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

1 Einleitung

Diese Software ermöglicht es dem Team StarCraft e.V. Daten von ihrem Formula-Student-Fahrzeug Daten in weicher Echtzeit auszulesen und zu visualisieren. Hierfür werden auf einer dSPACE MicroAutoBox II die Daten zu Blöcken gebündelt, per Ethernetverbindung an einen Embedded-PC gesendet und von diesem per UMTS/GPRS an eine Datenbank gesendet, die Daten der letzten 10 Stunden speichern soll. Die Darstellung erfolgt dann über eine Webseite, die die einzelnen Daten aus der Datenbank ausliest und darstellt. Die Webseite soll sich vor allem durch Übersichtlichkeit und leichte Bedienbarkeit auszeichnen, sowie ein Nutzersystem bereitstellen. Das Projekt ist daher viergeteilt in die Abschnitte MicroAutoBox II, Embedded-PC, vServer und Webseite, da in all diesen Abschnitten durch unser Team Arbeiten verrichtet werden. Im Folgenden wird erklärt, wie die Daten zwischen den einzelnen Abschnitten ausgetauscht werden und wie die Abschnitte dies bewerkstelligen.

2 Randbedingungen

Hier Text einfügen!

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.

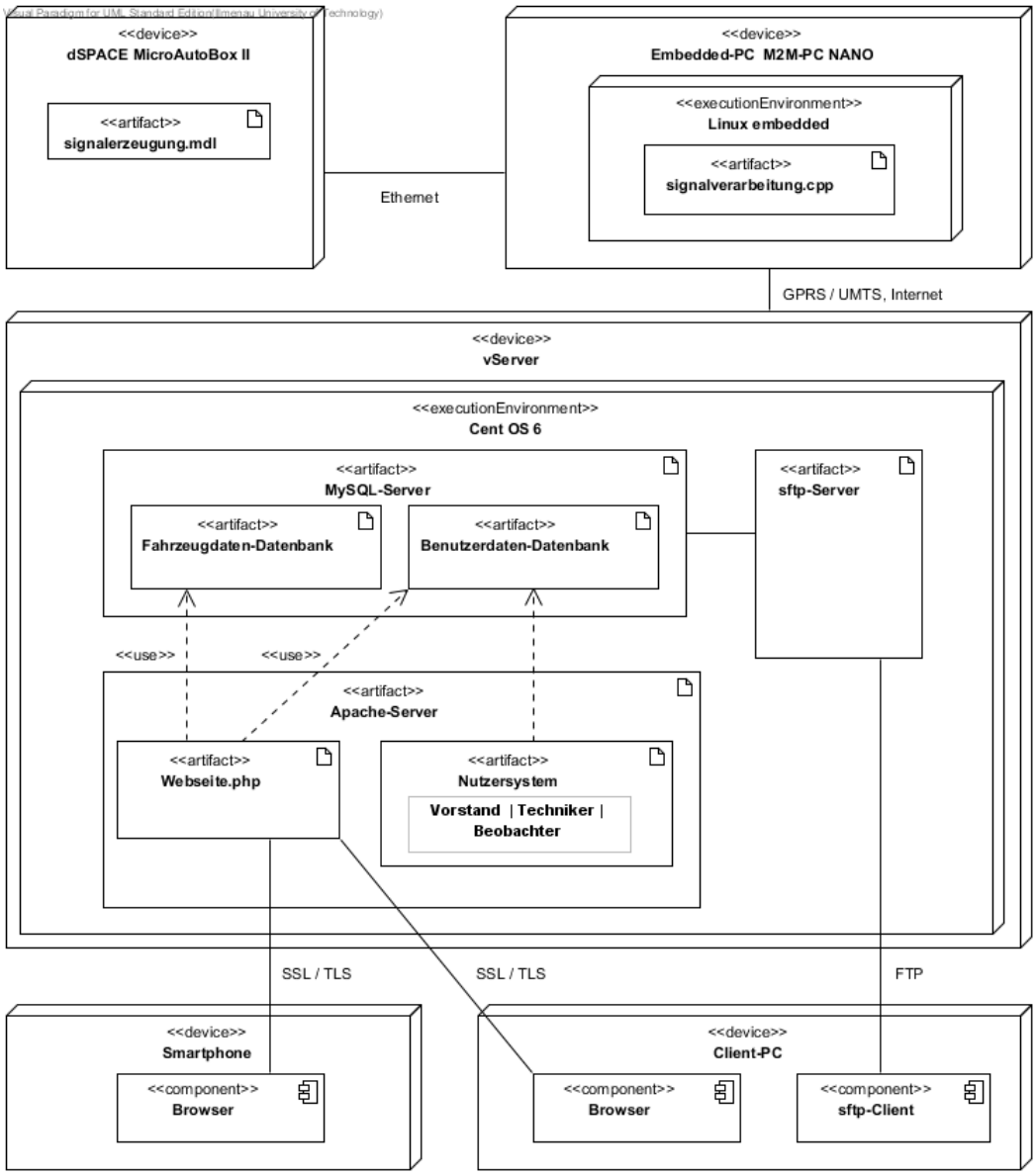

Abbildung 3.1: Gesamtsystem als Verteilungsdiagramm

3.1.1 dSPACE MicroAutoBox II

Bei der dSPACE MicroAutoBox II handelt es sich um ein Steuergerät mit Echtzeiteigenschaften, welches fest im Forumla-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 entgegennimmt, 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*. Für die eigentliche Übertragung zum virtuellen Server über UMTS* sind zwei in der Programmiersprache C++ selbst entwickelte Programme - das Server-Programm und das Client-Programm - zuständig. Beide Programme laufen eigenständig auf dem Embedded-PC und werden automatisch beim Systemstart ausgeführt. Ein von der eigentlichen Übertragung ausgeschlossenes und selbst entwickeltes Programm [namens AAAA] garantiert das Ausführen beider Programme und startet die Programme bei einem auftretenden Absturz oder schwerwiegenden Fehler neu. Dabei wird während des Vorgangs das Server-Programm jedem UDP-Packet des Datenstroms* ein Zeitstempel* hinzugefügt, wobei später ankommende Packete ignoriert und erst nach dessen Fertigstellung ein neues Packet behandelt werden **(Klären?)