Abschlusspräsentation 25.08.2008

Projektgruppe 522: AutoLab

Eine Experimentierplattform für automotive Softwareentwicklung

technische universität dortmund

Fakultät für Informatik

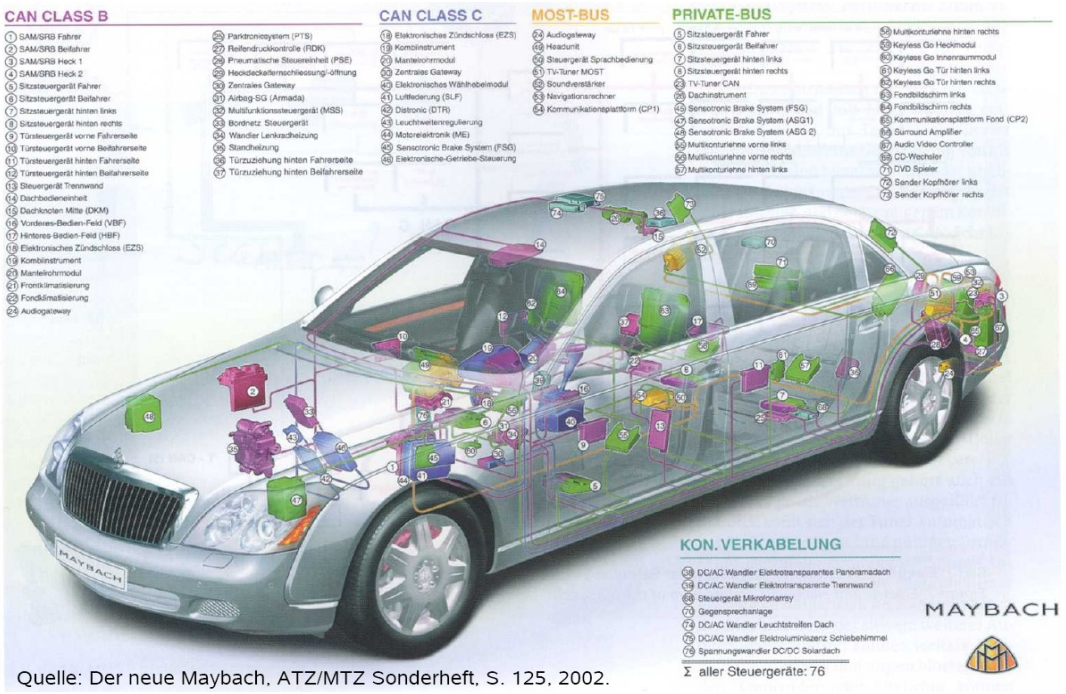
Arbeitsgruppe
Eingebettete Systemsoftware

Inhalt

- Projektbeschreibung und Team der PG522: AutoLab
- Arbeit der Projektgruppe
  - Fahrzeugnetz
  - Steuer- und Messumgebung
  - Benutzerschnittstelle
- Ziele der PG
- Mögliche Erweiterungen

Einleitung

- Moderne Fahrzeuge enthalten viel Elektronik und Software -> sind für Informatiker interessant
  - Kommunikation über Busse, insbesondere CAN-Bus
    - Nachrichten, Priorisierung, Laufzeiten, IDs...
  - Mehrere Steuergeräte als integrierte, spezialisierte Rechner
    - Codegröße
    - Effizienz
    - Energie-verbrauch



Quelle: Der neue Maybach, ATZ/MTZ Sonderheft, S. 125, 2002.

Das Projekt

- AutoLab: Eine Experimentierplattform für automotive Softwareentwicklung
- Entwurf und Inbetriebnahme eines Laboraufbaus eines typischen **Fahrzeugnetzes** auf Basis eines CAN-Busses
- Einbindung in aktuellen wissenschaftlichen Kontext
  - Forschung im Bereich automotive Systemsoftware
- Anwendungsbeispiele
  - Messungen von z.B.
    - Lastverteilung
    - Busauslastung
    - Energieaufnahme

