A Hardware and Software Framework for Cognitive Automobiles

Moritz Werling[1], Matthias Goebl[3], Oliver Pink[2], and Christoph Stiller[2]

[1] Institute of Applied Computer Science/Automation

[2] Institut für Mess- und Regelungstechnik

University of Karlsruhe (TH), 76128 Karlsruhe, Germany

[3] Institute for Real-Time Computer Systems

Technical University of Munich (TU), 80333 Munich, Germany

`moritz.werling@iai.fzk.de`   `goebl@rcs.ei.tum.de`   `{pink, stiller}@mrt.uka.de`

Abstract— This paper describes the hardware and software framework of AnnieWAY, an autonomous vehicle successfully competing at all qualification stages up to the finals of the *DARPA Urban Challenge 2007* competition. Besides the hardware premises for by-wire steering, braking, throttle control and sensors, two frameworks for high-level decision making and low-level control are described in detail. Furthermore, this contribution points out AnnieWAY's safety features for both unmanned competition runs and manned testing. The system proves to be flexible, expandable, and robust at the same time.

Index Terms— hardware framework, software framework, autonomous, cognitive automobile, AnnieWAY, DARPA Urban Challenge

I. INTRODUCTION

In 2006 the German Research Foundation (DFG) started the Transregional Collaborative Research Center 28 - *Cognitive Automobiles* interconnecting research of Universities and research centers in Karlsruhe and Munich. Its main objective is the development of new methods for machine perception and machine behavior for automobiles - not only in theory but also in extensive simulation and in practice. For the latter, an experimental vehicle based on a VW Passat platform has been equipped with actuators, computers, microcontrollers and sensors (see also [4], [5]). In the *DARPA Urban Challenge 2007* competition, the vehicle was put to a first test by Team AnnieWAY, a spin-off of the Collaborative Research Center, and made it into the finals.

This paper gives a detailed description of the hardware and software framework that the different algorithms [6] for the competition were integrated into. The focus of the framework lies on the clarity of the architecture, the decoupling of components, interchangeability and expandability of algorithms, and by-passing of the hardware components for simulation and in-the-loop testing purposes.

The remainder of this paper is organized as follows: After a brief overview on the hardware components in Sec. II, the system architecture is presented in Sec. III which the interaction of the hardware and software components is based on. In Sec. IV the controller framework is described that transfers high-level decision into low-level actuator commands. As AnnieWAY's algorithms only communicate via a real-time database, Sec. V introduces its main features.

Sec. VI concludes this contribution with a description of the realized safety features.

II. HARDWARE COMPONENTS

A. Vehicle platform and actuators

For the sake of reliability and time constraints a Volkswagen Passat Variant 2.0 FSI was chosen to be directly equipped with by-wire steering, braking, and throttle control by the manufacturer (Fig. 1). This includes amongst others the installation of

- an electric steering motor
- a new prototype power brake unit
- an accelerator value simulator
- a larger alternator
- a LIN interface (e. g. turning lights)

All components are unobtrusively integrated into the car interior lining and communicated with through a CAN gateway which provides additional information available on the series CAN-BUS. The manufacturer's safety system also allows for a quick and safe changeover from automatic to manual control and vice-versa.

Since pulling into and backing out of a parking spot as well as K-turns are required in the competition, an additional longitudinal actuator was integrated into the selector lever case of the automatic transmission in order to engage drive, rear, and neutral gear.

B. Computing system hardware

The on-board computing system hosts all software for perception and decision making. It has to provide sufficient computing power and at the same time high data rates for sensor data acquisition and low latencies for interprocess communication. To fulfill these requirements a multicore multiprocessor computer system using opteron processors has been chosen. All opteron processors are interconnected by HyperTransport (HT) at a speed of 3.2 GByte/s each. This configuration can be regarded as a cluster-in-a-box, yet infrastructure components like disks and power supply are needed once only. Due to the employed high-efficent (HE) CPUs, the system requires about 350W in total from the DC/DC power supply.