**. Der hinzugefügte Zeitstempel beinhaltet die Systemzeit des Embedded-PC zum Zeitpunkt des Hinzufügens und dient dazu, die Aktualität der Daten bei dem Eintreffen auf dem virtuellen Server festzustellen. Anschließend leitet das Programm die Daten an das Client-Programm weiter, welches die empfangenen Daten über eine mobile Breitbandverbindung mittels UDP-Sockets an den virtuellen Server weiterleitet.

3.1.3 Virtueller Server

Der virtuelle Server wird vom den Provider 1&1 bereitgestellt und ist jederzeit über eine feste IP* erreichbar. Darauf wird eine 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 eben diese 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

Benutzerdaten-Datenbank

Fahrzeugdaten-Datenbank

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

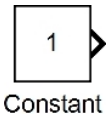

Abbildung 3.2: Constant-Block

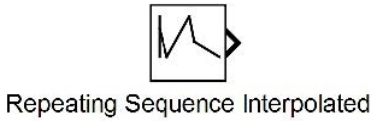

Abbildung 3.3: Repeating Sequence Interpolated - Block

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 X.Y. aufgeführten Datentypen zur Verfügung.

2) *„Repeating-Sequence-Interpolated" - Block*

   Im Gegensatz zum Costant-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

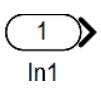

Abbildung 3.4: Inport-Block

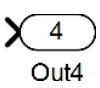

Abbildung 3.5: Outport-Block

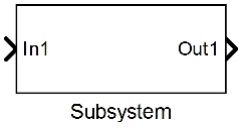

Abbildung 3.6: Subsystem

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 mit 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

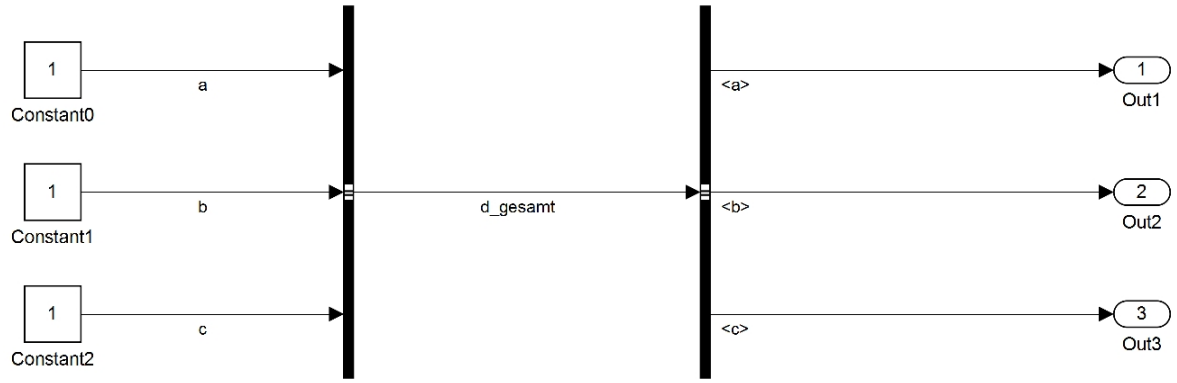

Abbildung 3.7: Bussystem bestehend aus einem Bus Creator (l.) und einem Bus Selector (r.)

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. X.Y. dargestellt aus dem Gesamtsignal d_gesamt wieder die Signale a, b und c herausgeführt.