Das Projekt - Beispielanwendung

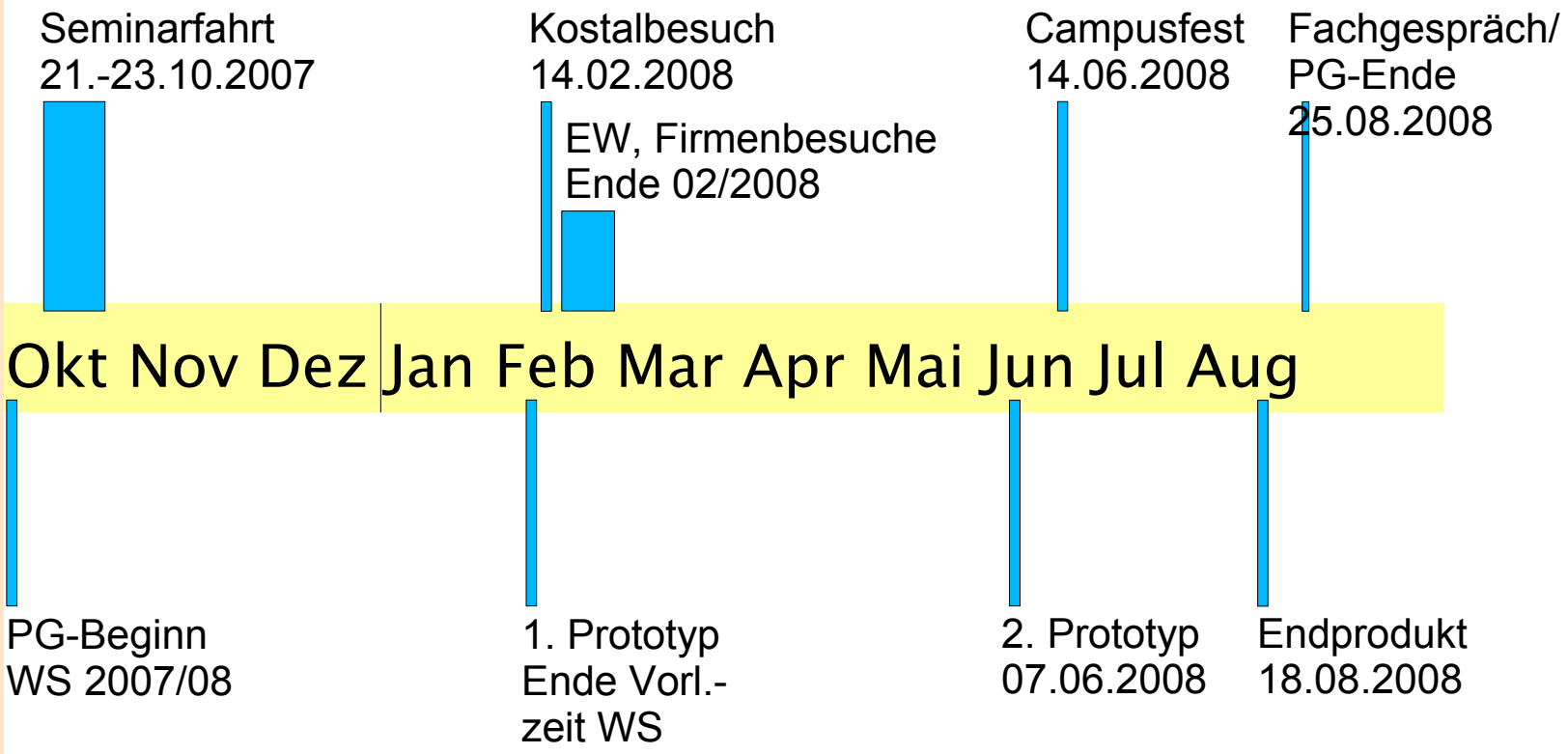
- Beispielanwendung (aus Einzelvorstellung):
  - Messung der Energieaufnahme und Busauslastung bei wechselseitig aktiven Knoten

Das Team

- 12 Studierende
  - Sabrina Hecke
  - Alexander Berger
  - Oliver Botschkowski
  - Stephan Braun
  - Florian Hohnsbehn
  - Christian Horn
  - Gregor Kaleta
  - Boris Konrad
  - Sebastian Kosch
  - Matthias Meier
  - Robert Neue
  - Thomas Romanek
- 4 Betreuer der Arbeitsgruppe ESS am Lehrstuhl 12
  - Prof. Dr.-Ing. Olaf Spinczyk
  - Dr. Michael Engel
  - Horst Schirmeier
  - Jochen Streicher

Zeitlicher Ablauf der Projektgruppe

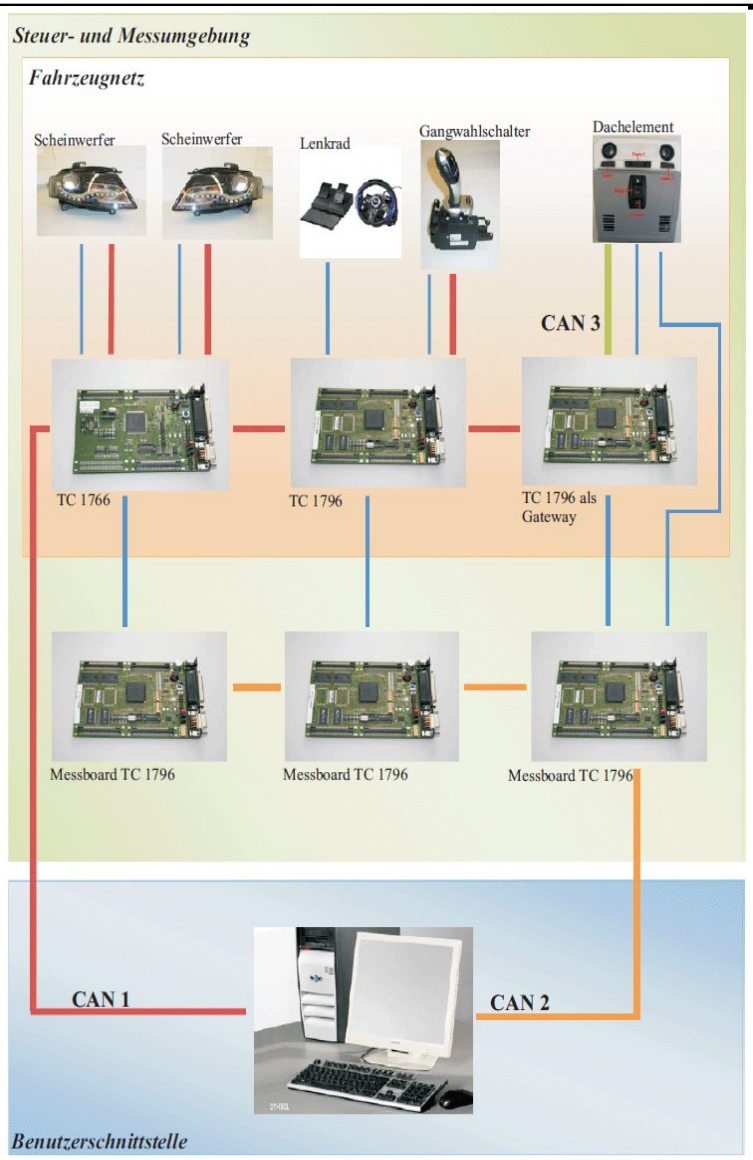
- WS 2007/2008 und SS 2008
- Iteratives Projektmanagement: 3 Prototypen



