

# Entwurfsdokument

## Service-Interface für ein Formula-Student-Fahrzeug

Technische Universität Ilmenau  
Softwareprojekt SS 2013  
Gruppe 19

Christian Boxdörfer  
Thomas Golda  
Daniel Häger  
David Kudlek  
Tom Porzig  
Tino Tausch  
Tobias Zehner  
Sebastian Zehnter

15.05.2013

betreut durch

Dr. Heinz-Dietrich Wuttke, TU Ilmenau  
Oliver Dittrich, fachlicher Betreuer Team StarCraft e.V.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Randbedingungen</b>	<b>4</b>
<b>3</b>	<b>Grobentwurf</b>	<b>6</b>
3.1	Architekturmuster und Systemzerlegung . . . . .	6
3.1.1	dSPACE MicroAutoBox II . . . . .	7
3.1.2	Embedded-PC . . . . .	7
3.1.3	Virtueller Server . . . . .	7
3.1.4	Webseite . . . . .	7
3.1.5	Datenbanken . . . . .	8
3.2	Simulink-Modell . . . . .	8
3.2.1	Verwendete Blöcke . . . . .	8
3.2.2	Signalgenerator . . . . .	13
3.2.3	Signalkollektor . . . . .	17
3.2.4	Enkodierung . . . . .	18
3.2.5	UDP-Schnittstelle . . . . .	18
3.3	Embedded-PC . . . . .	19
3.3.1	Empfangen der Fahrzeugdaten . . . . .	19
3.3.2	Hinzufügen eines Zeitstempels . . . . .	19
3.3.3	Senden der Fahrzeugdaten . . . . .	19
3.4	Virtueller Server . . . . .	20
3.4.1	Empfangen der Fahrzeugdaten . . . . .	21
3.4.2	Abfragen des Zeitstempels . . . . .	21
3.4.3	Daten dekodieren . . . . .	21
3.4.4	Daten verwerfen . . . . .	21
3.4.5	Daten in Datenbank schreiben . . . . .	21
3.5	Webseite . . . . .	22
3.5.1	Registrierung . . . . .	23
3.5.2	Anmeldung . . . . .	23
3.5.3	Passwort vergessen . . . . .	24
3.5.4	Benutzerverwaltung . . . . .	25
3.6	Datenbanken . . . . .	25
3.6.1	Nutzerdatenbank . . . . .	25
3.6.2	Fahrzeugdatenbank . . . . .	26
<b>4</b>	<b>Testdrehbuch</b>	<b>28</b>
4.1	Test auf Funktionalität . . . . .	28
4.2	Test auf Robustheit . . . . .	31
	<b>Glossar</b>	<b>32</b>
	<b>Abbildungsverzeichnis</b>	<b>38</b>
	<b>Anhang</b>	<b>39</b>
A.1	Datentypen in Simulink . . . . .	39
A.2	weitere Datentypen . . . . .	39
A.3	UML-Diagramme . . . . .	39

# 1 Einleitung

Diese Software ermöglicht es Team StarCraft e.V. ausgelesene Fahrzeugdaten ihres Formula-Student-Fahrzeugs in weicher Echtzeit an eine Webseite zu übertragen und dort geeignet zu visualisieren. Hierfür werden auf einer dSPACE MicroAutoBox II, auf der ein selbst erstelltes Simulink-Modell implementiert ist, die ausgelesenen Fahrzeugdaten ggf. aufbereitet und anschließend kodiert per UDP-Ethernetverbindung in einem *uint32*-Datenstrom an einen Embedded-PC gesendet. Dieser leitet daraufhin die Datenpakete mittels einer GPRS/UMTS-Verbindung an einen virtuellen Server weiter, der diese dekodiert, aufbereitet und an eine Datenbank übergibt, welche die Daten der letzten 10 Stunden speichern soll. Die Darstellung dieser Daten erfolgt schließlich über eine Webseite, die die einzelnen Fahrzeugdaten aus der Datenbank ausliest und geeignet visualisiert. Die Webseite selbst soll sich vor allem durch eine hohe Übersichtlichkeit und eine leichte Bedienbarkeit auszeichnen, sowie ein Nutzersystem bereitstellen. Das Projekt ist daher viergeteilt in die Abschnitte dSPACE MicroAutoBox II, Embedded-PC, virtueller Server und Webseite. Im Folgenden werden nun anhand des Datenflusses der Fahrzeugdaten der Datenaustausch bzw. die Kommunikation zwischen den einzelnen Abschnitten erläutert sowie die Ideen zu deren interner Realisierung anhand verschiedener Simulink-Modelle bzw. UML-Diagramme vorgestellt.

## 2 Randbedingungen

Folgende Randbedingungen werden an das System gestellt:

### 1) Simulink-Modell

- Die Kommunikation der MicroAutoBox II mit dem Embedded-PC wird aufgrund fehlender technischer Machbarkeit statt über eine API nun über eine UDP-Ethernet-schnittstelle realisiert.
- Vor der Übertragung der Datenpakete von der MicroAutoBox II ist es notwendig diese in einen *uint32*-Datenstrom zu konvertieren, da man bei einer Kommunikation der MicroAutoBox II via UDP mit externer Peripherie auf speziell für diesen Einsatzzweck konzipierte Blöcke der Firma dSPACE angewiesen ist.

### 2) Embedded-PC

- Die ursprünglich auf dem Embedded-PC vorgesehene Auswertung und Aufbereitung der Fahrzeugdaten muss nunmehr aus zweierlei Gründen auf den virtuellen Server ausgelagert werden:
  - 1) Durch die Kodierung des *uint32*-Datenstromes mittels des proprietären Simulink-Blocks der Firma dSPACE müssten die Daten nach einer möglichen Auswertung auf dem Embedded-PC erneut zu einem einzelnen Datenpaket gebündelt, kodiert und wiederum auf dem virtuellen Server dekodiert werden, was neben einem deutlichen Verkomplizieren der Software einen unnötigen Arbeitsaufwand gegenüber einer Auslagerung der Aufbereitung auf dem virtuellen Server zur Folge hätte.
  - 2) Die Anbindung des Embedded-PCs an die Fahrzeugdatenbank sollte ursprünglich über den MySQL Connector/C++ erfolgen. Da dieser jedoch keine Unterstützung für das UDP-Protokoll bietet und eine Kommunikation der beiden Komponenten via TCP/IP wegen eines möglicherweise auftretenden Paketstaus nicht eingesetzt werden kann, kommt nur eine Auslagerung des Arbeitsschrittes auf den virtuellen Server infrage. Zudem wäre bei TCP/IP das Zeitintervall zwischen einem verlorenem Datenpaket und dem erneuten Versenden von Datenpaketen gegenüber der Forderung nach weicher Echtzeit unter Umständen zu groß bemessen.

**3) Virtueller Server**

- Auf dem virtuellen Server müssen die Dienste MySQL, Apache, PHP, sftp und ssh installiert und funktionstüchtig sein.

**4) Webseite**

- Um auf die Webseite zugreifen zu können muss der virtuelle Server online sein.

**5) Organisatorisches**

- Da viele der von uns benötigten Komponenten wie z.B. der MicroAutoBox II lokal an die Maschinenhalle EIT gebunden sind, können bestimmte Arbeiten und Tests für das Softwareprojekt nur an dieser Örtlichkeit durchgeführt werden.
- Darüber hinaus existiert in der Maschinenhalle an dem uns zugewiesenen Arbeitsplatz nur ein Rechner mit der benötigten Software, welchen wir uns mit externen Nutzern teilen müssen. Dementsprechend sind exakte Absprachen bezüglich der Nutzung des Rechners bzw. der Rechnerkapazitäten erforderlich.
- Zudem ist eine Arbeit an diesem Arbeitsplatz ohne Beaufsichtigung nur durch eine vorherige Einweisung und Arbeitsschutzbelehrung möglich.

## 3 Grobentwurf

### 3.1 Architekturmuster und Systemzerlegung

Aufgrund der Komplexität des Systems, die aus einer Vielzahl an unterschiedlichen Plattformen und Funktionen resultiert, lässt sich die Architektur des Systems keinem vorgegebenen Architekturmuster zuordnen. Um dennoch einen Überblick über die einzelnen Systemkomponenten zu ermöglichen, wurden diese in dem folgenden Verteilungsdiagramm modelliert.

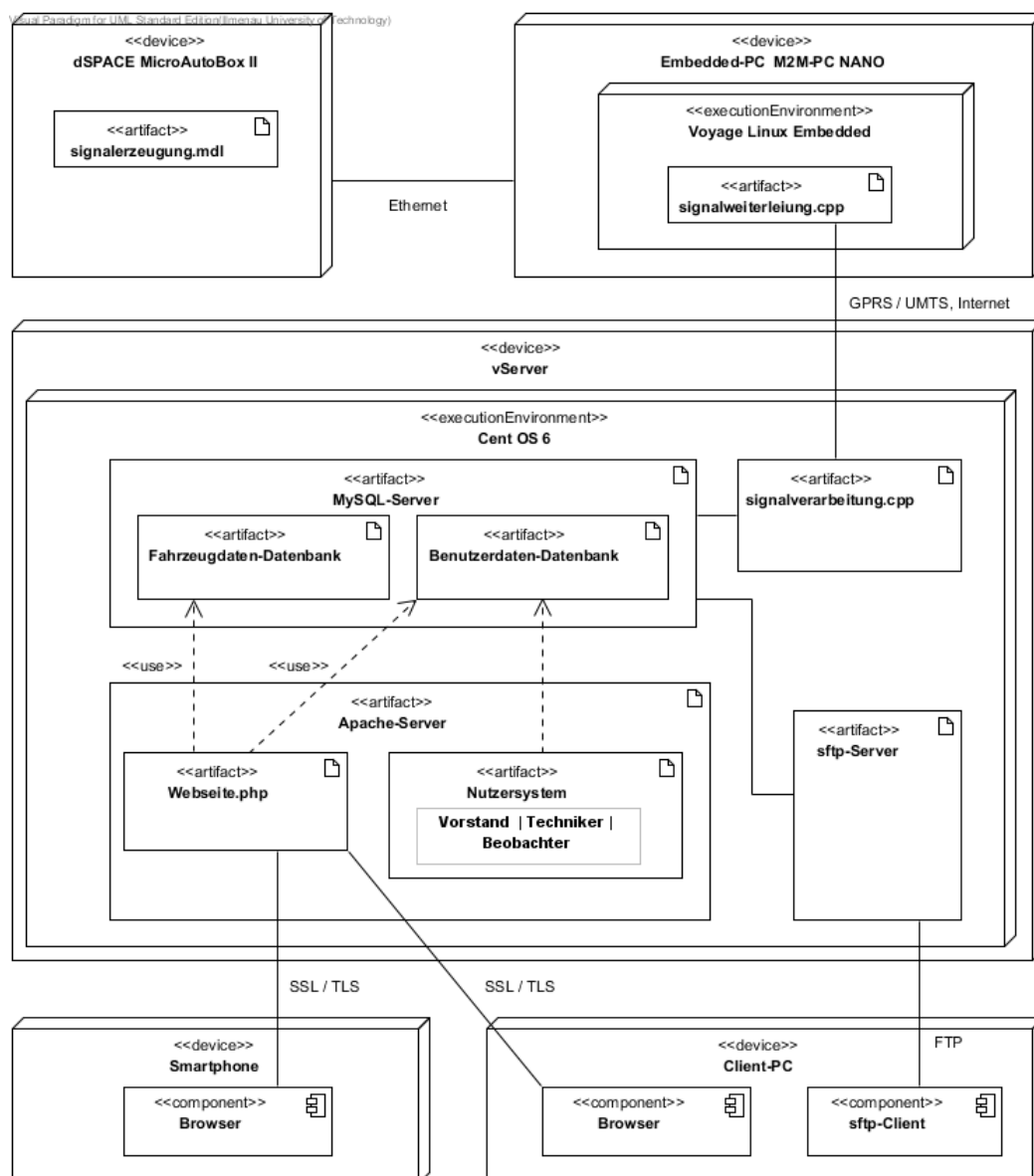


Abbildung 3.1: Gesamtsystem als Verteilungsdiagramm

### 3.1.1 dSPACE MicroAutoBox II

Bei der dSPACE MicroAutoBox II handelt es sich um ein Steuergerät mit Echtzeiteigenschaften, welches fest im Formula-Student-Fahrzeug verbaut ist und es ermöglicht, sämtliche Fahrzeugdaten aus ebendiesem auszulesen. Auf diesem Gerät wird ein Simulink-Modell implementiert, das die als Busarray empfangen Daten entgegen nimmt, aufbereitet, enkodiert und über eine UDP-Schnittstelle an einen Embedded-PC überträgt.

### 3.1.2 Embedded-PC

Die verwendete MicroAutoBox II sendet die zuvor aufbereiteten Fahrzeugwerte mit Hilfe des auf ebendieser implementierten Simulink-Modells per UDP-Sockets über einen kabelgebundenen Netzwerkanschluss (LAN) an den Embedded-PC M2M-PC NANO der Firma Adyna. Die Übertragung der Fahrzeugdaten von dem Embedded-PC zu dem virtuellen Server geschieht ebenfalls über UDP-Sockets. Das Programm läuft eigenständig auf dem Embedded-PC und wird automatisch beim Systemstart ausgeführt. Sobald ein UDP-Paket vollständig empfangen wurde, wird es mit einem Zeitstempel versehen, der die Systemzeit des Embedded-PC zum Zeitpunkt des Hinzufügens beinhaltet und dazu dient, die Aktualität der Daten bei dem Eintreffen auf dem virtuellen Server festzustellen. Anschließend werden die Daten über eine mobile Breitbandverbindung mittels UDP-Sockets an den virtuellen Server weitergeleitet. Für die eigentliche Übertragung zum virtuellen Server über eine GPRS/UMTS-Verbindung ist ein in der Programmiersprache C++ selbst entwickeltes Programm zuständig.