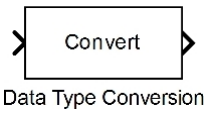

Abbildung 3.8: Convert-Block

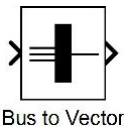

Abbildung 3.9: Bus to Vector - Block

Signal Attributes

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

Abbildung 3.10: Terminator-Block

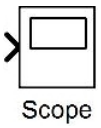

Abbildung 3.11: Scope

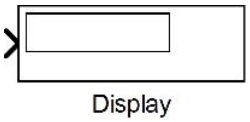

Abbildung 3.12: Display

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

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*

2) *„UDP Send" - Block*

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 per GPRS / UMTS an einen vServer 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.

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 verschieden Kategorien der jeweiligen Fahrzeugdaten repräsentieren.
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 X.Y 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 X.Y. gezeigt mittels Blöcken aus der Kategorie „Sources"(s. X.Y) 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

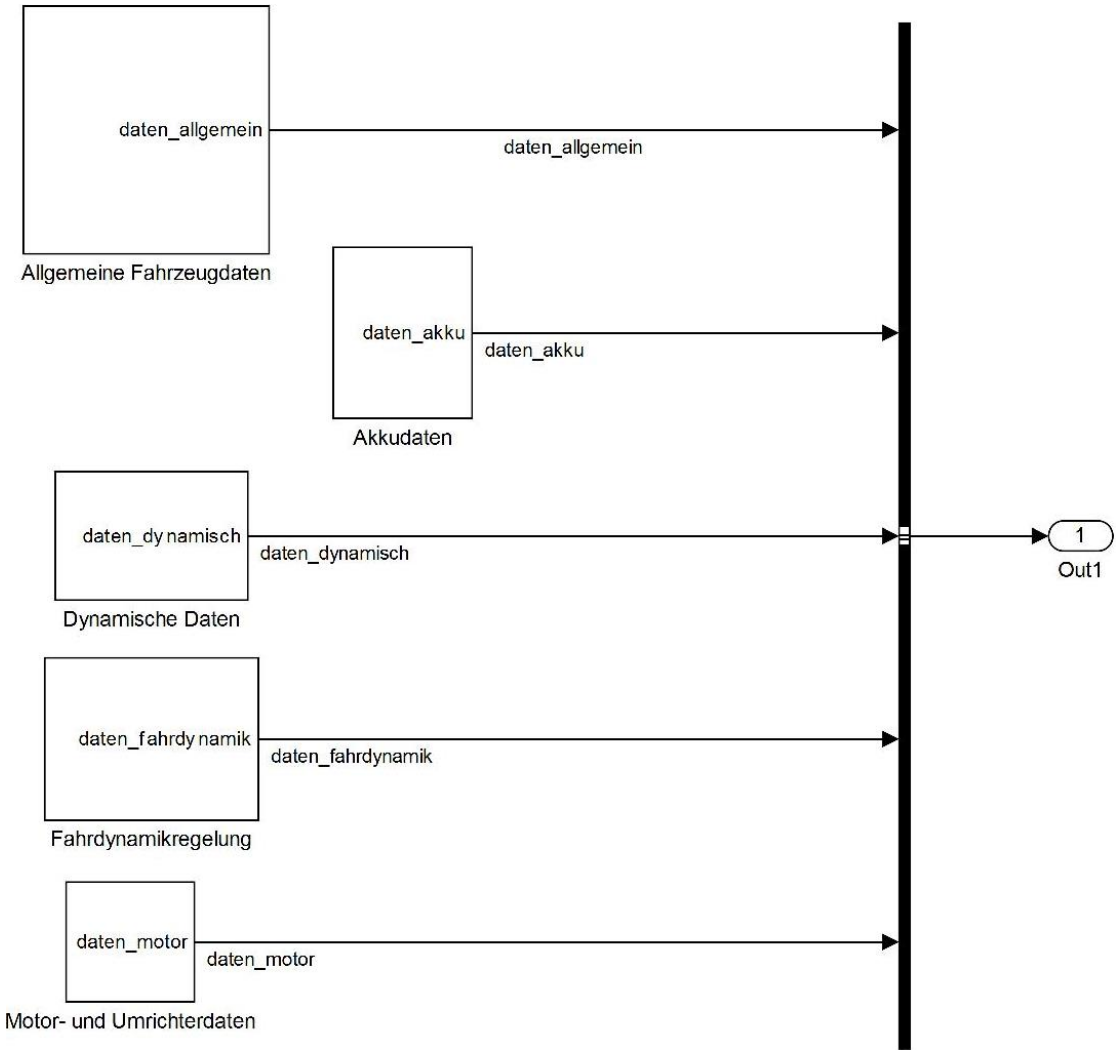

Abbildung 3.13: Subsysteme des Signalgenerators nach den Kategorien der Fahrzeugdaten

auf analoge Weise zu X.Y 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.