Eigene Anforderungen

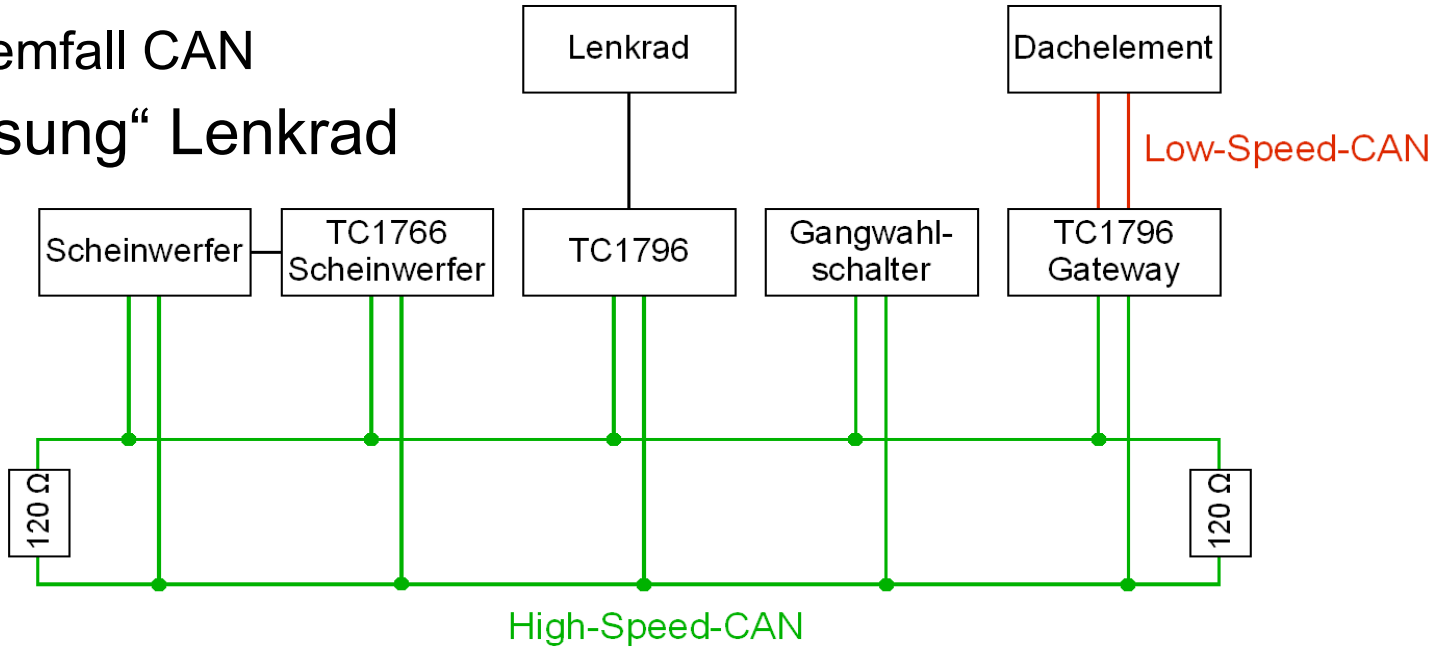
- Projektbeginn: Priorisierung und Erfassung der Anforderungen
- Bezug auf Anwendungsfälle
  - Demonstration
  - Lehre
  - Forschung (intern/extern)
  - Wartung
- Aufteilung in drei Kategorien
  - Fahrzeugnetz
  - Steuerumgebung
  - Messumgebung

Endprodukt


Steuer- und Messumgebung
Fahrzeugnetz
Scheinwerfer
Scheinwerfer
Lenkrad
Gangwahlschalter
Dachelement
CAN 3
TC 1766
TC 1796
TC 1796 als Gateway
Messboard TC 1796
Messboard TC 1796
Messboard TC 1796
CAN 1
CAN 2
Benutzerschnittstelle

Fahrzeugnetz

- Software
- Scheinwerfer
- Hardware
  - Gangwahlschalter
  - Dachelement
  - Problemfall SBC
  - Problemfall CAN
- „Notlösung“ Lenkrad