### 3.1.3 Virtueller Server

Der virtuelle Server wird von dem Provider 1&1 bereitgestellt und ist jederzeit über eine feste IP-Adresse erreichbar. Darauf wird eine in der Programmiersprache C++ entwickelte Software implementiert, die eingehende Pakete an einem bestimmten Port entgegennimmt, deren Nutzdaten dekodiert, validiert und in eine auf dem virtuellen Server laufende MySQL-Datenbank schreibt.

### 3.1.4 Webseite

Die Webseite übernimmt im Rahmen dieses Systems die Aufgaben der grafischen Benutzeroberfläche. Sie besitzt zwei Kernaufgaben: das Darstellen der Fahrzeugdaten, die sich in einer Datenbank auf dem Webserver befinden und das Vermeiden von Zugriffen unbefugter Personen auf ebendiese Daten. Dies wird durch ein Benutzersystem erreicht, welches nur ausgewählten Personen den Zugang zu den Informationen der Datenbank ermöglicht und diesen somit die Darstellung oder den Export selbiger Informationen zulässt.

### 3.1.5 Datenbanken

Auf dem virtuellen Server läuft ein MySQL-Server, der unserem Projekt zur Verfügung steht. Es werden zwei Datenbanken angelegt um Benutzerdaten und die Fahrzeugdaten aufzunehmen. Zusätzlich wird ein Benutzer mit zugehörigem Passwort eingerichtet, der alle Rechte auf die Datenbanken erhält und so dem Programm den Zugriff auf die Datenbanken zu ermöglichen. Für die Initialisierung der Datenbanken wird ein Skript erstellt.

#### Benutzerdaten-Datenbank

Diese Datenbank bildet die Grundlage für das optionale Benutzersystem, in ihr werden später die Nutzer und ihre zugehörigen Berechtigungen gespeichert. Zudem wird der Zeitpunkt des letzten Logins erfasst, um angemeldete Benutzer automatisch abmelden zu können.

#### Fahrzeugdaten-Datenbank

Diese Datenbank bildet die Schnittstelle zwischen dem C++ - Programm, welches die Fahrzeugdaten aus den UDP-Paketen kontinuierlich entgegennimmt und in die Datenbank speichert, sowie der Webseite, welche die Daten anzeigt. Die Tabelle soll mindestens zehn Stunden an Fahrdaten mit einem Intervall von einer Sekunde aufnehmen (36000 Datensätze), ältere Daten werden aus der Datenbank entfernt. Über die Webseite ist ein Export der Daten im CSV-Format möglich.

## 3.2 Simulink-Modell

### 3.2.1 Verwendete Blöcke

Um den Einstieg in unsere im folgenden aufgeführten Modellausschnitte zu erleichtern, werden im Verlauf dieses Abschnittes alle zur Erstellung des Simulink-Modells für die dSPACE MicroAutoBox II verwendeten Blöcke vorgestellt und ihre Funktionsweise kurz erläutert.

#### Sources



Constant

**Abbildung 3.2:**  
Constant-Block



Repeating Sequence Interpolated

**Abbildung 3.3:** Repeating Sequence Interpolated - Block

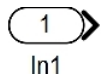


1) *Constant-Block*

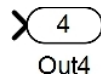
Der Constant-Block ermöglicht die Generierung eines reellen oder komplexen, konstanten Wertes. Je nach Modifikation der Einstellungen des Blocks wird es zudem ermöglicht, neben einem konstanten Skalar einen konstanten Vektor oder eine konstante Matrix als Eingangssignal bereitzustellen. Als Datentypen für das Eingangssignal stehen die unter A.1 aufgeführten Datentypen zur Verfügung.

2) *Repeating-Sequence-Interpolated - Block*

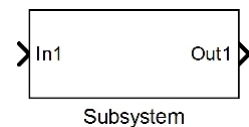
Im Gegensatz zum Constant-Block ermöglicht dieser Block die Erzeugung eines individuellen, sich periodisch wiederholenden und kontinuierlichen Signals mittels einer Interpolation anhand zuvor selbst definierter diskreter Zeit- und Funktionswerte, welche in zwei Vektoren gleicher Länge gespeichert sind.

**Ports & Subsystems**

**Abbildung 3.4:**  
Inport-Block



**Abbildung 3.5:**  
Outport-Block



**Abbildung 3.6:** Subsystem

1) *Inport-Block*

Dieser Block hat in unserem Modell die Aufgabe, die zuvor festgelegten Eingangssignale des Subsystems auf der Modellebene des Subsystems selbst zu repräsentieren. Darüber hinaus ist es in der Top-Level-Ebene des Systems mit diesem Block auch möglich, externe Eingangssignale aus dem Arbeitsbereich für das Modell bereitzustellen.

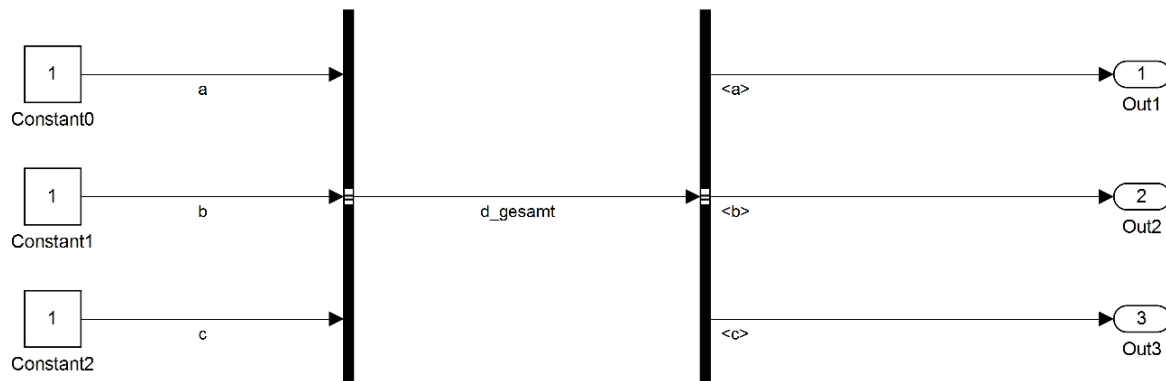
2) *Outport-Block*

Die Aufgabe des Outport-Blockes ist es, eine Verknüpfung vom aktuellen System zu einem Zielsystem außerhalb der Modellebene herzustellen.

3) *Subsystem*

Innerhalb eines Subsystems können verschiedene Blöcke zusammengefasst werden, was eine Strukturierung und Gliederung der Signalflüsse erleichtert und zudem eine deutlich übersichtlichere Darstellung des Modells zur Folge hat. Weiterhin ist es auch möglich, mehrere Subsysteme in einem Subsystem zusammenzufassen, um eine beliebige Tiefe innerhalb der Hierarchie eines Modells zu realisieren.

## Signal Routing



**Abbildung 3.7:** Bussystem bestehend aus einem Bus Creator (l.) und einem Bus Selector (r.)

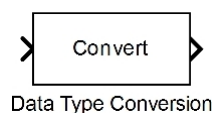
### 1) Bus Creator

Mit Hilfe eines solchen Bus Creators wird es ermöglicht, mehrere Signale (a, b, c) zu einem Gesamtsignal (d\_gesamt) zu bündeln.

### 2) Bus Selector

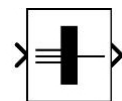
Umgekehrt erlaubt es der Bus Selector, aus einem Gesamtsignal wieder einzelne Signale zu selektieren und diese gesondert weiterzuleiten. So werden wie in Abb. 3.7 dargestellt aus dem Gesamtsignal d\_gesamt wieder die Signale a, b und c herausgeführt.

## Signal Attributes



Data Type Conversion

**Abbildung 3.8:**  
Convert-Block



Bus to Vector

**Abbildung 3.9:** Bus to Vector -  
Block

### 1) Convert-Block

Mit dem Convert-Block können verschiedene Anforderungen realisiert werden:

- Konvertierung eines Signals bzw. eines Signalvektors in einen anderen Datentyp, hierbei kann die Art der Rundung selbst festgelegt werden.
  - Umbenennung eines Signals bzw. eines Signalvektors.
- Dies hat den Zweck, dass man nach dem Convert-Block unabhängig vom angelegten Eingangssignal mit einem festen Datentyp und einem fest vergebenem Variablennamen in anderen Systemen bzw. auf anderen Plattformen arbeiten kann.

2) *Bus to Vector - Block*

Dieser Block konvertiert ein virtuelles Bussignal in ein Vektorsignal. Hierbei ist es erforderlich, dass alle am zu konvertierenden Bus anliegenden Signale im Datentyp, Signaltyp und im gewählten Sampling-Verfahren übereinstimmen.

**Sinks**

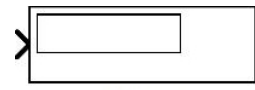
Terminator

**Abbildung 3.10:**  
Terminator-Block



Scope

**Abbildung 3.11:**  
Scope



Display

**Abbildung 3.12:** Dis-  
play

1) *Terminator*

Der Terminator dient dazu, die Signale, deren Ausgänge nicht mit Blöcken o.Ä. verbunden sind, abzuschließen.

2) *Scope*

Der Scope-Block stellt den Signalverlauf eines Signals über der Simulationszeit dar, weswegen er sich hervorragend dafür eignet ein erstelltes Modell auf dessen Korrektheit zu überprüfen.

3) *Display*

Der Display-Block zeigt den konkreten Wert eines Eingangssignals an. Um die Anzeige des Displays anzupassen, kann der Entwickler über die Einstellungen der Format-Parameter Einfluss darauf nehmen.

**Math Operations**

Gain

**Abbildung 3.13:** Gain-Block zur Verstärkung von Signalen

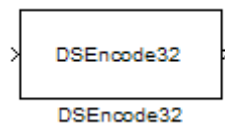
1) *Gain-Block*

Der Gain-Block multipliziert das Eingangssignal mit einer selbst gewählten Konstante, wobei das Eingangssignal selbst ein Skalar, ein Vektor oder eine Matrix sein kann.

### dSPACE-Blöcke

Im Folgenden sollen nun diejenigen Blöcke vorgestellt werden, welche speziell für eine Verwendung mit der MicroAutoBox II konzipiert wurden und u. a. für eine Kommunikation ebendieser mit dem Embedded-PC unerlässlich sind.

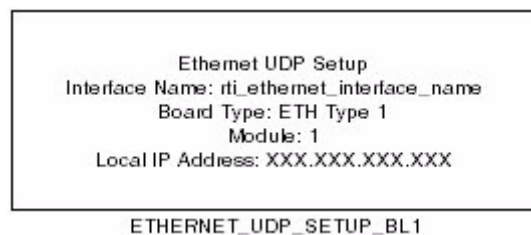
#### 1) *Encode32-Block*



**Abbildung 3.14:** Block zur Erzeugung eines *uint32*-Datenstroms

Der Encoder-Block DSEncode32 basiert auf einer sFunction, die von dSPACE implementiert wurde. Diese konvertiert einen Eingangsvektor in einen Datenstrom mit dem Datentyp *uint32*. Dieser Schritt ist notwendig, um die Daten für die Übertragung mittels einer UDP-Verbindung vorzubereiten, da die dafür vorgesehenen Blöcke nur Daten dieses Typs übertragen können.

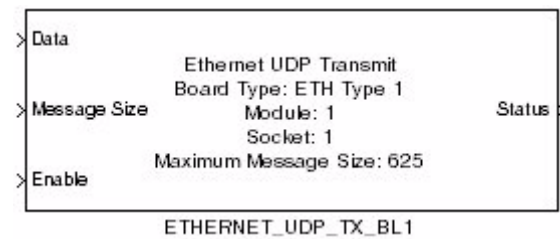
#### 2) *UDP-Setup-Block*



**Abbildung 3.15:** UDP-Block zur Einstellung der Netzwerkschnittstelle

Der Block ETHERNET\_UDP\_SETUP\_BL1 (Abbildung) dient zur Initialisierung der Netzwerkschnittstelle. Dort können lokale Portnummern für bis zu 4 Sockets pro Ethernet Interface vergeben und die IP-Adresse sowie Portnummer eines entfernten Geräts (hier: Embedded-PC) konfiguriert werden.

## 3) UDP-Send-Block

Abbildung 3.16: UDP-Block zum Senden eines *uint32*-Datenstroms

Das Senden der Daten erfolgt über den Block ETHERNET\_UDP\_TX\_BL1. Diesem wird ein Datenstrom vom Typ *uint32* sowie die Größe der zu übermittelnden Daten übergeben. Anschließend werden die Daten entsprechend der Konfiguration im ETHERNET\_UDP\_SETUP\_BL1 Block versendet.

## 3.2.2 Signalgenerator

Das Modell, welches auf der dSPACE MicroAutoBox II implementiert wird, besitzt zwei Subsysteme - den Signalgenerator und den Signalkollektor. Der Signalgenerator hat hierbei die Aufgabe, auf der MicroAutoBox II die für einen Systemtest benötigten Testsignale zu generieren, welche anschließend zum Embedded-PC via UDP-Schnittstelle gelangen und daraufhin über eine GPRS/UMTS-Verbindung an einen virtuellen Server gesendet werden. Dieses Subsystem wird jedoch nach einem erfolgreichen Test des Service-Interfaces durch ein Subsystem von Team StarCraft e.V. ersetzt, welches im Stande ist, die im Formula-Student-Fahrzeug verbauten Komponenten anzusprechen und somit im Unterschied zu dem aktuell verwendeten Signalgenerator statt künstlich erzeugter Daten die realen Daten auszulesen und an den Signalkollektor weiterzuleiten.