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. X.Y). Hierbei sollte jedoch beachtet werden, dass vor dem Compilieren des Simulink-Modells und der Implementierung ebendiesem auf der Micro-AutoBox 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

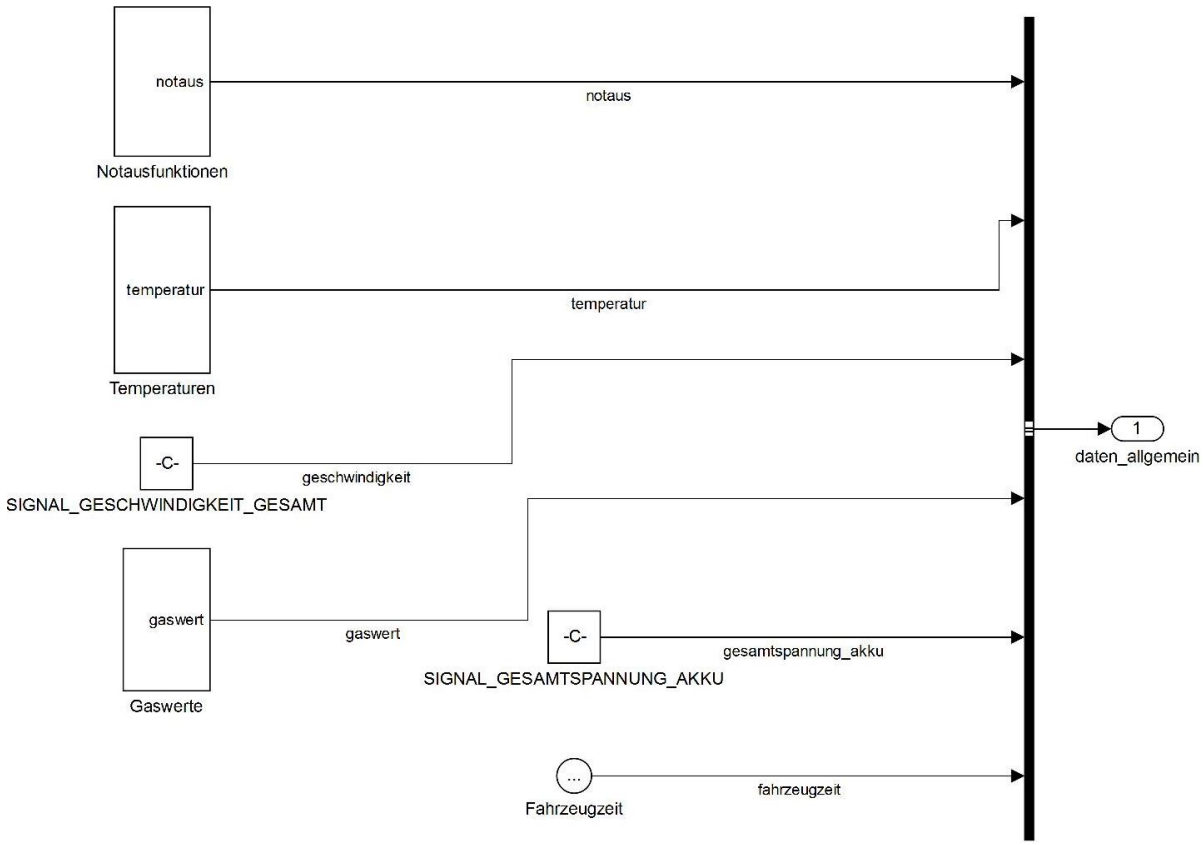

Abbildung 3.14: Subsystem „Allgemeine Daten" im Signalgenerator

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 .

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. X.Y) 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 (vgl. Abb. X.Y) zugeführt. Dort werden die nunmehr fünf Busarrays wieder über mehrere Busselektoren und Subsysteme hinweg in die 368 einzelnen Signale aufgelöst, welche somit einzeln aufbereitet werden können (s. Abb. X.Y). Während der Aufbereitung der Signale werden die

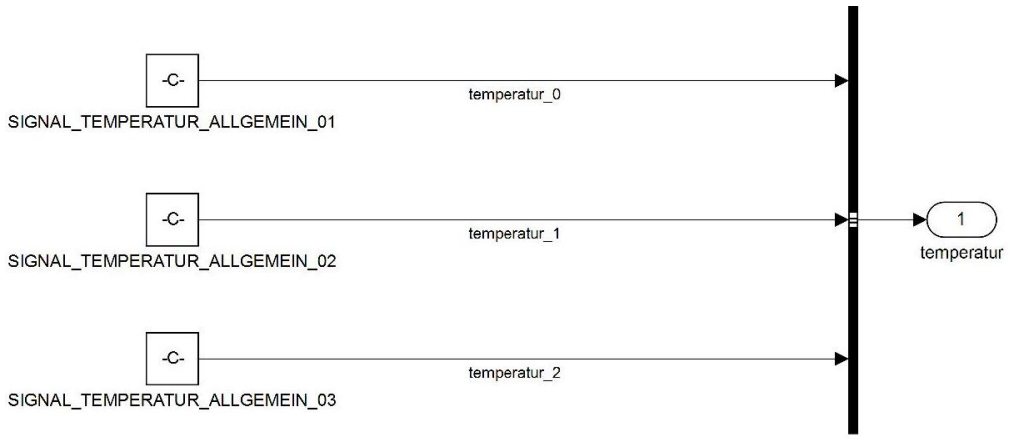

Abbildung 3.15: Signalerzeugung im Subsystem "Temperaturen"