① GPS antennas	④ alarm	⑦ warning lights
② 2D-lidar	⑤ e-stop buttons	
③ 1D-lidar	⑥ turn signal	

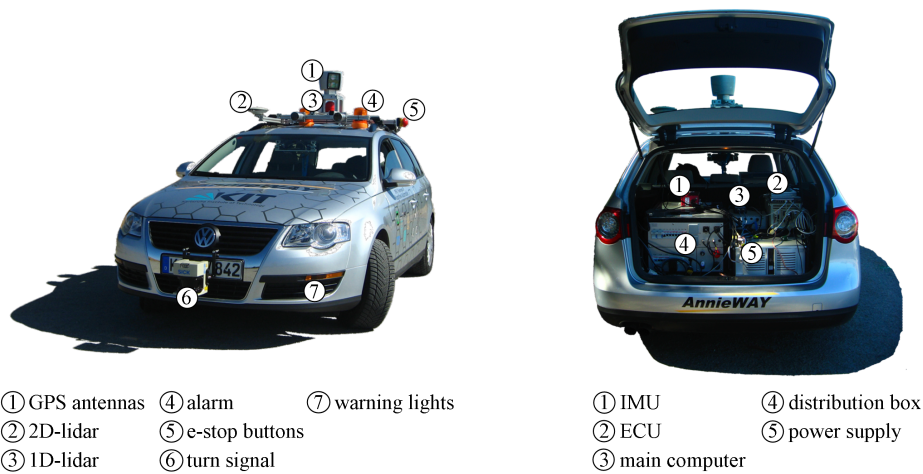| ① IMU	④ distribution box
② ECU	⑤ power supply
③ main computer	

Fig. 1. Hardware components

For persistent storage, system data and logging data are distinguished: The system files that seldom undergo changes are kept on a shock-proof flash disk. In contrast, for logging purposes a very fast raptor harddisk is used, providing 44-91 MB/s and sustaining accelerations up to 3g. For the unlikely case of a disk failure, another protection feature comes into effect: The disks are connected by hot-plug capable serial ATA (SATA) links. In contrast to parallel ATA, a hardware failure does not disturb the mainboard hardware function, but generates only software signals. In our arrangement, if the logging disks fail or the connection gets loose, only the logger process dies, leaving the rest of the software running and keeping the vehicle on course.

To cool the computing system, cold air from the standard air condition system is supplied, filtered by tissue-pads before entering the computer case.

C. Real-time microcontroller and distribution box

Since high level decision making runs at soft real time at best, a hard real time capable rapid prototyping environment, namely dSpace's Autobox, is used for the vehicle's lateral and longitudinal control. In addition to the processor board DS1005, the Autobox' setup provides reliable interface boards for digital I/Os (DS4002), CAN (DS4302), and ethernet (DS4502Eth).

The Autobox' digital I/Os are connected to the distribution box, which converts the TTL level into 12V via relay and provides enough space for reliable connectors.

D. DGPS/INS and wheel speed sensor

The *RT3003* Inertial and GPS Navigation System (IMU) is a six-axis inertial navigation strap down system that incorporates an L1/L2 RTK GPS receiver for position and a second GPS receiver for accurate heading measurements. Additional wheel speed input taken directly from the Passat's series wheel speed sensor in combination with the OmniSTAR correction signal, the system delivers better than 0.02m positioning and $0.1°$ heading under dynamic conditions.

The wheel speed information is directly taken from the series

sensor by tapping the signaling line and processing it to the required TTL specification via a simple electric circuit.

E. 2D and 1D lidars

AnnieWAY's perception mainly relies on three laser scanners: The *Velodyne HDL-64E* comprises 64 lasers that are mounted on upper and lower blocks of 32 lasers each and the entire unit spins at 10Hz. It inherently delivers a $360°$ horizontal and a $26.8°$ vertical field of view by providing about one million points per second with a angular resolution of $0.09°$ horizontally and 0.05m in distance. In addition to the 3D position the sensor signals reflectivity of each point. This allows to acquire information from visual textures, like lane marker information, similar to monoscopic image analysis.

Two additional *Sick LMS 291* 1D lidar scanners are mounted horizontally both on the front and rear bumper in order to observe obstacles that are too close to the vehicle for tracking by the Velodyne lidar. Since small parking spots where not an issue in the competition, the two Sick laser scanners were dismounted in the finals.