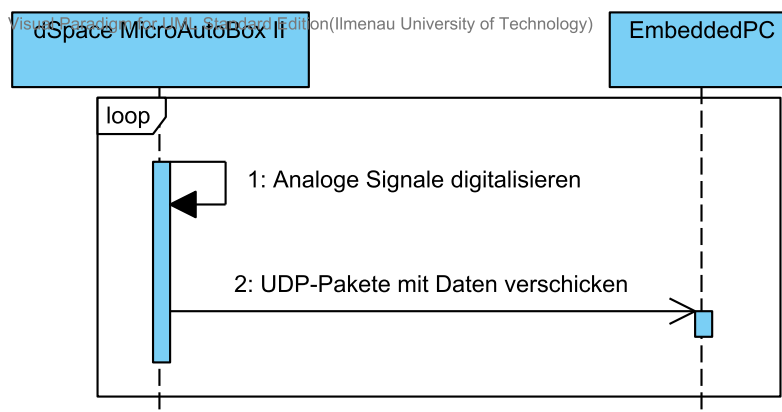
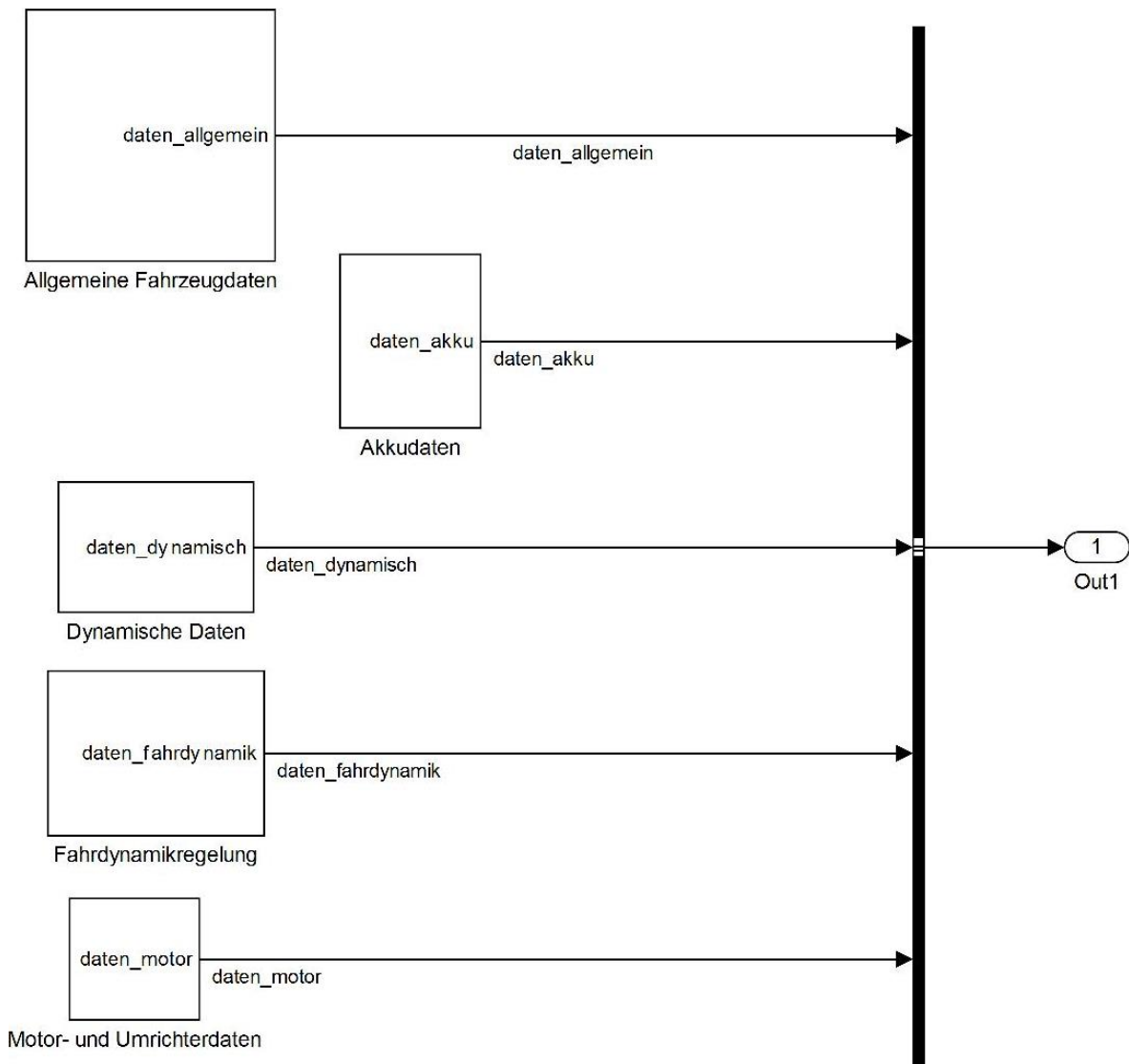


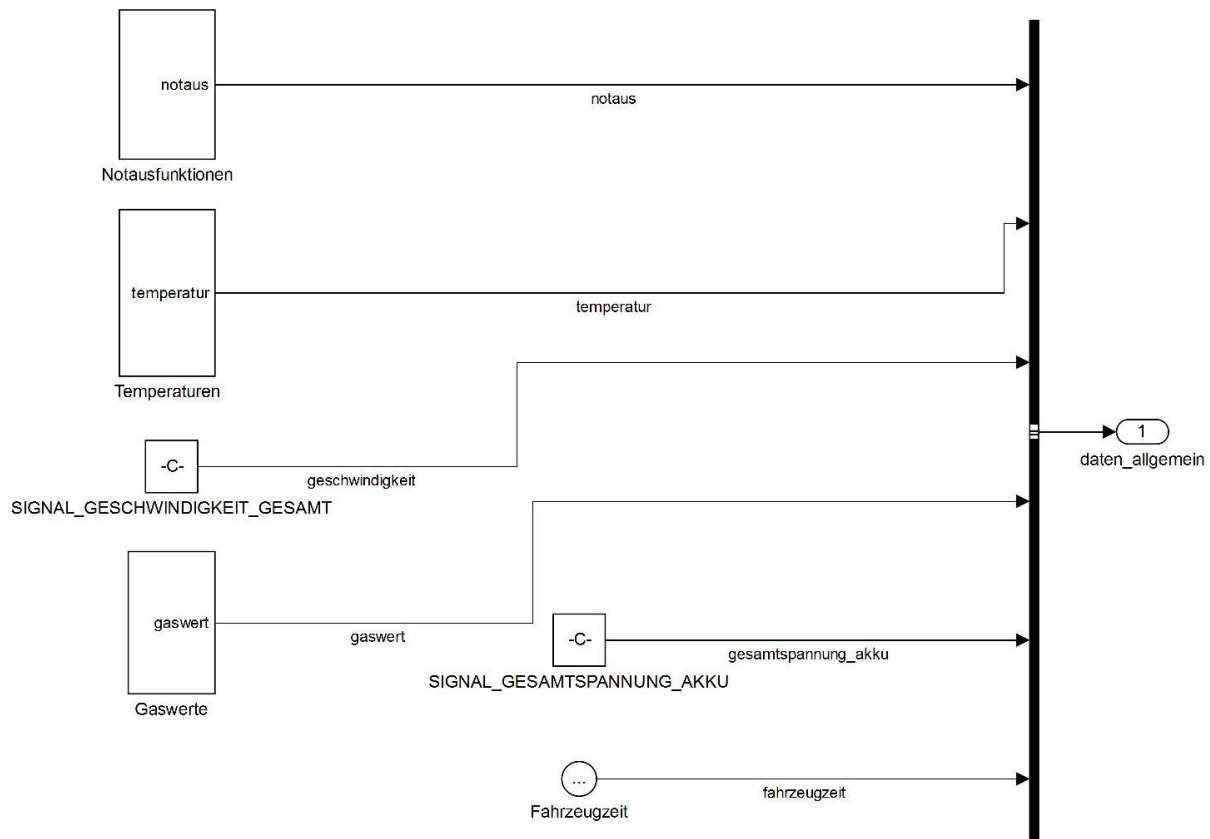
Abbildung 3.17: Sequenzdiagramm des auf der MicroAutoBox II implementierten Simulink-Modells

### Struktur des Subsystems

Um ein hohes Maß an Übersichtlichkeit und Modularität zu gewährleisten, wird an den Vorgaben von Team StarCraft e.V. orientierend das Subsystem des Signalgenerators nochmals in einzelne Subsysteme unterteilt, die die verschiedenen Kategorien der jeweiligen Fahrzeugdaten repräsentieren (s. Abb. 3.18):



**Abbildung 3.18:** Subsysteme des Signalgenerators nach den Kategorien der Fahrzeugdaten



**Abbildung 3.19:** Subsystem „Allgemeine Daten“ des Signalgenerators

Innerhalb dieser Subsysteme findet nun - ohne Beschränkung der Allgemeinheit am Subsystem „Daten Allgemein“ erläutert - die Signalerzeugung (Geschwindigkeit, Gesamtspannung des Akkus und der Fahrzeugzeit) statt, wobei wie in Abbildung 3.19 dargestellt die Signalerzeugung mehrere Signale der gleichen „Signalgruppe“ wiederum zu Subsystemen (Notausfunktionen, Temperaturen und Gaswerte) innerhalb des Subsystems „Allgemeine Daten“ zusammengefasst werden. Die eigentliche Erzeugung der Signale wird wie in Abbildung 3.20 gezeigt mittels Blöcken aus der Kategorie „Sources“ (s. Seite 8) realisiert. Hierbei wird je nach Input entweder ein Constant-Block mit dem Datentyp *boolean* zur Realisierung von Schaltern etc. (s. Notausfunktionen) oder ein Repeating Sequence Interpolated (RSI) - Block mit dem Datentyp *single* zur Modellierung der restlichen Fahrzeugkomponenten verwendet (s. Gaswerte). Der Datentyp *single* wurde deshalb gewählt, um auch wie von Team StarCraft e.V. gefordert als Input Werte mit mehreren Nachkommastellen realisieren zu können. Die auf analoge Weise in Abbildung 3.20 mit Constant- oder RSI-Blöcken erzeugten 386 Input-Signale werden schließlich im Umkehrschluss zur obigen Beschreibung über mehrere Bussysteme und Subsysteme zu einem Busarray im Signalgenerator zusammengefasst, welcher nunmehr dieses an den Signalkollektor übergibt.

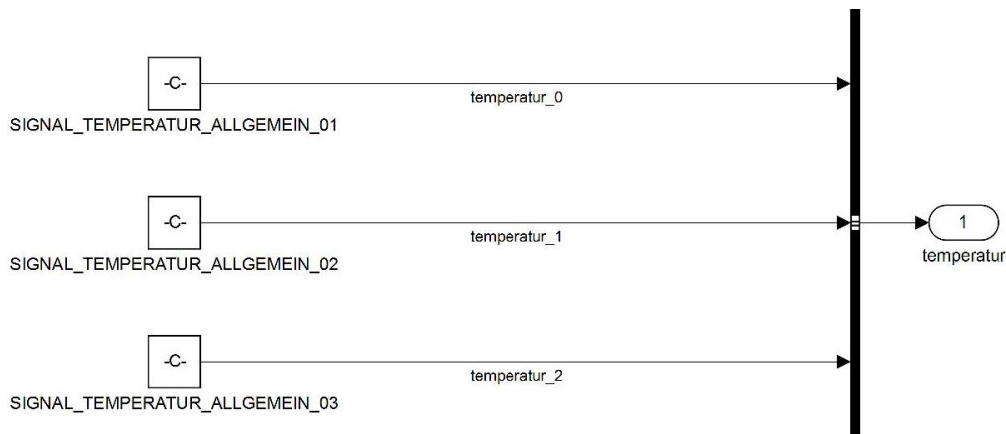


Abbildung 3.20: Signalerzeugung im Subsystem “Temperaturen”

### Config-Datei „signalgenerator\_microautobox.m”

Um eine mögliche Änderung der Testsignale zu vereinfachen und eine übersichtliche Darstellung aller Testsignale zu realisieren, sorgt eine Konfigurationsdatei in Matlab für die Spezifizierung der Testsignale des Simulink-Modells. Die in dem \*.m-File aufgeführten Parameter für die Zeit- und Funktionsvektoren dienen als Referenz für die signalerzeugenden Blöcke, welche über den Workspace von Matlab auf diese nach dem Ausführen der Datei zugreifen können (s. Abb. 3.21). Hierbei sollte jedoch beachtet werden, dass vor dem Kompilieren des Simulink-Modells und der Implementierung ebendiesem auf der MicroAutoBox II einmalig das \*.m-File ausgeführt werden muss. Demzufolge muss auch nach dem Ändern von Parametern die Datei neu ausgeführt werden, damit bei einer erneuten Implementierung des Simulink-Modells die geänderten Parameter korrekt übernommen werden können. Zudem wurde jeder Funktionsvektor mit dem Präfix „SIGNAL\_” und jeder Zeitvektor mit dem Präfix „TIME\_” versehen, um die Benennung der Parameter im \*.m-File zu standardisieren.