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. X.Y) 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. X.Y) in den Datentyp *int16* konvertiert, wodurch somit 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. X.Y.) gegeben, welcher das Busarray in einen Vektor umwandelt. Anschließend wird dieser Vektor einem Enkoder (s. X.Y) zugeführt.

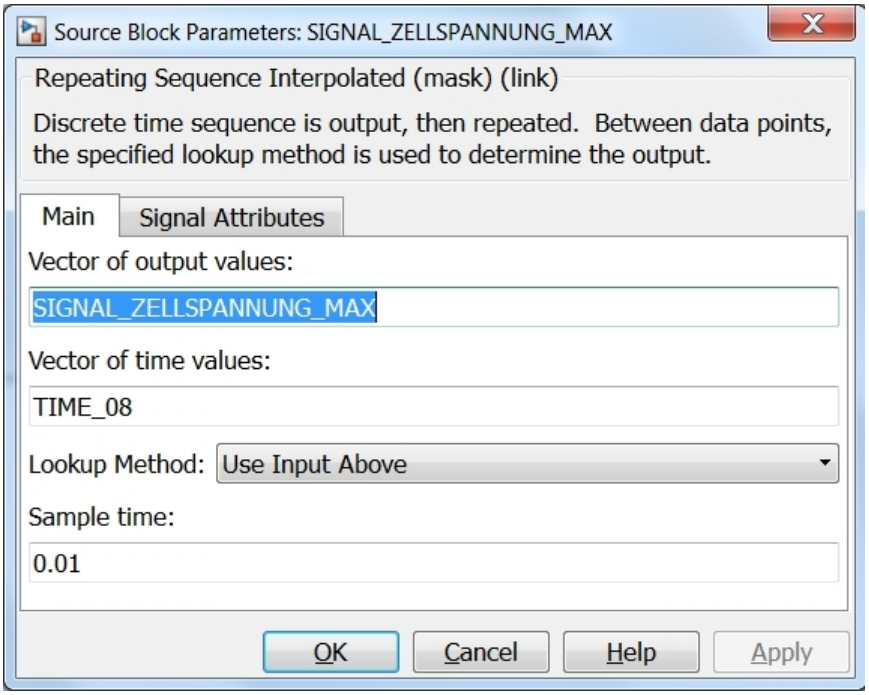

Abbildung 3.16: Referenzierung im RSI-Block

3.2.4 Enkoder-Block

Der Enkoder-Block basiert auf einer sFunction die von dSPACE implementiert wurde. Diese konvertiert einen Eingangsvektor in einen Datenstrom mit dem Datentyp *unit32*. Dieser Schritt ist notwendig um die Daten für die Übertragung mittels UDP vorzubereiten, da die dafür vorgesehenen Blöcke nur Daten dieses Typs übertragen können.

3.2.5 UDP-Schnittstelle

Die UDP-Schnittstelle wird mithilfe zweier Simulink-Blöcke realisiert. 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. Das Senden der Daten erfolgt über den Block ETHERNET_UDP_TX_BL1. Diesem wird ein Datenstrom vom Typ uint32 übergeben, sowie die Größe der zu übermittelnden Daten. Anschließend werden die Daten entsprechend der Konfiguration im ETHERNET_UDP_SETUP_BL1 Block versendet.