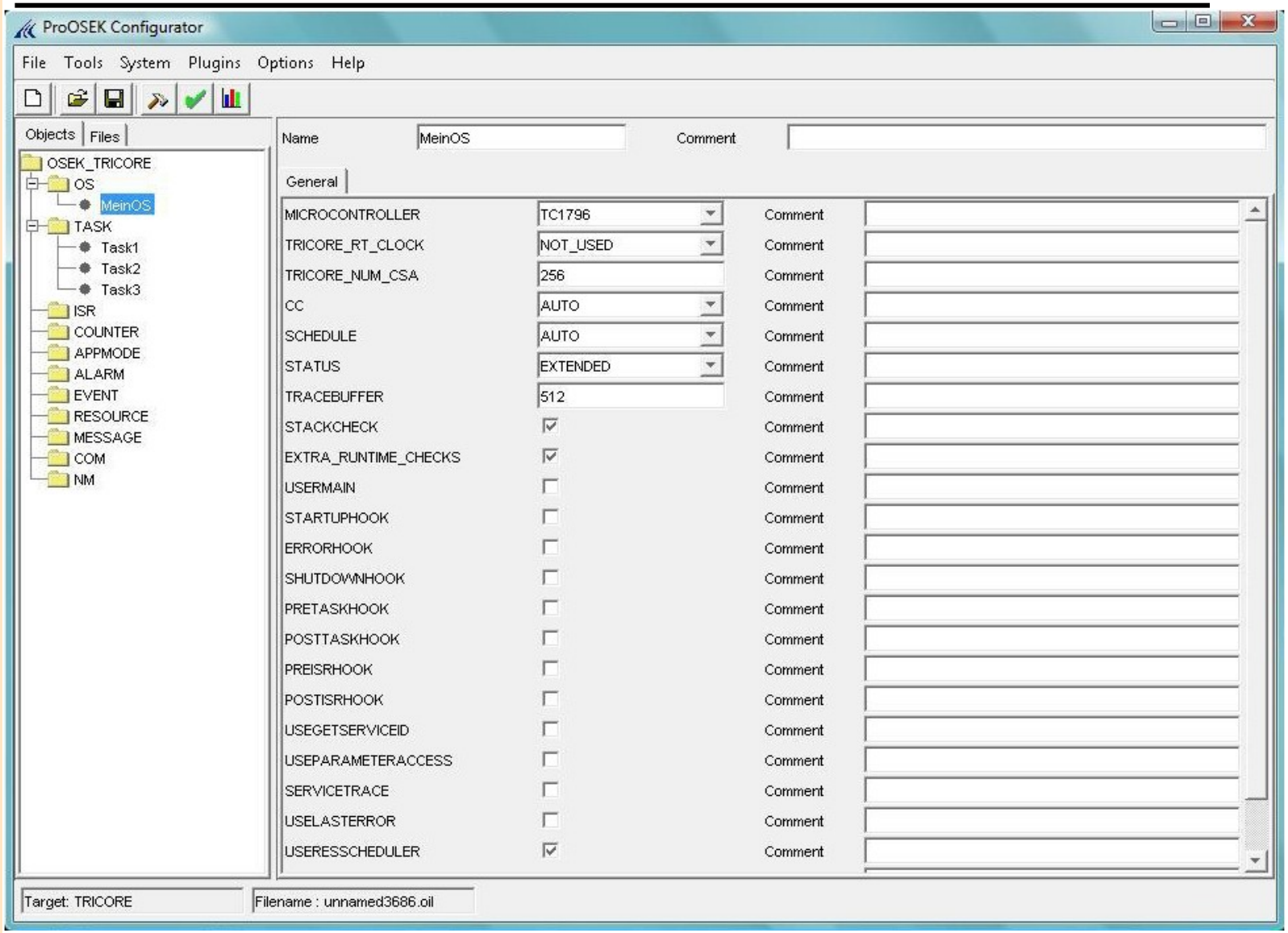


Software & OSEK

- Software (z.B. Scheinwerfersteuerung) auf den TriBoards ist in OSEK integriert

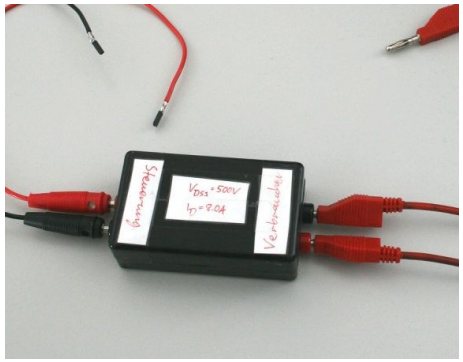
- OSEK – Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
- Projekt der deutschen und französischen Automobilindustrie
- Standardbetriebssystem für Fahrzeug Steuergeräte
- Ziele:
  - Portabilität und Wiederverwendbarkeit von Code auf verschiedenen Steuergeräten (Interfaces)
  - Dadurch Einsparung von Entwicklungskosten und -zeit
  - Skalierbar und Konfigurierbar auf die Anwendung (Effizienz)
- Momentaner Standard in der Automobilindustrie

OSEK - Konfigurator



Fahrzeugnetz - Scheinwerfer

- Inbetriebnahme der Scheinwerfer
  - Konfektionieren eigener Kabel für Strom und Datenleitungen
  - Entwurf und Design von „Transistorkästchen“ zum Schalten der Scheinwerferfunktionen
  - Senden von Initialisierungsnachrichten