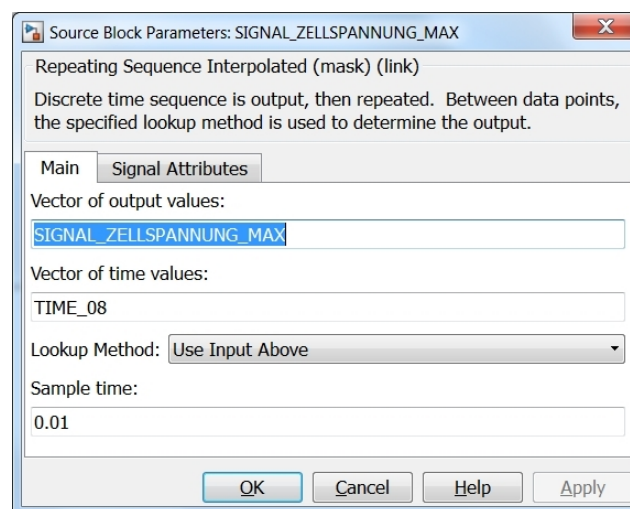


Abbildung 3.21: Referenzierung im RSI-Block



### 3.2.3 Signalkollektor

Der Signalkollektor stellt wie bereits erwähnt das zweite große Subsystem innerhalb des Simulink-Modells dar. Er hat die Aufgabe, den vom Subsystem „Signalgenerator“ (s. 3.2.2) oder von einem späteren Subsystem von Team StarCraft e.V. erhaltenen Busarray wieder in die einzelnen Signale zu unterteilen, ggf. geeignet aufzubereiten und diese dann an eine UDP-Schnittstelle innerhalb des Subsystems weiterzuleiten, welche die erzeugten Testsignale bzw. Testdaten an den Embedded-PC sendet.

#### Struktur des Subsystems

Das Subsystem selbst besitzt die folgende innere Struktur:

In einem ersten Schritt wird das Busarray wieder in den Kategorien A) - E) der Fahrzeugdaten entsprechenden Signalgruppen aufgeteilt und den fünf Subsystemen (analog zu Abb. 3.18) zugeführt. Dort werden die nunmehr fünf Busarrays wieder über mehrere Busselektoren und Subsysteme hinweg in die 386 einzelnen Signale aufgelöst, welche somit einzeln aufbereitet werden können (s. Abb. 3.22).

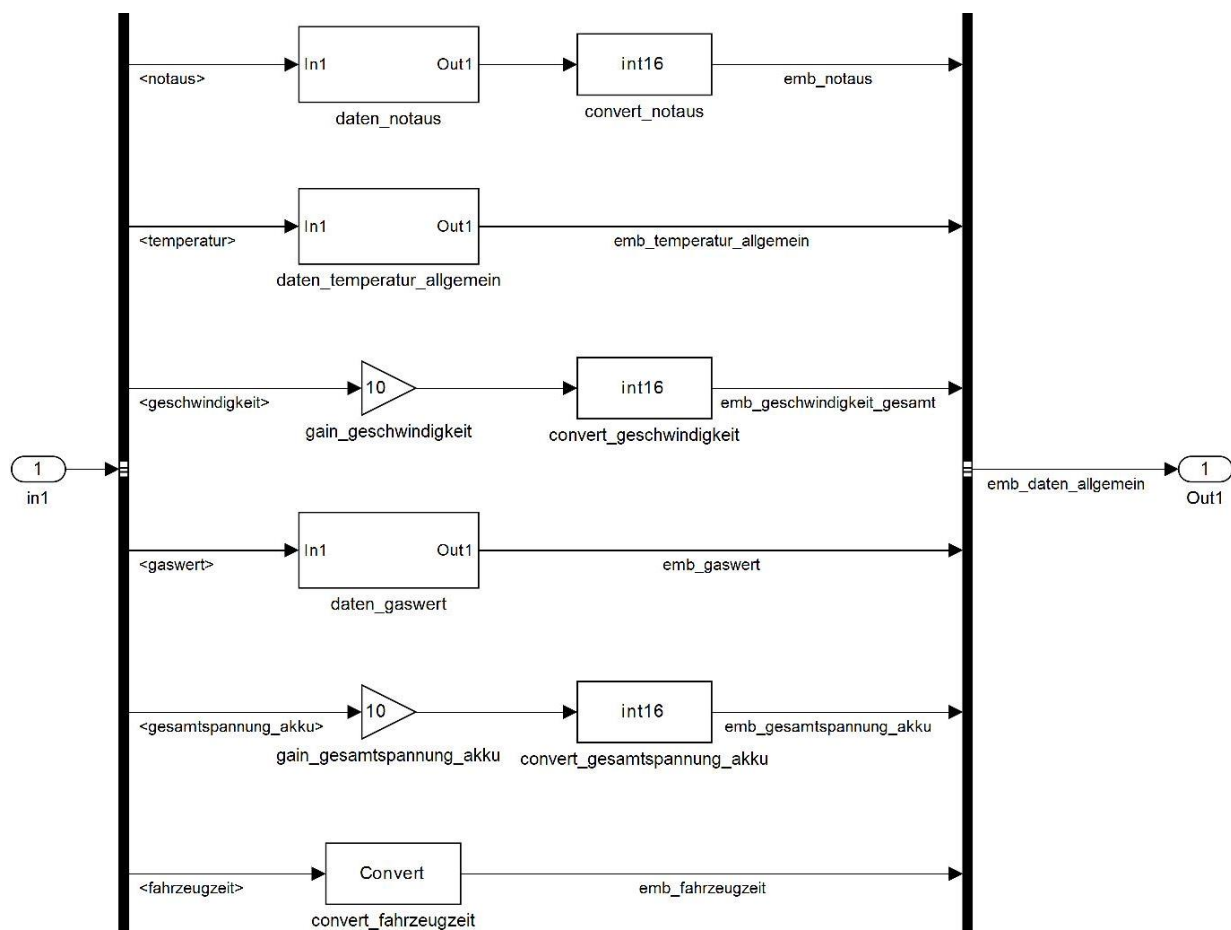


Abbildung 3.22: Signalaufbereitung im Subsystem „daten.allgemein“

Während der Aufbereitung der Signale werden die folgenden Schritte durchgeführt:

1) *Verstärkung der Signale*

Abhängig von der Anzahl der Nachkommastellen des Testsignals  $n$  mit  $n > 0$  wird dieses nun durch einen Gain-Block (s. Abb. 3.13) mit dem Faktor  $10^n$  multipliziert, um für den aktuellen Wert des Testsignals jeweils einen ganzzahligen Wert zu erhalten, was zur Vereinheitlichung der zu übertragenden Daten beiträgt.

2) *Konvertierung der Datentypen*

Die nunmehr ganzzahligen Werte werden von ihren Datentypen *boolean* oder *single* nun einheitlich mittels eines Convert-Blocks (s. Abb. 3.8) in den Datentyp *int16* konvertiert, wodurch alle Signale den gleichen Datentyp aufweisen.

3) *Umbenennung der Variablennamen*

Mit dem Convert-Block ist es zudem möglich, dem Ausgangssignal unabhängig vom Eingangssignal einen festen bzw. neuen Variablennamen zu vergeben. Dies ist insofern nützlich, da bei einem möglichen Zugriff auf die Daten bzw. die Variablen durch den Embedded-PC diese immer die gleichen Variablennamen besitzen, unabhängig davon ob der Signalgenerator durch das Subsystem von Team StarCraft e.V. ersetzt wurde oder nicht.

Nachdem die Daten anhand der obigen Schritte aufbereitet wurden, werden diese erneut durch mehrere Bus Creator und Subsysteme auf einem zentralen Bus Creator zu einem Busarray gebündelt und auf einen Bus to Vector - Block (s. Abb. 3.9) gegeben, welcher das Busarray in einen Vektor umwandelt. Anschließend wird dieser Vektor einem Encoder (s. Abb. 3.14) zugeführt.

### 3.2.4 Enkodierung

Die Enkodierung der Signale erfolgt in einem DSEncode32-Block von dSPACE (s. Seite 12). Dort werden die Signale so aufbereitet, dass sie über eine UDP-Schnittstelle übertragen werden können.

### 3.2.5 UDP-Schnittstelle

Die UDP-Schnittstelle wird mithilfe zweier Simulink-Blöcke realisiert (s. Seite 12).

## 3.3 Embedded-PC

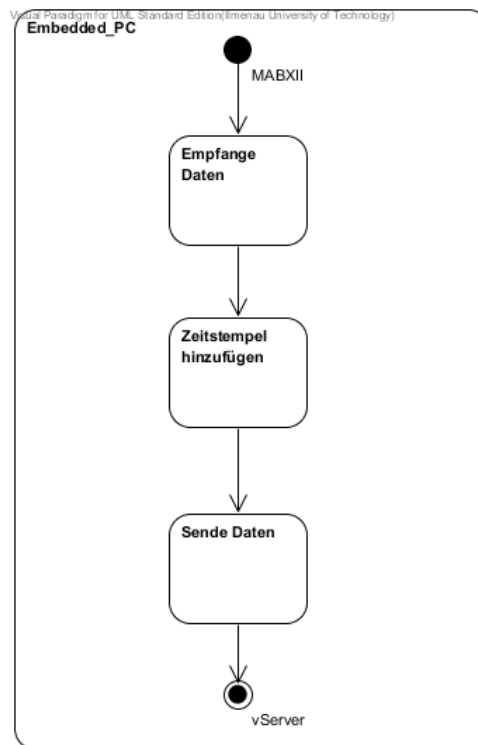


Abbildung 3.23: Aktivitätsdiagramm des Embedded-PCs

### 3.3.1 Empfangen der Fahrzeugdaten

Der Embedded-PC lauscht an einem fest definierten Port nach Paketen, welche zuvor von der MicroAutoBox II über eine Ethernet-Verbindung verschickt wurden. Die Übertragung erfolgt mittels des verbindungslosen Netzwerkprotokolls UDP und wird mithilfe von Unix Domain Sockets (SOCK\_DGRAM) realisiert.

### 3.3.2 Hinzufügen eines Zeitstempels

Da es nicht ausgeschlossen ist, dass sich Pakete bei der Übertragung überholen und/oder aufgrund einer schlechten Verbindung erst sehr spät das Ziel erreichen, erhält ein Paket sobald es entgegengenommen wird einen Zeitstempel, um im späteren Verlauf veraltete Pakete identifizieren zu können.

### 3.3.3 Senden der Fahrzeugdaten

Im Embedded-PC werden die Daten - bis auf das Hinzufügen eines Zeitstempels - nicht verändert. Sie werden also lediglich durchgereicht und mittels Unix Domain Sockets (SOCK\_DGRAM) an den virtuellen Server über eine GPRS/UMTS-Verbindung übertragen.

## 3.4 Virtueller Server

Im Folgenden wird die Funktionsweise des virtuellen Servers anhand eines Aktivitätsdiagrammes modelliert. Innerhalb des Programmes werden die Paketdaten in einem Datenobjekt gespeichert, in dem bereits grundlegende Funktionen (z.B. Abfrage des Embedded-PC-Zeitstempels) implementiert sind. Das Objekt wird in den einzelnen Schritten so aufbereitet, dass es letztendlich in die Datenbank geschrieben werden kann. Zur besseren Fehlerbehandlung in der Implementierungs- und Testphase wird eine Logging-Klasse eingerichtet. Diese wird im fertigen Programm nicht mehr zur Verfügung stehen, die Aktivierung erfolgt während der Kompilierung über Makros.

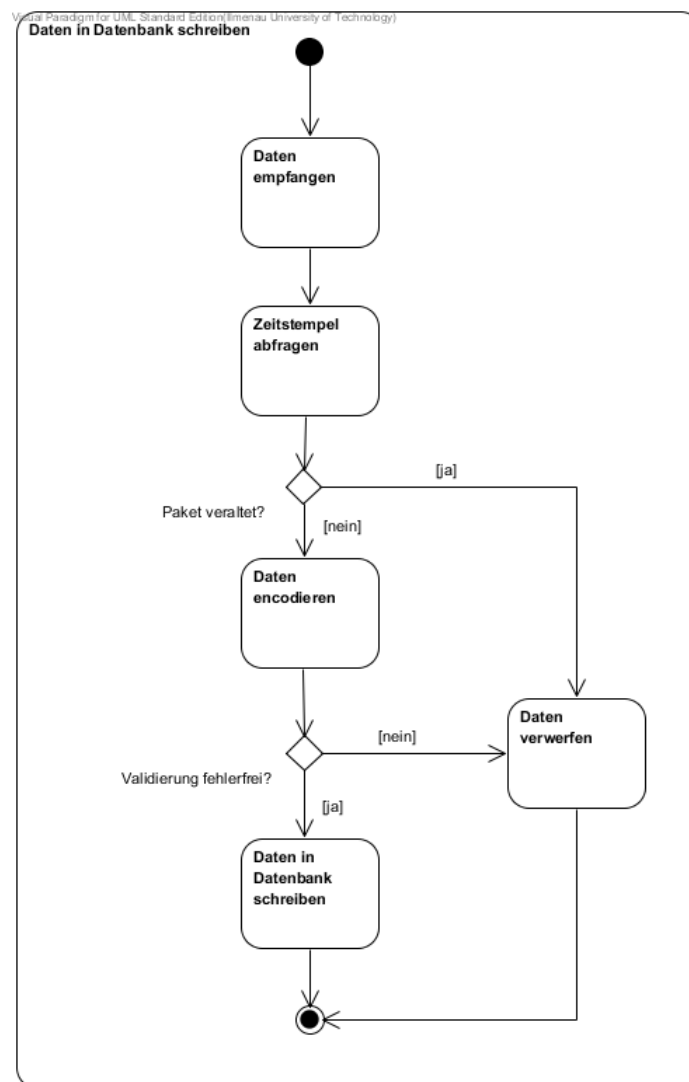


Abbildung 3.24: Aktivitätsdiagramm des virtuellen Servers

### 3.4.1 Empfangen der Fahrzeugdaten

Durch die starke Ähnlichkeit in der Funktionsweise mit 3.3.1 werden sich diese Abschnitte große, wenn nicht sogar alle Teile der Implementation teilen. Diese Maßnahme soll nicht nur dazu führen, dass Fehler in der Programmierung und Konzipierung reduziert werden, sondern auch, dass zusätzlicher Zeitaufwand vermieden wird.