F. E-stop system and warning devices

To ensure safe operation in autonomous mode, DARPA required every team to implement several safety features such as audible and visible warning signals and emergency stop devices. Key part of the emergency stop system is the *Omnitech Robotics Darpa Grand Challenge Safety Radio (DGCSR)* [1] which delivers two signals to remotely control the vehicle. The RUN/PAUSE signal switches between manual and autonomous operation. When in PAUSE mode, the vehicle decelerates to a safe stop and waits for reactivation. When in RUN mode, audible and visible warning signals are required.

Additionally, the DGCSR provides an ENABLE/DISABLE signal for emergency stop purposes which can also be triggered manually by two emergency stop buttons at both sides of the vehicle. When in DISABLE mode, the vehicle decelerates to a safe stop and the engine is shut off.

1081

To ensure safe operation even under software system crash, a hardware-only solution has been implemented. Our solution makes use of the manufacturer-equipped immobilizer system, which immediately switches off the engine when no ignition key is present. Instead of physically removing the key, a simple relay circuit is used to cut its power supply.

For emergency braking, another relay triggers the manufacturer-equipped electronic parking brake, which has a built-in emergency stop feature. When triggered at higher speeds, it automatically activates all vehicle brakes via the antilock brake system until the button is released or the vehicle has come to a complete stop [8]. As soon as the vehicle has come to a stop, the parking brake is activated automatically, safely keeping the vehicle in place even at slopes.

To keep the brake booster operational during emergency braking, shutting off the engine is delayed by one second. Fig. 2 shows the response of our vehicle speed after triggering the emergency brake system.
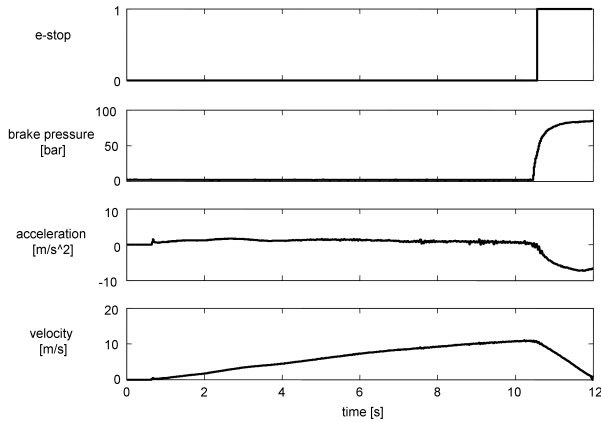

Fig. 2. Emergency brake maneuver

G. Power unit

In order to handle the increased power consumption of the vehicle, a larger alternator (220A) and a power unit was installed. The latter comprises 2 batteries, DC/AC and DC/DC converters, a charging device, fuses, and buffer condensators. With this installation AnnieWAY's computing system as well as actuators and sensors are independent of the car battery, even for hours of testing with a shut down engine.

III. System architecture and data transmission

In order to avoid extensive cabling between sensors, controllers, and actuators, a BUS system can be found in every modern vehicle. For the same and other reasons a real-time data base (RTDB) as described in Sec. V, has been introduced. Fig. 3 gives an overview on the complete system with emphasis on the inherent data transfer.

The flow of information from the sensors to the actors for vehicle control is shown. The LIDAR and IMU sensors are connected to the computing system with a 100 MBit/s

ethernet connection. The raw data is stored into the RTDB with the help of dedicated I/O modules. All algorithms from obstacle recognition to decision making [6] are executed on the computing system. The resulting path along with velocity information and commands like turning signals etc. are stored in the RTDB and immediately transmitted by a UDP bridge to the Autobox via ethernet. In turn, the Autobox sends vehicle sensor data paired with controller status information back.
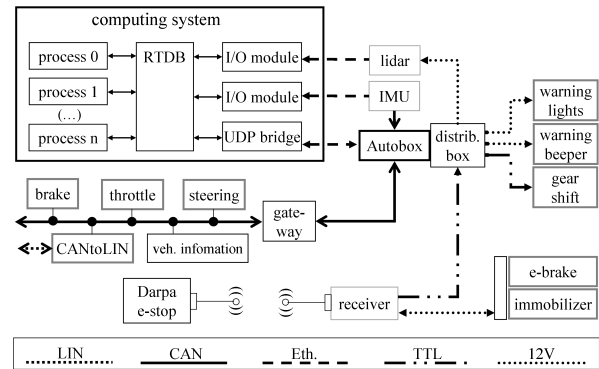