Fahrzeugnetz - Scheinwerfer

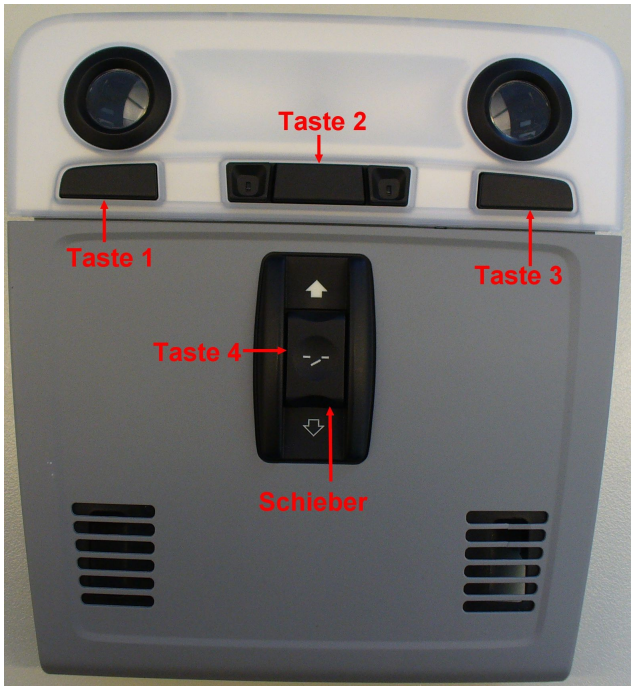
- Programmierung der Scheinwerfer
  - Iterativ in mehreren Schritten:
    - Per CANoe Generator
    - Nachrichten per Steuergerät
    - Steuergerätenachrichten als Reaktion auf äußere Einflüsse
    - Als „nacktes“ selbstständiges Programm mit allen Funktionen
    - Als Task eines Betriebssystems

Fahrzeugnetz - Gangwahlschalter

- Ansteuerung von LEDs
- Auslesen von Knöpfen
- Bestimmung der Hebelstellung
- Implementierung eines CAN-Treibers

Fahrzeugnetz - Dachelement

- Ansteuerung von LEDs und Lampen
- Auslesen von Tasten & Schieber
- Implementierung eines CAN-Treibers



Fahrzeugnetz - Problemfall SBC

- System Basis Chip
  - Aufgaben des SBC
    - versorgt µC mit Spannung und kontrolliert dadurch seinen Wach-Zustand
    - CAN-Transceiver
    - aktiviert zusätzliche Peripherie (LEDs)
  - Problem:
    - nach dem Einschalten deaktiviert der SBC den µC

Fahrzeugnetz - Problemfall CAN

- Problem:
  - Ein Aktor besitzt Low-Speed-CAN und alle weiteren Steuergeräte nutzen High-Speed-CAN-Transceiver
- Unterschiede:
  - Spannungsdifferenz zwischen den CAN-Leitungen
  - Terminierung des CAN-Bus
- Lösung:
  - CAN-Konverter
  - externer Low-Speed-CAN-Transceiver am Gateway

Fahrzeugnetz - Problemfall Lenkung

- Lenksäule konnte nicht in Betrieb genommen werden
- Ersatz: Handelsübliches Gaming-Lenkrad
- Anpassungen an unsere Bedürfnisse:
  - Schnittstelle für TriBoard erstellt

  → **Anbindung an Fahrzeugnetz**
  - Feinjustierung des Lenkwinkels (X-Y-Achse)
    - Unterteilung in Bereiche für Geradeaus- und Kurvenfahrt
    - Filterung von häufigen Peaks

  → **symmetrisches, stabiles und nicht zu sensibles Verhalten**

Lenkung (Fortsetzung)

- Anpassungen an unsere Bedürfnisse:
  - Definition von CAN-Nachrichten
    - Senden des Lenkradstatus:

CAN-ID: 220h

Bit	4	3	2	1	0
Bedeutung	Tasten	X-Achse	-	Y-Achse	-

    - Steuerung der Scheinwerfer mit dem Lenkrad:

CAN-ID: 501h

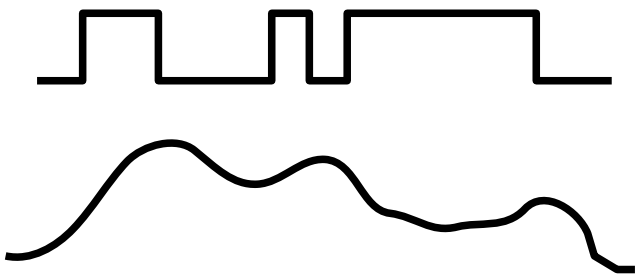
Bit	5	4	3	2	1	0
Bedeutung	Abblend-licht	Tagfahr-licht	Stand-licht	Fern-licht	Blinker rechts	Blinker links

    → **Kommunikation mit Scheinwerfern**
  - Hebelstellung bzw. Tastenbelegung:
    - Regelung des Blinkers, des Abblend- und Fernlichts

    → **Steuerung der Scheinwerferfunktionen**

Steuer- und Messumgebung

- Anforderungen an Steuer- und Messumgebung:
  - Messen digitaler Pegel
    - Steuersignale an/aus,...
  - Messen analoger Signale
    - Stromaufnahme



  - Messwerte per CAN-Bus an PC übertragen
  - „Messobjekte“ ein/ausschalten
- Vorüberlegung:
  - CAN-Bus Übertragungsrate: max. 1 Mbit/sec.
  - A/D-Wandler Auflösung: 10 Bit/Sample
    - bis zu 100.000 Samples/sec.