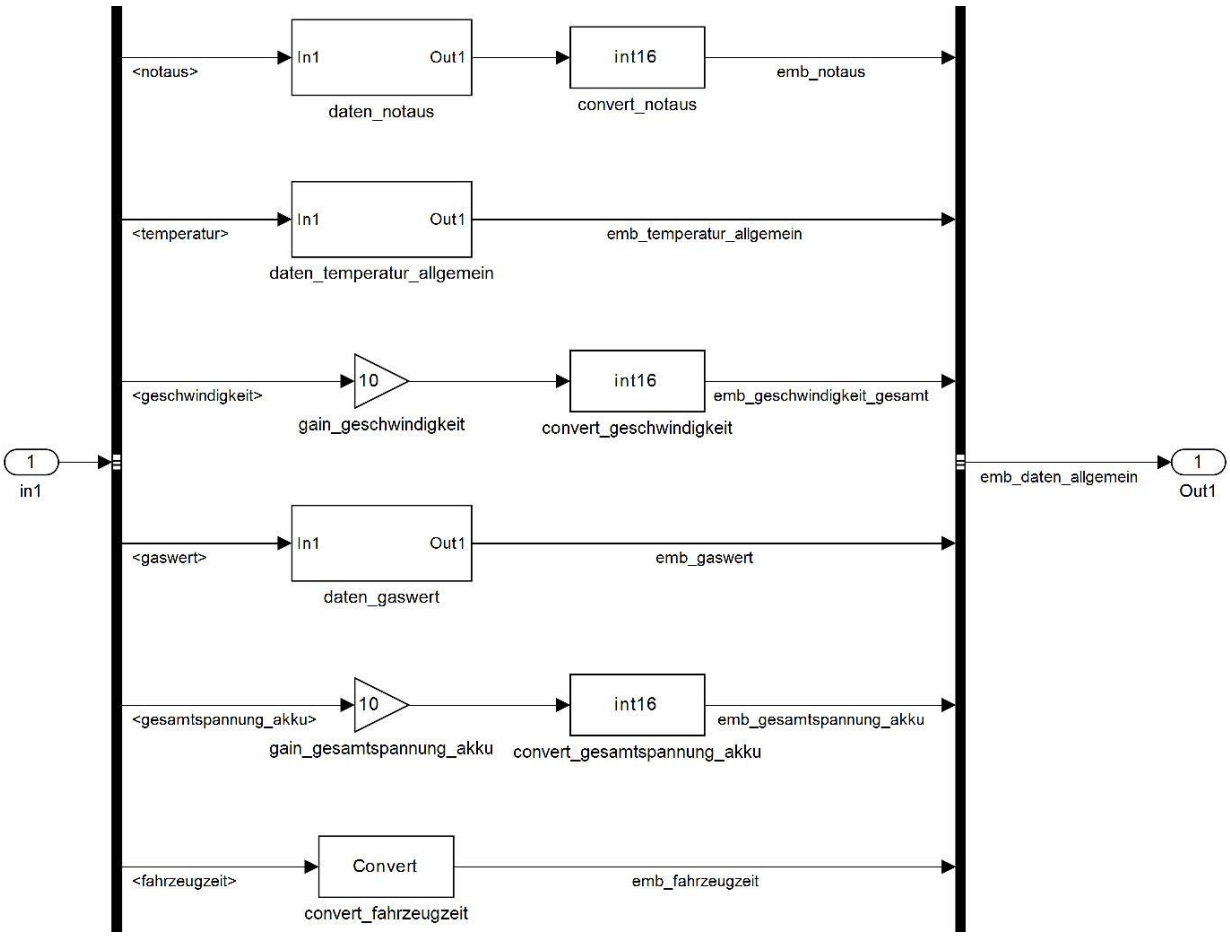

Abbildung 3.17: Signalaufbereitung im Subsystem „daten_allgemein"

3.3 Embedded-PC

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-Vebindung 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

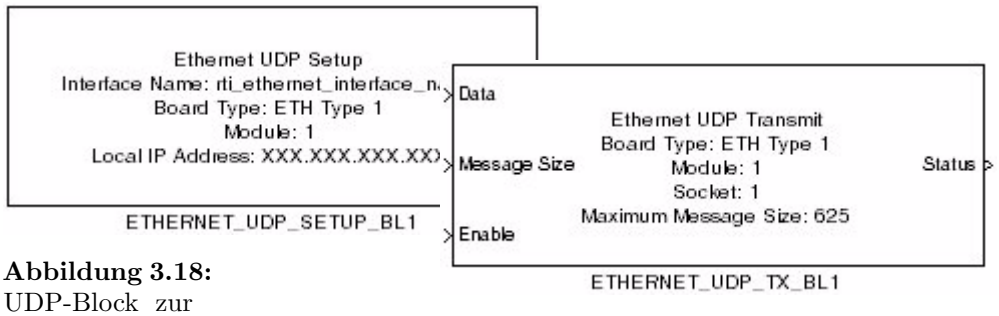

Abbildung 3.18: UDP-Block zur Einstellung der Netzwerkschnittstelle

Abbildung 3.19: UDP-Block zum Senden von uint32-Datenströmen

(SOCK_DGRAM) an den virtuellen Server über eine UMTS-Verbindung übertragen.

3.4 Virtueller Server

Im Folgenden wird die Funktionsweise des virtuellen Servers anhand eines Aktivitäts-Diagrammes modelliert.

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, sonder auch, dass zusätzlicher Zeitaufwand vermieden wird.

3.4.2 Abfragen 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 enkodieren

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) 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. Eine Fehlerkorrektur findet nicht statt. Die Daten liegen anschließend in einem Vektor bestehend aus int16-Werten vor.

3.4.4 Daten validieren

WOLLEN WIR DAS SICHER MACHEN? WAS WOLLEN WIR VALIDIEREN???