Fig. 3. System architecture and information flow diagram

The controller framework on the Autobox evaluates the received data, provides it to the respective controllers, and transfers their output to CAN and TTL signals as described in Sec. IV. The CAN signal is sent to a gateway, which filters the messages for safety reasons. The described hardware components of Sec. II are connected to the gateway CAN BUS as well as the CANtoLIN adapter handling head lights, wipers, turning signals, etc. The rest of the components, such as the beeper, the hazard lights and the DARPA e-stop system are connected to the distribution box controlled by the Autobox.

IV. Real-time controller framework

Since there is also a scientific focus on dynamic vehicle control, a universal real-time framework has been implemented in a MATLAB/SIMULINK environment as depicted in Fig. 6 which different control algorithms can be integrated into. The framework is set up in a way that algorithms implemented in Simulink or C can be validated online and offline as a matter of a mouse click without any modifications. For offline simulations the physical vehicle is automatically substituted by a veDYNA model [7], a highly accurate simulation of vehicle dynamics.

The framework comprises of
- a RTDB-interface,
- interfaces to the actuators and vehicle sensors,
- cascaded control algorithms,
- safety and debugging features.

Through the RTDB-interface information about the desired vehicle movement as well as LIN commands (e. g.

1082

turning signal) is transferred to the Autobox. In turn, sensor information from the vehicle's CAN BUS is provided to the RTDB.

The cascaded control algorithms for steering, braking, acceleration and gear shifting (see Figs. 4 and 5) take control over the hardware communication and enable the longitudinal and lateral controllers to simply use steering angle and acceleration as control output.
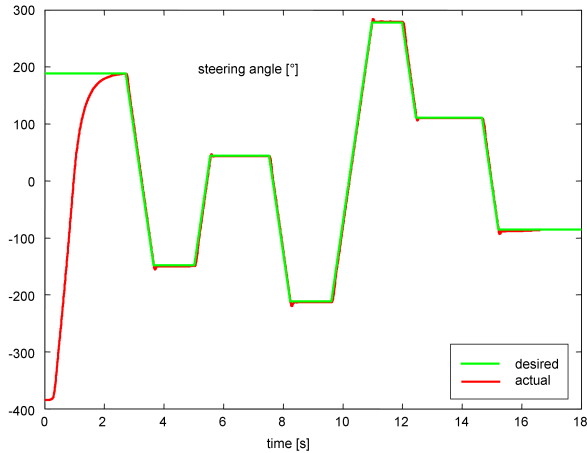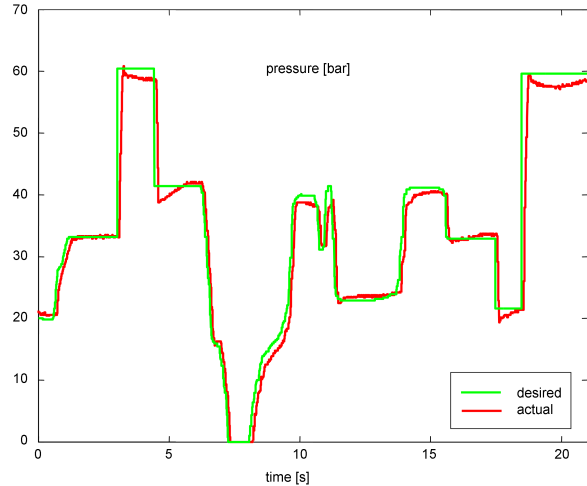

Fig. 4.   Steering control

Fig. 5.   Brake control

V. REAL-TIME DATABASE

The real-time database for cognitive automobiles (KogMo-RTDB)[2] acts as the central communication framework for high-level decision making on the computing system. Every process is connected only to the RTDB and to no other process. For interprocess communication the following method is applied:

- All processes have at least one logical database connection

- Every set of data is packed together into an "object", the smallest unit handled by the RTDB
- The sending process inserts an object with the relevant data into the RTDB
- The receiving process uses the RTDB-API to find the object and retrieve its data
- Whenever the sender has new data available, it immediately updates the object
- The receiver waits for new data and is informed as soon as it is submitted
- The data is automatically buffered within the RTDB for an adjustable time interval, so it is possible to retrieve older data, e.g. to combine lidar data points with the correct GPS-position at the time the measurement has been performed
- Tracked obstacles are inserted as individual objects, updated with each lidar rotation and removed when they are lost

Due to the efficient implementation of the real-time database, it takes only $8.3 \mu s$ to update an object and $6.4 \mu s$ to retrieve it. The full communication from one process to another takes $29.6 \mu s$ (see also [2], [3]). Due to the low overhead, a maximum of computing power is left for the perception and decision processes (AI modules).

A. Simulation and logging

Using the KogMo-RTDB architecture facilitates the exchange of arbitrary software modules. As long as the exchanged modules write the same objects into the RTDB, the other modules will not notice the difference. To test the AI modules, simulation modules were introduced that compute the dynamic motion of the own vehicle according to the low-level lateral and longitudinal control strategy and write the result into the vehicle status and position objects that would be written by the Autobox UDP bridge and the I/O processes from the GPS receiver. With this setup, multiple autonomous vehicles with multiple decision making processes can be simulated simultaneously as well as joystick driven cars, testing the correct interaction.

The centralized data storage offers the opportunity to record all data together. With the help of the RTDB recorder all objects can be logged to disk. Whenever a new object is created, destroyed, or an existing object is changed, the new state is recorded, including an absolute timestamp with a nanosecond resolution. Afterwards, the data log can be replayed in real-time or with a different speed into another RTDB, giving several options for simulation:

- By watching all replayed objects within the database, performance measurements for the involved modules can be taken.
- Only a selected set of objects can be replayed, so an algorithm can be tested multiple times under equal conditions.
- The results of a reworked algorithm can be recorded again together with the original source data.
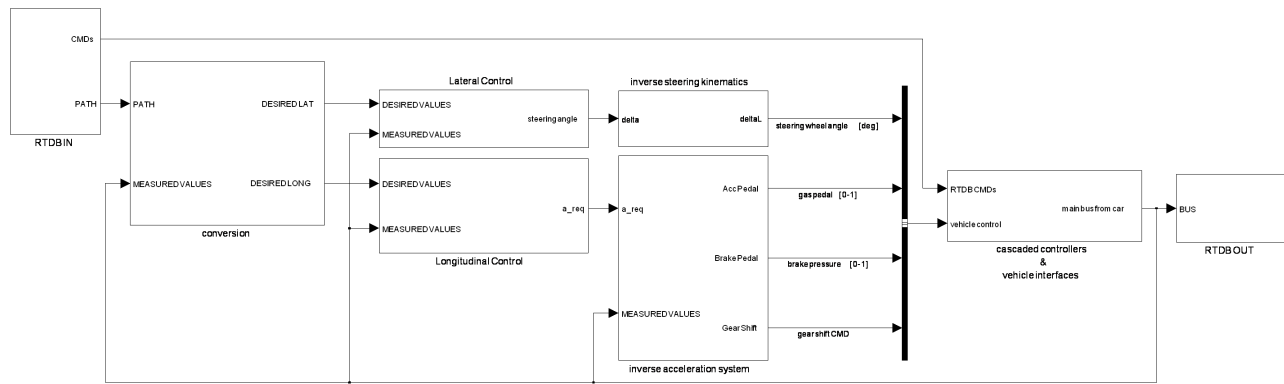- As raw data is also recorded, parameter modifications can be tested offline.

1083

Fig. 6.   MATLAB/SIMULINK controller framework

- The substitution of real-world sensors with the according simulation model is possible.