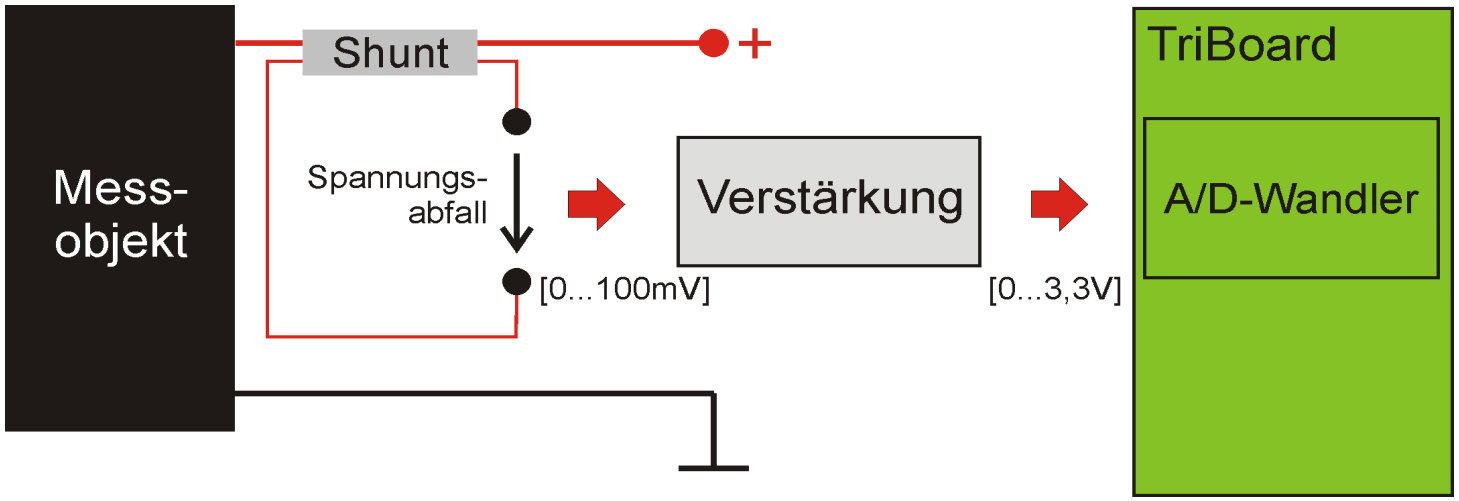
➔**Problem:** CAN-Bus zu langsam für LIVE Messwertübertragung

Steuer- und Messumgebung

- Lösung:
  - Messwerte zwischenspeichern
  - Jeden 1000ten Messwert direkt übertragen
  - Nach Ende der Messung Werte mit geringer Rate übertragen
- TriBoards als Messboards
  - A/D-Wandler zur Messung von analogen/digitalen Signalen
  - CAN on-Board
  - Speicher für Messwerte
- Messwerte bekommen Zeitstempel
  - CANoe erstellt Log-File
  - Eigenes Tool wertet Log aus
    - korrigiert Zeitverschiebung zwischen Messzeitpunkt und Zeitpunkt der Übertragung

Steuer- und Messumgebung

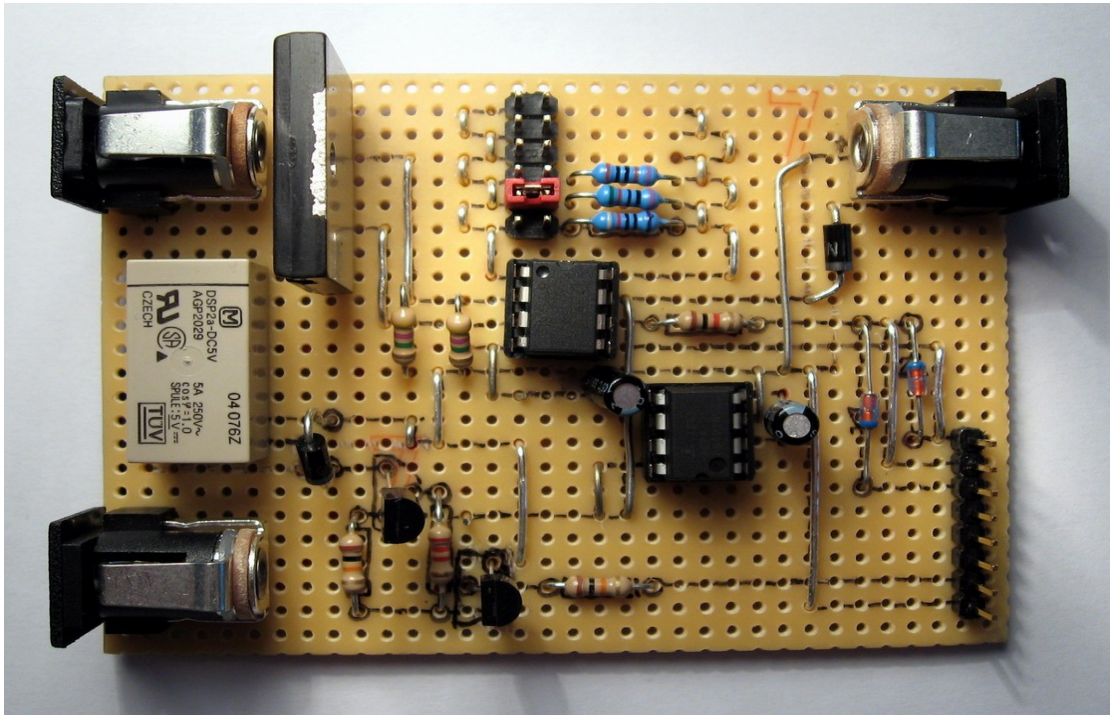
- Problem Strommessung
  - Prinzip: Spannungsabfall am Shunt messen