### 3.4.2 Abfragen des Zeitstempels

Hier wird der in 3.3.2 hinzugefügte Zeitstempel abgefragt. Um veraltete Pakete identifizieren zu können wird in einer Variable der aktuellste Zeitstempel zwischengespeichert und mit allen ankommenden Paketen verglichen. Pakete, die zu alt sind (ein genauer Wert hierfür muss bei praktischen Versuchen ermittelt werden) werden verworfen.

### 3.4.3 Daten dekodieren

Die in 3.2.4 enkodierten Daten werden in diesem Schritt wieder dekodiert und so aufbereitet, dass sie in die Datenbank geschrieben werden können. Der Algorithmus zur Dekodierung kann aus einer von dSPACE erstellten sFunction (C Code) DSDecode32 ermittelt werden. Treten beim Dekodiervorgang keine Fehler auf, gehen wir davon aus, dass das Paket korrekt übertragen wurde und führen bis auf das Prüfen der Wertebereiche keine weitere Fehlerkontrolle durch. Falls Fehler auftreten, wird das Paket verworfen und eine Fehlermeldung in den entsprechenden Eintrag in der Datenbank geschrieben. Eine Fehlerkorrektur findet nicht statt. Die Daten liegen anschließend in einem Vektor bestehend aus *int16*-Werten vor.

### 3.4.4 Daten verwerfen

Daten, die veraltet sind oder nicht dekodiert werden konnten, werden verworfen.

### 3.4.5 Daten in Datenbank schreiben

Die Anfragen an die MySQL-Datenbank erfolgen mittels des MySQL Connector/C++. Dabei handelt es sich um eine C++ API die es erlaubt mittels TCP- oder UNIX-Sockets die Verbindung zu einer MySQL-Datenbank herzustellen und Anfragen zu übergeben.

## 3.5 Webseite

Anhand des nachfolgenden Diagramms soll das Verhalten der Webseite exemplarisch dargestellt werden. Ein registrierter Benutzer ruft in einem Webbrowser die Seite des Service-Interfaces auf und loggt sich anschließend über das ihm angezeigte Formular in das System ein. Anschließend wird er auf die Seite der allgemeinen Fahrzeugdaten weitergeleitet. Über das Menü steht ihm die Auswahl frei, sich die nach unterschiedlichen Kategorien sortierten Fahrzeugdaten anzeigen zu lassen. Am Beispiel der Akkudaten ruft der Nutzer im Menü über einen Klick auf den Reiter „Akkudaten“ die Seite `akkudaten.php` auf. Dieses PHP-Skript öffnet eine MySQL-Verbindung und führt nach einem erfolgreichen Verbindungsaufbau eine MySQL-Anfrage aus, die die aktuellsten Akkudaten aus der Fahrzeugdatenbank ermittelt und schließt die Verbindung anschließend wieder. Diese ermittelten Daten werden nun auf unterschiedlichster Art und Weise auf der Webseite dargestellt, sei es nun z.B. als Tabelle oder als einzeln hervorgehobene und gekennzeichnete Sonderinformationen. Ein JavaScript, welches gleichzeitig im Hintergrund läuft, sorgt dafür, dass die Daten alle  $1000ms$  aktualisiert werden ohne die komplette Seite vollständig neu zu laden. Dabei - und unter Berufung auf die weichen Echtzeitanforderungen - wird die Aktualisierung mittels Timeout realisiert. Anstelle den Server alle  $1000ms$  aufzufordern neue Daten zu schicken, wird nach dem Erhalt der Daten  $1000ms$  gewartet bis eine neue Anfrage gestartet wird. Dies hat zum Vorteil, dass der Server nicht unnötig belastet wird. Der Benutzer hat jederzeit die Möglichkeit durch die Auswahl eines anderen Reiters im Menü sich andere Fahrzeugdaten darstellen zu lassen. Das Verhalten läuft analog ab. Vor dem Verlassen wird der Benutzer angehalten sich auszuloggen und seine aktuelle Sitzung somit zu beenden.

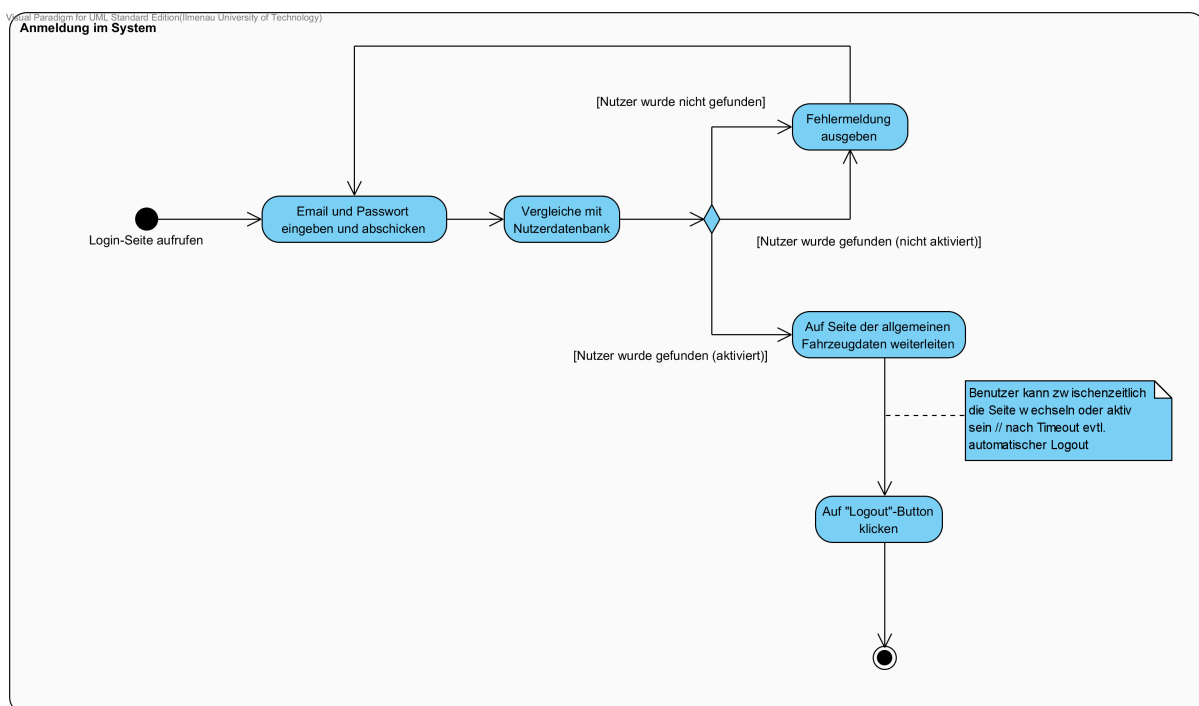


Abbildung 3.25: Aktivitätsdiagramm des Anmeldevorgangs

### 3.5.1 Registrierung

Durch das Aufrufen der Seite des Team StarCraft e. V. lässt sich das Service-Interface für das Formula-Student-Fahrzeug erreichen. Der Besucher erhält eine Aufforderung sich einzuloggen. Es können hierbei unterschiedliche Fälle eintreten (s. Abb. 3.25):

- *Fall 1: Der Besucher ist noch nicht registriert.*

Sollte er sich noch nicht registriert haben, kann er dies über einen Klick auf einen „Hier registrieren“ - Button tun. Der Besucher wird auf eine neue Seite weitergeleitet, die ihm mehrere Formularfelder zur Verfügung stellt. Neben dem Namen (Vor- und Nachname) wird er noch aufgefordert seine E-Mail-Adresse und ein Passwort eingeben. Das eingegebene Passwort muss aus mindestens sechs Zeichen (erlaubt sind Buchstaben in Groß- und /oder Kleinschreibung sowie Ziffern und Sonderzeichen, wobei von jeder Sorte mindestens eines enthalten sein muss) bestehen und muss einmal wiederholt werden. Anschließend kann der Besucher das Anmeldeformular durch einen Klick auf einen Button abschicken. Falls alle Eingaben den Vorgaben entsprechen, werden die Formulardaten in die Nutzerdatenbank eingetragen. Andernfalls wird der Nutzer aufgefordert sich mit anderen Daten zu registrieren. Dabei wird das Passwort in verschlüsselter Form und nicht im Klartext abgespeichert. Es wird zudem der Status des Benutzers auf „nicht aktiviert“ gesetzt. Anschließend wird eine E-Mail an den Vorstand generiert, die ihn auf die neue Registrierung im System hinweist. Der Vorstand erhält die Möglichkeit die Daten (mit Ausnahme des Passwortes) der registrierten Nutzer einzusehen und deren Status von „nicht aktiviert“ auf „aktiviert“ zu setzen oder einzelne Benutzer aus dem System zu löschen. Im Falle der Aktivierung des Accounts generiert das System eine E-Mail, die es an die eingetragene E-Mail-Adresse versendet und den Nutzer über die erfolgreiche Aktivierung seines Accounts hinweist.

### 3.5.2 Anmeldung

- *Fall 2: Der Benutzer ist bereits angemeldet.*

Zum Anmelden im System ruft der Nutzer die Login-Seite auf und wird anschließend aufgefordert seine E-Mail-Adresse und sein Passwort einzugeben. Durch einen Klick auf den Login-Button werden die eingegebenen Daten mit der Nutzerdatenbank verglichen. Sollte die Konstellation aus E-Mail-Adresse und dazugehörigem Passwort in der Nutzerdatenbank nicht entdeckt werden, gibt das System eine Fehlermeldung aus, die den Nutzer darauf hinweist, dass die von ihm eingegebenen Daten inkorrekt sind. Sollte das System hingegen einen passenden Eintrag finden, wird abhängig davon, ob der Nutzer bereits aktiviert ist, entweder eine Fehlermeldung (Fall „nicht aktiviert“) ausgegeben oder der Nutzer auf die Seite der allgemeinen Fahrzeugdaten weitergeleitet (Fall „aktiviert“). Der Nutzer kann nun auf der Seite nach Belieben, aber unter Berücksichtigung der ihm gegebenen Rechte, auf der Seite des Service Interfaces navigieren. Sollte in einer gewissen Zeitspanne keine Aktivität vom Benutzer erkannt werden, wird er automatisch aus dem System ausgeloggt. Ansonsten steht ihm die Möglichkeit nach getaner Arbeit offen, sich über den „Logout“ - Button auszuloggen und die Seite zu verlassen.



Abbildung 3.26: Aktivitätsdiagramm des „Passwort vergessen“ - Vorgangs

### 3.5.3 Passwort vergessen

- *Fall 3: Passwort vergessen*

Tritt der Fall ein, dass ein im System registrierter Benutzer sein Passwort vergisst, bietet das Nutzersystem ihm die Möglichkeit, sich ein neues Passwort zu setzen. Nach dem Aufruf der Login-Seite muss der Nutzer hierfür auf den Button „Passwort vergessen“ klicken und wird anschließend auf eine Seite weitergeleitet, die ihn auffordert seine E-Mail-Adresse einzugeben. Durch die Bestätigung der E-Mail-Adresse durch den Benutzer prüft das Nutzersystem im Hintergrund, ob ein Eintrag zu dieser E-Mail-Adresse existiert. Falls nicht, wird der Vorgang abgebrochen. Der Benutzer erhält aus Sicherheitsgründen kein Feedback darüber. Sollte ein entsprechender Nutzer existieren, generiert das System einen Token, den es mitsamt eines Aktivierungslinks an die angegebene E-Mail-Adresse verschickt. Auf der Aktivierungsseite wird der Nutzer erneut aufgefordert seine E-Mail-Adresse und den Token anzugeben. Des Weiteren kann er sich an dieser Stelle ein neues Passwort auswählen (selbe Bedingungen wie bei der erstmaligen Registrierung), welches anschließend das alte Passwort im System überschreibt.



### 3.5.4 Benutzerverwaltung

Der Vorstand besitzt als einziger Nutzer im System die Möglichkeit der Verwaltung aller registrierten Systemnutzer. Hierfür steht ihm neben den Übersichtsseiten für die einzelnen Kategorien an Informationen eine spezielle Seite „Benutzerverwaltung“ zur Verfügung. Diese Seite stellt die Daten aller angemeldeten Nutzer dar und gibt Auskunft über deren Status im System, also ob sie bereits aktiviert worden sind oder eine Aktivierung noch aussteht. Gelöschte Benutzer werden nicht angezeigt. Ein Aufruf der Seite der Benutzerverwaltung resultiert in einem Verbindungsaufbau mit der Nutzerdatenbank. Bei erfolgreichem Verbindungsaufbau erfolgt anschließend das Versenden eines MySQL-Query, das als Resultat alle im System befindlichen Nutzer ausgibt. Anschließend wird die MySQL-Verbindung wieder getrennt. Die Ausgabe beinhaltet dabei Informationen wie Vor- und Nachname, Rechtegruppe, Accountstatus und eine Information über den letzten Zeitpunkt einer Aktivität des Nutzers. Mittels verschiedener Buttons erhält der Vorstand die Möglichkeit alle Nutzer komplett zu verwalten, d.h. sie aus dem System zu entfernen, sie nach einer Registrierung zu aktivieren oder abzulehnen (und somit zu löschen). Weiterhin kann ihre Rechtegruppenzugehörigkeit festgelegt bzw. geändert werden. Ist der Vorstand fertig mit der Verwaltung der Nutzer, kann er entweder weiterhin auf der Seite navigieren und arbeiten oder sich ausloggen und die Seite im Anschluss verlassen.