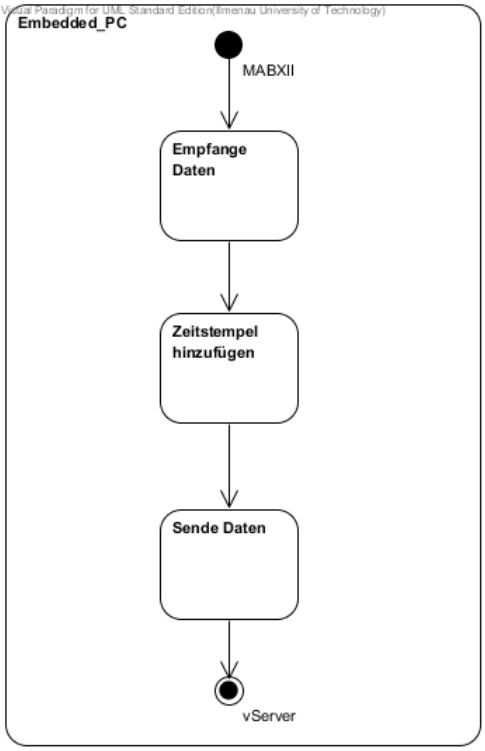

Abbildung 3.20: SOME TEXT

3.4.5 Daten verwerfen

Daten die veraltet, nicht dekodiert werden konnten oder die Validierung nicht bestanden haben werden verworfen.

3.4.6 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 im 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

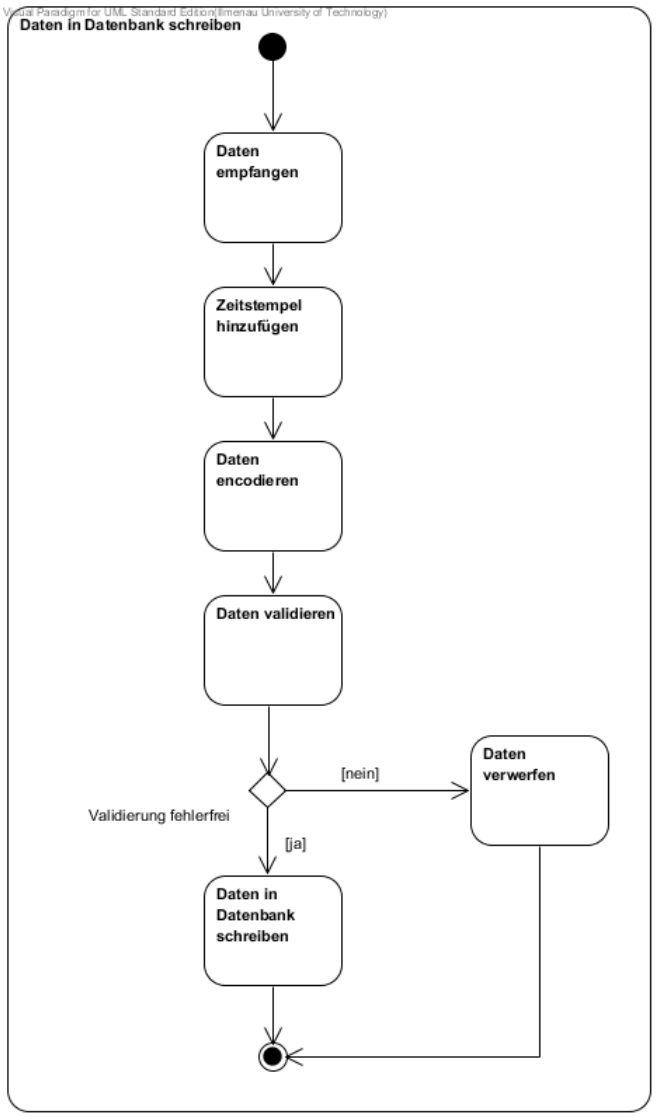

Abbildung 3.21: SOME TEXT

ermittelten Daten werden nun auf unterschiedlichster Art und Weise auf der Webseite dargestellt, sei es nun zum Beispiel 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.

3.5.1 Registrierung

Durch das Aufrufen der Seite des TeamStarcraft e. V. Teams lässt sich das Service Interface für das Formula Student Fahrzeug erreichen. Der Besucher erhält eine Aufforderung sich einzuloggen. Es können hierbei unterschiedliche Fälle eintreten:

- *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 Emailadresse und ein Password eingeben. Das eingegebene Passwort muss aus mindestens 6 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. Entsprechen alle Eingaben den Vorgaben 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 Email 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 Mail, die es an die eingetragene Emailadresse 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 Emailadresse und sein Passwort einzugeben. Durch einen Klick auf den Login-Button werden die eingegebenen Daten mit der Nutzerdatenbank verglichen. Sollte die Konstellation aus Emailadresse 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 äktiviert"). 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.