VI. SAFETY CONCEPTS

The most frequently activated safety measure that proved an important feature for testing has been the seamless switching between autonomous mode and manual interaction by a safety driver. As soon as either the gas or the brake pedal is applied, or the deviation between desired and measured steering angle exceeds a certain threshold, the system automatically switches to manual mode. To ensure safe operation when no safety driver is present, such as in the actual competition, several additional safety measures were implemented.

A. Manual hardware override

The manufacturer's emergency system allows a safety driver to regain control over the vehicle in two different ways: First, all actuators are designed so that they can always be overridden by a human operator. In case this is not sufficient, there are two easily accessible emergency buttons - one in the center console and another one next the brake pedal - which completely switch off all actuators and return control to the driver.

As this system was already integrated by the manufacturer, it is highly safe and reliable. In manual control, operation does not differ from a series-production car, so the vehicle can be used legally on the street.

B. Hardware emergency stop

For operation without a safety driver, a remotely controlled emergency system based on the DARPA-supplied Omnitech DGCSR is used. When activated, the vehicle has to come to a complete stop and, according to DARPA rules, shut off the engine. Our solution is based on the electronic parking brake and the electronic immobilizer system. This solution makes use of vendor-supplied components only, thus it does not need any additional hardware except for some relays and a time delay circuit and is therefore highly reliable. As another advantage, the vehicle can easily be recovered from DISABLE mode by simply removing the ignition key and restarting the engine.

C. Software emergency stop

In addition to the hardware emergency stop, the DGCSR signals is delivered to the Autobox, which itself also triggers the brakes via the CAN BUS when receiving a PAUSE or DISABLE signal. Thus even in the unlikely case of an error in the hardware emergency stop, the Autobox would ensure that the vehicle comes to a safe stop. Additionally, if no valid data is received from the main computing system for a certain amount of time, the Autobox internally switches to PAUSE state and thus brings the vehicle to a controlled stop. As soon as the main computer is sending valid data again, the system switches back to RUN mode and the vehicle continues its mission.

D. Software monitoring

In case of a detectable software misbehavior, it is safer for the unmanned vehicle to stop itself automatically rather than being ultimately e-stopped by the remote control, which inevitably leads to a disqualification in the competition. So a software monitoring system was designed to detect failures and to bring the vehicle to a safe stop. The monitoring system in Fig. 7 consists of several components:

1) Data watchdog: It indirectly watches the processes by looking at the data objects they write into the RTDB. For this analysis the watchdog supervises only one connection to the RTDB. The monitored processes do not have to fulfill any prerequisites, like dedicated debugging ports or the like. By looking at the *commit timestamp* of every object the watchdog can decide whether a process is able to meet its deadlines. If a hardware sensor fails, the I/O process (shown in Fig. 3) ceases from updating its sensor object, and the failure is observed by the watchdog. By looking at all *process objects* the watchdog can notice dead processes and restart them. In case of the death of a process, its objects are preserved for a predefined time (e.g. 5 seconds), so the data could be used for a possible emergency maneuver along the

1084

computing system

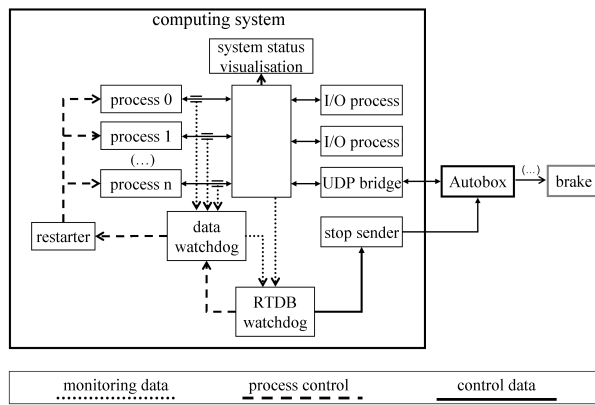monitoring data · · · · · · · · · process control – – – – – control data ——————

Fig. 7.   Monitoring system overview

previously calculated path. By adding object-type specific modules, the watchdog could look into the data part of objects and base its emergency reactions on semantic checks.

2) Process restarter: This component is used by the data watchdog to restart individual processes. At the system startup, it initially starts all processes. This also includes a full automatic start at a spurious reboot in case of a short power failure.