## 3.6 Datenbanken

### 3.6.1 Nutzerdatenbank

Die Aufgabe der Nutzerdatenbank ist das Aufbewahren aller Informationen der registrierten Benutzer. Sie besteht aus drei Tabellen (s. Abb. 3.27), welche im Folgenden erläutert werden sollen:



Abbildung 3.27: Entwurf der Nutzerdatenbank

- 1) *rechte*: In dieser Tabelle befinden sich sämtliche verfügbaren Rechte-Stufen, die sich aus einer eindeutigen ID und einer Bezeichnung in Textform zusammensetzen.
- 2) *benutzer*: Alle registrierten Nutzer werden in dieser Tabelle gespeichert. Dabei besitzt jeder Nutzer eine eindeutige ID. Diese wird durch automatisches Inkrementieren von der Datenbank erzeugt. Mit der E-Mail-Adresse zusammen bilden beide den Primärschlüssel zur eindeutigen Identifikation eines Nutzers. Des Weiteren gehören zu einem Benutzereintrag noch ein Vor- und ein Nachname. Das vom Benutzer bei seiner Registrierung gewählte Passwort wird in gehashter Form als String gespeichert. Ein einzelnes Bit gibt zudem Auskunft darüber, ob der Nutzer bereits vom Vorstand aktiviert wurde und somit Zugang zu den Fahrzeugdaten erhalten hat. Das Feld „rechte“ ist Fremdschlüssel zur Tabelle „rechte“ und beinhaltet eine der dort befindlichen IDs.
- 3) *online\_benutzer*: Diese Tabelle speichert Nutzer-IDs und Zeitstempel und gibt somit Auskunft über den Zeitpunkt der letzten Aktivität eines Nutzers.

### 3.6.2 Fahrzeugdatenbank

Die zweite Datenbank beinhaltet die aus dem Fahrzeug entnommenen Daten und bildet gleichzeitig auch die Schnittstelle zur Webseite. Sie besteht aus fünf Tabellen, die jeweils sämtliche Informationen einer Kategorie in sich kapseln (s. Abb. 3.28). Alle Tabellen besitzen über ihre Daten hinaus jeweils noch ein Feld, in dem sich ein Zeitstempel befindet und eines, das für Testzwecke Informationen speichern kann. Anhand des (im Fahrzeug erzeugten) Zeitstempels lassen sich die Datentupel der einzelnen Tabellen einander zuordnen. Im Folgenden werden die Informationen der einzelnen Tabellen erläutert:

<b>fahrdynamikregelung</b> <ul style="list-style-type: none"> <li>↕ Antriebschlupfregelung INT(3)</li> <li>↕ TorqueVectoring01 INT(3)</li> <li>↕ TorqueVectoring02 INT(3)</li> <li>↕ TorqueVectoring03 INT(3)</li> <li>↕ Lenkwinkel SMALLINT(3)</li> <li>↕ Zeitpunkt INT(8)</li> <li>↕ FehlerFeld VARCHAR(45)</li> </ul>	<b>motor_umrichterdaten</b> <ul style="list-style-type: none"> <li>↕ DCStrom DECIMAL(4,1)</li> <li>↕ DCSpannung DECIMAL(5,1)</li> <li>↕ Motortemperatur01 DECIMAL(3,1)</li> <li>↕ Motortemperatur02 DECIMAL(3,1)</li> <li>↕ Motortemperatur03 DECIMAL(3,1)</li> <li>↕ Motortemperatur04 DECIMAL(3,1)</li> <li>↕ Motortemperatur05 DECIMAL(3,1)</li> <li>↕ Motortemperatur06 DECIMAL(3,1)</li> <li>↕ Motortemperatur07 DECIMAL(3,1)</li> <li>↕ Motortemperatur08 DECIMAL(3,1)</li> <li>↕ Stromgrenze DECIMAL(4,1)</li> <li>↕ Maximalleistung DECIMAL(4,1)</li> <li>↕ Lüfterdrehzahl INT(4)</li> <li>↕ Lüfter SMALLINT(4)</li> <li>↕ Pumpe SMALLINT(4)</li> <li>↕ Wassertemperatur DECIMAL(4,1)</li> <li>↕ Zeitpunkt INT(8)</li> <li>↕ FehlerFeld VARCHAR(45)</li> </ul>	<b>dynamische_daten</b> <ul style="list-style-type: none"> <li>↕ XGeschwindigkeit DECIMAL(4,1)</li> <li>↕ YGeschwindigkeit DECIMAL(4,1)</li> <li>↕ ZGeschwindigkeit DECIMAL(4,1)</li> <li>↕ XBeschleunigung DECIMAL(4,1)</li> <li>↕ YBeschleunigung DECIMAL(4,1)</li> <li>↕ ZBeschleunigung DECIMAL(4,1)</li> <li>↕ XGierrate DECIMAL(4,1)</li> <li>↕ YGierrate DECIMAL(4,1)</li> <li>↕ ZGierrate DECIMAL(4,1)</li> <li>↕ DrehzahlVRL DECIMAL(5,1)</li> <li>↕ DrehzahlVRR DECIMAL(5,1)</li> <li>↕ DrehzahlHRL DECIMAL(5,1)</li> <li>↕ DrehzahlHRR DECIMAL(5,1)</li> <li>↕ Wassertemperatur01 DECIMAL(4,1)</li> <li>↕ Wassertemperatur02 DECIMAL(4,1)</li> <li>↕ Bremsdruck SMALLINT(3)</li> <li>↕ Bremskraft DECIMAL(4,1)</li> <li>↕ Bremsposition DECIMAL(4,1)</li> <li>↕ Federweg TINYINT(3)</li> <li>↕ Gaspedalstellung DECIMAL(4,1)</li> <li>↕ Lenkwinkel SMALLINT(3)</li> <li>↕ Zeitpunkt INT(8)</li> <li>↕ FehlerFeld VARCHAR(45)</li> </ul>	<b>allgemeine_fahrzeugdaten</b> <ul style="list-style-type: none"> <li>↕ StatusNotaus BINARY(10)</li> <li>↕ Temperatur01 SMALLINT(4)</li> <li>↕ Temperatur02 SMALLINT(4)</li> <li>↕ Temperatur03 SMALLINT(4)</li> <li>↕ Geschwindigkeit DECIMAL(4,1)</li> <li>↕ Gaswert01 TINYINT(3)</li> <li>↕ Gaswert02 TINYINT(3)</li> <li>↕ AkkuGesamtspannung SMALLINT(5)</li> <li>↕ AktuelleFahrzeugzeit INTEGER(10)</li> <li>↕ Zeitpunkt INT(8)</li> <li>↕ FehlerFeld VARCHAR(45)</li> </ul>	<b>akkudaten</b> <ul style="list-style-type: none"> <li>↕ zellaten001 INT</li> <li>...</li> <li>↕ zellaten144 INT</li> <li>↕ MaxZellspannung DECIMAL(4,3)</li> <li>↕ MinZellspannung DECIMAL(4,3)</li> <li>↕ StromLadegerät DECIMAL(5,1)</li> <li>↕ Balancing BINARY(144)</li> <li>↕ Zelltemperatur01 DECIMAL(4,1)</li> <li>...</li> <li>↕ Zelltemperatur48 DECIMAL(4,1)</li> <li>↕ Zeitpunkt INT(8)</li> <li>↕ FehlerFeld VARCHAR(45)</li> </ul>
--	--	--	---	---

Abbildung 3.28: Entwurf der Fahrzeugdatenbank

- 1) *fahrdynamikregelung*: Diese Tabelle beinhaltet, wie der Name bereits verrät, alle Informationen bezüglich der Fahrdynamikregelung. Sie hält aktuelle Informationen über den Status der Antriebsschlupfregelung, dem Torque-Vectoring in  $x$ -,  $y$ - und  $z$ -Richtung sowie den Lenkwinkel.
- 2) *motor\_umrichterdaten*: Alle Daten, die den Motor und Umrichter betreffen, werden hier gespeichert. Informationen über den aktuell anliegenden Strom oder die aktuell anliegende Spannung sowie die Informationen aller acht Temperatursensoren des Motors lassen sich hier wiederfinden. Ebenso auch Informationen, die die Kühlung des Motors betreffen, wie Lüfterdrehzahlen, Pumpenleistung, Wassertemperatur und eingestellte Stromgrenzen.
- 3) *dynamische\_daten*: Alle Informationen, die die dynamischen Aspekte des Fahrzeugs betreffen, sind hier enthalten. Dies sind beispielsweise die Geschwindigkeiten, Beschleunigungen und Gierraten jeweils in  $x$ -,  $y$ - und  $z$ -Richtung, genau wie die Drehzahlen jedes einzelnen Rads. Desweiteren finden sich hier auch Informationen zu den Bremsen, wie Bremsdruck, -kraft und -position wieder. Auch der Lenkwinkel, der Federweg und die Gaspedalstellung ergänzen das Set der Daten.
- 4) *allgemeine\_fahrzeugdaten*: Die wichtigsten und allgemeinsten Informationen des Fahrzeugs befinden sich in dieser Tabelle. Neben den wichtigen Informationen wie z.B. den Status der einzelnen Notaus-Schalter des Fahrzeuges, finden sich hier die Werte der drei wichtigsten Temperatursensoren, die aktuelle Fahrgeschwindigkeit, die Gaswerte der beiden Elektromotoren sowie die Gesamtspannung des Akkus und die aktuelle Fahrzeit.
- 5) *akkudaten*: Dies ist die letzte und zugleich auch die umfangreichste Tabelle. Hier befinden sich sämtliche Informationen über die im Fahrzeug verbauten Akkus. Neben globaleren Informationen wie die maximale und minimale Zellspannung und dem Strom bzw. der Spannung zum Ladegerät, finden sich die Zelldaten aller 144 Akkuzellen sowie die Temperaturen aller 48 Temperatursensoren wieder. Das Feld „Balancing“ gibt darüber hinaus Informationen darüber, ob für die einzelnen Zellen die Balancing-Option aktiviert oder deaktiviert ist.

## 4 Testdrehbuch

Im Folgenden wird der Ablauf der durchzuführenden Test geschildert. Dabei wird im ersten Teil darauf getestet, dass unter optimalen Umständen alle Funktionen korrekt ausgeführt werden. Im zweiten Teil wird anschließend das System auf dessen Robustheit geprüft, wobei auftretende Grenzfälle erkannt und korrekt behandelt werden sollen.

### 4.1 Test auf Funktionalität

#### **1) In Matlab/Simulink werden Testwerte generiert, die innerhalb der vereinbarten Wertebereiche liegen**

1. /TM100/ Es wird geprüft, dass alle Daten korrekt gebündelt und an die Ethernet-Schnittstelle der MikroAutoBox II weitergeleitet werden.
2. /TE120/TE300/  
Am Embedded-PC wird der Empfang der Datenpakete überprüft.
3. /TE100/ Bei einem Neustart durch Unterbrechung der Stromzufuhr des Embedded-PC soll dieser selbstständig die Verbindung zur MicroAutoBox II wiederaufbauen, sowie das GPRS/UMTS-Modul aktivieren und die Verbindung zum Webserver erneut etablieren.
4. /TE200/ Die Signalqualität und Stabilität der GPRS/UMTS-Verbindung wird im Ruhezustand, bei Bewegung und bei großer Entfernung zum Sendemast bzw. bei schlechter Empfangsqualität überprüft.
5. /TE300/ Es wird zu TE200 auf dem virtuellen Server überprüft, ob die Testdaten vollständig und korrekt über eine GPRS/UMTS-Verbindung an ebendiesen übermittelt wurden.
6. /TW200/ Nach mehr als 10 Stunden kontinuierlichen Datentransfers sollen die Fahrzeugdaten der letzten 10 Stunden in der Datenbank vorliegen. Den Ausgangspunkt für dieses Zeitintervall stellt den Zeitpunkt der Datentnahme dar.
7. /TW210/ Der korrekte Export der in der Datenbank enthaltenen Fahrzeugdaten als CSV-Datei wird auf dessen Fehlerfreiheit überprüft.
8. /TW100/TW110/  
Auf der Webseite wird die korrekte Visualisierung und Aufbereitung der Daten überprüft. Darin ist ebenfalls eine Überprüfung der fehlerfreien Darstellung auf den zuvor spezifizierten Browsern eingeschlossen.

**II) Unabhängig davon können die Verwaltungsprozesse auf der Webseite überprüft werden:**

1. /TW440/      Beim Aufrufen der Webseite wird der Nutzer zur Anmeldung aufgefordert.
2. /TW450/      Bei einem nicht registrierten Nutzer schlägt der Anmeldeversuch fehl und eine Fehlermeldung wird ausgegeben (Nutzer nicht gefunden).
3. /TW451/      Ein registrierter Nutzer meldet sich mit E-Mail und Passwort an, ist aber noch nicht freigeschaltet (Status-Bit == 0): Der Anmeldeversuch schlägt fehl und eine Fehlermeldung wird ausgegeben (Nutzer noch nicht aktiviert).
4. /TW452/      Ein registrierter Nutzer meldet sich mit E-Mail und Passwort an (Status-Bit == 1): der Vorgang verläuft ohne Fehler.
5. /TW400/TW410/TW420/TW421/TW430/  
Eine Registrierung mit E-Mail und Passwort (mind. sechs Zeichen) verläuft ohne Fehler. Der Nutzer wird daraufhin auf die Startseite weitergeleitet.
6. /TW400/TW410/TW420/TW421/TW430/  
Bei erfolgreichem Registrierungsversuch wird der Vorstand über das System per E-Mail informiert und die Daten der Registrierung in die Nutzerdatenbank geschrieben. Das Status-Bit des Nutzers ist 0.
7. /TW400/TW410/TW420/TW421/TW430/  
Es wird darauf getestet, das beim Akzeptieren durch den Vorstand der Nutzer auch tatsächlich für die Anmeldung freigeschaltet wird. Das Status-Bit wird auf 1 gesetzt. Der Nutzer wird per E-Mail über dessen Freischaltung informiert.
8. /TW400/TW410/TW420/TW421/TW430/  
Es wird darauf getestet, dass beim Abweisen eines Registrierungs-gesuches durch den Vorstand auch alle Daten des Registrierungs-vorgangs aus der Nutzer-Datenbank gelöscht werden.
9. /TW400/TW410/TW420/TW421/TW430/  
Beim Zuweisen von Rechten an registrierte Nutzer werden diese Änderung korrekt in die Benutzerdatenbank übernommen.
10. /TW460/      Registrierte Nutzer, die mit Rechten ausgestattet wurden, können diese ohne Einschränkungen nutzen.