- Spannungsabfall verstärken auf A/D-Wandler-Eingangsbereich

Steuer- und Messumgebung

- Problemlösung mit selbst erstellter Hardware: **Strommessplatine**
  - Verstärkt Spannungsabfall am Shunt auf [0V...3,3V]
  - Relais zum Ein/Ausschalten des „Messobjektes“



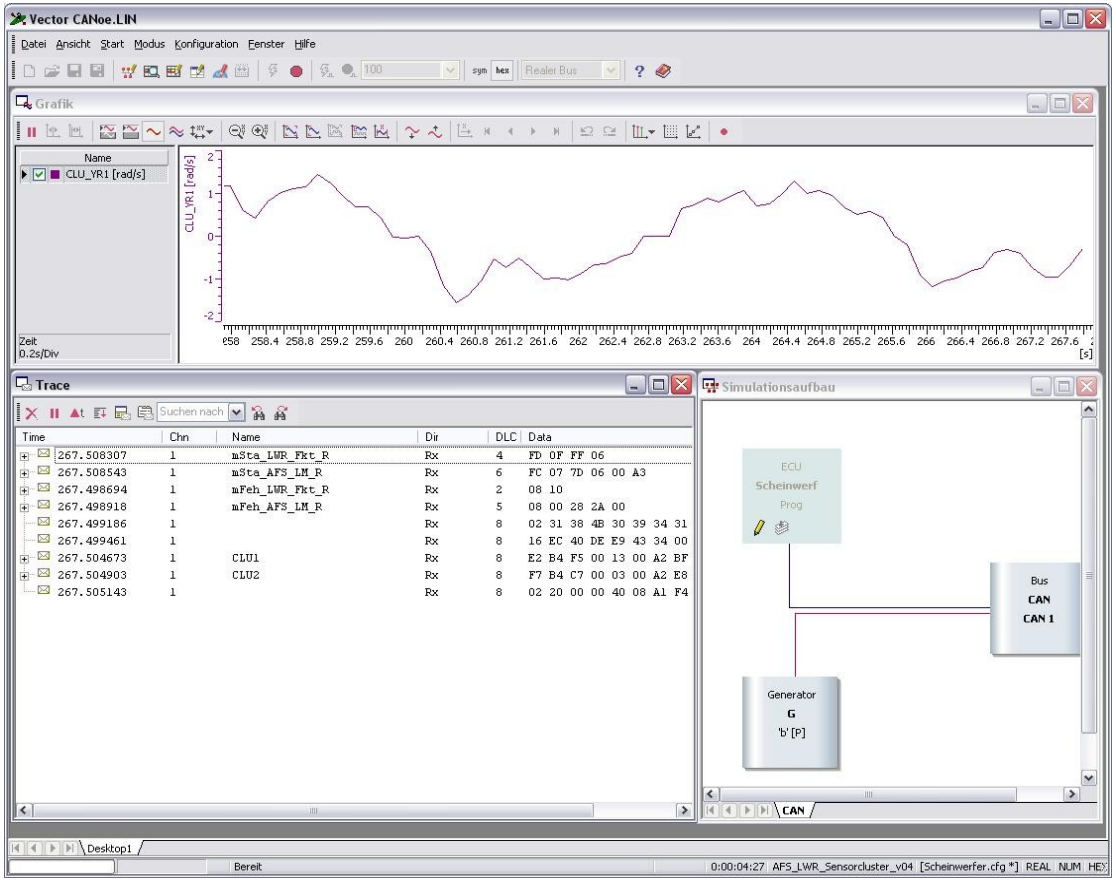
Benutzerschnittstelle

- CAN-Karte im Windows-PC
- Kommerzielle Softwarelösung: CANoe
  - Restbussimulationen
  - Simulation von CAN-Nachrichten
  - Überwachen und Auslesen des Busses
  - Visualisierung von eingegangenen Nachrichten
  - Replay-Funktion (Wiedereinspielen von Logfiles)
- Empfangen der Messdaten ebenfalls über CANoe
- Umwandlung der Messdaten aus Logfiles mittels selbst erstelltem Tool
- Fahrsimulation TORCS zur Demonstration
- Betrieb auf Linux-Notebook
  - CAN-Controller für Linux

CANoe

- CANoe: Schnittstelle zur Visualisierung der CAN-Nachrichten

  - Darstellung der Signalwerte
  - Zusammen-fassung von Signalen in Sichten
  - Simulation des Restbusses (Parallelentw.)



CANoe

- Eigene Datenbank in CANoe für die Signale eingepflegt
  - Einheitliche und verbindliche Datenbasis für alle Steuergeräte
  - Komfortable Anzeige von Signalen im CANoe

- Eigenes Konvertertool
  - Wandelt per CANoe aufgenommene Messdaten (CAN-Nachrichten) in ein von CANoe wieder abspielbares Format um
  - Sortiert die Signale zur besseren Darstellung

Torcs

- TORCS - The Open Racing Car Simulator
- Open Source Rennsimulator
- Von uns als Demonstrationsapplikation genutzt
- Musste entsprechend angepasst werden, um CAN Nachrichten zu verstehen
  - Nachricht vom Lenkrad
    - Pedalstellung
    - Knopfdruck
    - Lenkradauslenkung
  - Nachricht vom Gangwahlschalter
    - Informationen über aktuellen Modus
- Sendet selbst Status-Nachricht

Vorgehen und erfüllte Minimalziele

- Anforderungsspezifikation, Aufteilung in einzelne Umgebungen, Festlegung des Leistungsumfangs