3) Stop sender: In case of serious troubles, e. g. if the immediate restart of a process fails, the vehicle has to come to a safe stop as soon as possible. For reliability, a separate module sends an emergency brake signal to the Autobox as outlined in Sec. II-F and does neither depend on the UDP bridge nor the RTDB, which could potentially be the reason for the failure.

4) RTDB watchdog: Also, the unlikely case that the RTDB or the data watchdog crashes has to be treated. The RTDB watchdog is independent from the RTDB. It periodically tries to connect to the RTDB. It also checks for the data watchdog. If a check fails, the RTDB watchdog immediately stops the whole software system and triggers an emergency brake maneuver using the *stop sender*. If the data watchdog is unable to restart a process or the maximal amount of restarts is reached, it terminates itself, likewise causing an emergency stop. After the stop, the whole software system is restarted.

VII.   Results and Conclusions

The proposed framework has been implemented in the vehicle base and has been extensively tested in simulations. In particular, the seamless migration capabilities of software from simulation to in-the-loop testing and to on-vehicle operation without necessity for any software changes was found to expedite the engineering process and increase the robustness of the experiments. AnnieWAY has successfully competed in all qualification stages where it demonstrated its ability to autonomous driving in urban traffic including the following:

- driving in lane obeying speed limits and safety distances to preceding vehicles
- passing of static obstacles
- u-turn and k-turn
- compliance with legal order of precedence at intersections
- obstacle field and parking lot navigation
- dynamic rerouting
- merging into moving traffic
- emergency braking maneuvers

AnnieWAY was among the 11 finalists that were selected by DARPA. In the finals the vehicle navigated through a large portion of the terrain and stayed among the few finalists that drove collision-free. It finally came to a software deadlock situation that still lead AnnieWAY into a safe mode but caused its stop. That situation had required rebooting of the main computer which was not admitted. Although the competition revealed some room for improvements of the different algorithms, AnnieWAY's hardware and software framework fulfilled all requirements in terms of simplicity, flexibility, expandability, and robustness and will not be significantly modified for future research activities.

VIII.   Acknowledgements

The authors gratefully acknowledge the collaboration of all partners from University of Karlsruhe (TH), Technical University of Munich (TUM), and University of the German Federal Armed Forces, Munich. This work would not have been possible without the ongoing research of the Transregional Collaborative Research Centre 28 *Cognitive Automobiles*. Both projects cross-fertilized each other and revealed significant synergy.

References

[1] DARPA. *Grand Challenge Safety Radio - User Manual*, 2.3 edition, July 2007.
[2] Matthias Goebl and Georg Färber. A Real-Time-capable Hard- and Software Architecture for Joint Image and Knowledge Processing in Cognitive Automobiles. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 734–740, 2007.
[3] Matthias Goebl and Georg Färber. Eine realzeitfähige Softwarearchitektur für kognitive Automobile. In K. Berns and T. Luksch, editors, *Autonome Mobile Systeme 2007*, Informatik Aktuell, pages 198–204. "Springer-Verlag", October 2007.
[4] J. Schröder, U. Müller, and R. Dillmann. Smart Roadster Project: Setting up Drive-by-Wire or How to Remote-Control your Car. *Intelligent Autonomous Systems 9: IAS-9*, 2006.
[5] Christoph Stiller, Georg Färber, and Sören Kammel. Cooperative cognitive automobiles. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 215–220, Istanbul, Turkey, June 2007.
[6] Christoph Stiller, Sören Kammel, Benjamin Pitzer, Julius Ziegler, Moritz Werling, Tobias Gindele, and Daniel Jagszent. Team AnnieWAY's autonomous system. In Gerald Sommer and Reinhard Klette, editors, *Robot Vision, Lecture Notes in Computer Science 4931, Springer Berlin, Heidelberg*, pages 248–259, Auckland, New Zealand, February 2008.
[7] TESIS DYNAware GmbH, Baierbrunner Strae 15, 81379 Munich, Germany. *veDYNA - Professional Real-Time Vehicle Dynamics Simulation Environment*, 2007.
[8] Volkswagen. *VW Passat - User Manual*, November 2006.

1085