- 11. /TW421/    Wird ein Nutzer durch den Vorstand gelöscht, so werden alle Daten zu diesem Nutzer aus der Nutzer-Datenbank gelöscht.
  
- 12. /TW470/    Wird die „Passwort vergessen“ - Funktion genutzt und eine E-Mail-Adresse eingegeben, die in der Benutzerdatenbank steht, dann wird eine E-Mail an die Adresse gesendet. Inhalt der E-Mail ist ein zufälliges Token, sowie ein Link zur Passwortänderung.
  
- 13. /TW480/    Es wird geprüft, ob der Link aus /TW470/ gültig ist und auf die Aktivierungsseite der Passwortwiederherstellung führt.
  
- 14. /TW490/    Wird auf der Aktivierungsseite der Passwortwiederherstellung die E-Mail-Adresse, das Token und das neue Passwort eingegeben, soll die Änderung korrekt in die Benutzerdatenbank übernommen werden.

**III) Folgende Tests müssen auf dem virtuellen Server durchgeführt werden:**

- 1. /TS010/    Die Dekodierung der vom Embedded-PC über eine GPRS/UMTS-Verbindung versendeten Datenpakete verläuft ohne Probleme.
  
- 2. /TS020/    Es wird anschließend getestet, ob die nunmehr dekodierten Datensätze anschließend korrekt aufbereitet werden (für die Durchführung der Aufbereitung s. 3.4.3).
  
- 3. /TS030/    Es wird darauf geprüft, dass bei längerer Laufzeit des Programms für die Paketverarbeitung der Speicherbedarf nicht unbegrenzt steigt.

## 4.2 Test auf Robustheit

### I) In Matlab/Simulink werden Testwerte generiert, die außerhalb der vereinbarten Wertebereiche liegen

1. /TR010/ Das für die Plausibilitätsüberprüfung auf dem virtuellen Server zuständige Modul (s. 3.4.3) erkennt die Werte, welche außerhalb des Wertebereiches liegen und leitet diesbezüglich eine Warnmeldung an die Webseite weiter.
2. /TR020/ Auf der Webseite wird daraufhin die aufgetretene Überschreitung der Wertebereiche bei den jeweiligen Werten für den Nutzer kenntlich gemacht.

### II) Tests auf der Webseite

1. /TW400/TW410/TW420/TW421/TW430/  
Eine Registrierung mit E-Mail und Passwort, welches weniger als sechs Zeichen aufweist, resultiert in einer Fehlermeldung.
2. /TR030/ Bei einer Anmeldung mit inkorrektter Email-Adresse und/oder falschem Passwort soll fehlschlagen und eine Fehlermeldung ausgegeben werden.
3. /TR040/ Meldet sich ein noch angemeldeter Nutzer erneut an, soll dies fehlschlagen und eine Statusmeldung über den aktuellen Zustand (angemeldet / nicht angemeldet) des Nutzers ausgegeben werden.
4. /TR050/ Falls dem Nutzer, nachdem er den „Passwort vergessen“ - Button ausgewählt hat, bei der Eingabe der E-Mail-Adresse zu Übersendung des Tokens und des Links zur Aktivierungsseite ein Fehler unterläuft, soll dies fehlschlagen und der Nutzer eine Statusmeldung darüber erhalten, dass seine eingegebene E-Mail-Adresse nicht im System existiert.

### III) Tests auf dem virtuellen Server:

1. /TS040/ Wenn durch äußere Einflüsse das Programm zum Absturz gebracht wurde, soll sich dieses Neustarten lassen können.

# Glossar

<b>Ajax</b>	Ajax ist ein Akronym für die Wortfolge „Asynchronous JavaScript and XML“. Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server.
<b>Apache-Server</b>	Apache (HTTP) Server bezeichnet eine modulare Webserversoftware.
<b>API</b>	Eine API (application programming interface) ist eine Programmierschnittstelle, die Funktionalitäten einer Software anderen Programmen zur Verfügung stellt.
<b>Architekturmuster</b>	Ein Architekturmuster ist eine aus Erfahrung entstandene und optimierte Schablone für häufig wiederkehrende Probleme, anhand derer man Software entwerfen kann.
<b>Balancing</b>	Der Begriff Balancer, zu deutsch (Zellen-Ladungszustands-) Ausgleicher oder Ausgleichsregler, bezeichnet ein elektrisches Gerät, das die gleichmäßige Ladung aller Zellen innerhalb eines Akkupacks gewährleistet.
<b>bidirektional</b>	Datenübertragung, die in beide Richtungen funktioniert.
<b>boolean</b>	Ein Datentyp für die Darstellung der Werte „wahr“ und „falsch“.
<b>Browser</b>	Ein spezielles Computerprogramm, das für die Darstellung von Webseiten oder Daten verwendet wird.
<b>Busarray</b>	Ein Busarray ist eine gebündelte und zusammengefasste Anzahl von Bussen.
<b>CAN-Bus</b>	Der CAN-Bus (Controlled Area Network) ist ein asynchrones, serielles Bussystem und gehört zu der Klasse der Feldbusse.
<b>CSS</b>	Cascading Style Sheets sind eine deklarative Sprache für Stilvorlagen (engl. stylesheets) von strukturierten Dokumenten. Sie werden vor allem zusammen mit HTML eingesetzt.
<b>CSV</b>	Das Dateiformat CSV steht für den englischen Begriff Comma-separated values (gelegentlich auch Character-separated values) und beschreibt den Aufbau einer Textdatei zur Speicherung oder zum Austausch einfach strukturierter Daten.



<b>C++</b>	C++ ist eine objektorientierte Programmiersprache.
<b>Datenbank</b>	Eine Datenbank ermöglicht die permanente Speicherung von Daten. Die genaue Realisierung der Aufbewahrung wird von der Datenbank verwaltet, die Daten kann man mit speziellen Anfragen extrahieren.
<b>DC</b>	Die Abkürzung DC (direct current) steht für Gleichstrom.
<b>Dongle</b>	Ein Gerät (hier: USB-Stick), das an einen Computer o.ä. angeschlossen wird, um eine Verbindung zu kontrollieren. Dies dient zum Datenschutz und zur Kontrolle von gültigen Lizenzen.
<b>Embedded-PC</b>	Ein modular aufgebauter Industrierechner, dessen besonderer Fokus auf Kompaktheit liegt.
<b>Ethernet</b>	Übertragungstechnologie für Daten durch kabelgebundene Netze.
<b>Fremdschlüssel</b>	Eintrag in der Tabelle einer Datenbank dessen Inhalt nur Werte annehmen darf, die in einer definierten anderen Tabelle vorkommen.
<b>Gierrate</b>	Die Gierrate bezeichnet die Geschwindigkeit der Drehung eines Fahrzeugs um die Hochachse.
<b>GPRS</b>	Der General Packet Radio Service ist eine Übertragungstechnologie für die Mobilfunkübertragung von Daten (langsamer als UMTS).
<b>GPS</b>	Das Global Positioning System ist eine Technologie zur Positionsbestimmung und Zeitmessung.
<b>GUI</b>	Das Graphical User Interface (zu deutsch: Benutzeroberfläche) ist die grafische Schnittstelle zwischen Software und dem Benutzer.
<b>Hashing</b>	Als Hashing bezeichnet man einen Vorgang, bei dem mittels einer Hashfunktion (auch Streufunktion) ein Eingabewert auf einen Ausgabewert abbildet. Man kann auf diese Weise sehr schnell an Daten kommen oder auch auf diese Weise Daten durch die Hashfunktion verschlüsselt speichern.
<b>HTML</b>	Die Hypertext Markup Language ist eine textbasierte Auszeichnungssprache zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks in Dokumenten und im Internet.
<b>Interface</b>	Schnittstelle, über die Daten von verschiedenen Komponenten ausgetauscht werden können.

<b>IP</b>	Ein verbindungsloses Internetprotokoll, welches die Grundlage der Kommunikation im Internet darstellt.
<b>JavaScript</b>	JavaScript ist eine Skriptsprache, die hauptsächlich für dynamisches HTML in Webbrowsern eingesetzt wird.
<b>Latenz</b>	Laufzeit von Signalen, die sich aus der Differenz von dem Absenden und Ankommen von Daten ergibt.
<b>Makro</b>	Ein Makro ist eine zusammengefasste Abfolge von Befehlen oder Deklarationen, die bei Aufruf sequentiell abgearbeitet werden.
<b>Matlab/Simulink</b>	Software, die hier verwendet wird, um die von der MicroAutoBox II bereitgestellten Daten zu verpacken und an den Embedded-PC weiterzuleiten.
<b>Matlab/Simulink-Block</b>	Mit der blockorientierten und graphischen Oberfläche von Simulink werden Gleichungen in Form von (Übertragungs-)Blöcken eingegeben und dargestellt.
<b>MicroAutoBox II</b>	Ein Echtzeitsystem, welches für schnelles Funktions-Prototyping in Fullpass- und Bypass-Szenarien geeignet ist.
<b>MySQL</b>	MySQL ist eines der weltweit am weitesten verbreiteten relationalen Datenbankverwaltungssysteme. Es bildet die Grundlage für viele dynamische Webauftritte.
<b>MySQL Connector/C++</b>	Diese Software ermöglicht einen Zugriff auf eine MySQL-Datenbank. Dabei wird ein Programm in der Sprache C++ eingebunden.
<b>MySQL-Query</b>	Eine für das MySQL-System geschriebene Anfrage in der Anfragesprache SQL zur Definition der gewünschten darzustellenden Datensätze.
<b>MySQL-Server</b>	Der MySQL-Server verwaltet die einzelnen Datenbanken und ermöglicht es, mehrere von diesen nebeneinander führen zu können. Es ist daher gleichzusetzen mit dem Begriff „MySQL-Datenbankverwaltungssystem“.
<b>Nutzersystem</b>	Verwaltung von Personen und deren Rechte im Rechtesystem.
<b>PHP</b>	PHP ist eine Skriptsprache, die hauptsächlich zur Erstellung dynamischer Webseiten oder Webanwendungen verwendet wird.

**PID-Regler**

Ein PID-Regler besteht aus 3 Teilen, einem P-Anteil, einem I-Anteil und einem D-Anteil. PI steht für proportional integral wirkend (wie beim PI-Regler) und D steht für differentiell wirkend.

**Port**

Der Port ist Bestandteil der Protokolle UDP und TCP.

**Primärschlüssel**

Primärschlüssel werden von Fremdschlüsseln referenziert. Dienen zur Eindeutigen Bestimmung von Tupeln in einer Tabelle.

**Recht**

Hier: Synonym für Erlaubnis.

**Rechtesystem**

Vergabe von Rechten für definierte Aktionen.

**Sampling**

Sampling ist Englisch für „Abtastung“. Hierbei wird ein kontinuierliches Signal in festgelegten Zeitabständen abgetastet und der Wert bestimmt.

**Service-Interface**

Ein Service Interface ist eine Mensch-Fahrzeug-Schnittstelle, die es dem Menschen ermöglichen soll auf alle wichtigen Daten zuzugreifen.

**SFTP**

Das Secure File Transfer Protocol ist ein Protokoll, das es ermöglicht, über eine verschlüsselte Verbindung Daten zwischen zwei Geräten auszutauschen.

**Skript**

Ein Skript ist eine mit einer Skriptsprache wie PHP oder JavaScript geschriebene Datei, die von einem Programm ausgeführt werden kann.

**Smartphone**

Ein Smartphone (zu dt. intelligentes Telefon) beschreibt die aktuelle Generation von Mobilfunktelefonen, deren Eingabe auf einem berührungsempfindlichen Bildschirm erfolgt.

**Socket**

Ein Socket ist eine plattformunabhängige Schnittstelle, mit der man ein Netzwerkprotokoll implementieren kann. Dies ermöglicht es Rechnern, über Systemgrenzen hinweg miteinander zu kommunizieren.

**SSL/TLS**

Secure Sockets Layer/Transport Layer Security (zu Deutsch: Transportsicherheitsschicht) TLS ist die Weiterentwicklung von SSL, einem Protokoll zur verschlüsselten Datenübertragung im Internet.

**Tablet-PC**

Ein tragbarer, flacher Computer in besonders leichter Ausführung mit einem Touchscreen-Display, wobei im Unterschied zu Notebooks auf eine physische Tastatur verzichtet wird.

**TCP/IP**

TCP/IP ist ein verbindungsorientiertes Kommunikationsprotokoll im Internet.