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 Emailadresse einzugeben. Durch die Bestätigung der Emailadresse durch den Benutzer prüft das Nutzersystem im Hintergrund, ob ein Eintrag zu dieser Emailadresse 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 Emailadresse verschickt. Auf der Aktivierungsseite wird der Nutzer erneut aufgefordert seine Emailadresse 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 "Benutzerverwaltungßur 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) oder ihre Rechtegruppenzugehörigkeit festzulegen bzw. zu ändern. 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 Benutzerdaten-Datenbank

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

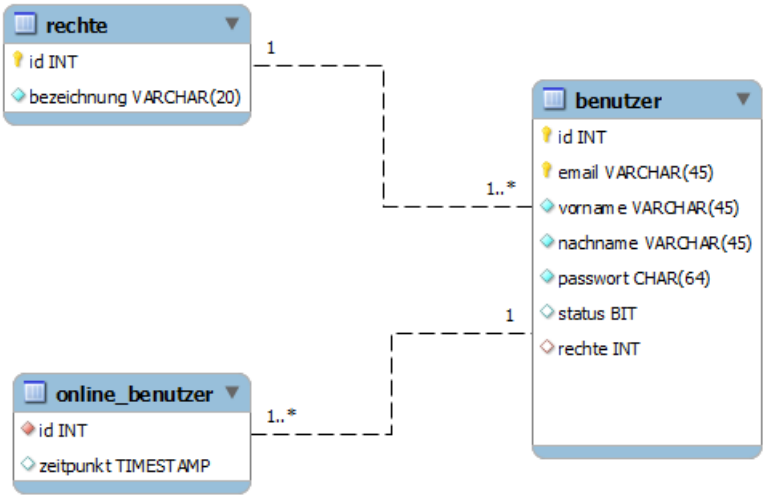

Abbildung 3.22: Entwurf der Fahrzeugdatenbank

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 Emailadresse zusammen bilden beide den Primärschlüssel zur eindeutigen Identfikation 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 "rechteïst 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 Fahrzeugdaten-Datenbank

Die zweite Datenbank beinhaltet die aus dem Fahrzeug entnommen 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. X.Y). 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:

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.

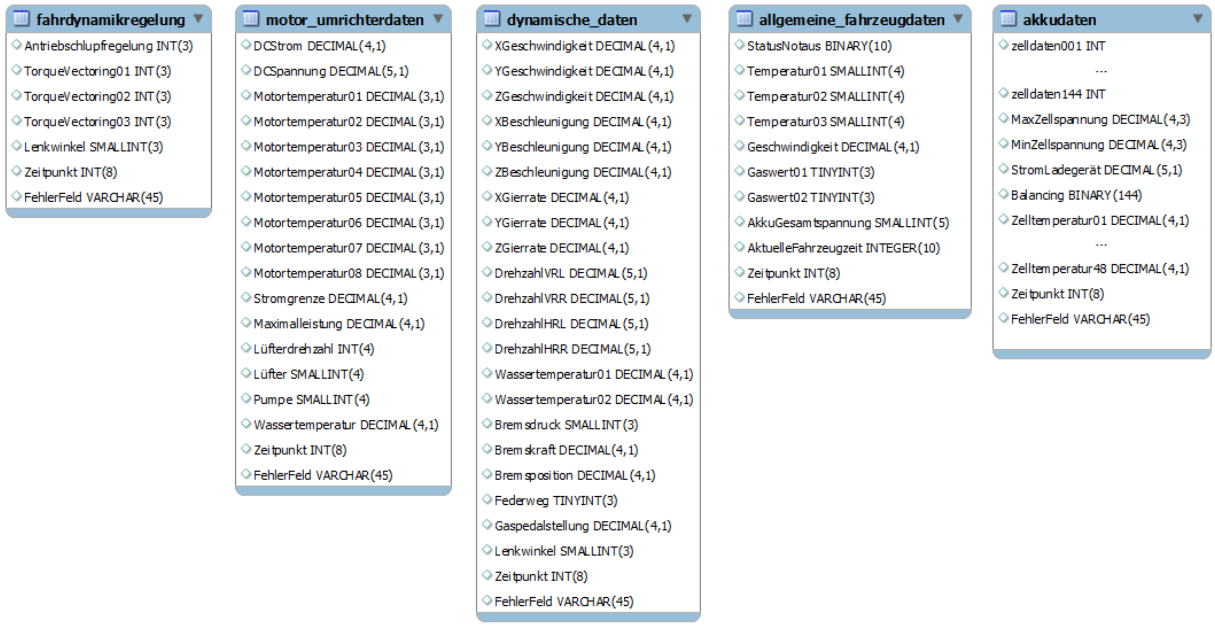

Abbildung 3.23: Entwurf der Fahrzeugdatenbank

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 zum Beispiel 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 Fahrzeugzeit.

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

Glossar

Abbildungsverzeichnis

Anhang