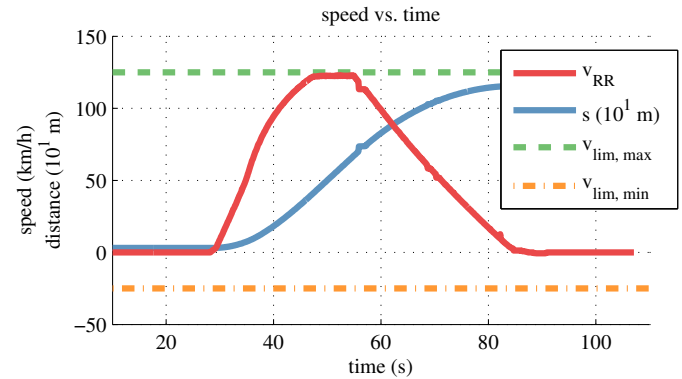


Figure 9. Vehicle speed (red) and travelled distance (blue). The VCU ensures that the speed is always between -25 km/h and +125 km/h.

Fig. 10 (top) shows the states of the high level VCU state machines. When the driver activates the drivetrain at  $t = 8$  s, the global mode switches from zero (parking mode) to ten (drive mode). This activates the drive mode state machine, which goes through several states in which the drivetrain systems are started up. When it reaches State no. 22, the vehicle is ready to go. The internal state of the BMS ECU and the state of the BMS driver module inside the VCU software are shown in Fig. 10 (bottom). The BMS module state machine is activated by the drive mode state machine. It goes through a few states, which control the startup procedure of the BMS. This startup includes checking the isolations state and going through the HV contactor closing procedure. The BMS driver module state transitions are requested by the VCU drive mode based on the gear lever position. If the BMS driver module would detect any errors, it would open the contactors and inform the higher level software by going into error mode. The shutdown procedure starting at  $t = 95$  s is analog to the startup procedure.

Fig. 11 shows the torque request, some torque limitations due to SOAs, and the actual motor torque: The driver requests torque by pressing the accelerator pedal. A pedal position above 30 % means a positive torque request, a position below 30 % means negative torque. The actual motor torque follows the request until  $t = 34$  s. Until this time, all drivetrain parameters are within their SOAs. Thus, the requested torque

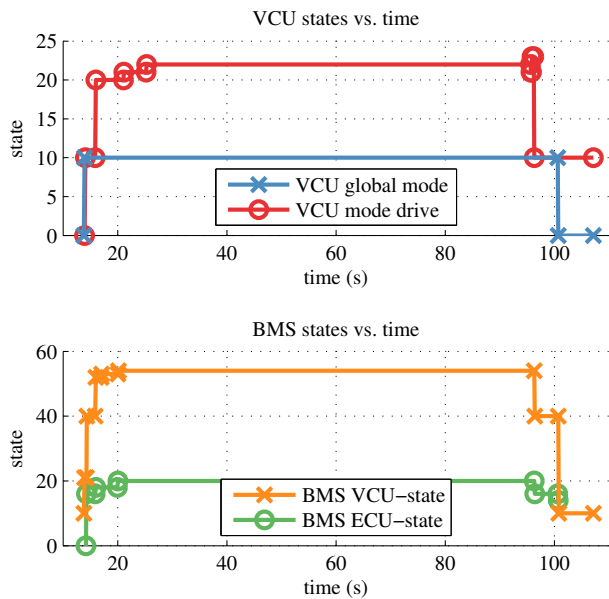


Figure 10. Top: Selected global mode (blue), state of drive mode controller (red); Bottom: State of BMS ECU (green), state of BMS driver module (orange).

is not derated by any derating stage within the high level VCU or within the motor driver module. Around  $t = 34$  s, the pedal is pressed further, which increases the torque request. As the vehicle speed is below the defined top speed of 125 km/h, the speed-based torque limiter is not activated. The increased battery current, though, leads to a battery voltage drop. To ensure that the voltage stays into its SOA, the request is reduced in the battery limiter stage (orange). The actual torque is below the request at this time, as the motors cannot deliver full power at the current speed. Around  $t = 43$  s, the vehicle speed reaches its SOA edge region. The driver request is still 100 % at this time, but the request is gradually derated to ensure that the top speed limit is not exceeded (purple).