<b>Timeout</b>	Eine festgelegte Zeitgröße, die eine Aktion auslöst, wenn für einen mit dem Timeout festgelegten Zeitraum bestimmte Ereignisse ausgeblieben sind.
<b>Token</b>	Ein Token dient als Sicherheitsmaßnahme zur zusätzlichen Authentifizierung einer Aktion im Internet (beispielsweise dem Setzen eines neuen Passworts) und besteht meist aus einer Aneinanderreihung von Buchstaben und Ziffern.
<b>Torque Vectoring</b>	Mit dem „gezielt eingesetztem Drehmoment“ (so die wörtliche Übersetzung von „Torque Vectoring“) lenkt das Fahrzeug noch spontaner und direkter in die Kurve ein, außerdem bleibt es deutlich länger spurstabil.
<b>Touchscreen</b>	Ein Touchscreen (zu dt. berührungsempfindlicher Bildschirm) beschreibt einen Bildschirm, auf dem gleichzeitig Inhalte angezeigt werden können und Eingaben am Telefon durch Berührung erfolgen können.
<b>UDP</b>	UDP ist ein verbindungsloses Protokoll für Datenübertragungen zwischen zwei Anwendungen durch ein Netzwerk hindurch.
<b>UDP-Paket</b>	UDP zerlegt die zu verschickenden Daten in einzelne Datenpakete, die dann mit Zusatzinformationen versehen verschickt werden.
<b>UML</b>	Unified Modeling Language (dt. Vereinheitlichte Modellierungssprache) ist eine standardisierte, graphische Modellierungssprache, die zur Spezifikation, Aufbau und Konstruktion von Software genutzt wird. In ihr enthalten sind aufgrund der Vielfältigkeit von Softwareproblemen viele verschiedene Diagrammtypen, die sich zur Problembeschreibung beziehungsweise deren Lösung unterschiedlich gut eignen.
<b>Umrichter</b>	Hierbei handelt es sich um einen Stromrichter, der aus einer Wechselspannung eine in der Frequenz und Amplitude unterschiedliche Wechselspannung generiert.
<b>UMTS</b>	Universal Mobile Telecommunications System ist eine Übertragungstechnologie für die Mobilfunkübertragung von Daten (schneller als GPRS).
<b>virtueller Server</b>	Eine Servervirtualisierung ist eine Softwarelösung, um mehrere virtuelle Server (auch vServer) auf einer Servermaschine laufen zu lassen. Dabei wird nur vorgegeben, dass der virtuelle Server ein richtiger Server ist, der getrennt von den anderen Instanzen läuft. Man kann den virtuellen Server allerdings wie einen normalen behandeln. Bei dem unserem Team vorliegenden vServer wird vom Provider 1&1 das Betriebssystem Cent OS 6 emuliert.

**Voyage Linux**

Voyage Linux ist ein Debian-basiertes Linux-Betriebssystem für einen Embedded-PC.

**Webseite**

Eine Webseite ist ein meist interaktives Dokument, das auf einem Webserver liegt und aus dem Internet erreichbar ist.

**Webserver**

Ein Webserver ist ein Computer mit einer Serversoftware, der Dokumente beispielsweise an Webbrowser übermittelt.

**weiche Echtzeit**

Das System soll Daten in einem festgelegten Zeitrahmen zur Verfügung stellen, es muss aber nicht gewährleistet werden, dass es das immer tut.

# Abbildungsverzeichnis

3.1	Verteilungsdiagramm Gesamtsystem . . . . .	6
3.2	Constant-Block . . . . .	8
3.3	Repeating Sequence Interpolated - Block . . . . .	8
3.4	Inport-Block . . . . .	9
3.5	Outport-Block . . . . .	9
3.6	Subsystem . . . . .	9
3.7	Bussystem . . . . .	10
3.8	Convert-Block . . . . .	10
3.9	Bus to Vector - Block . . . . .	10
3.10	Terminator-Block . . . . .	11
3.11	Scope . . . . .	11
3.12	Display . . . . .	11
3.13	Gain-Block . . . . .	11
3.14	Block zur Erzeugung eines <i>uint32</i> -Datenstroms . . . . .	12
3.15	UDP-Block zur Einstellung der Netzwerkschnittstelle . . . . .	12
3.16	UDP-Block zum Senden eines <i>uint32</i> -Datenstroms . . . . .	13
3.17	Sequenzdiagramm Simulink-Modell . . . . .	13
3.18	Subsysteme des Signalgenerators . . . . .	14
3.19	Subsystem „Allgemeine Daten“ des Signalgenerators . . . . .	15
3.20	Subsystem „Temperaturen“ . . . . .	16
3.21	Referenzierung der Config-Datei . . . . .	16
3.22	Signalaufbereitung im Subsystem „daten_allgemein“ . . . . .	17
3.23	Aktivitätsdiagramm Embedded-PC . . . . .	19
3.24	Aktivitätsdiagramm vServer . . . . .	20
3.25	Aktivitätsdiagramm Anmeldevorgang . . . . .	22
3.26	Aktivitätsdiagramm Passwort vergessen - Vorgang . . . . .	24
3.27	Entwurf Nutzerdatenbank . . . . .	25
3.28	Entwurf Fahrzeugdatenbank . . . . .	26
A.1	Sequenzdiagramm Webseite . . . . .	40
A.2	Aktivitätsdiagramm Registrierungsvorgang . . . . .	41

# Anhang

## A.1 Datentypen in Simulink

Simulink unterstützt die folgenden von Matlab unterstützten Datentypen:

- *double* (double-precision floating point nach IEEE Standard 754 mit 64 bit Speicherbedarf pro Zahl)
- *single* (single-precision floating point nach IEEE Standard 754 mit 32 bit Speicherbedarf pro Zahl)
- *int8*, *uint8*, *int16*, *uint16*, *int32*, *uint32* (signed/unsigned 8-, 16- bzw. 32-bit integer)

Darüber hinaus kennt Simulink noch den Datentyp

- *boolean* (0 oder 1, wird intern als *uint8* behandelt)

Quelle: Angermann — Beuschel — Rau — Wohlfahrt:  
„MATLAB - Simulink - Stateflow: Grundlagen, Toolboxes, Beispiele“  
Oldenbourg Verlag München

## A.2 weitere Datentypen

Folgende Datentypen kommen in C++ oder MySQL-Datenbanken zu den bereits oben genannten Typen hinzu:

- Ganzzahldatentypen: *decimal*, *smallint*, *tinyint* (nach Genauigkeit absteigend sortiert)
- zeichenbasierte Datentypen: *char* (Abspeicherung eines Zeichens), *String* (Abspeicherung einer Zeichenfolge), *varchar* (Datenbankenäquivalent zu *String*), *binary* (Abspeicherung einer Kette von Binärwerten)

Weiterhin kann eine MySQL-Datenbank noch folgenden Datentyp behandeln:

- *bit* (Speicherung einer Binärzahl)

## A.3 UML-Diagramme

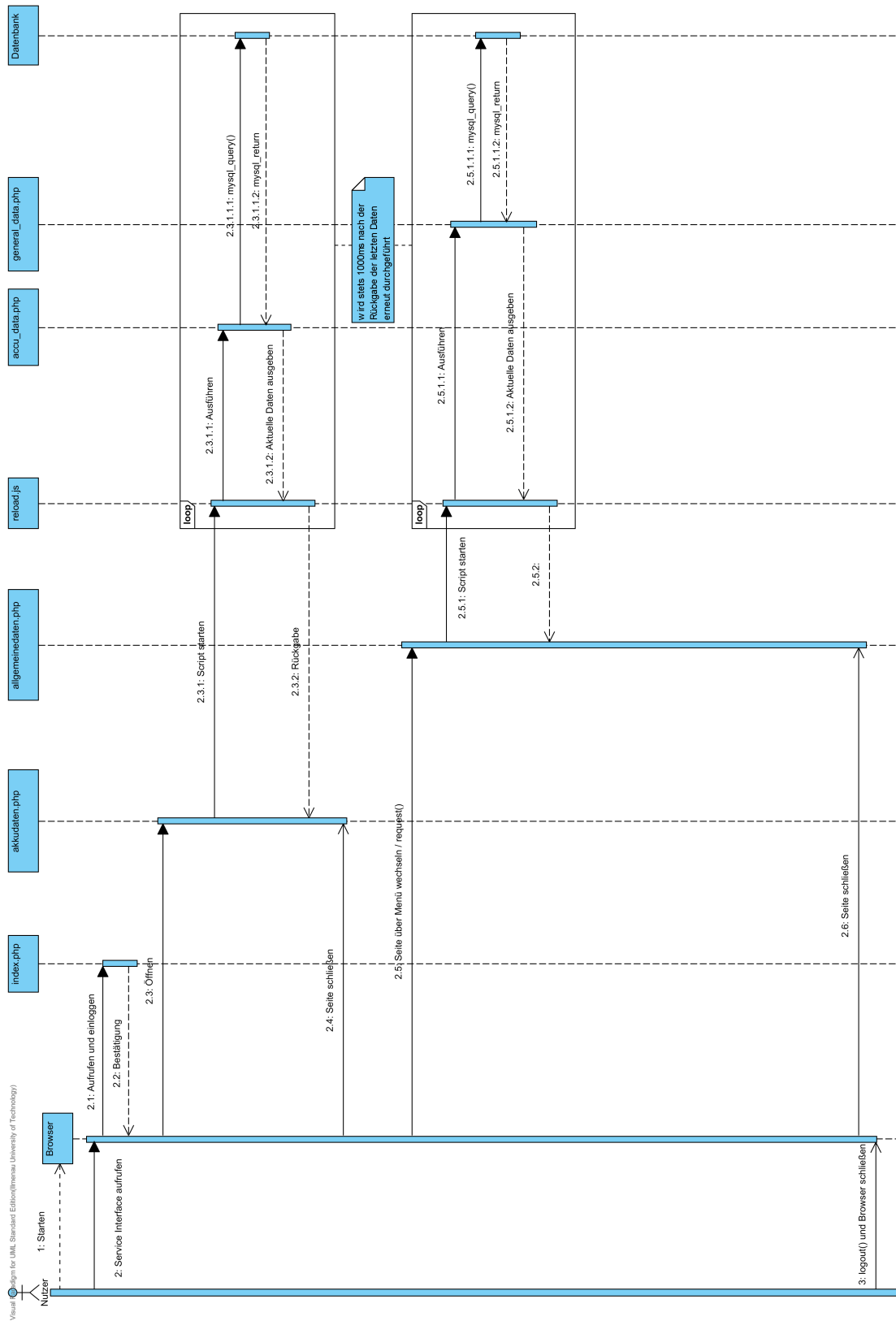


Abbildung A.1: Sequenzdiagramm der Webseite



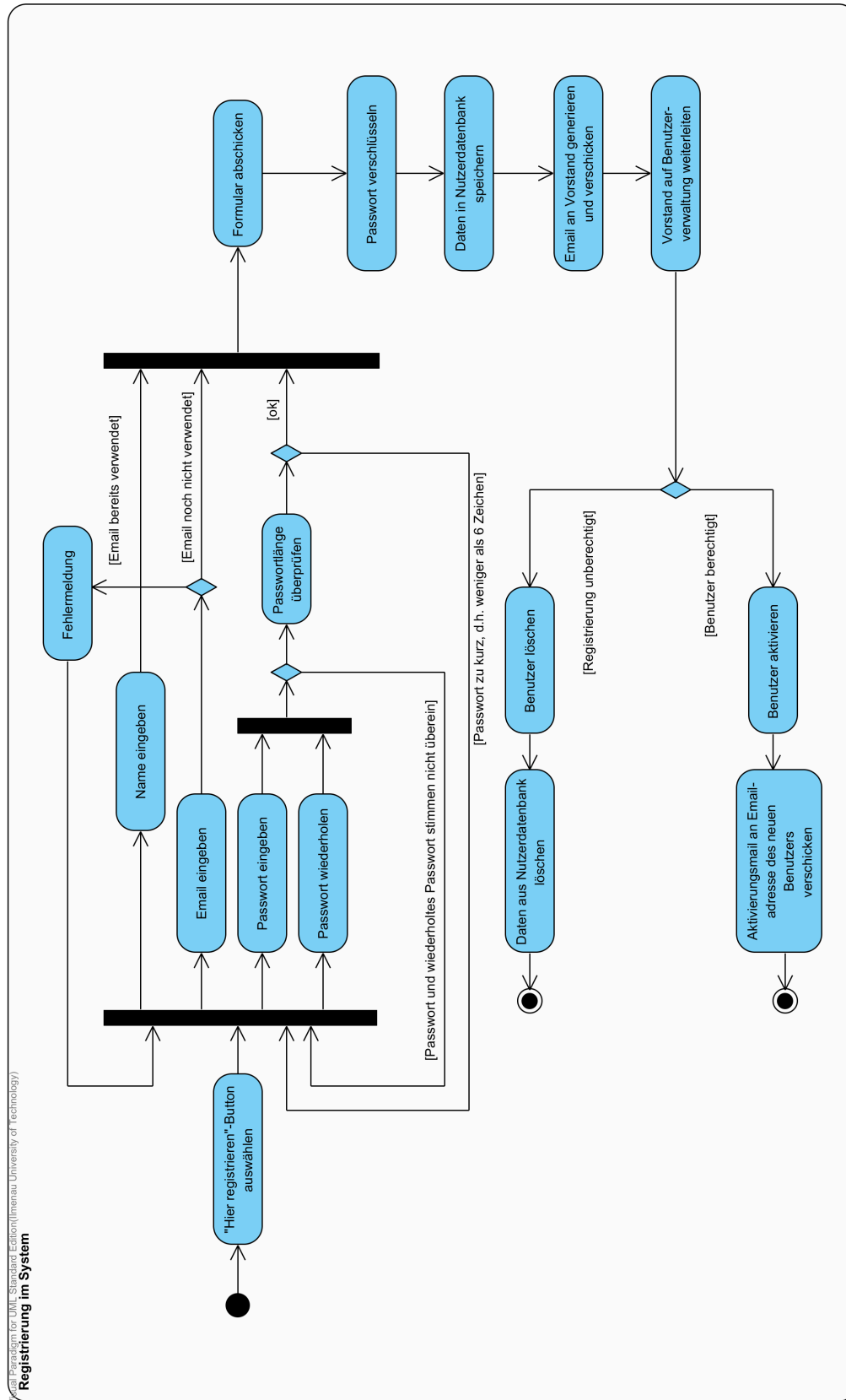


Abbildung A.2: Aktivitätsdiagramm des Registrierungsprozesses