  → **1. Erstellung eines Konzeptes/detaillierten Entwurfs**
- Analyse der Hard- und Software-Komponenten, Erstellen der Infrastruktur, Vernetzung der Steuergeräte, Testdurchläufe

  → **2. Inbetriebnahme eines Versuchsaufbaus**
- Entwurf/Implementierung der Umgebungen in Hard- und Software, Messwert- und Steuerungsverarbeitung

  → **3. Konzeption einer Steuerungs- und Messumgebung**
- Anpassen der Software, Einbettung der Komponenten, Entwurf/Implementierung einer Beispielanwendung, Visualisierung der Messwerte, Abstimmung der Steuerungssignale

  → **4. Entwurf einer Demonstratoranwendung**
- Protokollierung der Treffen, Dokumentation wichtiger (Zwischen-) Ergebnisse, Erstellung des Zwischen- und Endberichts

  → **5. Wissenschaftliche Dokumentation der Ergebnisse**

Erfüllte optionale Ziele

- Integration eines Lenkrads, Anpassen der Rennsimulation „TORCS“

→ **Verbesserung des Demonstrators, M-M-Interface**

- Einsatz zusätzlicher Steuergeräte und Messboards

→ **Verbesserung des modularen Aufbaus**

- Aufsplitten der Steuersoftware auf mehrere Steuergeräte

→ **Implementierung verteilter Fahrzeugfunktionen**

- Hinzufügen eines Low-Speed-CAN-Bus, Einrichten eines Gatewayknotens

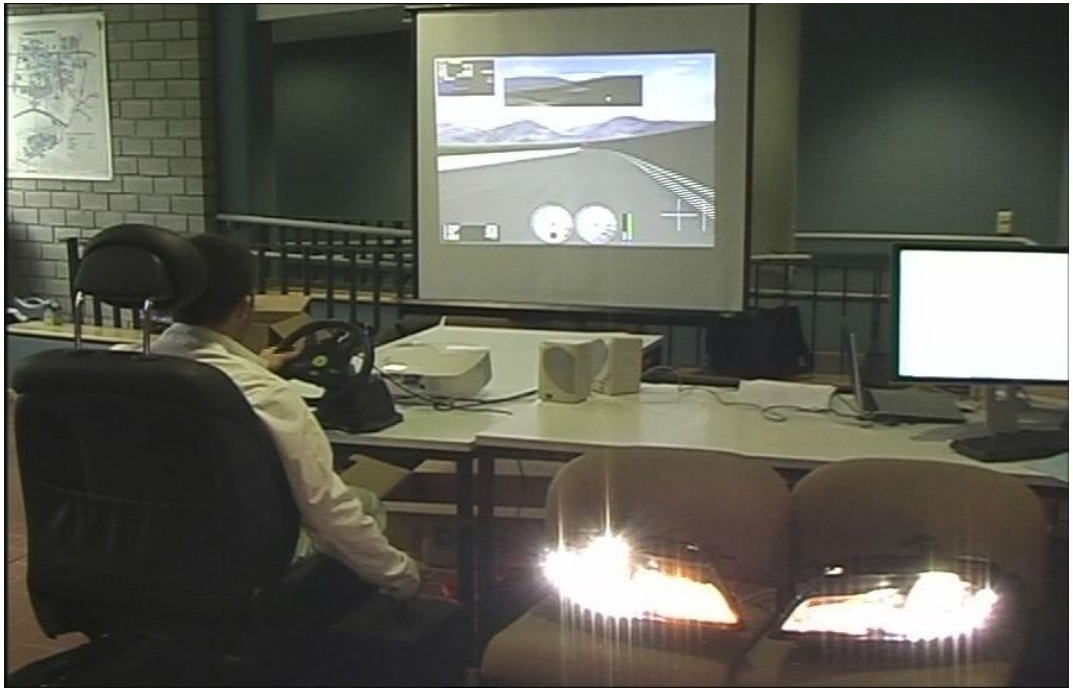
→ **Anbindung weiterer Hardwarekomponenten (z.B. Dachmodul)**

- Auslesen der Messwerte mittels CANoe, automatische Log-Konvertierung

→ **Verbesserte Visualisierung von Aufzeichnungen**

Campusfest

- 14.06.2008
- Öffentliche Vorführung unseres Aufbaus
- Hilfsmittel: Demonstration mit TORCS



Mögliche Erweiterungen

- Weitere Aktoren sind bereits vorhanden
  - Schiebedach
  - Lenksäulenmodul
  - Tür
- Ermöglichung von Remote-Zugriff über das Internet mittels IP-Steckdosen und geeigneter Betriebssw.
- Anpassung der Messsoftware auf konkrete Experimente und Einsatz im Rahmen solcher
- **Fortbestand** des Projektes **gesichert:**
  Einsatz als Lehrversuch im Rahmen des neuen **DLR School Lab** in Dortmund

Vielen Dank!

Die Teilnehmer der Projektgruppe 522 AutoLab möchten sich herzlich für die Unterstützung, die Zusammenarbeit und die tolle Atmosphäre bei den PG-Betreuern Prof. Dr.-Ing. Olaf Spinczyk, Dr. Michael Engel, Horst Schirmeier und Jochen Streicher sowie allen Mitarbeitern des Lehrstuhls 12 am Fachbereich für Informatik der TU Dortmund bedanken!

Partner

Ebenfalls ganz herzlichen danken wir unseren Partnern die unsere PG so intensiv unterstützt haben!

Fragen