The accelerator pedal is lifted at  $t = 54$  s. The torque request at zero percent pedal position is -20 Nm. This value is requested until the vehicle speed is below five km/h. Between five km/h and zero km/h, the torque request is reduced to zero Nm to make sure the vehicle cannot turn reverse due to negative torque.

## VII. CONCLUSIONS

An architecture for a vehicle control unit software that addresses the special challenges imposed by the creation of an electric vehicle powertrain that is composed solely from technology demonstrators is presented. The software is separated in different levels that help to structure the software development process.

The implementation with MATLAB-SIMULINK and a rapid prototyping hardware platform prove expedient: Quick reactions to issues with ECUs were always possible and ECU software functions that were dysfunctional could be replaced by software running on the VCU with little effort compared to the

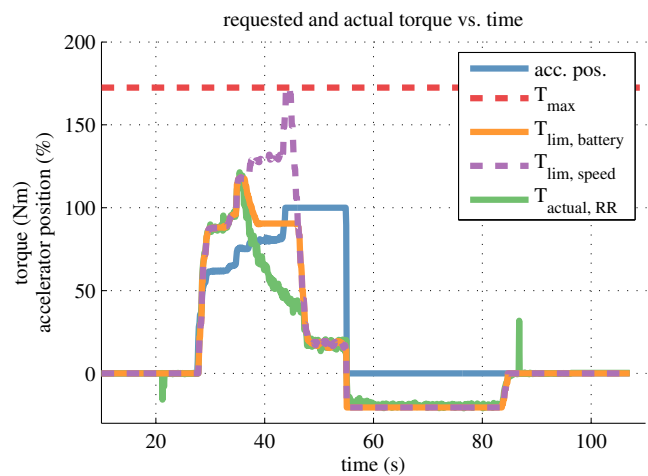


Figure 11. Accelerator position (blue), torque request after speed limiter stage (purple), torque request after battery voltage limiter stage (orange), and actual motor torque (green) of the rear right motor.

effort that fixes in the often hand-coded ECU firmware would have required. The driver modules were not only integrated into the VCU, but were also used in the preceding component testing and function verification phases. Besides the sports-car project described in this paper, the VCU software has proven its versatility in a subcompact car conversion project.

The experimental vehicle is currently being tested on both roller dynamometer and test track in preparation of road approval. Afterwards, the vehicle will be used for system level performance and efficiency experiments. Future plans also include the integration of additional power electronics systems, such as an inductive charging system and a DC charging system.

## REFERENCES

- [1] M. März, A. Schletz, B. Eckardt, S. Egelkraut, and H. Rauh, "Power electronics system integration for electric and hybrid vehicles," in *Proc. Integrated Power Electronics Systems (CIPS), 6th International Conference on*, Mar. 2010.
- [2] M. Hofmann, A. Schletz, K. Domes, M. März, and L. Frey, "Modular inverter power electronic for intelligent e-drives," in *Proc. Electric Drives Production Conference (EDPC), 2nd International*, Oct. 2012.
- [3] H. Rauh, M. März, L. Frey, and C. Sülthrop, "Energy optimized implementation of climatization systems in electric vehicles with integrated drive components," in *Conference on the Future of Automotive Technology*, Munich, 2014.
- [4] G. Gut, C. Allmann, M. Schurius, and K. Schmidt, "Reduction of Electronic Control Units in Electric Vehicles Using Multicore Technology," in *Multicore Software Engineering, Performance, and Tools*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, V. Pankratius, and M. Philippsen, Eds. Berlin, Heidelberg: Springer, 2012, vol. 7303, pp. 90–93.
- [5] D. Reinhardt and M. Kucera, "Domain controlled architecture: A new approach for large scale software integrated automotive systems," in *Proc. Third International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2013)*, Barcelona, Feb. 2013.
- [6] S. Gandhi and S. P. Brewerton, "Techniques and measures for improving domain controller availability while maintaining functional safety in mixed criticality automotive safety systems," SAE International, Warrendale, PA, Tech. Rep. 2013-01-0198, Apr. 2013.